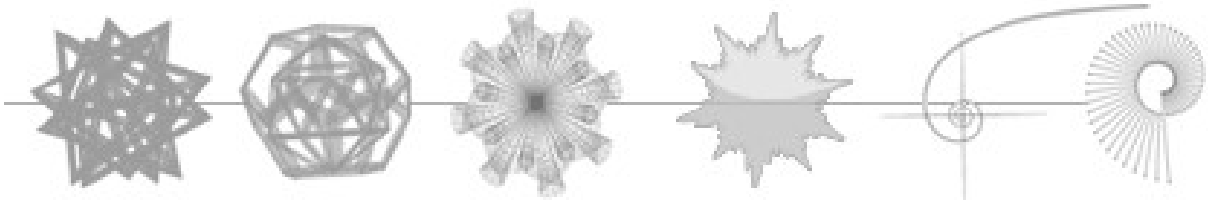
МИНИСТЕРСТВО НА ОТБРАНАТА

НВУ ‘В. ЛЕВСКИ’

Факултет „Артилерия, ПВО и КИС”



ДИСКРЕТНИ СТРУКТУРИ

Маргарита Василева

Шумен
2008

Учебникът по “Дискретни структури” има за цел да запознае студентите с фундаменталните понятия, методи и алгоритми на дискретната математика, които представляват теоретическа и методическа основа на информационните технологии. Изложеният материал позволява да се формира понятиен апарат и да се дадат знания по: теория на множествата и комбинаториката; общите и специални комбинаторни методи за решение на дискретни задачи; теория на размитите множества като теоретична основа на моделите на изкуствения интелект; основните дискретни структури: графи, дървета, мрежи, намиращи приложение в операционните системи, компютърните мрежи, структурите от данни, алгоритмите и др., някои екстремални задачи върху графи и алгоритми за решението им, мрежово планиране и управление на проекти.

Съдържанието е съобразено със съвременните тенденции за обучение на специалисти в областта на компютърните науки и отговаря на препоръките за преподаване на тази дисциплина, изложени в документа Computing Curricula 2001.

Учебникът е предназначен за обучение на студенти и курсанти в образователно-квалификационна степен “бакалавър” и „магистър” по специалностите: “Компютърни системи и технологии” и „Компютърна техника и технологии”.

ISBN 978 - 954 - 9681 - 32 - 1

МАРГАРИТА ВЛАДИМИРОВА ВАСИЛЕВА

ШУМЕН, 2008.

СЪДЪРЖАНИЕ

ПРЕДГОВОР

Фундаменталните обекти, които изучава математиката са математическите структури, построени върху множества. Наличието на непрекъснати и дискретни множества води до условно разделяне на математиката на непрекъсната (класическа) и дискретна.

Дискретната математика е съвкупност от математически теории, изучаващи свойствата на дискретните структури, които възникват както в математиката, така и в нейните приложения. Дискретните структури, представляващи абстрактни математически обекти, построени върху множества, са обект на изследване в дискретната математика. Тяхна определяща характеристика е дискретността, проявяваща се в строежа на обекта или в динамиката на неговото изменение.

Ролята и мястото на дискретната математика и необходимостта от нейното изучаване се определят от три основни фактора:

1. Дискретната математика предоставя формализирани средства за построяване и анализ на модели в съвременната наука.
2. Дискретната математика може да се разглежда като теоретически основи на информатиката.
3. Задачите на дискретната математика са тясно свързани с компютърната проблематика и естествено се представят чрез различни алгоритми.

Връзката между непрекъснатост и дискретност е обяснил Г. Кантор. Той разглежда непрекъснатите геометрични обекти (прави, повърхности, геометрични тела) и дискретните обекти (числа, логически величини, стъпки на алгоритъм и др.) като абстрактни обекти, построени върху обект, който не е нито дискретен, нито непрекъснат. Такъв обект е множеството, което може да има както

непрекъсната, така и дискретна природа. Дискретни са множествата, чиито елементи са изолирани, т.е за всеки елемент съществува околност, не съдържаща други елементи. Ако това не е изпълнено, множеството е непрекъснато. Тази несъществена разлика между непрекъснатите и дискретни множества води до принципно различни свойства на дискретните и непрекъснати функции и отношения върху множества.

Непрекъснатата математика включва онези математически теории, които се базират на идеята за граница и непрекъснатост. В дискретната математика се използват следните идеализации и свързаните с тяхното използване понятия:

- потенциална безкрайност;
- плътност (например плътност на множеството на рационалните числа);
- възможност за дефиниране на линейни наредби, в които се задават отношения като следващ елемент, предшестващ елемент и др.
- процедури като математическа индукция и изчерпващо търсене, базирани на отношението на съседство.

Към дискретната математика се отнасят дисциплините:

- теория на множествата;
- теория на графите;
- комбинаторен анализ;
- математическа логика;
- теория на автоматите;
- теория на формалните езици и граматики;
- теория на кодирането;
- теория на дискретните системи за управление;
- раздели от теория на числата, обща алгебра, изчислителна математика и др.

Учебникът по “Дискретни структури” има за цел да запознае студентите с фундаменталните понятия, методи и алгоритми на дискретната математика, които представляват теоретическа и методическа основа на информационните технологии и включва теми, които не се изучават в други дисциплини, но са необходими за пълноценната подготовка на специалисти в областта на компютърните науки. Изложеният материал позволява у обучаемите да се формира понятиен апарат и да се дадат знания по: теория на множествата и комбинаториката; общите и специални комбинаторни методи за решение на дискретни задачи; теория на размитите множества като теоретична основа на моделите на изкуствения интелект; основните дискретни структури: графи, дървета, мрежи, намиращи приложение в операционните системи, компютърните мрежи, структурите от данни, алгоритмите и др., някои екстремални задачи върху графи и алгоритми за решението им, мрежово планиране и управление на проекти.

Съдържанието е съобразено със съвременните тенденции за обучение на специалисти в областта на компютърните науки и отговаря на препоръките за преподаване на тази дисциплина, изложени в документа Computing Curricula 2001, разработен в рамките на международен проект от професионалните организации ACM и IEEE Computer Society [32, 33, 34].

Материалът е структуриран в пет отделни глави, всяка от които третира основни въпроси от съответните математически теории.

В първа глава **„Дискретни множества и модели”** се изучават елементи от теорията на множествата. Разгледани са основните понятия от теорията на множествата, начините за задаване на множества, операциите сечение, обединение, допълнение и декартово произведение. Въведено е понятието мощност на крайните и безкрайни множества и свързаните с него

теореми, позволяващи да се определят мощностите на натуралния ред на числата и на множеството от точки в интервала (0,1), служещи като еталон на изброимо и неизброимо множество съответно. В края на главата са разгледани основните методи за работа с множества: метод на математическата индукция, принцип на Дирихле и принципът за включване и изключване.

Във втора глава **„Крайни множества и комбинаторни обекти”** се разглеждат елементи на комбинаториката. Описани са основните комбинаторни конфигурации: вариации, комбинации и пермутации, дадена е комбинаторна интерпретация на правилото за сумата и правило за произведението. Разгледани са специалните комбинаторни методи: метод на рекурентните редици и метод на производящите функции. За илюстрация на теоретичния материал са използвани числата на Фибоначи и фракталите, намиращи приложение в програмирането и компютърната графика. Особено внимание е отделено на метода на изчерпващото търсене като най-общ комбинаторен метод за решение на NP-пълни задачи.

В трета глава **„Размити множества и модели”** са изложени основни понятия от теорията на размитите множества, представляваща формализиран апарат за моделиране в условията на неопределеност и построяване на размити модели в неструктурирани области.

В четвърта глава **„Теория на NP-пълните задачи”** се разглеждат основни моменти от теорията, която предоставя теоретична база за оценка на ефективността на алгоритмите и анализ на необходимите за решението на дадена задача изчислителни ресурси. Последователно са изложени основните понятия, класовете асимптотична сложност, теоремата на Кук, както и някои известни NP-пълни задачи. Включването на тази глава е свързано с необходимостта от оценка на временната сложност на някои алгоритми върху графи, както и с необходимостта от

създаване на теоретична основа за сравнение на алгоритми в дисциплината „Синтез и анализ на алгоритми”.

В пета глава **„Дискретни структури. Графи, дървета, мрежи"** се разглеждат основните понятия, задачи, методи и алгоритми от теория на графите. Особено внимание е отделено на Ойлеровите и Хамилтонови графи, свързани с известните задачи за Кьонингсбергските мостове и търговския пътник. Въведени са паросъчетанията като специален вид графи, описана е задачата за назначенията и унгарския метод за нейното решение.

В шеста глава **„Мрежовото планиране и управление”** е изложена същността на мрежовите модели. Даден е алгоритъм за построяване на мрежови графици. Като теоретична основа на мрежовото планиране е разгледан методът на критическия път.

В учебника са включени много решени примери и задачи за самоподготовка, които служат за илюстрация на теоретичния материал и могат да подпомогнат усвояването му. За да се направи връзка с професионалното направление, в словесното описание на задачите е използвана предимно компютърна терминология.

Учебникът отговаря по теми и съдържание на учебната дисциплина “Дискретни структури”, предназначена за обучение на студенти и курсанти в образователно-квалификационна степен “бакалавър” и „магистър” по специалностите: “Компютърни системи и технологии” и „Компютърна техника и технологии”. Дисциплина е от задължителния блок учебни дисциплини и се базира на знанията на обучаемите от средния курс и от дисциплината “Висша математика". Тъй като дискретните структури са фундаментална основа на информатиката, то дисциплината поставя теоретичните основи, върху които се изгражда по-нататъшната подготовка на обучаемите като софтуерни и хардуерни специалисти.

При разработката на учебника са използвани материали от лекциите на проф. В. А. Носов и доц. Д. М. Черняк от Московския

Държавен Университет „М. В. Ломоносов”, доц. В. Б. Гисин от Финансовата Академия на РФ, курса на А. Х. Шень и Н. К. Верещагин по аксиоматична теория на множествата, електронните курсове на Интернет Университета по Информационни Технологии, изключително пълната и професионално написана книга по алгоритми на П. Наков и П. Добриков, както и много други Интернет източници и достъпни печатни издания. Терминологията в пета глава е съобразена с разработения под ръководството на В. А. Евстигнеев и В. Н. Касьянов от Новосибирския Държавен Университет речник по теория на графите. Много от включените графични визуализации на различни дискретни структури са създадени със софтуерния продукт *Wolfram Demonstrations Project* [35], предназначен за динамична илюстрация на концепции в науката, технологиите, математиката, изкуството, финансите и други области. Част е от «Wolfram Mathematica” - най-пълната математическа софтуерна система за научни изследвания, инженерни разработки и обучение, разработвана от фирмата „Wolfram Research”. Продуктът е предназначен за свободно разпространение. Всички източници са включени в литературата и цитирани на съответните места.

В заключение искам да благодаря на рецензентите доц. д-р Начо Василев от Военна Академия „Г.С.Раковски” и доц. д-р инж. Станимир Станев от Шуменския Университет „Еп. К. Преславски” за любезното им съдействие, както и на колегите ми от катедра „Компютърни системи и технологии” на Националния Военен Университет за доброжелателното отношение и отзивчивост.

Маргарита Василева

глава

1

ДИСКРЕТНИ МНОЖЕСТВА И МОДЕЛИ

- *Теория на множествата. Основни понятия*
- *Крайни и безкрайни множества*
- *Задаване на множества*
- *Известни множества*
- *Операции над множествата*
- *Свойства на операциите върху множества*
- *Множества и изображения*
- *Мощност на безкрайните множества*
- *Изброими и неизброими множества*
- *Метод на математическата индукция*
- *Принцип на Дирихле*
- *Принцип за включване и изключване*

1.1. ТЕОРИЯ НА МНОЖЕСТВАТА - УНИВЕРСАЛЕН ЕЗИК НА СЪВРЕМЕННАТА НАУКА

“Днес ние знаем, че ако говорим логически, можем да изведем почти цялата съвременна математика от един източник - теория на множествата.

Н. Бурбаки

Създаването на теорията на множествата е връх в процеса на формализация на математическите науки. Тя е единна теория, предоставяща формализми за описание на обекти не само във всички други математически теории, но и в цялата съвременна наука. Терминологията в теория на множествата е основа на съвременния математически език.

Като математическа дисциплина теорията на множествата е създадена от немския математик Г. Кантор (1845-1918) и понякога се нарича “наивна”, поради използване на неформализирани, интуитивно ясни представи. Липсата на достатъчна строгост е предпоставка за вътрешна противоречивост на теорията и води до възникване на някои парадокси, свързани с множествата. През 1901 г. Бертран Ръсел открил парадокс в теория на множествата, наречен *антиномия на Ръсел*. Това събитие разтърсило и отрезвило всички, занимаващи се с изграждането на основите на математиката, и принудило учените да преразгледат и „*формализират много понятия, които до момента са изглеждали очевидни и не нуждаещи се от формално описание*” [4]. Хилбет намира изход, който запазва основното съдържание на Канторовата теория. Той предлага множествата да се разглеждат като обекти, удовлетворяващи някакви аксиоми, които се избират по такъв начин, че не допускат възникване на противоречия.

Кризата, предизвикана от парадокса на Ръсел, създава предпоставки за развитие на теорията. Най-сериозните разработки са извършени през първата половина на XX век и са насочени към създаване на по-строги аксиоматики, които не допускат съществуването на парадоксалното множество на Ръсел. Цермело (*Zermelo*) и Френкел (*Fraenkel*) създават така наречената *аксиоматична теория на множествата*, в която са формализирани основните понятия и построения. Тази теория се нарича ZF теория в чест на нейните създатели. Ако към аксиоматиката на ZF теорията добавим аксиомата за избор (choice), получаваме ZFC теория на множествата [28].

В средата на 20 век **Никола́ Бурбаки́** (*Nicolas Bourbaki* - общ псевдоним на математическа общност от 30 математика, най-известните от които са Клод Шевалие, Андре Вейл, Анри Картан, Жан Диедонне, Лоран Шварц, Шолем Манделброт) формализират съвременната математика, като поставят в основата и формалната теория на множествата. В продължение на 25 години те издават поредица от математически трактати, в които всички доказателства са представени с изрядна строгост. Главните категории в изследването на Бурбаки са „формалните структури” в цялата математика.

Интензивните изследвания довеждат до получаване на значими резултати и възникване на нови научни направления. Някои математици (Брауер, Ръсел, Уайтхед) търсят разрешение на кризата в ограничаване на правилата на логиката и в методите на извършване на математическите доказателства. През 1932 г. Курт Гьодел доказва, че всяка достатъчно сложна аксиоматична теория е непълна и нейната непротиворечивост е недоказуема. Алън Тюринг полага основите на формалната теория на алгоритмите,

формализирайки математически определенията за компютър и алгоритъм. През 1939 г. Алонсо Чърч забелязал универсалната идея за изчислимост.

Тези открития имат фундаментално значение за формализацията на математиката и поставят началото на научни дисциплини с голямо практическо значение.

1.2. ОСНОВНИ ПОНЯТИЯ

Понятието множество е едно от фундаменталните и най-общи математически понятия, което не се определя чрез други понятия, а се описва и обяснява. Съдържанието и смисълът му се разкриват при изучаване на теорията на множествата.

Съгласно Кантор, ***множеството*** X е съвкупност от обединени по някакъв признак определени и ясно различими обекти на нашата интуиция или интелект, които възприемаме като едно цяло. Тези обекти се наричат ***елементи*** на множеството. Всяко множество напълно се определя от своите елементи. От това определение следва, че:

Множеството, като съвкупност от обекти, се разглежда като отделен обект. В говоримите езици съществуват понятия както за отделните елементи, така и за техните съвкупности. Например множеството от студенти може да се разглежда като един обект – група, множеството от птици – ято, множеството от избиратели – електорат, съвкупността от компютърни компоненти – хардуер, съвкупността от програми – софтуер, и др.

Елементите на множеството са обекти от произволно естество, продукт на нашата интуиция или интелект. Могат да бъдат както реално съществуващи физически обекти, така и абстрактни

построения. Не се поставят ограничения върху вида на елементите, техния брой или свойства. Множеството може да съдържа книги, компютърни компоненти, точки, прави, геометрични обекти... В частност, елементите на дадено множество могат да бъдат също множества. Например изчислителната система разглеждаме като съвкупност от хардуер (множество от компютърни компоненти) и софтуер (множество от програми).

Определени означава, че ако е дадено множество и някакъв обект, съществува начин да се определи дали обектът принадлежи, или не принадлежи на множеството.

Различими означава, че за всеки два обекта, разглеждани като елементи на дадено множество, съществува начин да се определи дали те съвпадат, или са различни.

Множествата обикновено се означават с големи латински букви – A, B, X, Y…, а техните елементи – с малки - a, b, x, y…

1.2.1. Принадлежност на елемент към множество

Отношението ***принадлежност*** е отношение между множествата и техните елементи. Множествата се състоят от елементи. Означаваме $x \in A$, ако елементът x принадлежи на множеството A, т.е. A съдържа елемента x. Ако елементът x не принадлежи на множеството A (A не съдържа x), използваме означението $x \notin A$. Например процесорът принадлежи на хардуера и не принадлежи на софтуера. Ако A *е* множеството на целите числа, които се делят на 5, то $5 \in A$, $10 \in A$, но $7 \notin A$.

Празно множество се нарича множеството, което не съдържа нито един елемент. Означава се чрез символа $\varnothing$. За произволен елемент x е в сила $x \notin \varnothing$.

1.2.2. Подмножества

Понятието подмножество служи за отразяване на отношения между множества. Множеството A е ***подмножество*** на множеството B, ако всеки елемент на A принадлежи и на B. В такъв случай казваме, че множеството A се съдържа (се включва) в множеството B, и означаваме $A \subseteq B$, или B съдържа A, което означаваме чрез $B \supseteq A$. Отношението на включване притежава следните свойства:

- рефлексивност: $A \subseteq A$;

- симетричност: ако $A \subseteq B$ и $B \subseteq A$, то $A = B$;

- транзитивност: ако $A \subseteq B$ и $B \subseteq C$, то $A \subseteq C$.

Ако множеството A не се съдържа в множеството B, използваме означението: $A \not\subset B$. За да докажем, че $A \not\subset B$, е достатъчно да намерим елемент на множеството A ($a \in A$), който не принадлежи на множеството B ($a \notin B$).

За произволно множество $A \neq \emptyset$ празното множество $\emptyset$ и самото множество A са негови подмножества, т.е. $\emptyset \subseteq A$ и $A \subseteq A$. Подмножествата на множеството A, различни от $\emptyset$ и A, се наричат ***собствени подмножества*** на A.

Множеството A е собствено подмножество на множеството B, ако $A \subseteq B$ и съществува поне един елемент $x \in B$, такъв, че $x \notin A$. В този случай се използва означението за строго включване $A \subset B$, или $B \supset A$. Понякога в литературата не се съблюдава различието на знаците за операциите за включване $\subset$ и $\subseteq$, тъй като се предполага, че използваната операция е смислово ясна.

За операциите $\subset$ и $\subseteq$ са в сила следните твърдения:

Ако $A \subseteq B$ и $B \subset C$, то $A \subset C$.

Ако $A \subset B$ и $B \subset C$, то $A \subset C$.

Всеки елемент на множеството A определя негово подмножество, т.е., ако $a \in A$, то $\{a\} \subseteq A$.

В теоретичните постановки се приема, че всички множества са подмножества на ***универсалното множество*** U, наричано още *универсум.*

Чрез 2^A се означава множеството на всички подмножества на множеството A. Ако A = {a, b, c}, то:

$2^A = \{\varnothing, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$.

Всяко крайно множество A *с* n *е*лемента, притежава 2^n подмножества, което ще бъде доказано по-късно.

1.2.3. Равенство на множества

Отношението “равенство” е отношение между множествата. Две множества A и B са ***равни*** (A = B) тогава и само тогава, когато съдържат едни и същи елементи. В противен случай множествата са ***различни*** ($A \neq B$).

Множествата {0, 1}, {1, 0} и {0,1,1} са равни, защото съдържат едни и същи елементи, а множествата {{0,1}} {0,1} не са равни, тъй като първото съдържа един елемент {0,1}, който е множество, а второто е двуелементно множество от числа.

Отношението “***равенство***” е отношение на еквивалентност и притежава свойствата *рефлексивност*, *симетричност* и *транзитивност*.

Ако едновременно $A \subseteq B$ и $B \subseteq A$, то множествата A и B съдържат едни и същи елементи и следователно A = B. Равенството на множества се доказва чрез доказване на включванията $A \subseteq B$ и $B \subseteq A$, т.е. доказваме следното:

1. ако $a \in A$, то $a \in B$;

2. ако $a \in B$, то $a \in A$.

Пример 1.1. Нека X е множеството на всички положителни четни цели числа, а У – множеството на положителните цели числа, представими във вид на сума от две положителни нечетни числа. За да докажем, че X = У, постъпваме по следния начин:

1. Нека $x \in X$, тогава $x = 2m = (2m-1)+1$, т.е. $x \in У$.

2. Нека $x \in У$, т.е. $x=(2p-1)+(2q-1) = 2(p+q-1)$, т.е. $x \in X$.

Следователно X = У.

ЗАДАЧИ ЗА САМОПОДГОТОВКА

1. Съвпадат ли множествата $\varnothing$, $\{\varnothing\}$, $\{\varnothing, \{\varnothing\}\}$?
2. Равни ли са множествата A и B и ако не са равни, защо?

   A = {2, 4, 6}, B = {2, 4, 7};

   A = {1, (2, 3), 4, 5, 6}, B = {1, 2, 3, 4, 5, 6};

   A = {1, 2, 3, 1, 2, 3}, B = {1, 2, 3};

   A = {1, {2, 3}, 4},B = {1,{3, 2}, 4};
3. В какво отношение се намират множествата A, B, C?

   A = {1, 3}; B = {x: x - нечетно число}; C = {x: x-4x+3=0};

   A = {2, 5}; B = {x; x - цяло число }; C = {x: x- 7x +10=0}.
4. Ако X и Y са множества, кои от изразите са неверни и защо?

   $1 \in \{a, b, \{1\}\}$;

   $X \in \{a, b, \{x\}\}$;

   $\{X, Y\} \in \{x, y\}$;

   $\{X, Y\} \in \{x, y, \{x, y\}\}$;

   $X \in \{a, b, x\}$;

   $X \in \{X, Y\}$.

5. Свързани ли са множествата A и B с отношение на включване?

   A = {a, b, d}, B = {b, d, a, c};

   A = {a, c, d, e}, B = {a, c, e};

   A = {c, d, e}, B = {a, c};

   A = {a, (c, d), e}, B = {a, e, (c, d), k}.

1.3. КРАЙНИ И БЕЗКРАЙНИ МНОЖЕСТВА

Всички множества се делят на две големи групи: крайни и безкрайни.

Множествата, съдържащи краен брой елементи, се наричат ***крайни*** множества. Броят на елементите на крайното множество A се означава чрез |A| и се нарича ***мощност*** на множеството. При това, не се интересуваме, дали този брой е известен, или не, важното е, че съществува. Очевидно, $|\varnothing| = 0$. Ако A съдържа n елемента, то |A| = n. Например множеството от решения на квадратното уравнение $x^2 + px + q = 0$ е крайно. Мощността му |A| = 0 при D < 0, |A| = 1 при D = 0 и |A| = 2 при D > 0.

Ако множествата A и B са равни, то мощностите им съвпадат, т.е. |A| = |B|. В такъв случай казваме, че множествата са ***равномощни***. Мощност на множество и равномощни множества са две различни понятия. Можем да знаем, че две множества са равномощни, и въпреки това да не знаем тяхната мощност. Например знаем, че двама души разполагат с еднакъв брой банкноти по 10 лева, но не знаем колко е броят на банкнотите. Ако A е подмножество на B, $A \subseteq B$, то очевидно за мощностите им е изпълнено: $|A| \leq |B|$.

Ако A и B са крайни множества, за мощностите им е в сила една и само една от следните релации:

$|A| = |B|$, или $|A| < |B|$, или $|A| > |B|$.

Безкрайни са множествата, които съдържат безкраен брой елементи. Множеството от решения на квадратното неравенство $x^2+px+q \leq 0$ е крайно при $D \leq 0$ и безкрайно при $D > 0$.

ЗАДАЧИ ЗА САМОПОДГОТОВКА

1. Информацията в компютърните системи се представя (кодира) чрез определен брой двоични разряди. Определете мощността на множествата от кодове, в случай че се използва 8-битов (16-битов, 32-битов, 64-битов, ...) двоичен код.
2. Какви са следните множества:

   A – множеството на всички четни числа;

   B – множеството на планетите на Слънчевата система;

   C - множеството от решения на уравнението (5x-3) = 0?
3. Да се докаже, че множеството на всички подмножества на дадено крайно множество от n елемента съдържа 2^n елемента.

1.4. ЗАДАВАНЕ НА МНОЖЕСТВА

1.4.1. Чрез изброяване на елементите

Крайните множества могат да се зададат чрез изброяване на елементите им. Например:

A = {1, 3, 5, 7, 9} – множество от нечетните десетични цифри;

B = {-1, +1} – множество от решенията на квадратното уравнение $x^2 – 1 = 0$;

C = {0, 1, 2, 3, 4, 5, 6} – множество от остатъците при деление на цели числа на 7;

D = {MB, CPU, RAM, SVGA} – множество от компютърни компоненти.

Ако крайното множество X се състои от елементите x_1, x_2, x_3 … x_n, ще го означаваме чрез: $X = \{x_1, x_2, \ldots x_n\}$.

По такъв начин могат да се задават и безкрайните множества, ако е известен алгоритъмът за пораждане на елементите им. Например:

A = {2, 4, 6, 8, …2n, ….} – множеството на четните числа;

B = {2, 4, 8,...2^n ...} – множество, съдържащо степените на 2;

C = {0, 1, 4, 9, 16, …n^2…} – множество от квадратите на целите числа.

1.4.2. Чрез указване на свойство, което елементите удовлетворяват

Множеството може да се зададе като съвкупност от елементи на известно множество, удовлетворяващи някакво свойство, например множество на четните числа се състои от цели числа, които се делят на две.

В общия случай, при зададено свойство P и множество X, A се дефинира като множество от всички елементи на X, притежаващи свойството P. Използват се следните означения:

$A = \{x \mid x \in X$ и притежава свойството P $\}$;

$A = \{x \in X \mid P(x)\}$;

$A = \{x \mid P(x)\}$.

Например множеството на числата, които се делят на три, може да бъде зададено по един от следните начини:

A = {x | x – цяло число, което се дели на 3};

A = {x $\in$ Z | x се дели на 3}, Z - множество на целите числа.

Празното множество $\varnothing$ може да бъде зададено чрез:

$\varnothing = \{x \in X \mid x \neq x\}$ или $\varnothing = \{x \mid x \neq x\}$.

Прилагането на тази схема води до парадокса на Ръсел, свързан с формално построимото множество от всички множества:

A = {x | x – множество}.

Парадокс на Ръсел

Парадоксът на Ръсел е свързан с една формално построима конструкция – множество, което съдържа всички мислими множества. Възниква въпрос, съдържа ли това множество себе си? Положителният отговор води до възникване на абстрактни конструкции с рекурсивно влагане на обекти с неограничена дълбочина, които могат да имат теоретично значение, но съмнителна практическа полезност. Множествата, които най-често разглеждаме, не съдържат себе си като свой елемент. Например множеството на естествените числа не е естествено число и, следователно, не е елемент на N, т.е. N не съдържа себе си в качеството на свой елемент. В теоретичен план обаче можем да разглеждаме множества, които се явяват свои елементи.

Формалното изложение на парадокса на Ръсел се състои в следното:

Множеството A ще наричаме ***правилно***, ако не съдържа себе си. В противен случай A се нарича ***неправилно*.** Нека R е множеството на всички правилни множества:

$R = \{x \mid x \notin x\}$.

Съгласно определението произволно множество $A \in R$ тогава и само тогава, когато $A \notin A$. Ако вместо A приложим това твърдение върху R, то ще получим парадокс: $R \in R$ тогава и само тогава, когато $R \notin R$.

Лесно се доказва, че R не е нито правилно, нито неправилно.

Допускаме, че R е правилно множество. Тогава то не трябва да съдържа себе си, т.е. $R \notin R$. Но съгласно допускането R е правилно и, следователно, принадлежи на множеството на всички правилни множества R, т.е. $R \in R$.

Ако R е неправилно, то съдържа себе си като свой елемент, следователно, $R \in R$, но R по определение съдържа само правилни множества, т.е. $R \notin R$.

Следователно, множеството на всички правилни множества R не може да бъде нито правилно, нито неправилно.

1.4.3. Чрез еднозначно описание на елементите

Например множеството A може да се зададе по следния начин:

$1 \in A$

Ако $a \in A$, то $2a + 1 \in A$ и $3a + 1 \in A$.

Започвайки от 1 и последователно прилагайки (2), можем да получим всеки елемент на множеството A. Някои негови елементи са: 1, 3, 4, 7, 9, 10, 15,....

1.4.4. Чрез задаване на характеристична функция на множеството

Характеристична функция χ_A: $A \to \{0, 1\}$ на множеството $A \subset U$ се задава по следния начин:

$\chi_A (a) = 1$, ако $a \in A$ и $\chi_A (a) = 0$, ако $a \notin A$.

Броят на елементите на множеството A може да се зададе като сума от стойностите на неговата характеристична функция:

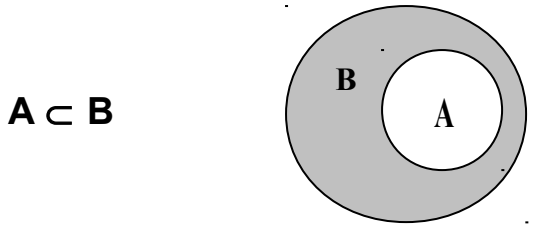
$$|A| = \sum_{u \in U} \chi_A(u)$$

1.4.5. Чрез операции над известни множества

На базата на известни множества можем да получаваме нови, като прилагаме върху тях описаните по-долу операции върху множества.

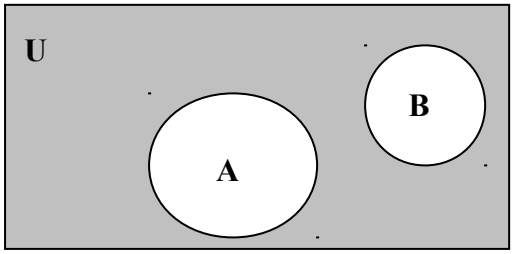
1.4.6. Геометрично представяне. Диаграми на Ойлер–Вен

Диаграмите на Ойлер служат за геометрично представяне на множествата, отношенията между тях и операциите върху множества чрез кръгове.



Фиг. 1.1. Диаграми на Ойлер

Вен въвежда изобразяването на универсалното множество U чрез правоъгълник, в който се разполагат разглежданите множества.



Фиг. 1.2. Диаграми на Вен

1.5. ИЗВЕСТНИ МНОЖЕСТВА

Множество на естествените числа:

N = {0, 1, 2, 3, …}, (*the set of nonnegative integers*) наричано още ***натурален ред*** на числата. В съвременната математика съществуването му се приема за базов постулат [4]. Елементите на N се наричат ***естествени (натурални) числа***. Натуралният ред се задава чрез множеството N и изображението за непосредствено следване φ:

φ: N $\rightarrow$N, $\varphi(x) = x^{\circ}$,

и удовлетворява следните аксиоми, наричани ***аксиоми на натуралния ред***:

Множеството N съдържа елемент, който не следва никой елемент от N и се означава чрез 0, т.е. $0 \in N$ и $0 \neq \varphi(x)$ за $\forall\, x \in N$.

Изображението за непосредствено следване φ е ***инективно***, т.е., ако $\varphi(x) = \varphi(y)$, то $x = y$ за $\forall\, x, y \in N$.

Единственото подмножество на множеството N, което съдържа 0 и заедно с всеки свой елемент съдържа и непосредствено следващия го елемент, е самото множество N.

От аксиомите следва, че редицата:

$0, 0^{\circ} = \varphi(0), 0^{\circ\circ} = \varphi(0^{\circ}), 0^{\circ\circ\circ} = \varphi(0^{\circ\circ})$ …

не съдържа повтарящи се елементи. Ако допуснем, че такива елементи съществуват, например 0 = 0, то съгласно аксиома 2 0 = 0 = $\varphi(0)$, което противоречи на първата аксиома. Множеството N е единственото множество, което може да бъде построено в съответствие с трите аксиоми. Започвайки от елемента 0, можем да получим за краен брой стъпки произволен елемент $x \in N$, повтаряйки изображението φ.

Занимателен е фактът, че името на една от най-известните „търсачки” в Интернет е свързано с хипотетично съществуващото най-голямо естествено число, наречено Google.

Множество на целите положителни числа:

P = {1, 2, 3, …} (*the set of positive integers*) съдържа всички цели положителни числа, подмножество на натуралния ред на числата.

Множество на целите числа:

Z = {... –3, -2, -1, 0, 1, 2, 3, …} (*the set of integers*) включва целите отрицателни числа, нулата и целите положителни числа.

Множество на рационалните числа:

Q (*the set of rational numbers*) - съдържа числата, представими във вид на несъкратими дроби a/b ($b \neq 0$), където a и b са цели числа.

Множество на реалните числа:

R (*the set of real numbers*) съдържа всички рационални и ирационални числа.

Множество на комплексните числа:

C (*the set of complex numbers*) съдържа числа от вида a+bi, където a и b са реални числа, а i е имагинерната единица $i^2 = -1$.

Очевидно $P \subset N \subset Z \subset Q \subset R \subset C$.

Множество на простите числа:

{2, 3, 5, 7, 11, 13, 17, …} съдържа всички цели положителни числа, които се делят само на себе си и на 1.

ЗАДАЧИ ЗА САМОПОДГОТОВКА

1. Докажете, че множеството на естествените числа е безкрайно.
2. Докажете, че множеството на простите числа е безкрайно.
3. Докажете, че в редицата от естествени числа 5, 8, 11, 14, 17, 20…. има безкрайно много прости числа.
4. Да се състави алгоритъм за последователно генериране на първите 100 по големина елементи на множеството A, зададено по следния начин:
   1. $1 \in A$
   2. Ако $a \in A$, то $2a + 1 \in A$ и $3a + 1 \in A$.
5. Черни дупки в редицата на простите числа. Ще наричаме черна дупка интервал от числовата права, не съдържащ прости числа. Да се докаже, че за $\forall$ n > 0 в редицата на простите числа има дупки с размер по-голям от n.

   Решение: Нека е дадено числото n. Разглеждаме следния ред числа:

   m!+2 , m!+3 , m!+4 , … , m!+m.

   Всички тези числа са съставни. Например числото m!+k се дели на k, тъй като m! се дели на k и самото k се дели на себе си (за $2 \leq k \leq m$).

   В този ред се съдържат (m - 1) числа. Полагаме m-1=n+1 и получаваме m = n+2. Т.е. получихме черна дупка с размер, по-голям от даденото число n.

   Жозеф Бертран, изследвайки таблицата на простите числа в интервала от 1 до 6 000 000, е открил, че между числата n и 2n-2, където n>3, се съдържа поне едно просто число

(по-късно доказано от П. Л. Чебишев). Това означава, че простите числа не са толкова редки.

1.6. ОПЕРАЦИИ НАД МНОЖЕСТВАТА

Дефинират се операциите сечение, обединение, разлика, допълнение и декартово произведение на множества, които се прилагат върху множества и дават като резултат множество.

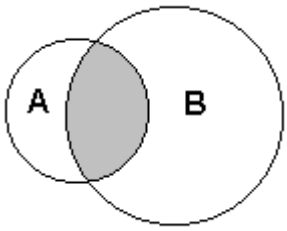
1.6.1. Сечение

Сечението $A \cap B$ на множествата A и B се състои от елементи, които принадлежат и на двете множества:

$A \cap B = \{x \mid x \in A \text{ и } x \in B\}$.

Ако $A = \{a, b, c\}$ и $B = \{b, c, d\}$, то $A \cap B = \{b, c\}$.

Геометрично сечението на множества се представя по следния начин:



Фиг. 1.3. Сечение на множества

Непосредствено от определението следва, че:

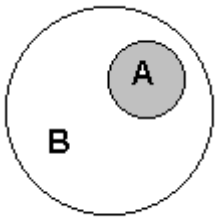
- ако A и B нямат общи елементи, то сечението им е празното множество $A \cap B = \square$. В такъв случай казваме, че множествата A и B не се пресичат;

- $A \cap B \subseteq A$ и $A \cap B \subseteq B$, при това сечението $A \cap B$ е най-голямото множество, което принадлежи на множествата A и B едновременно;

- за броя на елементите в сечението (мощността на сечението) $|A \cap B|$ е в сила: $|A \cap B| \leq |A|$ и $|A \cap B| \leq |B|$;

- ако $C \subseteq A$ и $C \subseteq B$, то $C \subseteq A \cap B$;

- $A \cap B = A$ тогава и само тогава, когато $A \subseteq B$.



Фиг. 1.4. Сечение на множество и подмножество

Операцията сечение може да се обобщи за произволен брой множества. Ако са дадени множествата A_i, $i \in I$ (I може да бъде както крайно, така и безкрайно множество), то сечението им се състои от елементи, които принадлежат на всяко едно от множествата. Използват се означенията:

$A_1 \cap A_2 \cap A_3 \ldots \cap A_n = \bigcap_{i=1}^{n} Ai$ при крайни множества I;

$A_1 \cap A_2 \cap A_3 \ldots \cap A_n \cap \ldots = \cap A_i \; i \in I$ при безкрайно I.

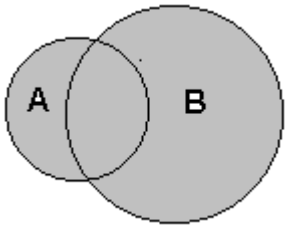
1.6.2. Обединение

Обединението A U B на множествата A и B се състои от елементи, които принадлежат поне на едно от множествата:

$A \cup B = \{x \mid x \in A$ или $x \in B\}$.

Ако $A = \{a,b,c\}$ и $B = \{b,c,d\}$, то $A \cup B = \{a,b,c,d\}$.

Геометрично обединението се представя чрез следната диаграма:



Фиг. 1.5. Обединение на множества

За обединението е в сила:

$A \in A \cup B$ и $B \in A \cup B$, при това обединението е най-малкото от всички множества, съдържащи множествата A и B едновременно.

Обединението на множеСтвата A_i, $i \in I$, се състои от елементи, които принадлежат поне на едно от множествата, и се означава:

$A_1 \cup A_2 \cup A_3 \ldots \cup A_n = \bullet \overset{n}{_{i=1}} Ai$ при крайни множества I;

$A_1 \cup A_2 \cup A_3 \ldots \cup A_n \ldots \cup \ldots = \cup A_{i \in I}$ при безкрайни множества I.

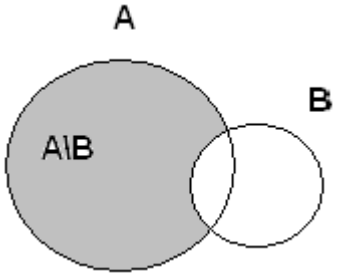
1.6.3. Разлика

Разликата $A \setminus B$ на множествата A и B се състои от елементи, които принадлежат на A, но не принадлежат на B.

$A \setminus B = \{x \mid x \in A \text{ и } x \notin B\}$.

Ако $A = \{a, b, c\}$ и $B = \{b, c, d\}$, то $A \setminus B = \{a\}$.

Геометричното представяне на разликата е следното:



Фиг. 1.6. Разлика на множества

От определението за разлика следва, че:

$A \setminus B \subset A$ и $(A \setminus B) \cap B = \varnothing$.

Разликата $C = A \setminus B$ е най-голямото множество, за което:

ако $C \subset A$ и $C \cap B = \varnothing$, то $C \subset A \setminus B$

$A \setminus B = A$ тогава и само тогава, когато $A \cap B = \varnothing$;

$A \setminus B = \varnothing$ тогава и само тогава, когато $A \subseteq B$.

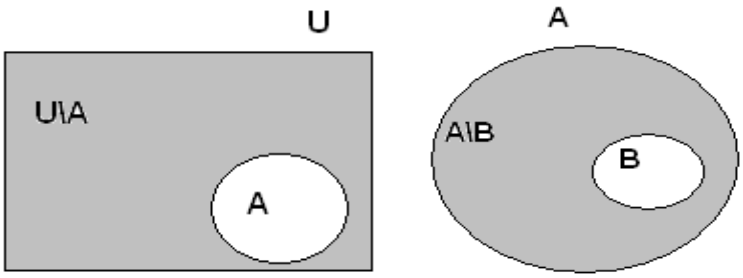
1.6.4. Допълнение

Разликата $U \setminus A$ се нарича ***допълнение*** на A до U и се бележи чрез $\overline{A}$. Допълнението $\overline{A}$ съдържа онези елементи на универсалното множество U, които не принадлежат на A. Очевидно е изпълнено:

$\overline{\varnothing} = U$ и $\overline{U} = \varnothing$

$A \cup \overline{A} = U$ и $A \cap \overline{A} = \varnothing$

Ако $B \subset A$, то разликата $A \setminus B$ се нарича допълнение на B до A.



Фиг. 1.7. Допълнение на множества

Операциите върху множества могат да се зададат и чрез операции върху техните характеристични функции:

Сечение: $\chi_{A \cap B} = \chi_A(u) \times \chi_B(u)$, $u \in U$;

Обединение: $\chi_{A \cup B} = \chi_A(u) \oplus \chi_B(u)$, $u \in U$ ($\oplus$ - събиране по модул 2);

Допълнение: $\chi_{A \setminus U} = 1 - \chi_A(u)$, $u \in U$.

1.6.5. Декартово произведение на множества

Декартово или пряко произведение $A \times B$ на множествата A и B се нарича множеството на всички наредени двойки (a,b), където $a \in A$ и $b \in B$. Може да се зададе по следния начин:

$$A \times B = \{(a, b) | \ a \in A, b \in B\}.$$

Всички елементи на декартовото произведение $A \times B$ при крайни множества $A = \{a_1, a_2, \ldots a_n\}$ и $B = \{b_1, b_2, \ldots b_m\}$ могат да се зададат чрез $n \times m$ матрица от наредени двойки. Елементите на първото множество A се разполагат по хоризонтал, а елементите на B – по вертикал по следния начин:

(a_1, b_1) (a_1, b_2) ..... (a_1, b_m)

(a_2, b_1) (a_2, b_2) ..... (a_2, b_m)

.....

(a_n, b_1) (a_n, b_2) ..... (a_n, b_m)

Правилото е приложимо и при безкрайни множества A и B. В такъв случай се задава само горната лява част на матрицата:

(a_1, b_1) (a_1, b_2) ..... (a_1, b_m) .....

(a_2, b_1) (a_2, b_2) ..... (a_2, b_m) .....

.....

(a_n, b_1) (a_n, b_2) ..... (a_n, b_m) .....

.....

Операцията може да се обобщи за произволен брой множества. Декартово произведение на множествата $A_1, A_2, \ldots, A_n$ е множество, съдържащо наредени n-торки от вида $(a_1,a_2,\ldots,a_n)$, наричани още ***кортежи***, където $a_1 \in A_1$, $a_2 \in A_2$, …, $a_n \in A_n$. Означава се чрез $A_1 \times A_2 \times \ldots \times A_n$ и може да се зададе и по следния начин:

$$A_1 \times A_2 \times \ldots \times A_n = \{ (a_1, a_2,\ldots,a_n) \mid a_i \in A_i \text{ за } \forall \ i = 1, 2, \ldots n\}$$

В частен случай, ако множествата A_i, за $\forall$ i = 1, 2, …n съвпадат, то произведението $A^n = A \times A \times ... \times A$ се нарича ***декартова степен***.

$A^0 = \varnothing$ (приема се)

$A^1 = A$

$A^2 = A \times A$

$A^3 = A \times A \times A$ и т.н.

Например елементите на $R^2 = R \times R$ са наредените двойки реални числа (x, y), x, y $\in$ R.

От определението за декартово произведение следва:

$(A \cap B) \times C = (A \times C) \cap (B \times C)$;

$(A \cup B) \times C = (A \times C) \cup (B \times C)$;

$(A \setminus B) \times C = (A \times C) \setminus (B \times C)$.

Ако A_1 и A_2 са крайни множества, съдържащи съответно n_1 и n_2 елемента, то за мощността на декартовото им произведение е в сила:

$|A_1 \times A_2| = |A_1| \, |A_2| = n_1 n_2$.

Пример 1.2. Едно приложение на декартовото произведение е в релационните бази данни, където информацията се представя таблично:

Факултетен номер	Фамилия	Група	Курс
2130	Иванов	522	2
3047	Василев	518	1
1735	Димитров	571	3

Всеки ред на таблицата представлява кортеж и принадлежи на декартовото произведение $A \times B \times C \times D$ на множествата:

- A – множество на допустимите факултетни номера;

- B – множество на допустимите фамилии;

- C – множество, съдържащо допустими номера на групи;

- D – множество на курсовете {1, 2, 3, 4, 5}.

Това представяне на информацията позволява на операциите върху таблици да се съпоставят операции върху множества.

Пример 1.3. Множеството от клетки върху шахматната дъска може да се разглежда като декартово произведение на множеството от стълбове {a, b, c, d, e, f, g, h} и множеството от редове {1, 2, 3, ...8 }. Всяка клетка се представя чрез наредена двойка, например (a, 1), (d, 3)... Аналогично, всяка карта от колодата може да се представи като елемент на декартовото произведение на множествата:

{♠, ♣, ♥, ♦} и {A, K, D, J, T, 9, 8, 7, 6, 5, 4, 3, 2}.

ЗАДАЧИ ЗА САМОПОДГОТОВКА

1. Определете множествата $A \cap B$, $A \cup B$, $A \setminus B$, $B \setminus A$:

   - ако A = (-2, 1], B = [-1, 2];

   - ако A = {1, 2, 3, 4, 5}, B = {2, 4, 6}.

2. Ако:

   A = {1, 2, 3, 4, 5},

   B = {0, 1, 2, 4},

   C = {1, - 2, 3, - 4, 5},

   B = {-1, 0, 1},

какви елементи има множеството $(A \cap B) \cup (C \cap D)$?.

3. Докажете, че от $A \cap B = C$ следва $C \subset A$ и $C \subset B$.

4. Да се докаже, че $|A \times B| = |A| . |B|$.

1.7. СВОЙСТВА НА ОПЕРАЦИИТЕ ВЪРХУ МНОЖЕСТВА. АЛГЕБРА НА МНОЖЕСТВАТА

Операциите върху множества удовлетворяват съотношения, наречени ***закони.*** Доказателството на тези твърдения се извършва на базата на въведените определения.

Закони за идемпотентност:

$A \cap A = A$

$A \cup A = A$

Закони за нулата и единицата:

$A \cap \varnothing = \varnothing$

$A \cup \varnothing = A$

$A \cap U = A$

$A \cup U = U$

Закони за комутативност:

$A \cap B = B \cap A$

$A \cup B = B \cup A$

Закони за асоциативност:

$(A \cap B) \cap C = A \cap (B \cap C)$

$(A \cup B) \cup C = A \cup (B \cup C)$

Закони за поглъщане:

$A \cap (A \cup B) = A$

$A \cup (A \cap B) = A$

Закони за дистрибутивност:

$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$

$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$

Закони за допълнението:

$A \cap A = \overline{\varnothing}$

$A \cup A = \overline{U}$

Закон за инволюцията (отрицание на отрицанието):

$A = \overline{\overline{A}}$

Закони на Де Морган:

$\overline{(A \cup B)} = \overline{A} \cap \overline{B};$

$\overline{(A \cap B)} = \overline{A} \cup \overline{B};$

Пример 1.4. За да докажем верността на асоциативния закон за сечение на множества, трябва да докажем равенството на множествата в лявата и дясната част чрез следните две включвания:

1. $(A \cap B) \cap C \subseteq A \cap (B \cap C)$ – нека $x \in (A \cap B) \cap C \Rightarrow x \in (A \cap B)$ и $x \in C \Rightarrow x \in A$ и $x \in B$ и $x \in C \Rightarrow x \in A$ и $x \in B \cap C \Rightarrow x \in A \cap (B \cap C)$.

2. $(A \cap B) \cap C \supseteq A \cap (B \cap C)$ – $x \in A \cap (B \cap C) \Rightarrow x \in A$ и $x \in B \cap C \Rightarrow x \in A$ и $x \in B$ и $x \in C \Rightarrow x \in A \cap B$ и $x \in C \Rightarrow x \in (A \cap B) \cap C$.

От (1) и (2) $\Rightarrow (A \cap B) \cap C = A \cap (B \cap C)$.

Пример 1.5. По същия начин постъпваме при доказателство на дистрибутивния закон (6)

$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$.

1. $A \cap (B \cup C) \subset (A \cap B) \cup (A \cap C)$ – нека $x \in A \cap (B \cup C) \Rightarrow x \in A$ и $x \in (B \cup C)$. Допускаме, че $x \in B \Rightarrow x \in A \cap B \Rightarrow x \in (A \cap B) \cup (A \cap C)$. Същото бихме получили при допускането, че $x \in C$.

2. $A \cap (B \cup C) \supset (A \cap B) \cup (A \cap C)$ – нека $x \in (A \cap B) \cup (A \cap C) \Rightarrow x \in (A \cap B)$ или $x \in (A \cap C)$. Допускаме, че $x \in (A \cap B)$ (или $x \in (A \cap C)$, което ще даде същия резултат) $\Rightarrow x \in A$ и $x \in B \Rightarrow x \in A$ и $x \in (B \cup C) \Rightarrow x \in A \cap (B \cup C)$.

ЗАДАЧИ ЗА САМОПОДГОТОВКА

1. Като се използват свойствата на операциите над множества, да се провери верността на твърденията (с апостроф (‘) е означено допълнението на съответното множество):
    1. $A \supseteq A \cup B \cap A$
    2. $A \supseteq A \cap B \cup B$
    3. $B \supseteq A \cup B$
    4. $B \supseteq A \cap B \cup A' \cap B$
    5. $A \supseteq A \cap B \cup A' \cap B$
    6. $A \cap B \cup B \cap C \supseteq A \cup C$
    7. $A \cup A \cap B \supseteq B$
    8. $A \cap B' \cup B \supseteq B' \cup A$

1.8. МНОЖЕСТВА И ИЗОБРАЖЕНИЯ

Нека са зададени две множества A и B. Казваме, че е зададено ***изображението*** f на множеството A в множеството B:

f: $A \to B$,

ако на всеки елемент $a \in A$ е съпоставен елемент $f(a) \in B$. A се нарича дефиниционна област на изображението, а B – множество от стойности. Ако изображението f съпоставя на елемента $a \in A$, елемент $b \in B$, използваме означението $b = f(a)$.

Изображението f: $A \to A$ се нарича ***тъждествено***, ако $f(a) = a$ и се означава чрез Id_A (от английски *identical* – тъждествен).

Две изображения f: A $\rightarrow$B и g: A' $\rightarrow$B' са ***равни*** тогава и само тогава, когато A = A', B = B' и f(a) = g(a) за $\forall$ a $\in$ A. В противен случай изображенията се наричат различни.

Множеството от елементи f(a) $\in$ B се нарича ***образ*** на A при изображението f: A $\rightarrow$ B и се означава чрез f(A) или Im $\boldsymbol{f}$ (от английски Image – образ). От това определение следва, че уравнението f(x) = b има решение x $\in$ A тогава и само тогава, когато b $\in$ Im f**.**

Пример 1.6. Изображенията:

f_1: [0, 1] $\rightarrow$[0, 1] Im f = [0, 1];

f_2: [0, 1] $\rightarrow$R Im f = [0, 1];

f_3: R $\rightarrow$R Im f = {x $\in$ R | x $\geq$ 0},

които на всеки елемент от дефиниционната област съпоставят неговия квадрат, са различни, поради несъвпадение на дефиниционните им области и множествата от стойности.

За построяване на обобщени оператори за сечение и обединение в теория на размитите множества се използват т.нар. триъгълни норми и конорми, които се дефинират като изображения:

T:[0,1] $\times$ [0,1] $\rightarrow$[0,1],

удовлетворяващи допълнително зададени условия.

Изображението f: A $\rightarrow$B се нарича ***инективно изображение*** (*инекция*), ако от $a_1 \neq a_2$ следва $f(a_1) \neq f(a_2)$. Ако f е инективно изображение на A в B, то уравнението f(x) = b има не повече от едно решение x $\in$ A. В горния пример $\boldsymbol{f}_1$ и f_2 са инективни изображения, а f_3 не е, тъй като $f(-1) \neq f(1)$.

Изображението f: A $\rightarrow$ B се нарича ***сюрективно изображение*** (*сюрекция*), ако Im f = B. Ако f е сюрективно

изображение, то уравнението f(x) = b има решение при произволно b $\in$ B. Изображението f_1 е сюрекция, а f_2 и f_3 не са.

Изображението f: A $\rightarrow$B се нарича ***биективно изображение*** (*биекция*), ако f е инективно и сюрективно. Такова е изображението f_1. В случай на биективно изображение, уравнението f(x) = b има единствено решение за всяко b $\in$ B.

За произволни крайни множества *A* и *B* с еднакъв брой елементи, |*A*| = |*B*| и произволно изображение *f*: *A* $\rightarrow$ *B* следващите твърдения са равносилни:

f е инективно;

f е сюрективно;

f е биективно.

1.9. МОЩНОСТ НА БЕЗКРАЙНИТЕ МНОЖЕСТВА

Възприетото определение за мощност на крайните множества не може да се пренесе по аналогия върху безкрайните множества. При сравняване на безкрайни множества възникват проблеми. Отговорът на въпроси от типа: “Кои числа са по-вече - натуралните или целите?”, “Къде има по-вече точки – в отсечката или в цялата права?” е значително по-сложен. Интуитивно ясният принцип, че частта е по-малка от цялото, е неприложим.

Мощност на множествата

Две множества A и B (крайни или безкрайни) са ***равномощни***, ако може да се установи взаимно еднозначно съответствие между елементите им, т.е. ако съществува биективно изображение f: A $\rightarrow$B. В такъв случай казваме, че множествата имат равен брой елементи или са еквивалентни. Взаимната еднозначност

между елементите означава, че на всеки един елемент от първото множество съответства точно един елемент от второто, и обратно. За да се докаже, че две множества имат еднаква мощност, е достатъчно да се зададе взаимно еднозначно съответствие между елементите им.

Отношението ***равномощност*** е отношение на еквивалентност между множества и притежава следните свойства:

- ***рефлексивност*** – всяко множество е равномощно на себе си;

- ***симетричност*** – ако A е равномощно на B, то B е равномощно на A;

- ***транзитивност*** - ако A е равномощно на B и B е равномощно на C, то A е равномощно на C.

От определението следва, че множеството на естествените числа е равномощно на множеството на четните числа и на множеството на нечетните числа, което се вижда от следните съответствия:

{1, 2, … n, …} {1, 2, … n, …}

{2, 4, … 2n, ...} {1, 3, … (2n-1) …}

Аналогично можем да дефинираме съответствия между естествените числа и техните квадрати, кубове и т.н. Следователно, множеството на естествените числа е еквивалентно (равномощно, има еднакъв брой елементи) на всяка своя безкрайна част.

Не така очевидни са следните твърдения:

Пример 1.7. Множеството 2^N на всички подмножества на естествените числа N е равномощно на множеството от безкрайните редици от нули и единици. Действително, на всяко подмножество A на N съпоставяме характеристичната му функция,

представляваща редица от нули и единици, при това i–тият елемент на редицата е 1, ако i $\in$ A и 0 в противен случай. Например:

000 … 0 … $\leftrightarrow \varnothing$;

111 … 1 … $\leftrightarrow$ N;

1010101 … 0101 ... $\leftrightarrow$ {0, 2, 4, … 2n, …..} – множество на четните числа;

010101010...1010 … $\leftrightarrow$ {1, 3, 5, 7, … 2n+1, …} – множество на нечетните числа;

011101000 ... 000 .... $\leftrightarrow$ {1, 2, 3, 5} и т.н.

По такъв начин на всяко подмножество съпоставяме редица, и обратно, откъдето следва, че двете разглеждани множества са равномощни.

Пример 1.8. Множеството на всички безкрайни редици, съдържащи числата 0, 1, 2 и 3, е равномощно на множеството от безкрайните редици от нули и единици.

В случая биекцията е следната: на всяка безкрайна редица, съдържаща числата 0, 1, 2 и 3, съпоставяме безкрайна редица от нули и единици, получена чрез замяна на 0, 1, 2 и 3 с техния двоичен запис – 00, 01, 10, 11, и обратно. Например:

11111111111 ... $\leftrightarrow$ (01)(01)(01)(01)(01)(01) .....

22222222222 ... $\leftrightarrow$ 101010101010 .....

123123123123 ... $\leftrightarrow$ 01101101101101101 ..... и т.н.

Пример 1.9. Множеството на всички точки от интервала (0, 1) е равномощно на множеството от безкрайните редици от нули и единици.

Доказателство: Всяко число в интервала (0, 1) е представимо във вид на безкрайна десетична дроб. На всяко число съпоставяме редица от нули и единици, получена чрез заместване

на всяка десетична цифра с нейния двоичен запис. Следователно двете множества са равномощни.

Тъй като множеството на всички подмножества на естествените числа N е равномощно на множеството от безкрайните редици от нули и единици (пример 1), което от своя страна е равномощно на множеството на всички точки от интервала (0, 1) (пример 3), съгласно транзитивността на отношението еквивалентност следва, че множеството на всички точки от интервала (0, 1) е равномощно на множеството на всички подмножества на естествените числа N. Тогава:

$|(0, 1)| = |2^N|$

Няколко теореми обосновават възможността да се сравняват мощностите на безкрайните множества.

Теорема 1.1 (Теорема на Кантор–Бернщтейн). Ако съществуват инективни изображения:

$f: A \rightarrow B$ и

$g: B \rightarrow A$,

то множествата A и B са равномощни, т.е., ако $|A| \leq |B|$ и $|B| \leq |A|$, то $|A| = |B|$.

Теорема 1.2 (Теорема на Цермело). Ако A и B са произволни (крайни или безкрайни) множества, то за мощностите им е изпълнено едно и само едно от следните три условия: или $|A| < |B|$, или $|B| < |A|$, или $|A| = |B|$.

Теорема 1.3 (Теорема на Кантор). Мощността на произволно множество A е строго по-малка от мощността на множеството 2^A от всички негови подмножества, т.е.:

$|A| < |2^A|$.

Теоремата показва, че винаги можем да намерим множество, чиято мощност е по-голяма от мощността на дадено множество. Прилагайки това правило, можем да получим множества с произволно големи мощности.

Доказателство: Изображението $x \in A \rightarrow \{x\} \in 2^A$, което на всеки елемент x от A съпоставя едноелементно подмножество {x}, е инективно изображение на A в $2^A \Rightarrow |A| \leq |2^A|$. Ще покажем че $|A| \neq |2^A|$. Допускаме противното, $|A| = |2^A| \Rightarrow$ съществува биективно отображение $f: A \rightarrow 2^A$. Задаваме множеството D по следния начин:

$D = \{x \in A \mid x \notin f(x)\}$.

D съдържа онези елементи на A, които не съвпадат с образите си при изображението f, от което следва, че:

$x \in D \Leftrightarrow x \notin f(x)$.

Тъй като f е биективно, съществува елемент $d \in A$, чийто образ е D, т.е. f(d)=D. Ако в горната формула заменим x с d, получаваме противоречие:

$d \in D \Leftrightarrow d \notin D$,

дължащо се на допускането, че $|A| = |2^A|$. От доказаните твърдения:

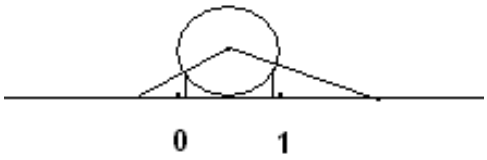
$|A| \leq |2^A|$ и $|A| \neq |2^A|$,

следва, че:

$|A| < |2^A|$.

ЗАДАЧИ ЗА САМОПОДГОТОВКА

1. Да се докаже, че интервалът (0, 1) е равномощен на реалната права.



Фиг. 1.8. Равномощност на интервала [1, 0] и реалната права

Упътване: Дефинирайте биекция между точките на интервала и точките на реалната права, като използвате рисунката.

2. Докажете, че интервалът [0, 1] е равномощен на интервала [0, 2].
3. Докажете, че интервалите [a, b] и [c, d] са равномощни.
4. Докажете, че броят на редиците от нули и единици с дължина n е равен на броя на подмножествата на n – елементното множество {0, 1, 2, 3, …. (n – 1)}.
5. Докажете, че множеството на всички безкрайни редици, съдържащи числата 0, 1 и 2, е равномощно на множеството от безкрайните редици от нули и единици.
6. Докажете, че множествата N и Z са равномощни.
7. Две крайни множества са равномощни, ако мощностите им са еднакви. Докажете, че A и B са равномощни тогава и само тогава, когато съществува биекция f A $\rightarrow$B.

1.10. ИЗБРОИМИ И НЕИЗБРОИМИ МНОЖЕСТВА

1.10.1. Мощност на изброимите множества

Множеството A е ***изброимо***, ако е равномощно с естествения ред на числата N. Съгласно определението за равномощни множества това означава, че може да се установи взаимно еднозначно съответствие между елементите на множеството и естествените числа. С други думи, елементите на изброимо множество могат да се номерират. A може да се представи във вид на краен или безкраен списък:

$A = \{a_0, a_1, a_2, \ldots a_n, \ldots\}$,

чрез биекцията f: $N \rightarrow A$, която на $n \in N$ съпоставя елемента $a_n \in A$.

Произволно изображение $N \rightarrow \{0, 1, 2, \ldots (n - 1)\}$ не може да бъде инективно, тъй като натуралният ред N е безкраен, а интервалът [0, (n-1)] е краен и съдържа точно n елемента. Следователно мощността на N е по-голяма от мощността n на всеки краен интервал:

$|N| > |[0, (n-1)]| = n$, за всяко естествено n.

Мощността на естествения ред на числата N се означава чрез $\aleph_0$ и служи като еталон за мощност на изброимите множества. Натуралният ред е в някакъв смисъл възможно най-голямото изброимо множество. Всяко множество, равномощно с N, има мощност $\aleph_0$.

За доказване на изброимостта на произволно множество понякога е достатъчно да се зададе правило за номериране на елементите на множеството без пропуски и повторения.

Пример 1.10. Множеството на четните числа е изброимо, можем да ги представим чрез списък 0, 2, 4, 6, ... 2n, … Биекцията е: $n \rightarrow 2n$. По същите съображения е изброимо и множеството на нечетните числа.

Пример 1.11. Множеството на целите числа Z е изброимо. Достатъчно е да разположим целите числа в два реда и да номерираме по стълбове по следния начин:

0, 1, 2, 3, 4. .....

-1, -2, -3, -4, ...

Мощността на Z е $\aleph_0$.

Пример 1.12. Множество рационалните числа Q е изброимо. Достатъчно е да разположим рационалните числа в таблица по следния начин:

Знам. / Числ.	1	2	3	…..	m	…..
0	$\frac{0}{1}$	$\frac{0}{2}$	$\frac{0}{3}$	…..	$\frac{0}{m}$	…..
1	$\frac{1}{1}$	$\frac{1}{2}$	$\frac{1}{3}$	…..	$\frac{1}{m}$	…..
2	$\frac{2}{1}$	$\frac{2}{2}$	$\frac{2}{3}$	…..	$\frac{2}{m}$	…..
3	$\frac{3}{1}$	$\frac{3}{2}$	$\frac{3}{3}$	…..	$\frac{3}{m}$	…..
…..	…..	…..	…..	…..	…..	…..
n	$\frac{n}{1}$	$\frac{n}{2}$	$\frac{n}{3}$	…..	$\frac{n}{m}$	…..
…..	…..	…..	…..	…..	…..	…..

Обхождаме по диагонал (или по квадрати), в резултат получаваме следния списък:

0/1, 0/2, 1/1, 0/3, 1/2, 2/1, ...,

в който след краен брой стъпки ще се появи всяка дроб. Премахваме съвпадащите елементи и номерираме.

В сила са следните теореми, свързани с изброимите множества.

Теорема 1.4. Всяко подмножество B на изброимото множество A е изброимо или крайно.

Действително, нека A е изброимо и следователно представимо във вида: A = {a_0, a_1, a_2, … a_n, …}, а B е негово подмножество. В редицата a_0, a_1, a_2, … a_n, … зачеркваме всички елементи, които не принадлежат на B, без да променяме реда. Ако B е крайно, ще останат краен брой не зачеркнати елементи и безкраен в противен случай. Резултатът номерираме, спазвайки последователността на елементите.

Теорема 1.5. Всяко безкрайно множество A съдържа изброимо подмножество.

Конструирането на изброимо подмножество на множеството A се извършва по следната схема: от A извличаме един елемент, който бележим с b_0, от A \ {b_0,}, извличаме втори елемент, който означаваме с b_1, и т.н... Възможността за такова извличане се дава от аксиомата за избор. Процесът няма да прекъсне, тъй като A е безкрайно по условие. В крайна сметка ще получим изброимото подмножество B = { b_0, b_0, b_0, … b_0, ..}.

От теорема (1.5) следва, че мощността на изброимото множество не е по-голяма от мощността на произволно безкрайно множество, т.е. тази мощност е най-малката от мощностите на безкрайните множества.

Теорема 1.6. Обединението на краен или изброим брой крайни или изброими множества е крайно или изброимо.

Нека са зададени изброим брой изброими множества. Тогава самите множества и техните елементи можем да номерираме по следния начин:

$A_0 = \{a_{00}, a_{01}, a_{02}, \ldots a_{0n}, \ldots\}$

$A_1 = \{a_{10}, a_{11}, a_{12}, \ldots a_{1n}, \ldots\}$

$A_2 = \{a_{20}, a_{21}, a_{22}, \ldots a_{2n}, \ldots\}$

$A_3 = \{a_{30}, a_{01}, a_{32}, \ldots a_{3n}, \ldots\}$

……..

Ако множествата са краен брой или част от тях са крайни, номерацията някъде ще прекъсне. Обхождаме елементите по квадрати (възможно е и друго обхождане, например по диагонали). При това на всяка стъпка обхождаме краен брой елементи. В резултат за обединението получаваме крайна или безкрайна редица от елементи, които можем да номерираме последователно с естествените числа (съвпадащите елементи изключваме):

$a_{00}, a_{01}, a_{11}, a_{10}, a_{10}, a_{02}, a_{12}, a_{22}, a_{21}, a_{20}, \ldots$

От теоремите следва валидността на следните манипулации с мощности:

$\aleph_0 + n = \aleph_0$ - обединението на изброимо и крайно множество е изброимо;

$\aleph_0 + \aleph_0 = \aleph_0$ - обединението на две изброими множества е изброимо. Например множеството на четните и множеството на нечетните числа са две непресичащи се подмножества на натуралния ред с мощност $\aleph_0$, чието обединение дава натуралния ред със същата мощност;

$\aleph_0 \times \aleph_0 = \aleph_0$ - обединението на изброим брой изброими множества е изброимо.

1.10.2. Мощност на неизброимите множества

Освен изброими, съществуват и ***неизброими*** множества. Множеството на всички подмножества на натуралния ред на числата има мощност, строго по-голяма от мощността на

натуралния ред съгласно теоремата на Кантор и, следователно, не е изброимо. Може да се докаже неизброимостта на много известни множества.

Пример 1.13. Интервалът (0, 1) е неизброимо множество от точки.

Допускаме противното. Тогава точките от интервала могат да се номерират с естествените числа и:

$(0, 1) = \{ a_0, a_1, a_2, \ldots a_n, \ldots\}$,

при това посоченият списък съдържа всички числа.

Всяко число от интервала (0, 1) може да се запише като безкрайна десетична дроб от вида $a_n = 0, a_{n0}\, a_{n1}\, a_{n2}, \ldots$ Разглеждаме числото $x = 0, \eta_0, \eta_1, \eta_2 \ldots, \in (0,1)$, построено по следния начин: всяка цифра η_n от {0, 1, 2, … 9} е избрана така, че $\eta_0 \neq a_{00}$, $\eta_1 \neq a_{11}$, $\eta_2 \neq a_{22}$, ... Тъй като множеството $\{ a_0, a_1, a_2, \ldots a_n, \ldots\}$ съдържа всички елементи на интервала (0, 1), то би следвало x да съвпадне с някое от числата. Това не е вярно, тъй като по построение x се различава от всяко число поне по една цифра. Полученото противоречие се дължи на допускането, че (0, 1) е изброимо множество. Следователно, множеството от точки в интервала (0, 1) е неизброимо.

Използваният метод за доказателство се нарича ***диагонален метод на Кантор***. Чрез него се доказват много твърдения, свързани с безкрайните множества.

Мощността на (0, 1) се нарича ***мощност на континиума*** (от английски *continuous* - непрекъснат) и се бележи с “C”. Множествата, равномощни с (0, 1), се наричат ***континуални.*** Континуални са множеството от точки върху права, множеството на реалните числа, множеството от точки върху равнината, множествата от точките на

непрекъснати геометрични обекти и много други, срещани в математическата практика множества. Вече доказахме, че

$$|(0, 1)| = 2^{|N|} \Rightarrow C = 2^{\aleph_0}.$$

Чрез $\aleph_1$ се означава мощността на най-малкото неизброимо множество. Предизвикалата много спорове хипотезата на континиума гласи, че $\mathbf{C = \aleph_1}$, т.е. (0, 1) е най-малкото неизброимо множество. За съжаление, това твърдение не може да се докаже. И положителното (т.е. интервалът (0, 1) е най-малкото неизброимо множество, доказано от Коен), и отрицателното решение на континиум хипотезата (равносилно на съществуване на множества с мощности, по-малки от континиума, доказано от Хедел) не противоречи на аксиоматиката на теорията на множествата.

Понякога в литературата вместо интервала (0, 1) се разглежда интервалът [0, 1], получен от чрез добавяне на крайните точки. Може да се докаже, че (0, 1) и [0, 1] са равномощни.

Пример 1.14. Чрез диагоналния метод на Кантор може да се докаже, че множеството 2^N от всички подмножества на натуралния ред е неизброимо.

Допускаме противното. Тогава всички елементи на множеството от всички подмножества на натуралния ред могат да се подредят в следния списък:

$A_0, A_1, A_2, A_3, \ldots A_n, \ldots\ldots$

Вече разгледахме по какъв начин на всяко подмножество на N може да се съпостави безкрайна редица от нули и единици, наричана още ***характеристична редица*****.** Тъй като A_n са подмножества на натуралния ред, то на всяко съпоставяме редица от нули и единици, като означаваме членовете по следния начин:

$A_0 \rightarrow a_{00}, a_{01}, a_{02}, \ldots a_{0n}, \ldots ;$

$A_1 \rightarrow a_{10}, a_{11}, a_{12}, \ldots a_{1n}, \ldots ;$

$A_2 \rightarrow a_{20}, a_{21}, a_{22}, \ldots a_{2n}, \ldots ;$

……..

$A_n \rightarrow a_{n0}, a_{n1}, a_{n2}, \ldots a_{nn}, \ldots ;$

……..,

където $a_{ij} = 1$, $i, j \in N$, ако $j \in A_i$ и нула в противен случай.

Ще покажем, че съществува редица η_0, η_1, η_2 ..., не попадаща в горния списък, а следователно, не съответстваща на нито едно от изброените множества. Полагаме $\eta_0 = 1 - a_{00}$, η_1, $= 1 - a_{11}$, $\eta_2 = 1 - a_{22}$, и т.н. Така конструираната редица не съвпада с нито една от изброените в списъка редици: от първата се различава по първия елемент, от втората – по втория и т.н... Следователно, тя не принадлежи на списъка, а заедно с нея и не принадлежи на списъка и съответното й множество. Този факт противоречи на допускането, че списъкът е пълен, т.е че множеството от всички подмножества на натуралния ред е изброимо. С това доказахме, че множеството от всички подмножества на N е неизброимо.

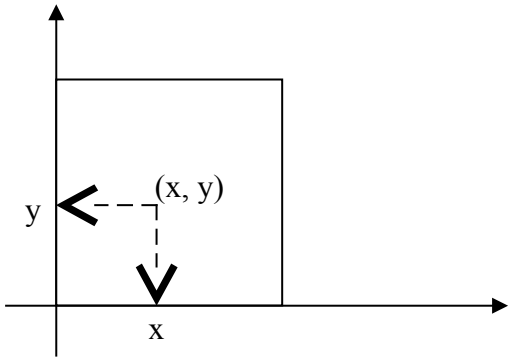
Пример 1.15. По същия начин може да се докаже по-общото твърдение: за произволно множество A множеството 2^A от всички негови подмножества е неизброимо.

Допускаме противното: A е изброимо и всички негови подмножества са подредени в списък $A_0, A_1, A_2, A_3, \ldots A_n, \ldots$ Конструираме множеството B по следния начин: $b_n \in B \Leftrightarrow b_n \notin A_n$, за $\forall\ b_n \in A$ (възможно ли е това?). Очевидно $B \subseteq A$ и $B \neq A_n$ за $\forall\ n$, което противоречи на допускането, че списъкът е пълен. Следователно, множеството 2^A от всички подмножества на A е неизброимо.

Като се използват операциите върху мощностите, може да се докаже, че квадратът е равномощен на интервала, независимо от интуитивната представа, че в квадрата има повече точки от интервала. Самият Кантор бил шокиран след доказателство на това твърдение. В писмо до математика Дедекинд той пише: „ Аз виждам това, но не вярвам в това”.

Доказателството на Кантор е следното: Без ограничение на общността разглеждаме квадрата $[0, 1] \times [0,1]$ и интервала [0,1].

За простота разглеждаме вътрешните точки на отсечката и квадрата. Всяка точка от квадрата задаваме чрез нейните координати (x, y), които записваме като безкрайни десетични дроби от вида:



$x = 0,\alpha_1\alpha_2\alpha_3 \ \dots \ \alpha_n \dots$

$y = 0,\beta_1\beta_2\beta_3 \ \dots \ \beta_n \dots$

На тази точка от квадрата съпоставяме точката z от интервала (0, 1), получена по следния начин:

$z = 0,\alpha_1\beta_1\alpha_2\beta_2\alpha_3\beta_3 \ \dots \ \alpha_n\beta_n \dots$

Ясно е, че на различни точки от квадрата съответстват различни точки от отсечката. Установихме взаимно еднозначно съответствие между точките на квадрата и част от точките на отсечката, следователно той има не по-голяма мощност. Но не и по-малка, от което следва, че двете мощности съвпадат.

Има и по-елегантно доказателство. За мощностите на квадрата и отсечката е изпълнено:

$$|[0, 1] \times [0,1]| = C \times C = 2^{\aleph_0} \times 2^{\aleph_0} = 2^{\aleph_0 + \aleph_0} = 2^{\aleph_0} = C = |[0,1]|.$$

ЗАДАЧИ ЗА САМОПОДГОТОВКА

1. Като използвате теорема 3 и метода на нейното доказателство, докажете, че множеството на рационалните числа Q е изброимо.
2. Докажете, че множеството на реалните числа R е неизброимо.
3. Ако B = {0,1}, докажете, че множеството на думите с безкрайна дължина върху B е неизброимо.
4. Интересна интерпретация на свойствата на изброимите множества дава полският фантаст С. Лем в книгата „Записките на Йон Тихи”. Тази занимателна задача, цитирана в много работи по теория на множествата, може да се формулира така:

   Галактическият хотел „Космос” разполага с неограничен брой стаи. В даден момент всички са заети.

   • Неочаквано пристига президентът на космическата федерация и пред администратора възниква задачата да го настани, при условие че няма свободни стаи. Интерпретацията в термините на теория на множествата е следната: ако към изброимо множество добавим един елемент, полученото множество е изброимо.

   • Пристига космическа делегация от 100 души, как те да бъдат настанени, т.е какво ще се получи, ако към изброимо множество добавим краен брой елементи?

   • За космически конгрес безброй галактики изпращат по един представител. Как да се настанят делегатите, при

условие че хотелът е пълен? Интерпретация: какво ще получим при обединение на две безкрайни изброими множества?

- Във връзка с ремонт се затварят безброй много галактически хотели. Техните наематели чакат настаняване в хотел „Космос”. Как да постъпим? Интерпретация: какъв е резултатът от обединението на изброим брой изброими множества?

За решение на проблемите на администратора е достатъчно да се укаже такова преномериране на стаите в хотел „Космос”, при което ще се освободи стая за всеки новопристигнал делегат.

5. Като използвате, че мощностите на (0, 1) и R съвпадат, докажете, че множеството от точки на реалната права е неизброимо. Кое от свойствата на релацията „еквивалентност” на множества трябва да се използва?
6. Докажете, че интервалите [a, b] и [c, d] съдържат еднакъв брой точки.

1.11. МЕТОД НА МАТЕМАТИЧЕСКАТА ИНДУКЦИЯ

В науката се използват два метода на изследване - ***индуктивен*** (от частното към общото) и ***дедуктивен*** (от общото към частното). Общоприето е индуктивните методи да се свързват с експерименталния подход, а дедуктивните - със строг математически подход.

Експерименталният подход в математическите разсъждения не е приемлив, тъй като дадено твърдение може да е вярно в

редица частни случаи и невярно в общия. Пример за това е многочленът:

$P(x) = x^2 + x + 41.$

Ойлер допуска, че стойностите му при x – естествено число, са прости числа. Действително:

x	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**…**	**39**
P(x)	**41**	**43**	**47**	**53**	**61**	**71**	**83**	**97**	**113**	**131**	**151**	**173**	**197**	**…**	**1601**

Допускането е вярно за x = 0, 1, 2, …. 39, но при x = 40 получаваме $P(40) = 40^2 + 40 + 41 = 1681 = 41^2$.

Математическата индукция е дедуктивен метод за доказване на свойства на естествените числа и на други множества, равномощни с множеството на естествените числа. Липсата на горна граница в множеството на естествените числа дава възможност да се доказват твърдения, които са верни за всички естествени числа. Методът на математическата индукция се базира на аксиомата за индукция от теория на множествата, която може да се зададе по следния начин:

Принцип на пълната математическа индукция. Нека P е твърдение относно натуралните числа n, което искаме да докажем и:

1. P е вярно при n = 0;

2. от верността на P при n = k следва, че твърдението е в сила и при n = k + 1.

Тогава твърдението P е вярно за всички естествени числа n.

Доказателствата по метода на математическата индукция включват следните стъпки:

1. Предпоставка на индукцията - проверяваме, че твърдението е вярно за n = 0 (или за някаква начална стойност n = n_0.> 0).

2. Индуктивно предположение - допускаме, че твърдението е вярно за n = k.

3. Индуктивна стъпка - доказваме, че твърдението е изпълнено за n = k+1.

Съгласно аксиомата за математическата индукция, твърдението е вярно за всяко естествено число $n \geq n_0$.

Пример 1.16. Да се пресметне сумата на първите n нечетни числа:

$S_n = 1 + 3 + 5 + ...+ (2n – 1)$.

Решение*:* Очевидно $S_1 = 1$, $S_2 = 4$, $S_3 = 9$,… Предполагаме, че търсената сума $S_n = n^2$. Ще докажем по метода на математическата индукция, че твърдението е изпълнено за всяко $n \in N$.

1. Предпоставка: твърдението е вярно за n = 1.

2. Индукционна хипотеза: допускаме, че твърдението е вярно за n = k, т.е.

$S_k = 1 + 2 + 3 + … + (2k -1) = k^2$.

3. Индуктивна стъпка Ще докажем, че твърдението е изпълнено за n = k+1 . Действително:

$S_{k+1} = 1 + 2 + 3 + … + (2k -1) + (2(k+1) – 1) = k^2 + 2k +1 = (k + 1)^2$

Съгласно принципа на математическата индукция, твърдението е вярно за $\forall n \in N$.

Пример 1.17. Да се докаже, че всяко крайно множество A с n елемента притежава 2^n подмножества.

Доказателство [4]**:** 1. При n = 0 твърдението е вярно. В този случай A е празното множество (единственото множество, съдържащо 0 елемента) и притежава точно едно подмножество – $\varnothing$, т.е. при n = 0 броят на подмножествата на A е $2^0 = 1$.

2. Допускаме, че всяко множество с n = k елемента притежава точно 2^k подмножества.

3. Ще докажем, че множество с n = k + 1 елемента притежава 2^{k+1} подмножества. Нека *A е* произволно множество с n = k + 1 елемента. Тъй като k + 1 > 0, то множеството A не е празно и съдържа поне един елемент a $\in$ A. Този елемент разбива подмножествата на A на два непресичащи се класа :

– клас U = {X $\subset$ A | a $\in$ X}, включващ всички подмножества на A, съдържащи a, и

– клас V = {Y $\subset$ A | a $\notin$ Y} на подмножествата на A, несъдържащи елемента a.

– U $\cap$ V = $\varnothing$, U $\cup$ V = A

Разглеждаме множеството A°, получено от A чрез премахване на елемента a: A° = A \ {a}. Подмножествата на множеството A° са точно онези подмножества на множеството A, несъдържащи елемента a, следователно A°= V . Множеството A° съдържа k елемента. Съгласно допускането, то притежава 2^k подмножества, откъдето следва, че $|V| = 2^k$.

Разглеждаме следните изображения:

U $\rightarrow$V: на $\forall$ X $\in$ U $\rightarrow$X \ {a} $\in$ V и

V $\rightarrow$U: на $\forall$ Y $\in$ V $\rightarrow$Y U {a} $\in$ U.

Те осъществяват взаимно еднозначно съответствие между елементите на множествата U и V, поради което $|U| = |V| = 2^k$. Тогава за броя на подмножествата на множеството A получаваме:

$|A| = |U \cup V| = |U|+|V|=2^k +2^k = 2.\ 2^k = 2^{k+1}$,

с което твърдението е доказано.

ЗАДАЧИ ЗА САМОПОДГОТОВКА

1. Да се докаже, че при $n \geq 1$ е в сила:

$$1^3 + 2^3 + 3^3 + \ldots n^3 = \frac{n(n+1)^2}{4}.$$

2. Да се пресметне сумата:

$$S_n = 2^0 + 2^1 + 2^2 + \ldots 2^n$$

3. Да се докаже по метода на математическата индукция че:

$$1 + 2 + 3 + \ldots + n = \frac{n(n+1)}{2}.$$

$$1^2 + 2^2 + 3^2 + \ldots + n^2 = \frac{n(n+1)(2n+1)}{6}$$

4. Да се докаже, че ако n е естествено, то $n^2 + n$ е четно число.

5. Да се докаже, че за всяко $n \geq 0$ е изпълнено $n! \geq 2^n$

6. Докажете, че n различни прави върху равнината я разбиват на части, които могат да бъдат оцветени в бяло и черно по такъв начин, че съседните области да бъдат оцветени в различни цветове.

7. Ако редицата $\{S_n\}$ е дефинирана по следния начин:

$$S1 = 1,\ Sn = S1 + S2 + S3 + \ldots + Sn-1,\ n \geq 2,$$

докажете, че общият й член допуска представяне във вида:

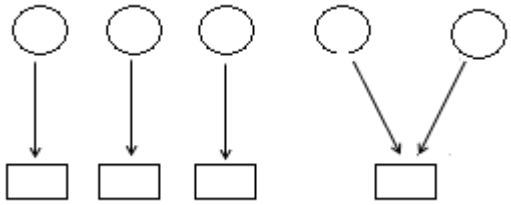
$$S_n = 2^{n-2} \text{ за всяко } n \geq 2$$

1.12. ПРИНЦИП НА ДИРИХЛЕ

Принципът на Дирихле, който намира приложение при решение на много дискретни задачи върху крайни множества, се основава на следното твърдение:

Нека A и B са крайни множества и |A| > |B|. Тогава за всяка функция f: A → B съществуват $a_1 \neq a_2 \in A$, такива, че $f(a_1) = f(a_2)$.

Принцип на Дирихле *(на чекмеджетата):* Нека X е множество с n елемента (предмети), а Y множество с m елемента (чекмеджета) и n > m. Както и да поставим всички предмети в чекмеджетата, поне в едно чекмедже ще има поне 2 предмета:



ЗАДАЧИ ЗА САМОПОДГОТОВКА

1. Докажете, че в един клас от 32 ученици има поне двама, родени на едно и също число.
2. Докажете, че има поне двама потребители на Internet, притежаващи еднакъв брой книги.
3. Докажете, че за произволни 5 числа от интервала [1, 8], винаги има две, чиято сума е равна на девет.
4. Докажете, че от произволни 1001 цели числа има поне две, разликата на които се дели на 1000.
5. При проверка на тест по дискретни структури се оказало, че точно един студент е допуснал 13 грешки, а останалите

– по-малко. Ако в групата има 30 студенти, да се докаже, че поне трима са допуснали равен брой грешки.

6. Докажете, че от произволни n + 1 цели числа има поне две, разликата на които се дели на n.

1.13. ПРИНЦИП ЗА ВКЛЮЧВАНЕ И ИЗКЛЮЧВАНЕ

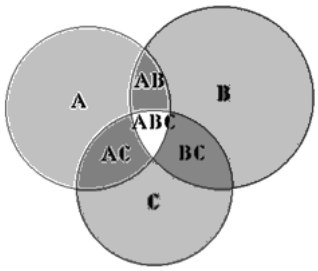
Принципът (формулата) за включване и изключване дава възможност да се пресметне мощността на обединението на няколко множества, ако са известни техните мощности и мощностите на всички сечения. За две и три множества формулата изглежда по следния начин:

$|A \cup B| = |A| + |B| - |A \cap B|;$

$|A \cup B \cup C| = |A| + |B| + |C|$

$- |A \cap B| - |A \cap C| - |B \cap C|$

$+ |A \cap B \cap C|;$



Формулата може да се обобщи при произволен брой множества:

$$|A_1 \cup A_2 \cup A_3 \ldots \cup A_n| = \sum_i |A_i| - \sum_{i<j} |A_i \cap A_j|; + \sum_{i<j<k} |A_i \cap A_j \cap A_k| - \ldots$$

$$\ldots + (-1)^{n-1} | A_1 \cap A_2 \cap A_3 \ldots \cap A_n |$$

Доказателство на валидността на формулата може да се извърши както по метода на математическата индукция, така и чрез използване на характеристичните функции на множествата [28].

Нека множествата A_i, I = 1, 2, … n са подмножества на някакво множество U (в частност на универсалното множество) с характеристични функции $\chi_{Ai}(u)$, i = 1, 2, ….n, $u \in U$.

Но броят на елементите на дадено множество може да се получи като сумираме единиците в стойността на характеристичната му функция:

$$|A_1 \cup A_2 \cup A_3 \ldots \cup A_n| = \sum_u \chi_{A1 \cup A2 \ldots \cup An}(u),\ u \in U$$

Съгласно закона на де Морган:

$$A_1 \cup A_2 \cup A_3 \ldots \cup A_n = \overline{A_1 \cup A_2 \cup A_3 \ldots \cup A_n} = \overline{A_1} \cap \overline{A_2} \cap \overline{A_3} \ldots \cap \overline{A_n}$$

Последното равенство, изразено в термините на характеристичните функции, изглежда по следния начин:

$$\chi_{A1 \cup A2 \ldots \cup An} = 1 - \chi_{\overline{A1} \cap \overline{A2} \ldots \cap \overline{An}} =$$

$$= 1 - (1 - \chi_{A1})(1 - \chi_{A2})\ldots(1 - \chi_{An}) =$$

$$= \sum_i \chi_{Ai} - \sum_{i<j} \chi_{Ai}\chi_{Aj} + \sum_{i<j<k} \chi_{Ai}\chi_{Aj}\chi_{Ak} - \cdots\cdots (-1)^{n-1}\chi_{A1}\chi_{A2}\ldots\chi_{An}$$

Като сумираме двете страни на горния израз по $u \in U$, за мощността на обединението получаваме точно формулата за включване и изключване.

Заключение

Теория на множествата предоставя формален апарат за построяване на модели не само в математиката, но и в много други научни и приложни области.

ЗАДАЧИ ЗА САМОПОДГОТОВКА

1. Група студенти от специалност КСТ изучават три избираеми дисциплини: дискретни структури (ДС), компютърни мрежи (КМ) и информационни технологии (ИТ). При разпределението се оказало, че 33 са избрали ДС, 35 – КМ, а 37 – ИТ. При това, 13 са записали едновременно два курса - ДС и КМ, 17 – КМ и ИТ, а 15 – ИТ и ДС. Ако общият брой на студентите е 65, колко са се записали за трите курса едновременно?
2. В многопроцесорна система са включени 92 микропроцесора. При решение на дадена задача 19 микропроцесора обработват текстова информация, 17 – графическа, 11 - числова, 12 микропроцесора едновременно обработват графическа и текстова, 7 - текстова и числова, 5 - графическа и числова, а част от микропроцесорите едновременно обработват графическа, текстова и числова информация. Колко микропроцесора са универсални, ако при решаване на задачата 67 микропроцесора не са включени в обработката?
3. На изпит по Дискретни структури две групи студенти трябва да решат две задачи. След проверка на резултатите се оказало, че 49 студента са решили поне по една задача. Колко са студентите, решили двете задачи, ако в първата група има 34, а във втората – 28 студента?

глава

2

КОМБИНАТОРНИ МЕТОДИ И МОДЕЛИ

2.1. КОМБИНАТОРНИ ОБЕКТИ И ЗАДАЧИ

2.1.2. Предназначение на комбинаториката

Комбинаториката е математическа дисциплина, която се занимава с изучаване на свойствата на комбинаторните конфигурации – конструкции, построени от елементите на някое (обикновено крайно) множество в съответствие със зададени правила. Комбинаторният анализ предоставя теоретичната база за изследване на свойствата на комбинаторните конфигурации, оценка на броя им, алгоритми за тяхното генериране.

Първите научни изследвания по комбинаторика принадлежат на италианските учени Дж. Кардано, Н. Тарталя (1499-1557), Г. Галилей (1564-1642) и френските учени Б. Паскал (1623-1662) и П. Ферма.

Началото на комбинаториката като наука е поставено през XVIII век и е свързано с възникването на теорията на вероятностите и необходимостта да се пресмята броят на различни комбинаторни конфигурации.

За първи път терминът *комбинаторика* е въведен през 1666 год. от немския учен Г. Лайбниц в неговата работа "*За изкуството на комбинаториката*", която поставя началото на развитието на комбинаториката като математическа наука. Значителен принос за развитието на комбинаториката има Л. Ойлер. Появата и развитието на изчислителната техника създава предпоставки за повишен интерес и развитие на комбинаториката, поради възможността за решаване на сложни и многоразмерни комбинаторни задачи.

Комбинаторните задачи са свързани с:

- избор на елементи с определени свойства от дадено крайно множество;

- разполагането на тези елементи в определен ред в съответствие със зададени правила;

- оценка на броя на възможните конфигурации от даден тип.

Комбинаторните методи намират приложение в теория на графите, теория на вероятностите, криптография, разпознаване на образи, икономика, теория на информацията, разработка на алгоритми, експертните системи и много други теоретични и приложни направления.

2.1.2. Основни правила на комбинаториката

Решението на много комбинаторни задачи се основава на две фундаментални правила, наричани съответно правило за сумата и за произведението.

Правило за сумата. Ако A_1 и A_2 са непресичащи се крайни множества, съдържащи съответно n_1 и n_2 елемента, то обединението им $A_1 \cup A_2$ съдържа $n_1 + n_2$ елемента:

$$| A_1 \cup A_2 | = | A_1 | + | A_2 | = n_1 + n_2, A_1 \cap A_2 = \varnothing.$$

Ако множествата се пресичат, то:

$$| A_1 \cup A_2 | = | A_1 | + | A_2 | - | A_1 \cap A_2 |, A_1 \cap A_2 \neq \varnothing.$$

Правилото важи за произволен, но краен брой непресичащи се крайни множества:

$$| A_1 \cup A_2 \cup ... \cup A_k | = | A_1 | + | A_2 | + \ldots + | A_k | = n_1 + n_2 + \ldots + n_k$$

Формулата може да се обобщи при произволен брой слагаеми с непразно сечение с помощта на т.нар. *принцип на включване и изключване,* доказан в (1.13). В комбинаторен вариант той се формулира по следния начин:

Принцип на включване и изключване. Нека е дадено крайно множество с N елемента и са формулирани свойства a_1, a_2, … a_n, които елементите могат да притежават или да не притежават. Въвеждаме означенията:

N(a1), N(a2), … N(an), … - брой на елементите, притежаващи свойството a_1, a_2, … a_n съответно;

N(a1'), N(a2'), … N(an'), … - брой на елементите, не притежаващи свойството a_1, a_2, … a_n съответно;

N(aiaj) - брой на елементите, притежаващи едновременно свойствата a_i и a_j …

За броя $N(a_1' a_2' \dots a_n')$ на елементите, не притежаващи нито едно от свойствата a_1, a_2, … a_n, е в сила:

$$N(a_1' a_2' \dots a_n') = N - N(a_1) - N(a_2) - \dots - N(a_n)$$
$$+ N(a_1 a_2) + N(a_1 a_3) + \dots + N(a_{n-1} a_n) + \dots$$
$$+ (-1)^n N(a_1 a_2 \dots a_n)$$

Ще дадем доказателство по метода на математическата индукция (в (1.13) доказателството използва характеристични функции на множества).

Доказателство:

1. При n = 1 формулата N(a1') = N - N(a1) е вярна, тъй като броят на елементите, не притежаващи свойството a1, можем да получим, като от общия брой N извадим броя на елементите, притежаващи a1.

2. Допускаме, че формулата е вярна за n-1 свойства:

$$N(a_1' a_2' \dots a_{n-1}') = N - N(a_1) - N(a_2) - \dots - N(a_{n-1})$$
$$+ N(a_1 a_2) + N(a_1 a_3) + \dots + N(a_{n-2} a_{n-1}) + \dots \quad (1)$$
$$+ (-1)^{n-1} N(a_1 a_2 \dots a_{n-1})$$

3. Ще докажем нейната справедливост за n свойства на обектите.

Съгласно (1) имаме:

$$N(a_1' a_2' \dots a_{n-1}' a_n') = N(a_1' a_2' \dots a_{n-1}') - N(a_1' a_2' \dots a_{n-1}' a_n) \qquad (2)$$

Прилагаме формулата за съвкупността $N(a_n)$

$$N(a_1' a_2' \dots a_{n-1}' a_n) = N(a_n) - N(a_1 a_n) - N(a_2 a_n) - \dots - N(a_{n-1} a_n)$$

$$+ N(a_1 a_2 a_n) + \dots + N(a_{n-2} a_{n-1} a_n) + \qquad (3)$$

$$+ \dots + (-1)^{n-1} N(a_1 a_2 \dots a_{n-1} a_n)$$

След заместване на (1) и (3) в (2) получаваме искания резултат.

Пример 2.1. В студентска група има 25 студенти. От тях 15 знаят езика Паскал, 10 - езика Си и 14 - езика Бейсик. Освен това, 7 студенти знаят Паскал и Си, 10 студенти - Паскал и Бейсик, 8 студенти - Си и Бейсик, а 5 студенти знаят и трите езика. Колко студенти не знаят нито един от трите езика за програмиране?

Решение: При решение на такъв тип задачи често срещан проблем е определянето на обектите и техните свойства. В случая обектите са студентите, а свойствата – знание на езици. Въвеждаме следните означения:

a_1 - свойство "знае Паскал";

a_2 - свойство "знае Си";

a_3 - свойство "знае Бейсик".

Съгласно условието на задачата: $N = 25$, $N(a_1) = 15$, $N(a_2) = 10$, $N(a_3) = 14$, $N(a_1,a_2)=7$, $N(a_1,a_3)=10$, $N(a_2,a_3)=8$, $N(a_1,a_2,a_3)=5$.

По формулата за включване и изключване за броя на студентите, незнаещи нито един от дадените езици, получаваме:

$$N(a_1',a_2',a_3') = 25 - (15 + 10 + 14) + (7+10+8) - 5 = 6.$$

Пример 2.2. Да се намери броят на правилните несъкратими дроби със знаменател 100.

Решение: Тъй като $100 = 2^2.5^2$, то числителят на правилна несъкратима дроб със знаменател 100 трябва да бъде цяло число в интервала от 1 до 100, което не се дели на 2 и 5. Нека X_2, X_5 и X_{10} са множества от числа в интервала от 1 до 100, които се делят съответно на 2, на 5 и на 10. Очевидно:

$|X_2| = 100/2 = 50$

$|X_5| = 100/5 = 20$

$|X_{10}| = 100/10 = 10.$

Тъй като $X_2 \cap X_5 = X_{10}$, то съгласно правилото за сумиране получаваме:

$|X_2 \cup X_5| = 50 + 20 – 10 = 60.$

Следователно, в интервала от 1 до 100 има 60 числа, които се делят на 2 или 5, и 100 – 60 = 40 числа, взаимно прости с числото 100, т.е. между правилните дроби със знаменател 100 точно 40 са несъкратими.

Правило за произведението. Разглеждаме наредени набори (кортежи) от вида $(a_1, a_2,\ldots, a_k)$ със зададена дължина k. Предполагаме, че елементът a_1 може да бъде избран по n_1 различни начини; при фиксирано a_1 елемент a_2 може да бъде избран по n_2 начина; при фиксирани a_1 и a_2 елементът a_3 може да бъде избран по n_3 начина и т.н., при фиксирани $a_1, a_2,\ldots, a_{k–1}$ елементът a_k може да бъде избран по n_k начина. Тогава броят на различните кортежи е равен на произведението $n_1.n_2.\ldots.n_k$. Този резултат е свързан с мощността на декартовото произведение на крайни множества, разгледан е в (1.6).

Пример 2.3. Да се намери броят на различните трицифрени числа, несъдържащи еднакви цифри, и броят на различните трицифрени числа, съдържащи поне две еднакви цифри.

Решение: Нека a_1, a_2, a_3 са цифрите на трицифрените числа. Първата цифра a_1 може да се избере по 9 начина (изключваме 0 като водеща цифра) от елементите на множеството {1, 2, 3, 4, 5, 6, 7, 8, 9}. При фиксирана първа цифра, втората a_2 може също да се избере по 9 начина, като се взема произволно число от 0 до 9, с изключение на a_1. При фиксирани първа и втора цифра, третата може да се избере по 8 начина. По правилото на произведението броят на трицифрените числа, несъдържащи еднакви цифри, е 9.9.8 = 648. Броят на трицифрените числа е 900. Всяко от тях или съдържа две еднакви цифри, или не съдържа. Следователно, има 900 – 648 = 252 различни трицифрените числа, които имат поне две еднакви цифри.

Пример 2.4. За отваряне на сейф се използва комбинация от 4 цифри. Каква е вероятността случайно да се отвори сейфът?

Решение: Всяка цифра от комбинацията за отваряне на сейфа може да бъде избрана по 10 начина в интервала от 0 до 9. В комбинацията участват 4 цифри и съгласно правилото за умножение общият брой комбинации ще бъде 10 х 10 х 10 х 10 = 10 000. Само една от тези комбинации може да отвори сейфа, следователно, вероятността да я улучим е 1/10 000 = 0.0001.

ЗАДАЧИ ЗА САМОПОДГОТОВКА

1. Правилна комбинация за отваряне на сейф е тази, в която всички цифри са различни. Каква е вероятността случайно да се отвори сейфът?

2. Фирма за продажба на компютри има в наличност 8 вида дънни платки, 5 вида процесори и 3 вида видеоконтролери. Колко различни компютърни конфигурации може да асемблира фирмата?
3. По колко начина могат да се разположат два топа на шахматната дъска, така че да не се бият?
4. В група от 35 студенти 11 посещават семинар по компютърни мрежи, 20 – по синтез и анализ на алгоритми, а 10 не посещават нито единия, нито другия. Колко студенти посещават двата семинара едновременно?
5. На заседание на парламента 13 депутата се записали за изказване. По колко начина може да се списък? Решете задачата при условие, че двама депутати са заели 3 и 9 място в списъка.
6. Колко са петзначните числа, съдържащи поне една четна цифра? *Определете броя на числата, не съдържащи нито една четна цифра.
7. Ако се използва азбука от две букви A = {X, Y}, колко е броят на думите с дължина 5 символа? Каква е връзката на тази задача с представянето на информацията в компютрите?
8. В даден поток има общо 67 студента. От тях 48 знаят английски, 35 – немски, а 27 знаят и двата езика. Колко студенти не знаят нито английски, нито немски?
9. За кодиране на символи в компютрите се използват 8 бита. Колко символа можем да използваме? Международната кодова таблица Unicode представя символите чрез 2 байта. Колко е максималният брой символи, които могат да се представят?

10. Колко различни 10-разрядни троични числа можем да запишем?
11. В щафета 100 + 200 + 400 + 800 метра треньорът може да избере за първия участък един от общо 6 спринтьори, за втория – един от 3, на третия – един от 7 и на четвъртия - точно един. По колко начина могат да бъдат разпределени спортистите?
12. Колко различни автомобилни номера могат да бъдат съставени, ако номерата съдържат две букви (от 20 допустими) и четири цифри?
13. По колко начина може да се запълни фиш за участие в играта 6 от 49? А 5 от 35? Колко фиша трябва да попълним, за да си гарантираме четворка?
14. Подредете 5 лъва и 4 тигра в редица, така че да няма два тигъра, разположени един след друг.
15. (Лондонски университет, на изпит по ДС за първи курс). При анкетиране на 10 000 клиенти се оказало, че 5 010 обичат шоколад, 3 470 обичат бонбони, а 4 820 – пасти. При това, 500 обичат всичко, 1 000 обичат шоколад и бонбони (възможно и пасти), 840 – шоколад и пасти (и може би бонбони), а 1 410 обичат бонбони и пасти (и евентуално шоколад). Оказало се, че последното число е грешно. Нека вярното е X. Намерете X, ако всеки клиент обича поне един продукт. Уви! Последното допускане не е вярно. Има клиенти, които не харесват нито един от продуктите. В такъв случай е невъзможно да се определи X еднозначно. Намерете максимално възможното X. А минималното?

16. В един район на Англия изследвали N = 10 000 ученици. Установили следните здравословни проблеми:
    - A – физически недостатъци, P(A) = 877;
    - B – нервност, P(B) = 1 086;
    - C – умствено изоставане, P(C) = 789.
    - Освен това: N(AB) = 336, N(BC) = 455.

    Докажете, че някои умствено изостанали ученици нямат физически недостатъци и намерете минималния им брой.

2.2. ОСНОВНИ КОМБИНАТОРНИ КОНФИГУРАЦИИ

Комбинаторни конфигурации (или *съединения*) се наричат групи от елементите на дадено крайно множеството A, образувани по определени правила. Ако в дадена конфигурация се допуска повторение на елементите на множеството, тя се нарича ***конфигурация с повторения,*** противен случай – ***конфигурация без повторения***. Броят на елементите, участващи в дадена конфигурация, се нарича ***клас на конфигурацията***. Например: 1122 е съединение от четвърти клас с повторения, а 123 е съединение от трети клас без повторения. Ако редът на елементите в групата е съществен, то 123 и 321 са различни конфигурации.

Основните комбинаторни конфигурации са вариации, комбинации и пермутации. При разглеждането им приемаме, че елементите им се избират от дадено крайно множество A = { a_1, a_2, a_3,...., a_n } с n елемента. Поради съществуване на биекция A $\leftrightarrow$ {1, 2, 3, … n}, без ограничение на общността можем да разглеждаме конфигурации от елементите на множеството {1, 2, 3, … n}.

2.2.1. Вариации

Вариациите са наредени конфигурации от елементите на базовото множество A с дължина k (т.е. съдържащи k елемента). Могат да бъдат:

- вариации без повторения;

- вариации с повторения.

Вариации без повторения на n различни елемента a_1, a_2, ..., a_n от k-ти клас се наричат всички наредени съединения, съдържащи k елемента и различаващи се едно от друго или поне по един от елементите или по техните места.

Ако означим с V_n^k броя на вариациите от n елемента k-ти клас без повторения, то:

$$V_n^k = n(n-1)...(n-k+1) = \frac{n!}{(n-k)!}.$$

Например вариациите от 5 елемента 3-ти клас са 60 на брой:

1 2 3 1 3 2 2 1 3 2 3 1 3 1 2 3 2 1
1 2 4 1 4 2 2 1 4 2 4 1 4 1 2 4 2 1
1 2 5 1 5 2 2 1 5 2 5 1 5 1 2 5 2 1
1 3 4 1 4 3 3 1 4 3 4 1 4 1 3 4 3 1
1 3 5 1 5 3 3 1 5 3 5 1 5 1 3 5 3 1
1 4 5 1 5 4 5 1 4 5 4 1 4 5 1 4 1 5
2 3 4 2 4 3 3 2 4 3 4 2 4 2 3 4 3 2
2 3 5 2 5 3 3 2 5 3 5 2 5 2 3 5 3 2
2 4 5 2 5 4 5 2 4 5 4 2 4 5 2 4 2 5
3 4 5 3 5 4 5 3 4 5 4 3 4 5 3 4 3 5

Посочените вариации представляват наредени тройки от елементите на базовото множество {1, 2, 3, 4, 5}, различаващи се една от друга или по наредбата на елементите, или поне по един елемент.

Пример 2.5. Студенти изучават 10 дисциплини. Всеки ден в програмата са включени 3 различни дисциплини. По колко начина може да се състави програма за един ден?

Решение: Тъй като редът на дисциплините има значение, трябва да определим броя на вариациите от 10 елемента 3-ти клас без повторения:

$V_{10}^{3} = \frac{10!}{(10-3)!}$ = 720 начина.

Вариации с повторения на n различни елемента a_1, a_2, a_3,...., a_n от k-ти клас са всички наредени конфигурации с повторение на елементите на даденото множество, съдържащи k елемента и различаващи се една от друга или по местата на отделните елементи, или поне по един от елементите.

Броят на вариациите $\widetilde{V}_n^k$ от n елемента k-ти клас с повторения съвпада с начините на запълване на k квадратчета, като за тази цел се използват n елемента, които могат да се повтарят. Съгласно правилото за произведение:

$\widetilde{V}_n^k = n^k$.

Пример 2.6. Допустима парола за достъп до базата с данни на дадено предприятие може да съдържа четири английски букви. Колко са възможните пароли?

$\widetilde{V}_{26}^{4} = 26^4 = 456976$

2.2.2. Комбинации

Комбинации от n елемента a_1, a_2, a_3,...., a_n k-ти клас се наричат конфигурации, съдържащи по k елемента и различаващи се една от друга поне с един елемент. За разлика от вариациите, при комбинациите наредбата на елементите е без значение, т.е. комбинациите 123 и 213 са равни.

Комбинациите могат да бъдат комбинации без повторения и комбинации с повторения.

Комбинации без повторения на n различни елемента от k-ти клас се наричат всички ненаредени конфигурации без повторения, съдържащи k елемента и различаващи се една от друга поне с по един от съставящите ги елементи.

Броят на комбинациите от n елемента k-ти клас без повторения се означава като C_n^k и се намира по следната формула:

$$C_n^k = \frac{V_n^k}{k!} = \frac{n!}{k!(n-k)!} = \frac{n(n-k)...(n-k+1)}{1.2...k}.$$

Очевидно:

$$C_n^0 = 1 \qquad C_n^1 = n \qquad C_n^n = 1 \qquad C_n^k = C_n^{n-k}$$

Числата C_n^k се наричат ***биномни коефициенти***, тъй като присъстват в бинома на Нютон:

$(a + b)^n = a^n + C_n^1 a^{n-1}b + C_n^2 a^{n-2}b^2 + \ldots + C_n^{n-1} ab^{n-1} + b^n$

При разглеждане на крайни множества доказахме, че ако A има n елемента, то множеството от всички подмножества на A има мощност $|2^A| = 2^n$. От друга страна:

$|2^A| = C_n^0 + C_n^1 + C_n^1 + ... + C_n^n \Rightarrow$

$C_n^0 + C_n^1 + C_n^1 + ... + C_n^n = 2^n$

Ако n = 5, k = 3, комбинациите без повторение от елементите на множеството {1, 2, 3, 4, 5} са 10 на брой:

(123), (124), (125), (134), (135), (145), (234), (235), (245), (345)

Пример 2.7. В дадена група има 15 студенти. По колко начина могат да се изберат 5 делегати за участие в конференция?

Решение: Трябва да се изберат 5 от общо 15 души. Тъй като наредбата няма значение, търсената конфигурация е комбинация от 15 елемента 5-ти клас без повторения. Съгласно формулата, броят на различните комбинации е:

$$C_{15}^{5} = \frac{15!}{5!(15-5)!} = 3003 .$$

Комбинации с повторения на n различни елемента от k-ти клас се наричат всички ненаредени конфигурации от елементите на дадено множество с повторения, съдържащи k елемента и различаващи се едно от друго поне с по един от съставящите ги елементи.

Например, ако е дадено множеството от елементи {a,b,c,d}, n = 4, при k = 2 комбинациите без повторение от втори клас са:

(a,b), (a,c), (a,d), (b,c), (b,d), (c,d),

а комбинациите с повторение са:

(a,a), (a,b), (a,c), (a,d), (b,b), (b,c), (b,d), (c,c), (c,d), (d,d).

Броят $\widetilde{C}_n^k$ на комбинациите от n елемента k-ти клас с повторения се намира по следната формула:

$$\widetilde{C}_n^k = C_{n+k-1}^k = \frac{(n+k-1)!}{k!(n-1)!} .$$

Пример 2.8. В книжарница се продават 5 марки дискети. По колко начина могат да се купят 10 дискети?

Решение:

$\widetilde{C}_5^{10} = C_{5+10-1}^{10} = C_{16}^{10} = \frac{14!}{10!4!} = 1001$ начина.

2.2.3. Пермутации

Пермутации от n елемента a_1, a_2, a_3,..., a_n се наричат наредени конфигурации от всички елементи на базовото множество, които се различават една от друга само по реда на елементите.

Пермутациите биват два вида - пермутации без повторения и пермутации с повторения.

Пермутация без повторение се нарича всяка n-елементна наредена конфигурация без повторение на елементите на дадено множество.

Броят P_n на пермутациите без повторение от n елемента се задава чрез формулата:

$P_n = 1.2.3 \ldots n = n!$

Например всички възможни пермутации от елементите на множеството {a, b, c} с 3 елемента са:

(a b c), (a c b), (b a c), (b c a), (c a b), (c b a).

Пример 2.9. С цел засекретяване на информация се използва шифър (наричан шифър на Цезар), при който всяка буква от една азбука се заменя с друга буква от същата азбука. Колко схеми на кодиране могат да се съставят с буквите на българската азбука?

Решение: Броят на различните схеми съвпада с броя на пермутациите без повторения -1. Защо?

$P_{30} = 30! = 2\ 652\ 528.10^{26} - 1$ начина.

Независимо от големия брой начини за замяна, този начин на кодиране е ненадежден. Оказва се, че може да се определи честотата на срещане на всяка буква от азбуката в текстовете на даден език. Използването на тази информация може да съкрати значително броя на проверяваните варианти.

Пермутация с повторение се нарича всяка n-елементна наредена конфигурация с повторение на елементите на дадено множество.

Нека означим с $P_n(n_1, n_2, \ldots n_k)$ броя на пермутациите с повторения от n елемента. Допускаме, че съвпадащите елементи са разпределени в k групи. Първата група съдържа n_1 равни елемента, втората - n_2 елемента и т.н. Последната k-та група съдържа n_k съвпадащи елемента. Броят на тези пермутации с повторение е:

$$P_n(n_1, n_2, \ldots, n_k) = \frac{n!}{n_1!.n_2!,\ldots,.n_k!}$$

Пример 2.10. Колко различни „думи" ще получим, като разместим буквите в думата *математика*?

Решение:

$P_{10}(3,2,2,1,1) = \frac{10!}{3!.2!.2!.1!.1!} = 151\ 200$ различни думи.

ЗАДАЧИ ЗА САМОПОДГОТОВКА

1. Пресметнете броя на вариациите и комбинациите без повторения от 7 елемента 3-ти клас.
2. По време на кандидатстудентска кампания деканът назначава комисия от двама души: председател и член от общо 6 души административен персонал. С цел избягване на корупция, деканът се стреми всеки ден да определя различни членове на комисията. Колко различни комисии може да назначи? Ако кампанията продължава 90 дни, колко души трябва да бъдат определени за участие в комисията?

*Комисиите (a – председател, b – член) и (b – председател, a – член) приемаме за различни, тъй като участващите служители съвпадат, но заемат различни длъжности.

3. На тест за интелигентност в полицията дават 10 въпроса с възможни отговори {да, не}. Тестващата програма засича времето, но не ограничава броя на опитите. Даден кандидат, за когото въпросите се оказали доста сложни, решава да експериментира, въвеждайки всички възможни комбинации от отговори. Колко време му е необходимо, ако една комбинация от отговори въвежда за 1 минута.

4. Вратата на една каса, съдържаща секретни материали, се отваря, след като се набере код от 4 позиции, всяка от които съдържа една от 5-те букви {a, b, c, d, e}. По колко различни начина може да се набере този код? Каква е вероятността за случайно отваряне на касата?

5. След разпит на свидетели се установило, че номерът на автомобила, причинил катастрофата, съдържа 4 различни цифри, от които първата е 1, а последната 9. Колко автомобила трябва да проверят служителите на КАТ?

   *Липсващите цифри могат да бъдат 0, 2, 3, 4, 5, 6, 7, 8. Редът им има значение, тъй като 1569 и 1659 са два различни валидни номера. Следователно, броят на автомобилите за проверка ще бъде равен на броя вариации от 8 елемента втори клас без повторения.

6. Слушателите от едно класно отделение са 12 души. По колко начина от тях могат да се изберат 4 души за участие в щафета 4x100 m?

7. Колко различни „думи” ще получим, като разместим буквите в думата *комбинаторика?*

8. На конференция всеки новопристигнал делегат се ръкува с всички в залата. Колко са делегатите, ако общият брой ръкостискания е 78?

9. Колко са различните 7-разрядни десетични числа?

10. Преподавателите от дадена катедра трябва да изберат ръководител на катедрата, заместник и отговорник по връзки с медиите. По колко начина може да се осъществи изборът, ако общият брой на преподавателите е 17?

11. Колко са вариантите, ако за ръководител на катедрата има 3 подходящи кандидатури, за заместник – 7, а за отговорник – 5?

12. Седем студенти се явяват на изпит. За да няма недоволни, решават да изберат случаен вариант на подреждане. Колко е общият брой варианти?

13. В лотария, свързана с конни надбягвания, участниците попълват талон, в който указват реда на финиширане на конете. За да си гарантира печалба, вманиачен зрител решава да попълни всички възможни варианти. Колко талона трябва да попълни, ако в състезанието участват 7 коня? Какъв е броят на талоните, ако той разполага с информация за първото и последно място?

14. Търговски пътник трябва да обходи общо 6 града, тръгвайки от първия и връщайки се в него. По колко различни начина може да осъществи това, ако между всеки два града има пряк път?

2.3. СПЕЦИАЛНИ КОМБИНАТОРНИ МЕТОДИ

2.3.1. Метод на рекурентните уравнения

Методът на рекурентните уравнения дава възможност решението на дадена комбинаторна задача, отнасяща се до n обекта, да се сведе към решение на аналогична задача с по-малък брой обекти с помощта на някаква зависимост, наричана рекурентна.

Рекурентни уравнения

Редицата $u_0, u_1, u_2, \ldots u_n, \ldots$ се нарича ***рекурентна редица*** от ред k, ако съществува зависимост, изразяваща общия член на редицата u_n чрез k предходни члена:

$$u_n = f(u_{n-1}, u_{n-2}, u_{n-3}, \ldots u_{n-k}),\ 1 \leq k < n \qquad (1)$$

За еднозначно определяне на членовете на редицата е необходимо задаването на първите ѝ k члена $u_0, u_1, u_2, \ldots u_{k-1}$. Понякога зависимостите от вида (1) в литературата се наричат рекурентни зависимости или рекурентни съотношения. Рекурентната връзка отразява зависимостта между членовете на редицата, но е неудобна за работа, тъй като получаването на произволен член е свързано с пресмятане на всички предхождащи го членове. Затова при работа с рекурентни редици стремежът е да се изрази общият член в явен вид чрез n.

Редицата от факториали 1, 1, 2, 6, 24, 120, 720…, n!,… може да се зададе чрез рекурентното уравнение от първи ред (всеки следващ член се определя еднозначно от предходния):

$$u_{n+1} = (n + 1)u_n, \quad u_0 = 1.$$

Известно е представянето на общия член на тази редица чрез n:

$u_{n+1} = (n + 1)! = 1.2.3 \ldots (n+1)$.

2.3.2. Линейни рекурентни уравнения

Ако зависимостта (1) е линейна, общият член на редицата може да се представи чрез ***линейно рекурентно уравнение*** от ред k във вида:

$$u_n = a_1u_{n-1} + a_2u_{n-2} + \ldots + a_ku_{n-k}, \quad (2)$$

където a_i, i = 1, 2,…, k, са независещи от n константи.
Такива са изучаваните в средното училище аритметична и геометрична прогресии:

$u_{n+2} = 2u_{n+1} – u_n$ - от втори ред

$u_{n+1} = qu_n$ - от първи ред.

За да получим представяне на общия член на редицата u_n чрез n, което позволява директното му пресмятане при зададено n, записваме (2) във вида:

$$u_n - a_1u_{n-1} - a_2u_{n-2} - \ldots - a_ku_{n-k} = 0,$$

и му съпоставяме полинома:

$$P_k(x) = x^k - a_1x^{k-1} - a_2x^{k-2} - \ldots - a_k,$$

който се нарича ***характеристичен полином*** на линейното рекурентно уравнение (2). Тъй като е от k-та степен, може да има k реални или комплексни корени (някои от които могат да съвпадат). Нека λ е корен на характеристичния полином $\Rightarrow P_k(\lambda) = 0$. В сила са следните твърдения:

Теорема 2.1. Редицата $u_0, u_1, u_2, \ldots u_n, \ldots$ с общ член:

$u_n = c\lambda^n$, c – произволна константа,

удовлетворява линейното рекурентно уравнение (2).

Трябва да докажем, че:

$u_n - a_1u_{n-1} - a_2u_{n-2} -\ldots- a_ku_{n-k} =0$

Заместваме $u_n = c\lambda^n$, $u_{n-1} = c\lambda^{n-1}$ и т.н. и получаваме:

$c\lambda^n - a_1c\lambda^{n-1} - a_2\, c\lambda^{n-1} - ... - a_kc\lambda^{n-k} = c\lambda^{n-k}(\lambda^k - a_1\lambda^{k-1} - a_2\lambda^{k-2} -\ldots- a_k)$

$= c\lambda^{n-k} P_k(\lambda) \equiv 0.$

Теорема 2.2. Ако $\{u_n\}$.и.$\{v_n\}$ са линейни рекурентни редици, такава е и редицата $\{z_n\}$ с общ член, получен като линейна комбинация на членовете на редиците:

$z_n = \alpha\, u_n + \beta\, v_n$, където α и β са произволни константи.

За да докажем твърдението, трябва да докажем, че z_n удовлетворява (2), което се вижда след заместване и преобразуване на получения израз.

Теорема 2.3. Ако λ_1, λ_2, ... λ_k са прости корени на характеристичния полином на линейната рекурентна редица $\{u_n\}$, то общият й член може да се изрази във вида:

$u_n = c_1\lambda_1^n + c_2\lambda_2^n + c_k\lambda_k^n$, c_1, c_2, ... c_k – подходящи константи.

Доказателството следва от първите две теореми.

Теорема 2.4. Ако λ_1 с кратност r_1, λ_2 с кратност r_2,... λ_k с кратност r_k ($r_1 + r_2 + \ldots + r_k = k$) са кратни (реални или комплексни) корени на характеристичния полином на линейната рекурентна редица $\{u_n\}$, то общият й член може да се изрази във вида:

$$u_n = \sum_{i=1}^{s}(c_{i1} + c_{i2}n + \ldots c_{ir_i}n^{r_i-1})\lambda_i^n ,$$

където c_{ij} са подходящи (възможно комплексни) константи.

Частен случай на линейна рекурентна редица е редицата на Фибоначи.

2.3.3. Числа на Фибоначи

“Няма такава съвършена красота, която да не е с някакво своеобразие на пропорциите”.
Франсис Бейкън

"На всяка човешка дейност са присъщи три отличителни особености: форма, време и отношение - и всичките те се подчиняват на реда на Фибоначи".
Р. Елиот

Числата на Фибоначи са свързани със задачата за размножаване на популация зайци, дадена за пръв път от италианския математик Фибоначи през 1202 година в книгата “Liber Abaci” (Книга за изчисления). Този трактат, който за съжаление е изгубен, запознава европейците с арабските цифри и аритметичните действия в десетична бройна система и определя развитието математиката в Европа за няколко столетия напред.

Числата на Фибоначи намират голямо приложение в математиката, естествените науки, архитектурата, изкуството, икономиката, анализа на финансовите пазари и много други области. Имат връзка с известната геометрична задача за „златното сечение” и с такива геометрични обекти като фракталите.

Формулираната от Фибоначи задача е следната: Двойка зайци могат да произведат за един месец нов чифт зайци. Новата двойка дава първото си поколение след два месеца, след което продължава да се размножава всеки месец. Колко двойки зайци ще има след определено време?

Лесно се проследява изменението на броя в началните месеци: 1, 1, 2, 3, 5, 8, ... Означаваме чрез F(n) броя на двойките през месец n и разглеждаме ръста на популацията за 1 месец. Очевидно, през месец n броят на двойките е сума от възрастни и новородени. Броят на възрастните е F(n - 1), т.е. колкото двойки е

имало в края на предходния месец, а броят на новородените е равен на броя на двойките през n – 2-рия месец, тъй като половата зрелост настъпва след един месец, т.е:

$F(n) = F(n - 1) + F(n - 2)$.

Получената рекурентна зависимост при дадените начални условия $F(1) = 1$ и $F(2) = 1$ дава възможност за последователно пресмятане на ***числата на Фибоначи*** при произволно n. Изменяйки началните условия, можем да получим други редици, чиито елементи удовлетворяват същата рекурентна зависимост.

Характеристичният полином на редицата на Фибоначи има вида:

$$P_2(x) = x^2 - x - 1. \qquad (3)$$

Съгласно теорема 3 общият член допуска представяне във вида:

$$F(n) = c_1\lambda_1^2 + c_2\lambda_2^2,$$

където λ_1 и λ_2 са корени на характеристичния полином, а c_1 и c_2 са константи, определени от началните условия. При решаване на уравнението $P_2(x) = 0$, за корените получаваме:

$$\lambda_1 = \frac{1+\sqrt{5}}{2} \qquad \lambda_2 = \frac{1-\sqrt{5}}{2},$$

а за общия член:

$$F(n) = c_1\lambda_1^2 + c_2\lambda_2^2 = c_1\left(\frac{1+\sqrt{5}}{2}\right)^n + c_2\left(\frac{1-\sqrt{5}}{2}\right)^n.$$

Коефициентите c_1 и c_2 определяме от началните условия. Чрез заместване получаваме системата:

$$F(1) = c_1\left(\frac{1+\sqrt{5}}{2}\right)^1 + c_2\left(\frac{1-\sqrt{5}}{2}\right)^1 = 1;$$

$$F(2) = c_1\left(\frac{1+\sqrt{5}}{2}\right)^2 + c_2\left(\frac{1-\sqrt{5}}{2}\right)^2 = 1.$$

Решенията:

$c_1 = \frac{1}{\sqrt{5}}$ и $c_2 = -\frac{1}{\sqrt{5}}$

дават възможност да изразим в явен вид F(n) чрез n:

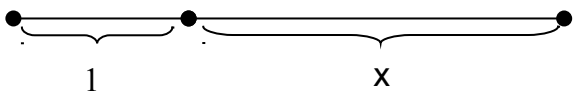
$F(n) = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n$.

Получената формула се нарича ***формула на Бине***. Независимо че във формулата участват ирационални числа, тя винаги дава целочислен резултат, равен на n–тото число на Фибоначи.

Числата на Фибоначи притежават интересни свойства и са свързани със занимателни факти от различни области:

Стойността на λ_1 се бележи с гръцката буква φ и се нарича „***златно сечение***", станало символ на пропорционалност и хармония в природата, изкуството, архитектурата и др. Пропорциите, определени от златното сечение, се възприемат като еталон за естетичност в най-различни области на човешката практика.

Златното сечение е свързано със следната геометрична задача: в отсечка с дължина x да се намери точка, която я разделя на две части по такъв начин, че по-голямата част се отнася към по-малката, както цялата отсечка към по-голямата част:



$\frac{x+1}{x} = \frac{x}{1} \Rightarrow x^2 - x - 1 = 0$

Положителният корен на уравнението е точно „*златното сечение*":

$\varphi = \frac{1+\sqrt{5}}{2} = 1.618...$

Лесно се доказват някои от известните свойства на φ:

$\varphi = 1 + \frac{1}{\varphi}$

$\frac{1}{\varphi} = \frac{1}{1.618} = \varphi - 1 = 0.618 = -\frac{1-\sqrt{5}}{2}$

$0{,}618^2 = 1 - 0{,}618$; $1{,}618^2 = 1 + 0{,}618$

$\varphi^n = \varphi^{n-1} + \varphi^{n-2}$ за $\forall\ n \in N$

$F(n+1)F(n-1) - F(n)\,F(n) = (-1)^n$

Отношенията на две съседни числа на Фибоначи:

$\frac{F(n)}{F(n-1)} \to 1.618$, $\frac{F(n)}{F(n+1)} \to 0.618$, при $n \to \infty$,

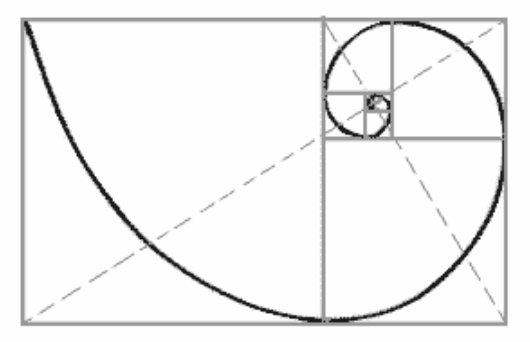
което може да се види на фиг. (2.1):

F(n)	F(n)/F(n-1)	F(n)/F(n+1)
1	1,000	1,000
1	1,000	0,500
2	2,000	0,667
3	1,500	0,600
5	1,667	0,625
8	1,600	0,615
13	1,625	0,619

Фиг. 2.1. Асимптотика на отношението на две съседни числа на Фибоначи

Ако построим квадрати със страни 1, 1, 2, 3, 5, ... равни на числата на Фибоначи по указания на схемата начин, и във всеки квадрат впишем четвърт окръжност, получаваме «*златната спирала*», която се среща много често в растителния и животински свят. Тази спирала е единствената, която запазва формата си при

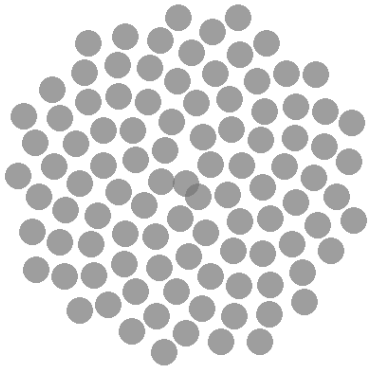
увеличаване на размерите. Това свойство може да се нарече "*самоподобие*" и има пряка връзка с фракталите.



Фиг. 2.2. Златната спирала

Златната спирала се наблюдава в много естествени животински, растителни и природни форми (2.2):

Фиг. 2.3. Спираловидни природни форми



Ако откъснете един ананас ще видите, че вътрешните прегради се разполагат спираловидно в две направления, като образуват 8 спирали в едната посока и 13 в другата, докато при боровите шишарки броят на спиралите най-често е 5 и 8.

Интересно е приложението на числата на Фибоначи за анализ и прогноза на финансовите пазари. Вълните, описващи колебанията на котировките на ценните книжа, съответстват на реда на Фибоначи и се наричат *вълни на Елиот* по името на Ралф Елиот, публикувал през 1939 г. серия статии във Financial World Magazine. В тях за първи път била представена теорията му, че движението на индекса Доу-Джонс се подчинява на определен ритъм, свързан с числата на Фибоначи.

2.3.4. Геометрия на самоподобието: фрактали

Фракталите са сложни, самоподобни геометрични обекти. Науката, която изучава фракталите и особеното им поведение се нарича ***фракталната геометрия***. Тя намира приложение в науката, техниката, компютърната графика, както и в много математически направления, като нелинейната динамика, теорията на хаоса и комплексните системи. Използва се при моделиране на сложни органични и геоложки системи, компресиране на данни и др. От особено значение за развитието на фракталната геометрия е

възможността за компютърна визуализация на изучаваните от нея обекти.

Фракталите са свързани с името на английския математик Бенуа Манделброт, който за пръв път се опитва да формализира представата за тези геометрични образувания, познати още от времето на Кантор. В книгата «Фрактална геометрия на природата», издадена през 1975 г., Манделброт описва съществената разлика между традиционните геометрични обекти и невъобразимо по-сложните формирования, каквито са фракталите, срещащи се в много природни форми, по следния начин: *„Защо често наричат геометрията "студена" и "суха" наука? Една от причините за това е неспособността ѝ да опише формата на облаците, планините, бреговата линия или дървото. Природата притежава не просто по-голяма сложност, а сложност от съвършено друго ниво”.* За описание на тези обекти Манделброт въвежда думата *фрактал.*

Фракталите са самоподобни геометрични обекти, които нямат ясна размерност. Самоподобността е видима при увеличаване на мащаба, когато се разкрива сходна картина. По принцип имат неравномерна (не гладка) форма и, следователно, не са обекти, изучавани от традиционната геометрия.

Наблюдават се три вида самоподобност:

Точна самоподобност — най-силната самоподобност. Фракталът е идентичен в различни мащаби. Фракталите, дефинирани от итеративни функции, често притежават това свойство.

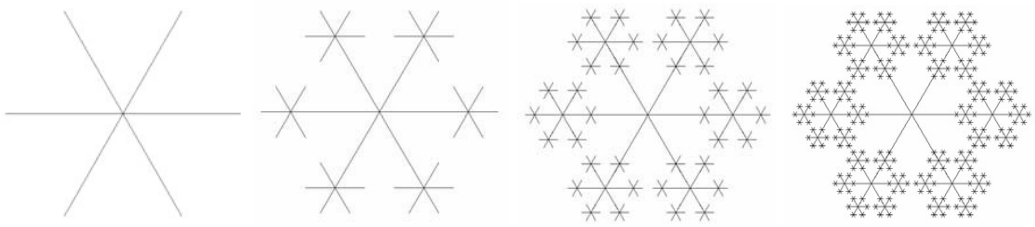
Квазисамоподобност — по-слаба форма на самоподобност, при която фракталът изглежда приблизително, но не напълно идентичен в различни мащаби. Квазисамоподобните

фрактали съдържат малки копия на целия фрактал в деформирани и изродени форми. Фракталите, дефинирани от рекурсивни функции, обикновено са квазисамоподобни, но не точно самоподобни.

Статистическа самоподобност — най-слабата форма на самоподобност. Фракталът има числени или статистически характеристики, които се запазват в различни мащаби. Най-убедителната дефиниция на „фрактал“ предполага някаква форма на самоподобност. Самата фрактална размерност е числена характеристика, която се запазва в различните мащаби. Случайните фрактали са пример за статистическа самоподобност.

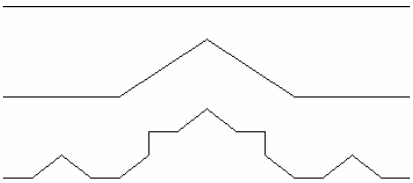
Примери за фрактали:

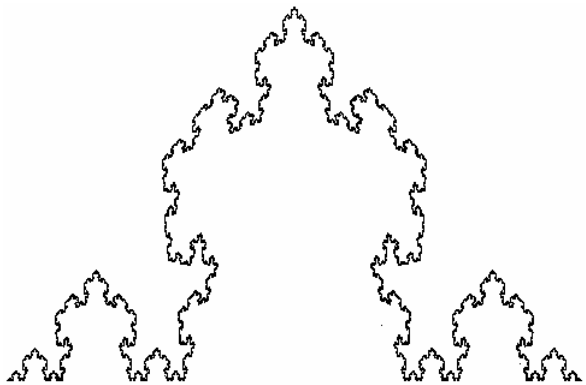
Снежинка на Кох: Получава се при последователно възпроизвеждане на шест симетрични лъча, започващи от обща точка.



Фиг.2.4. Снежинка на Кох

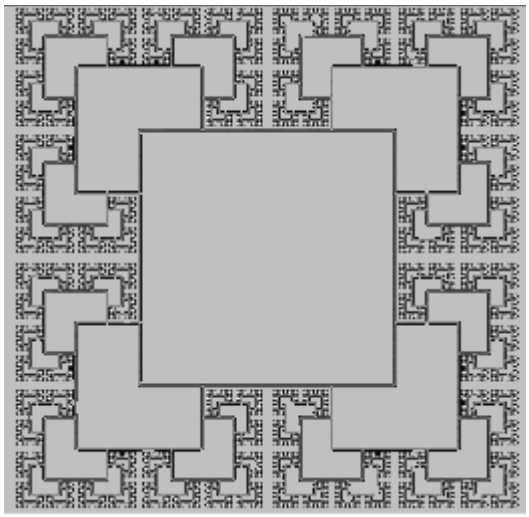
Крива на Кох: Дадена отсечка се дели на три равни части, средната част се заменя с начупена линия. Същото действие се повтаря за получените части. След многократни итерации се получава следната негладка форма:





Фиг. 2.5. Фрактал „Крива на Кох”

Множества на Кантор: Получават се при премахване на подобна част от някакъв геометричен обект. Например, разделяме отсечка на три равни части и „изрязваме” средата. Същото действие прилагаме към двете части. След безкраен брой итерации получаваме множество, което се нарича „прах” на Кантор. Друг известен фрактал се получава по следния начин: от даден квадрат изрязваме квадрат с двойно по-малка страна, разположен в средата. Разделяме квадрата на 4 равни части и повтаряме действието за всяка от тях.



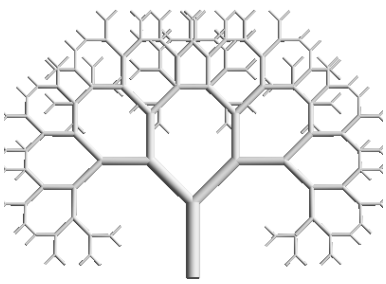
Фиг. 2.6. Фрактал „Множество на Кантор”

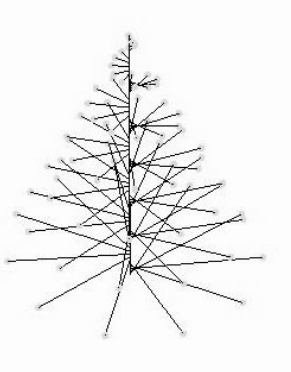
В резултат двумерното множество става едномерно – в достатъчно малка околност на всяка точка липсват други точки на множеството.

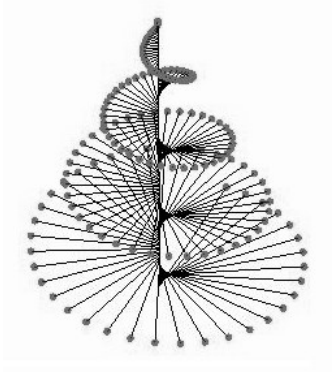
Много от програмите за визуализация на фрактали използват

като од ı на

прир ет







Фиг. 2.7. Програмно визуализирани фрактали

2.3.5. Метод на производящите функции

Методът на производящите функции е разработен от Ойлер през 50-те години на XVIII век. Производящите функции намират приложение при решаване на рекурентни зависимости, просто представяне на сложни комбинаторни редици, доказване на различни тъждества и много други комбинаторни задачи.

Функцията:

$$f(x) = \sum_{n=0}^{\infty} a_n x^n$$

се нарича ***производяща функция*** на редицата a_0, a_1, a_2, …a_n, … при условие, че безкрайният степенен ред в дясно на израза е сходящ. Ако редицата $\{a_n\}$ е крайна (т.е. членовете ѝ при някое $n > n_0$ са равни на нула), използваме означението:

$$f(x) = \sum_{n=0}^{n_0} a_n x^n$$

Например производяща функция на крайната редица 1, 2, 1, 0, 0, … е функцията:

$f(x) = 1x^0 + 2x^1 + 1x^2 = (1 + x)^2$.

Аналогично експоненциална производяща функция на редицата a_0, a_1, a_2, …a_n, … се нарича редът:

$f(x) = \sum_{n=0}^{\infty} a_n \frac{x^n}{n!}$

Съгласно формулата за бинома на Нютон, при a = 1, b = x получаваме:

$(1 + x)^n = 1 + C_n^1 x + C_n^2 x^2 + C_n^3 x^3 + \ldots + C_n^{n-1} x^n.$

Функцията:

$f(x) = (1 + x)^n$

е ***производяща функция за биномните коефициенти***.

Ще използваме метода на производящите функции за получаване на производящата функция на числата на Фибоначи:

0, 1, 1, 2, 3, 5, 8, 13, 21, …

$F(x) = 0x^0 + 1x^1 + 1x^2 + 2x^3 + 5x^4 + \ldots \Rightarrow$

$F(x) = F_0x^0 + F_1x^1 + F_2x^2 + F_3x^3 + F_4x^4 + \ldots$

Умножаваме двете страни на последното равенство на x и x^2 съответно, при което получаваме:

$xF(x) = F_0x^1 + F_1x^2 + F_2x^3 + F_3x^4 + F_4x^5 + \ldots$

$x^2F(x) = F_0x^2 + F_1x^3 + F_2x^4 + F_3x^5 + F_4x^6 + \ldots$

Сумираме двете равенства:

$xF(x) + x^2F(x) = F_0x^1 + (F_1 + F_0)\, x^2 + (F_2 + F_1)x^3 + (F_3 + F_2)\, x^4 + \ldots$

$= F_0x^1 + F_2x^2 + F_3x^3 + F_4x^4 + \ldots$

$= F_2x^2 + F_3x^3 + F_4x^4 + \ldots$

Към двете страни прибавяме x:

$F_0x^0 + F_1x^1 = x$, тъй като $F_0 = 0$, $F_1 = 1$.

Окончателно получаваме:

$x + xF(x) + x^2F(x) = F_0x^0 + F_1x^1 + F_2x^2 + F_3x^3 + F_4x^4 + \ldots = F(x) \Rightarrow$

$F(x) = x + xF(x) + x^2F(x) \Rightarrow$

Решаваме полученото уравнение спрямо F(x) и получаваме:

$F(x) = \frac{x}{1-x-x^2}$

Използвайки въведените означения, за ***производящата функция на редицата на Фибоначи*** получаваме следния компактен запис:

$$0 + x + x^2 + 2x^3 + 3x^4 + 5x^5 + \ldots = \sum_{n=0}^{\infty} F_n x^n = \frac{x}{1-x-x^2}$$

ЗАДАЧИ ЗА САМОПОДГОТОВКА

1. Какво трябва да знаем, за да определим еднозначно членовете на редиците:

   $x_n = x_{n-1} + x_{n-2} - x_{n-3}$

   $u_n = 2u_{n-1} - 4u_{n-2}$

   $a_n = a_{n-1} + 9a_{n-2} - 9a_{n-3}$

2. Задайте различни начални членове за редиците от зад. 1 и пресметнете първите няколко члена.

3. Определете общия член на редиците от задача 1.

4. Докажете, че общия член на рекурсивно дефинираната редица: $x_1 = 2$, $x_n = x_{n-1} + 2n$ допуска представяне във вида: $x_n = n(n+1)$ за всяко $n \in P$.

5. Докажете, че общия член на рекурсивната редица:

$x_n = 3x_{n-1} - 2x_{n-2}$, $n \geq 2$, $x_0 = 3$, $x_1 = 5$

се представя във вида:

$x_n = 2^n + 1$

6. Докажете разгледаните в (2.3.1.2) свойства на числата на Фибоначи.

7. Намерете редиците с производящи функции:

$$e^x = 1 + \frac{1}{1!}x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \ldots$$

$$\log(\frac{1}{1-x}) = x + \frac{1}{2}x^2 + \frac{1}{3}x^3 + \ldots$$

$$\sin(x) = x - \frac{1}{3!}x^3 + \frac{1}{5!}x^5 + \ldots$$

2.4. ОБЩИ КОМБИНАТОРНИ МЕТОДИ. МЕТОД НА ИЗЧЕРПВАЩОТО ТЪРСЕНЕ. МОДИФИКАЦИИ. ОЦЕНКА НА СЛОЖНОСТТА

2.4.1. Метод на изчерпващото търсене. Същност. Модификации

Методът на изчерпващото търсене (*exhaustive search*) е възможно най-общият, универсален метод за решение на комбинаторни задачи. Предполага пораждане на всички потенциални решения и проверка за допустимост на всяко от тях. Генерирането на множеството от възможни варианти се извършва чрез анализ на задачата и вида на допустимите решения. Съществено за практическата реализация на метода е пораждането на решения без пропуски и повторения.

Приложението на метода се възпрепятства от т.нар. „*комбинаторен взрив*”, свързан с много бързото (експоненциално) нарастване на броя на изследваните варианти при увеличение на размера на задачата. Например при играта на шах съществуват

10^{120} различни състояния на шахматната дъска и анализът на всички е практически неосъществим. „*Това число е по-голямо от броя на молекулите във Вселената или броя на наносекундите от момента на „големия взрив". Анализът на такъв брой състояния е извън възможностите на произволна изчислителна машина, чиито размери са ограничени до ограничена област на мирозданието, а реализацията трябва да завърши преди тази област да се подложи на разрушителното действие на ентропията"* [17].

Изчерпващото търсене има смисъл, ако множеството от възможните решения на задачата е крайно, т.е., ако подлежат на генериране и проверка краен брой варианти. Тогава по принцип може да се вземе всяко потенциално решение от това множество и да се провери дали удовлетворява условието на задачата. Ако задачата се решава ръчно, този брой е около 1 000 (примерно!), използването на компютър може да го увеличи до няколко милиона. Но потенциалният брой варианти на много реални задачи далеч надхвърля възможностите на изчислителната техника. Поради тази причина един от основните проблеми при използване на метода е съкращаване на вариантите до практически осъществими мащаби.

В случай, че се търси оптимално решение, се въвежда функция (*целева функция, критерий за оптималност, критерий за качество на решението*), която дава възможност за оценка на решението. Изборът на оптимално решение се базира на сравняване на стойностите на целевата функция за всяко генерирано решение. Изчерпването на всички възможни решения гарантира оптималността на получения резултат.

Метод на пораждане и проверка – вариант на метода на изчерпващото търсене, при който генерирането и проверката на

потенциалните решения се преустановява при получаване на допустимо решение. Може да се използва в случай, че се търси допустимо, а не оптимално решение.

Пример 2.11. Санкционирането на достъпа до информация в компютърните мрежи е свързано с използване на ключове. При дължина на ключа n-двоични разряда „отгатването" му е свързано с генериране и проверка за съвпадение на 2^n двоични вектора с дължина n. Задачата е типичен пример за реализация на метода на пораждане и проверка и следва следната схема:

1. Пораждане на поредния ключ.
2. Проверка за съвпадение. Ако ключът е търсеният, край. В противен случай – (1).

Започвайки от първия ключ 000 ... 0, следвайки схемата, след краен брой стъпки ще получим търсения. Всеки следващ ключ получаваме чрез добавяне на 1 в младшия двоичен разряд.

Размерът на пространството на състоянията при различни стойности на n е следният:

64 бита: 2^{64} = 18 446 744 073 709 551 616 възможни ключа;

128 бита: 2^{128} = 340 282 366 920 938 463 463 374 607 431 768 211 456 възможни ключа;

256 бита: 2^{256} = 115 792 089 237 316 195 423 570 985 008 687 907 853 269 984 665 640 564 039 457 584 007 913 129 639 936 възможни ключа.

От статистическа гледна точка средно трябва да проверим половината ключове, докато получим правилния. Всеки допълнителен бит удвоява количеството на възможните ключове и съответно времето за реализиране на изчерпващо търсене. Задачата допуска разбиване на пространството на състоянията на непресичащи се части, които могат да бъдат обхождани на

паралелно работещи компютри. Интересен пример за сравнение: на 22 август 1999: число с размер 155 десетични цифри (512 бита) било факторизирано за шест месеца изчисления. Използвани са около 600 изчислителни машини.

Метод на клоните и границите (*търсене с връщане, backtracking*) – вариант на метода на изчерпващото търсене, при който се генерират не всички, а част от решенията с цел намаляване на пространството на търсене. Отхвърлянето на варианти става на базата на евристики, използване на допълнителни условия, анализ на възможните решения и др.

Основен недостатък на тази група методи е, че не могат да гарантират получаване на решение за разумно време, ако времето за решение расте експоненциално при увеличаване на размера на задачата. Въпреки това, те са единствен метод за решение на много задачи (наричани NP пълни задачи, които ще бъдат разгледани по-късно), което обяснява интереса към тях и практическото им приложение.

2.4.2. Изчерпващо търсене и евристики. Практически аспекти

Най-общо, търсенето се осъществява чрез систематично обхождане на елементите на множеството, съдържащо всички потенциални решения на задачата. Това множество, структурирано по определен начин, се нарича ***пространство на състоянията на задачата*.**

Пространството на състоянията се задава чрез четворката [V, E, S, GD], където:

V – крайно, а следователно и изброимо множество от състояния в процеса на решение на задачата;

E - множество от допустими преходи от едно състояние в друго;

S - непразно множество от начални състояния на задачата;

GD - непразно множество от целеви състояния на задачата.

Може да има различна структура и да се описва чрез дърво, ацикличен граф, произволен граф с примки и кратни дъги и др., които ще бъдат разгледани по-късно. Структурата на пространството на състоянията влияе съществено върху процедурите за търсене, реализиращи систематично обхождане на елементите, и дава възможност да се анализира сложността на задачата.

„*Въпреки приложимостта на изчерпващото търсене при произволни пространства на състоянията, огромният им размер в интересните задачи прави подхода практически неизползваем*" [17]. Убедителен пример за това е шахматът. Ако методът гарантира получаване на резултат, може да се разработи винаги побеждаваща компютърна програма. Достатъчно е във всеки момент да се оценяват всички възможни състояния на шахматната дъска и да се следват ходове, които водят до победа. Изчерпателен анализ на практическите аспекти при разработка на процедури за търсене в различни пространства на състоянията е дадено в [17].

При решение на сложни реални проблеми, които най-често се решават в условията на неопределеност, експертите в дадена област реализират търсене в големи пространства на състоянията, като за целта използват както формализирани научни знания в дадена област, така и неформализирани съображения, почиващи на здравия разум, минал опит и т.н. Например в медицинската диагностика пространството на състоянията включва всички възможни диагнози, определени от наблюдаваните симптоми.

Действията на експертите демонстрират интелектуално поведение, водещо до намиране на приемливо решение, независимо от практически необозримия брой варианти. Специалистите използват търсене, но не в цялото пространство, а в негова ограничена част: лекарят анализира няколко възможни диагнози, шахматистът разглежда и оценява няколко хода. В практиката специалистите следват субективни правила, които насочват търсенето към онези части на пространството на състоянията, които по някакви причини изглеждат „обещаващи". Тези правила се наричат ***евристики*** (от гръцки *heuristics* – намирам) и са предмет на изследване в изкуствения интелект.

Евристиката е стратегия за изборно търсене в пространството на състоянията. Тя ограничава търсенето в практически обозрими части на пространството, които изглеждат перспективни, имат по-голяма вероятност за успех по някакви причини. Рестрикцията на пространството е свързана с използване на съществени знания за естеството на задачата. Евристиките оптимизират търсенето чрез намаляване на броя на анализираните варианти, но не винаги са гаранция за успех.

Въпреки очевидната универсалност, методът на изчерпващото търсене не е достатъчен за моделиране на интелектуалното поведение на експертите в процеса на търсене на решение, но е важен инструмент за проектиране на интелектуални програми.

От всичко казано дотук следва, че *„методът на изчерпващото търсене осигурява средства за формализиране на процеса на търсене на решение, а евристиките внасят интелект в този формализъм"* [17].

Пример 2.12. Задача за 8-те царици. В задачата се иска 8 царици да се разположат на шахматната дъска по такъв начин, че никои две да не стоят на един ред, един стълб и един диагонал. Пълното изчерпване предполага генериране на всички възможни позиции на 8-те царици върху шахматната дъска и проверка за всяка от тях дали удовлетворява условието. Първата царица можем да разположим на една от възможните 64 клетки. Ако сме фиксирали мястото на първата, за втората остават 63 възможни положения, за третата – 62 и т.н. Общият брой състояния е:

64.63.62.61.60.59.58.57 = 1,78463E+14.

Съвременните компютри могат да обработят такъв брой състояния. Но съществуват задачи, чиито пространства са изключително сложни и практически необозрими. Ефективните стратегии за обхождане в пространства с голяма размерност използват евристики, позволяващи да се намали броят на изследваните състояния. В задачата за цариците евристичният подход предполага следното: при поставяне на поредната царица да изключим позициите, които вече се „бият" от поставените до момента царици.

2.4.3. Стратегии на търсене

Търсенето в пространството на състоянията може да бъде реализирано в две направления: от изходните данни на задачата към целта и в обратно направление от целта към изходните данни.

Търсене от данните към целта (*права верига на обхождане, forward – напред*): първо се анализира началното състояние (съвпада ли с целта?), към него се прилагат допустимите правила за изменение на състоянието, в резултат се получава ново състояние на системата, което се анализира и т.н. Процесът на

обхождане се прекратява при достигане на целта или до ситуация, при която целта не е достигната, но движение напред не може да се осъществи.

Търсене от целта към данните (*обратна верига на обхождане, backward - назад*) – търсенето започва от целта, анализират се правилата за изменение на състоянието, които водят към нея, и условията за използването им. В резултат се формира нова цел, към която се прилага аналогичен подход. Търсенето в обратно направление продължава до достигане на началното състояние или установяване на факта, че задачата няма решение. Тази стратегия напомня движение в лабиринт в обратна посока, от крайното състояние към началното, за да проверим дали съществува път.

Изборът на стратегия за обхождане зависи от естеството на решаваната задача, структурата на пространството на състоянията, вида на данните и др.

Пример 2.13. Родословно дърво [17]. Задачата се състои в проверка на твърдението „Аз съм потомък на Томас Джеферсън". При права верига строим дървото, започвайки от корена „аз", двамата родители, техните родители и т.н. Ако разликата е 10 поколения, следвайки схемата, ще получим 2^{10} предшественици. Твърдението е вярно, ако Томас Джеферсън фигурира между тях, и невярно в противен случай. При обратна верига започваме от върха „Томас Джеферсън", анализираме неговите наследници, след това наследниците на наследниците... Ако допуснем, че средният брой наследници е 3, то при обратната верига ще се наложи да анализираме 3^{10} наследници. В тази конкретна задача обхождането „напред" е по-ефективно от обхождането „назад".

Изчерпващото търсене и неговите модификации са възможно най-общите методи за решение на дискретни комбинаторни задачи. Те са единствените методи за решение на NP – пълните задачи. Методът е еталон за неефективност, което не означава, че реализацията му е проста. Модификациите не водят до качествено изменение на поведението на метода при произволни входни данни, но могат съществено да ускорят работата в конкретен случай. Използването на евристики внася интелект в търсенето и позволява да се намали броят на изследваните варианти.

ЗАДАЧИ ЗА САМОПОДГОТОВКА

1. Задача за щастливите билети. Даден билет е щастлив, ако сумата от първите три цифри на шестцифрения му номер съвпада със сумата на последните три. Да се намерят номерата на всички щастливи билети.

   *Потенциалните решения са шестцифрените числа. Генерираме ги последователно и за всяко проверяваме изпълнението на условието.

2. Кафе машина работи с монети от 10, 20, 50 стотинки и 1 лев. Предлага различни напитки с цени 30, 40, 50 и 60 стотинки. Съставете алгоритъм за връщане на ресто в случай, че сумата превишава цената на напитката.

3. За какви стойности на булевите променливи X_1, X_2, X_3 и X_4 има стойност „истина” изразът:

$(X_1 \vee X_4) \wedge (\overline{X}_1 \vee X_2) \wedge (X_1 \vee \overline{X}_3) \wedge (\overline{X}_2 \vee X_3 \vee \overline{X}_4) \wedge (\overline{X}_1 \vee \overline{X}_2 \vee \overline{X}_3)$

4. Задача за раницата. Двама приятели решили да отидат на поход, събрали и претеглили всички необходими вещи. Как

да разделят предметите на две части по най-честния начин?

* Формално: дадено е множество от естествени числа, някои от които могат да съвпадат. Необходимо е да се раздели на две подмножества по такъв начин, че разликата от сумата им да бъде минимална.

5. Задача на Айнщайн. Има 5 къщи, оцветени в различен цвят. Във всяка къща живее един човек - немец, англичанин, швед, датчанин, норвежец. Всеки от живущите пие точно определена напитка, пуши определена марка цигари и отглежда определено животно.

   Въпрос: Кой отглежда в къщата си рибка?

   Когато Айнщайн измислил тази загадка, предполагал, че 98% от жителите на Земята не са в състояние да я решат. Потенциалните решения могат да се представят чрез дърво, всеки възел на което има по пет наследника. Решението е път, свързващ корена с листата. Общият брой възможни решения е 5.5.5.5.5 = 3125. Пълното изчерпване на множеството от допустими решения без компютър е трудна задача. Допълнителните условия позволяват „отсичане” на клони от дървото, което значително намалява броя на възможните варианти.

   <u>Допълнителни условия:</u>

   1. Англичанинът живее в червена къща.
   2. Шведът отглежда куче.
   3. Датчанинът пие чай.
   4. Зелената къща е от ляво на бялата.
   5. Живеещият в зелената къща пие кафе.
   6. Човекът, който пуши «Pall Mall», отглежда птица.

7. Жителят на средната къща пие мляко.

8. Жителят на жълтата къща пуши «Dunhill».

9. Норвежецът живее в първата къща.

10. Пушещият «Marlboro» живее около този, който има котка.

11. Този, който има кон, живее около този, който пуши «Dunhill».

12. Пушещият «Winfield» пие бира.

13. Норвежецът живее около синята къща.

14. Немецът пуши «Rothmans».

15. Пушещият «Marlboro» е съсед на този, който пие вода.

Задача: Да се построи пълното дърво на решенията. Като се използват допълнителните условия, да се изключат клоните, които не могат да бъдат потенциални решения.

6. Задача парламент (Задача от студентска олимпиада в Москва, 1998 г.) В съкратен вариант задачата е следната:

   Съгласно Конституцията на една островна държава, новият премиер-министър се назначава от президента, но трябва да бъде утвърден от парламента. Отчитайки опозиционните настроения в парламента по отношение на новата кандидатура, президентът решил да се обърне за помощ към някакъв финансов магнат.

   За подпомагане на избора, финансовият магнат определя някаква сума от X тугрики (валутата на държавата), която да послужи за убеждаване на депутатите да гласуват за предложения премиер. Парламентът се състои от N ($0 < N < 1000$) депутати, които винаги присъстват на всички заседания и винаги гласуват или „за”, или „против” (няма негласували или въздържали се). Предварително е

известно отношението на всеки депутат към предложения премиер. Това отношение се изразява чрез вероятността p^0_i ($0 <= p^0_i <= 1$), с която i-тият депутат гласува за кандидатурата. Ако за убеждаването на i-тия депутата се дадат K_i тугрика, депутатът обещава да поддържа кандидата, но тъй като гласуването е тайно, в действителност депутатът гласува „за” с вероятност p^1_i ($0 <= p^1_i <= 1$). На президента е ясно, че ако си осигури 70 % от гласовете на депутатите, кандидатурата ще бъде утвърдена.

От друга страна, президентът иска да икономиса част от предоставената сума. Затова е готов да заплати 10 % от икономисаната сума за най-добър вариант за убеждаване на депутатите. Експертна група е готова да направи необходимите разчети, но при условие, че във всеки случай ще получи минимум M тугрика (и максимум – 10 % от икономисаната сума). Ако сумата, предоставена от финансовия магнат е по-малка от M, групата отказва да прави разчет. Ако се окаже, че сумата е по-голяма от M, но недостатъчна за осигуряване на положителния вот (като се отчете заплащането на труда на групата), експертната група ще пресметне на какъв минимален процент гласували „за” може да разчита президентът. Целият остатък от сумата X, неизразходван за депутатите и за групата експерти, остава в разпореждане на президента.

Пълният текст на задачата може да се види на адрес: http://unicorn.cmc.msu.ru/problems/1998-1/c/.

***Указание:** Решение на задачата е списък от индексите на депутатите, на които трябва да се плати. За всеки депутат

има два варианта: или му плащат, или не. Множеството от възможни решения е набор от наредени N-торки, на i-тата позиция, в които има флаг "да се плати/да не се плати" за i-тия депутат. Лесно се съобразява, че "наредените N-торки" са N-разрядни двоични числа. Пространството на потенциалните решения ще съдържа всички възможни варианти, т.е. от $000...0_2$ до $111...1_2$. Изчерпващото търсене може да се осъществи по нарастване на елементите, като се започне от 0 и се завърши с 2^N-1.

7. Пресметнете времето за „разбиване” на 64–битов код, ако компютърът извършва 1 000 000 операции в секунда.

глава

3

РАЗМИТИ МНОЖЕСТВА И МОДЕЛИ

- *Теория на размитите множества*
- *Операции над размити множества*
- *Размито моделиране на реални задачи*

3.1. ТЕОРИЯ НА РАЗМИТИТЕ МНОЖЕСТВА

3.1.1. Теория на размитите множества. Същност

Решаването на реални задачи е свързано с формализация на обекта или системата и прилагане на абстрактен метод за търсене на решение върху построения математически модел. Процесът на решение неминуемо се съпровожда с голяма степен на абстракция, на идеализация на реалните обекти и явления, които често се оказват твърде сложни, за да се вместят в тесните рамки на традиционната логика, в която централна роля играят точните, строги, количествени разсъждения.

В същото време експертите в дадена област са способни да намират оптимални или близки до оптималните решения, независимо от сложността на задачата, големия брой решения, непълнотата на входните данни, неточността на критериите и др., произтичащи от неизчерпаемата сложност на реалния свят.

Анализът на мисловния процес на експерта в процеса на вземане на решение дава възможност да заключим, че човешкото мислене е способно да работи в условията на неопределеност, неточност от различен характер, да използва размити понятия и методи на разсъждение и да взема правилни решения в размита информационна среда

Опит за преодоляване на тези противоречия е създаването на *теорията на размитите множества на Заде* . Тя възниква като реакция на неудовлетвореност от методите на класическата математика, която принуждава специалистите да прибягват до изкуствена точност, неуместна при много реални задачи.

Теорията на размитите множества (ТРМ) представлява формален апарат за описание и анализ на сложни обекти и явления в условията на неопределеност от различен тип. Тя дава схема за решаване на проблеми, в които значителна роля за отчитане на факторите на неопределеност играе субективната оценка. Централно място в ТРМ заемат функциите за принадлежност, представляващи субективна мяра за съответствие на обект от дадено множество обекти към понятие, смисълът на което се формализира чрез размито множество.

Човешките разсъждения в процеса на вземане на решения са по-скоро приблизителни, отколкото точни. Построяването на модели на приближени разсъждения и използването им в компютърните системи се разглежда като "*един от най-важните проблеми на съвременната наука*". При това моделите, в които се използват размити формализми за моделиране на човешките разсъждения, дават по-добра познавателна имитация от моделите, в които се използват традиционни методи.

Предимствата на моделите, базирани върху ТРМ, се оценяват в две основни насоки:

От една страна, се базират върху идеята за размитост (неясност, неточност, неистинност, недостатъчност, противоречивост) на входните данни, което е напълно адекватно на реалните ситуации.

От друга страна, дават възможност за формализация и теоретична обосновка на неясните, неточни, размити човешки разсъждения в процеса на вземане на решения.

3.1.2. Основни понятия

Основен принцип в класическата теория на множествата е това, че определен елемент може или да принадлежи, или да не принадлежи на дадено множество. В ТРМ Заде се отказва от този принцип и въвежда понятието функция на принадлежност (*Membership function*), аналогична на понятието характеристична функция на множество, но приемаща стойности в интервала [0, 1].

Нека X е множество с елементи $x \in X$ и нека A да е подмножество на X, съдържащо елементи, които притежават някакво свойство p. Ако характеристичната функция $\mu_A(x)$ приема стойности в интервала [0, 1], то за елементите $x \in X$ по отношение на свойството p са възможни следните случаи:

$\mu_A(x) = 0$ - не притежава свойството p;

$\mu_A(x) \in (0,1)$ - притежава p в някаква степен;

$\mu_A(x) = 1$ - напълно притежава p.

Съвкупността от всички двойки $(x, \mu_A(x))$, за които $x \in X$, а $\mu_A(x)$ е функция, дефинирана върху X и приемаща стойности в интервала [0, 1], се нарича ***размито множество*** (РМ) A над множеството X.

Функцията $\mu_A(x):X—>[0, 1]$ се нарича ***функция на принадлежност***, а множеството от стойности, които тя приема - ***множество на принадлежност***. Множеството X се нарича ***базово множество***, а стойността на $\mu_A(x)$ - ***степен на принадлежност*** на елемента x към РМ A.

Размитото множество A над X се задава по един от следните начини:

$A = \{x \in X: \mu_A(x)\}$,

$A = \{x_1/\mu_A(x_1),...,x_i/\mu_A(x_i),...,x_n/\mu_A(x_n)\}$,

или таблично:

x_1	x_2	…	x_i	…	x_n
$\mu_A(x_1)$	$\mu_A(x_2)$	…	$\mu_A(x_i)$	…	$\mu_A(x_n)$

където x_1 ... x_n са елементи на X, а $\mu_A(x_i)$ е степента на принадлежност на елемента x_i към размитото множество A.

Празно размито множество се нарича размитото множество A над X, за което $\mu_A(x) = 0$ за всяко $x \in X$. Означава се с 0.

Универсално размито множество се нарича размитото множество A над X, за което $\mu_A(x) = 1$ за всяко $x \in X$. Означава се с U.

Носител на размито множество A над X се нарича обикновеното множество $\{ x: x \in X, \mu_A(x) > 0\}$. Означава се със supp A.

Размитото множество A над X се нарича ***нормално***, ако съществува $x \in X$, за който $\mu_A(x) = 1$. Ако едно размито множество A над X не е нормално, то се нарича субнормално.

Нека A и B са две размити множества над X.

Размитото множество A над X се нарича ***подмножество*** на размитото множество B над X, ако $\mu_A(x) \leq \mu_B(x)$ за всяко $x \in X$. Означава се с $A \subseteq B$.

Размитите множества се наричат ***равни***, A = B, ако $\mu_A(x) = \mu_B(x)$ за всяко $x \in X$. Ако A и B не са равни, те се наричат ***различни***. Означава се с $A \neq B$.

Ниво α на размитото множество A над X ще наричаме ***обикновено множество*** A^α, чиито елементи са от X и степента им на принадлежност към A не е по-малка от α, т.е.:

$$A^\alpha = \{ x : x \in X, \mu_A(x) \geq \alpha \}.$$

3.1.3. Операции върху размити множества

В съответствие с операциите над обикновени множества, се въвеждат операциите допълнение, сечение и обединение на размити множества. Аксиоматичното им задаване дава възможност да се дефинират по различен начин в реалните задачи, за да могат най-точно да отразят разнообразната семантика на логическите връзки “и”, “или”, “не” в практиката.

3.1.4. Операции на Заде

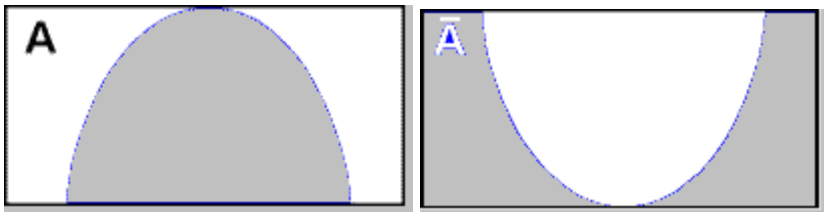
Заде предлага операциите допълнение, сечение, и обединение върху размити множества да се дефинират по следния начин:

Нека A, B и C са размити множества над X.

Допълнение на размитото множество A над X ще наричаме размитото множество B над X, ако:

$\mu_B(x) = 1 - \mu_A(x)$ за всяко $x \in X$.

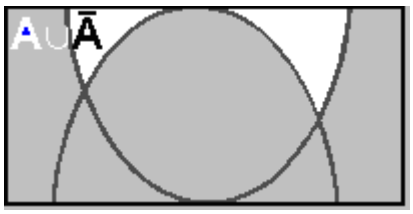
Означава се $B = \bar{A}$.



Фиг. 3.1. Допълнение на размито множество

Обединение на размитите множества A над X и B над X ще наричаме размито множество C над X, ако:

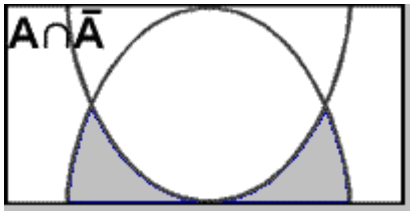
$\mu_C(x) = \max(\mu_A(x), \mu_B(x))$ за всяко $x \in X$. Означава се C = A U B.



Фиг. 3.2. Обединение на на размити множества

Сечение на размитите множества A над X и B над X ще наричаме размито множество C над X, ако:

μC(x) = min(μA(x), μB(x)) за всяко x¤ X. Означава се C = A ∩ B.



Фиг. 3.3. Сечение на размити множества

Мини-максните операции на Заде дават добри резултати, ако РМ са дефинирани върху един и същ универсум. В практиката обаче се налага да се извършват операции върху РМ, свързани с

различни универсуми, за които операциите на Заде не винаги дават възможност за адекватно моделиране на логическите връзки “и”, “или”, “не”.

3.1.5. Триъгълни норми и конорми

Един от подходите за построяване на обобщени оператори за сечение и обединение е задаването им в класа на триъгълните норми и конорми.

Триъгълна норма (Т-норма) се нарича функцията:

$T:[0,1] \times [0,1] \rightarrow [0,1]$,

удовлетворяваща следните условия:

1. ограниченост

$T(0, 0) = 0$ $T(1, \mu_A) = T(\mu_A,1) = \mu_A$

2. монотонност

$T(\mu_A, \mu_B) \leq T(\mu_C, \mu_D)$ ако $\mu_A \leq \mu_C$, $\mu_B \leq \mu_D$)

3. комутативност

$T(\mu_A, \mu_B) = T(\mu_B, \mu_A)$

4. асоциативност

$T(\mu_A, T(\mu_B, \mu_C))=T(T(\mu_A, \mu_B), \mu_C)$.

Частни случаи на триъгълните норми са:

- $\min(\mu_A, \mu_B)$
- $\mu_A.\mu_B$
- $\max(0, \mu_A + \mu_B - 1)$.

Триъгълна конорма (Т-конорма) се нарича функцията:

$T:[0, 1] \times [0, 1] \rightarrow [0, 1]$,

удовлетворяваща следните условия:

1. ограниченост

$T(1, 1) = 1$ $T(0, \mu_A) = T(\mu_A, 0) = \mu_A$

2. монотонност

$T(\mu_A, \mu_B) \geq T(\mu_C, \mu_D)$ ако $\mu_A \geq \mu_C, \mu_B \geq \mu_D)$

3. комутативност

$T(\mu_A, \mu_B) = T(\mu_B, \mu_A)$

4. асоциативност

$T(\mu_A, T(\mu_B, \mu_C)) = T(T(\mu_A, \mu_B), \mu_C)$.

Частни случаи на триъгълните конорми са:

- $\max(\mu_A, \mu_B)$
- $\mu_A + \mu_B - \mu_A.\mu_B$
- $\min(1, \mu_A + \mu_B)$.

В практиката триъгълните норми се използват за моделиране на операцията сечение, а конормите – за моделиране на операцията обединение на размити множества. Изборът е експертен и зависи от семантиката на използваните в лингвистичните модели размити данни и умозаключения.

3.1.6. Алгебрични операции над размити множества

Нека X е универсум, а A и B са РМ над X.

Алгебрична сума A + B на РМ A и B се определя чрез следната ФП:

$\mu_{A+B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x)\mu_B(x))$, за всяко $x \in X$.

Алгебрично произведение A.B на РМ A и B се задава чрез ФП:

$\mu_{A.B}(x) = \mu_A(x).\mu_B(x)$, за всяко $x \in X$.

На базата на операцията алгебрично произведение се дефинира и операцията ***степенуване*** на РМ. Ако $\alpha \geq 0$, то A^α се задава чрез ФП:

$\mu_{A^\alpha}(x) = (\mu_A(x))^\alpha$, за всяко $x \in X$.

Частни случаи на степенуване са:

концентриране – $CON(A) = A^2$

разтягане – $DIL(A) = A^{0.5}$,

които се използват за моделиране на лингвистични неопределености.

Ако $A_1, A_2, \ldots A_n$ са РМ над $X_1, X_2, \ldots X_n$ съответно, то декартово произведение на размитите множества $A = A_1 \times A_2 \times \ldots A_n$ се нарича РМ над $X = X_1 \times X_2 \times \ldots X_n$ с ФП:

$\mu_A(x_1,x_2,\ldots x_n) = \min(\mu_{A_1}(x_1), \mu_{A_2}(x_2),\ldots \mu_{A_n}(x_n))$, за $\forall x_i \in X_i$, i=1,2,… n.

3.2. РАЗМИТО МОДЕЛИРАНЕ НА РЕАЛНИ ЗАДАЧИ

3.2.1. Размити и лингвистични променливи

Размитите и лингвистични променливи се използват за моделиране на размити понятия и обекти.

Размитата променлива α се задава с тройката (α, X, A), където:

α е име на променливата;

X - дефиниционна област на α;

A - размито множество, описващо ограниченията, налагани върху стойностите на α.

Лингвистичните променливи (ЛП) се задават чрез (β, T, X, G, M), където:

β е име на лингвистичната променлива;

T е множество от стойности на ЛП, което се нарича терм-множество. Стойностите на ЛП са лингвистични терми. Всяка стойност на ЛП от T представлява име на РП, дефинирана върху универсума X;

X - дефиниционна област на β;

G – съвкупност от правила за генериране на нови значения на ЛП на базата на термите от T. Ако G(T) е множество от генерирани терми, то G(T) U T се нарича разширено терм-множество на ЛП;

M – съвкупност от правила за преобразуване на всяко ново значение на ЛП, получено по правилата G, в размита променлива.

3.2.2. Размити числа

Размитите числа намират голямо приложение за моделиране на размити множества, свързани с числов универсум. Въпросите по представянето и аритметиката на размити числа са детайлно и прецизно разработени в литературата, което облекчава използването им в практиката.

Под ***размито число*** се разбира РМ A върху множеството на реалните числа R, т.е. размита променлива върху реалната права с ФП:

$$\mu_A(x) : X \rightarrow [0,1]$$

Размитото число a е ***нормално***, ако:

$\max(\mu_A(x)) = 1.$

$x \in R$

Ако условието $\mu_A(x) = 1$ се изпълнява за точно една точка $x=a \in R$, т.е. $\mu_A(a) = 1$, то числото А се нарича ***унимодално***, а числото а се нарича ***мода***.

Подмножеството $R_A \in R$ се нарича ***носител на размитото число*** А, ако:

$R_A = \{x \mid \mu_A(x) > 0\}.$

Един от начините за задаване на размити числа е определянето им в класа на ***L-R функциите***. Това са функции на реални променливи, удовлетворяващи следните свойства:

$L(-x) = L(x),$

$R(-x) = R(x),$

$L(0) = R(0).$

В класа на L-R функциите унимодалното размито число А с мода а се задава чрез следната ***функция за принадлежност***:

$$\mu_A(x) = \begin{cases} L((a-x)/\alpha) \text{ при } x \leq a; \\ R((x-a)/\beta) \text{ при } x \geq a, \end{cases}$$

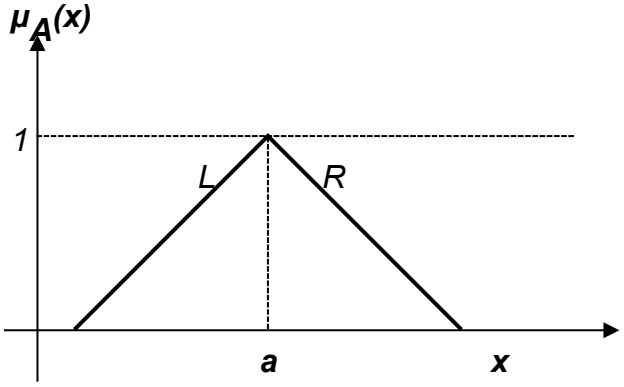
където а е мода, а $\alpha > 0$ и $\beta > 0$ са ляв и десен коефициент на размитост.

В литературата се предлагат различни аналитични представяния за L-R – функциите, както линейни, така и нелинейни. При зададени L и R унимодалното размито число А се задава чрез тройката $A = (a, \alpha, \beta)$, а толерантното размито число А се задава с

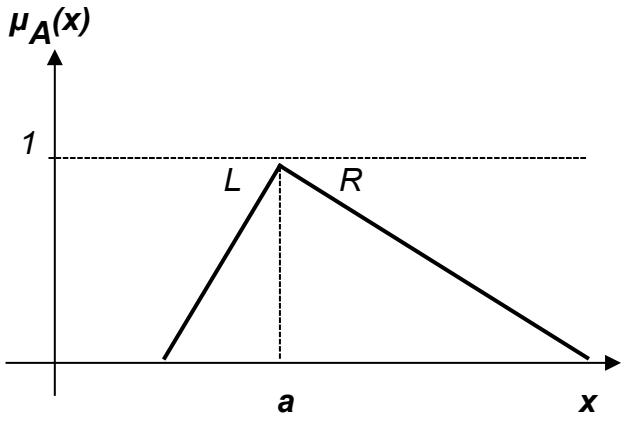
четворката A = (a_1, a_2, α, β), където [a_1, a_2] е интервал на толерантност, в който ФП $\mu_A(x)$ приема стойност 1 за $\forall$ x $\in$ [a_1, a_2].

Графичното представяне на размити числа може да бъде дадено по следния начин (в примерите използваме линейни функции, въпреки че могат да бъдат избрани произволни S-образни функции):

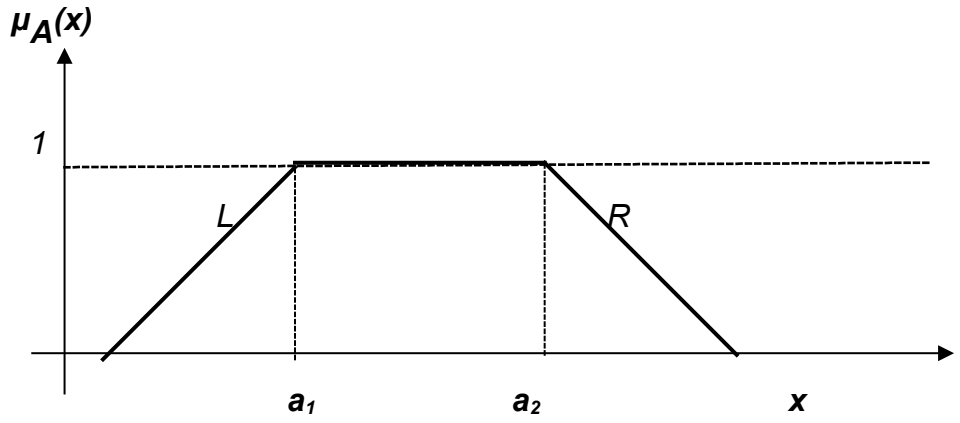
1. Унимодално размито число от вида A = (a, α, β), L= R:



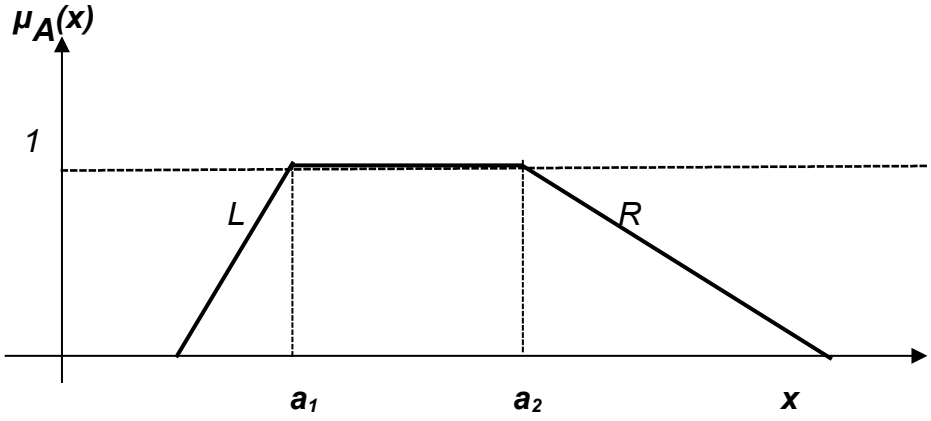
2. Унимодално размито число от вида A = (a, α, β), L ≠ R:



3. Толерантно размито число от вида A=(a_1,a_2,α,β), L= R:



4. Толерантно размито число от вида A=(a_1,a_2,α,β), L ≠ R:



3.2.3. Размити отношения

Нека $X = X_1 \times X_2 \times \ldots X_n$ е декартово произведение на базовите множества X_1, X_2,….X_n с базови променливи x_i съответно, а L е множество на принадлежност.

Размито отношение се нарича размито множество R над X със стойности в L:

$R: X \rightarrow L$

При n = 2 и L = [0,1] размито отношение между множествата X и Y и се нарича функцията $R: X \times Y \rightarrow [0,1]$, която съпоставя на

всяка двойка елементи (x,y), стойността $\mu_R(x,y) \in [0,1]$. Размитото отношение R: $X \times Y \rightarrow [0,1]$ се нарича размито отношение над X и се използва за моделиране на семантиката на отношения на сходство, подобие, близост, различие и други, подобни по смисъл лингвистични връзки между принадлежащи на един и същ клас обекти.

По аналогия с операциите върху РМ се дефинират и операции върху размити отношения. Нека R_1 и R_2 са размити отношения върху $X \times Y$.

Допълнение Ř на отношението R се нарича РМ над $X \times Y$ с ФП:

$$\mu_{\check{R}}(x,y) = 1-\mu_R(x,y)$$

Обединението $R_1 \cup R_2$ на размитите отношения R_1 и R_2 се задава чрез ФП:

$$\mu_{R1UR2}(x,y) = \mu_{R_1}(x,y) \vee \mu_{R_2}(x,y)$$

Сечение $R_1 \cap R_2$ на отношенията R_1 и R_2 се задава чрез ФП:

$$\mu_{R1\cap R2}(x,y) = \mu_{R_1}(x,y) \wedge \mu_{R_2}(x,y)$$

Алгебрична сума $R_1 + R_2$ на отношенията R_1 и R_2 се задава чрез ФП:

$$\mu_{R1+R2}(x,y) = \mu_{R_1}(x,y)+\mu_{R_2}(x,y)$$

Алгебрично произведение $R_1.R_2$ на отношенията R_1 и R_2 се задава чрез ФП:

$$\mu_{R1.R2}(x,y) = \mu_{R_1}(x,y).\mu_{R_2}(x,y)$$

Проекции на размити отношения се задават по следния начин:

Нека R е размито отношение над $X \times Y$, $R: X \times Y \to [0,1]$ с ФП:

$\mu_R(x,y): X \times Y \to [0,1]$.

Проекция на R върху X (първа проекция) се нарича РМ R_1' над X с ФП:

$$\mu_{R_{1'}}(x) = \vee_y \mu_R(x,y)$$

Проекция на R върху Y (втора проекция) се нарича РМ R_2' над X с ФП:

$$\mu_{R_{2'}}(y) = \vee_x \mu_R(x,y)$$

Величината $\vee_x \mu_{R_{1'}}(x) = \vee_y \mu_{R_{2'}}(y)$
се нарича ***глобална проекция*** на R.

От своя страна проекциите R_1' и R_2' на R определят в $X \times Y$ размити отношения $\check{R}_1'$ и $\check{R}_2'$ с ФП съответно:

$$\mu_{\check{R}_1}(x,y) = \mu_{R_{1'}}(x) \text{ за } \forall\, y \in Y$$

$$\mu_{\check{R}_2}(x,y) = \mu_{R_{2'}}(y) \text{ за } \forall\, x \in X$$

$\check{R}_1'$ се нарича цилиндрично продължение на R_1', а $\check{R}_2'$ – цилиндрично продължение на R_2'.

Нека:

$R_1: (X,Y) \to [0,1]$

$R_2: (Y,Z) \to [0,1]$

Композиция на размитите отношения R_1 и R_2 се нарича размито отношение над $X \times Z$, което се означава с $R_1 \bullet R_2$ и се задава със следната ФП:

$$\mu_{R1 \bullet R2}(x,z) = \vee_{y} (\mu_{R_1}(x,y) \wedge \mu_{R_2}(y,z)).$$

3.2.4. Размити импликации

Ако A и B са РМ над X и Y съответно, то под ***размита импликация*** "*ако A, то B*" се разбира начинът на задаване на размитото отношение R върху $X \times Y$, който съществено зависи от семантиката на лингвистично зададеното твърдение. Някои от използваните в литературата формализми за представяне на размити импликации чрез размити отношения и съответните им ФП са следните (проучването е направено от японските математици Танака, Фуками и Мидзумото, по-късно проф. Золотухин от Новосибирската лаборатория по РМ провежда експерименти и дава насоки за използването им в практиката):

Заключение:

Теорията на Заде е способ за формализация на неопределености от различен тип, възникващи при описание и изследване на сложни обекти и системи, което обуславя практическото й използване в най-различни области, някои от които са: теория на формалните езици и граматики, теория на алгоритмите, логика, разпознаване на образи, изкуствен интелект, оптимално управление, вземане на решения, прогнозиране, медицинска и промишлена диагностика, моделиране на финансовите пазари, психология, лингвистика, социология, икономика и др.

глава

4

ТЕОРИЯ НА NP ПЪЛНИТЕ ЗАДАЧИ

- **Изчислителна сложност на алгоритмите**
- **Теория на NP пълните задачи. Основни понятия**
- **Полиномиални и експоненциални алгоритми. Класове на сложност**
- **Характеристика на класа NP. Теорема на Кук**

4.1. ИЗЧИСЛИТЕЛНА СЛОЖНОСТ НА АЛГОРИТМИТЕ

Теорията на NP пълните задачи възниква в началото на 70-те години на XX век във връзка с необходимостта от теоретична оценка на изчислителната сложност на алгоритмите. Компютризацията на различни области на обществената практика води до възникване на широк клас задачи, които трябва да бъдат решени в условията на многоразмерност и времекритичност. Стремежът да се разработят достатъчно общи и ефективни алгоритми за решение на голям клас дискретни и комбинаторни задачи се сблъсква с определени трудности при практическата им реализация, свързани с голям разход на памет и машинно време, което ги прави практически неизползваеми. В същото време общата теория на алгоритмите изучава потенциалната възможност за алгоритмично решение на даден клас задачи, без да се отчитат необходимите за решението ресурси. Но наличието на алгоритъм за получаване на решение все още не значи, че решението може да бъде получено на практика. Възниква необходимост от по-задълбочено изследване на алгоритмите във връзка с практическата им реализуемост, която довежда до създаването на теорията на NP пълните задачи. Тя предоставя теоретична база за сравнение на алгоритмите, за оценка на тяхната ефективност и анализ на необходимите за решението на дадена задача изчислителни ресурси.

Пълно изложение на теорията на NP пълните задачи е дадено в книгата на M. Garey и D. Jonson [3], която и досега остава една от най-изчерпателните и достъпни книги по въпроса. В [19] са

дадени указания за практическа оценка на времето за работа на различни алгоритмични фрагменти.

4.2. ТЕОРИЯ НА NP ПЪЛНИТЕ ЗАДАЧИ. ОСНОВНИ ПОНЯТИЯ

4.2.1. Масова и индивидуална задача

Под ***масова задача*** Π ще разбираме въпрос, на който трябва да се даде отговор, т.е. да се получи решение.

Формално всяка масова задача Π се определя от:

1. Списък на параметрите на задачата (свободни променливи, стойностите на които не са зададени).
2. Формулировка на свойствата, които трябва да притежава решението.

Всяка ***индивидуална задача*** I се получава от масовата, след като на всички параметри се присвоят конкретни значения. По такъв начин масовата задача представя съвкупност от индивидуални задачи, получени от масовата при конкретни стойности на параметрите.

Масова е задачата за генериране на всички пермутации на числата 1,2,.....n, която зависи от един параметър. Индивидуалните задачи се получават при конкретни стойности на n, например n = 3, а решението включва наредените тройки:

{1, 2, 3}, {1, 3, 2}, {2, 1, 3}, {2, 3, 1}, {3, 1, 2}, {3, 2, 1}.

Под ***алгоритъм за решение*** на дадена задача ще разбираме дискретна и крайна редица от стъпки, следвайки която можем да получим решението.

Ще казваме, че даден алгоритъм решава масовата задача Π, ако той дава решение на всяка индивидуална задача, получена от Π при конкретни стойности на параметрите.

4.2.2. Размер на задачите

На всяка задача може да се съпостави число, което се нарича ***размер*** на задачата и характеризира обема от входни данни, необходими за нейното решаване. Размерът на задачата дава възможност за оценка на необходимата памет за съхранение на входните данни, междинните и крайни резултати, както и времето за решение. Например, ако сумираме елементите на матрица с размерност $n \times n$, ще са необходими n^2+1 клетки памет за съхранение на елементите и тяхната сума. Времето за решаване също ще зависи от размера на матрицата – по-големият размер предполага и по-голямо време за решение.

Представянето на данни е свързано с тяхното ***кодиране,*** т.е. представянето им във вид на крайни редици от символи на някаква азбука. При въвеждане на данни в компютъра се използва двоичната схема за кодиране, а азбуката съдържа два символа {0, 1}. Входната информация (числова, текстова, звукова, графична...) се представя чрез крайни редици от нули и единици, а интерпретацията им зависи от програмата, която ги обработва. Нека предположим, че за дадена масова задача Π е фиксирана схемата за кодиране, която на всяка индивидуална задача $I \in \Pi$ съпоставя редица от символи a(I) от някаква азбука A.

Размер (входна дължина) на задачата $I \in \Pi$ се определя като брой на символите в редицата, получена в резултат от прилагането на избраната схема за кодиране на масовата задача Π към индивидуалната задача I.

Независимо че съществуват различни „разумни” схеми за кодиране на данните на дадена задача, те водят до получаване на полиномиално еквивалентни входни дължини, т.е. за полиномиално време можем да преобразуваме входните данни на задачата от една схема на кодиране в друга.

Ако обработваме масив с n елемента, размерът на входните данни се определя от броя на дадените числа n. Ако на входа имаме матрица m $\times$ n, то дължината на входа се определя от броя m.n на елементите независимо от схемата на кодиране.

4.2.3. Изчислителна сложност на алгоритмите

Натрупаният опит при решение на многоразмерни дискретни задачи показва, че съществуването на алгоритъм не гарантира възможност решението да бъде получено на практика при произволни допустими стойности на входните параметри. Практическата реализация е свързана с разход на изчислителни ресурси, които могат да се окажат недостатъчни при голям обем на входните данни.

Очевидно, съществените ресурси, влияещи върху практическата реализируемост на алгоритмите и тяхната ефективност, са времето за работа на алгоритъма и необходимата памет за съхранение на данните.

При голям размер на задачата може да се окаже, че тя не може да бъде решена за “разумно” време или изисква памет за съхранение на данни, която не може да се предостави. Възниква необходимост от предварителна оценка на ***изчислителната сложност*** (*временна сложност, сложност по памет*) на алгоритмите в зависимост от размера на решаваните задачи. Обикновено ефективността на алгоритмите се свързва с

временната им сложност, тъй като времето е доминиращ фактор при оценка на възможността за практическата им реализуемост.

4.2.4. Временна сложност на алгоритмите

Временната сложност на алгоритмите е количествена оценка на времето за работа на алгоритъма в зависимост от размера на задачата. Ясно е, че времето за решение зависи от избраната схема за кодиране и от модела на изчислителното устройство. Без ограничение на общността ги приемаме за фиксирани. Най-често за модел на изчислително устройство при оценка на изчислителната сложност на алгоритмите се приема машината на Тюринг или машина с произволен достъп до паметта.

Временната сложност на алгоритмите се представя чрез функция, която на всяка входна дължина n поставя в съответствие максималното време за решение на индивидуалните задачи с тази дължина.

$$T(n) = \max_{I, |a(I)| = n} T(I),$$

където I е индивидуална задача с входен размер a(I), определен при фиксирана схема за кодиране, а T(I) е времето за работа на алгоритъма при решение на задачата I, при фиксирано изчислително устройство.

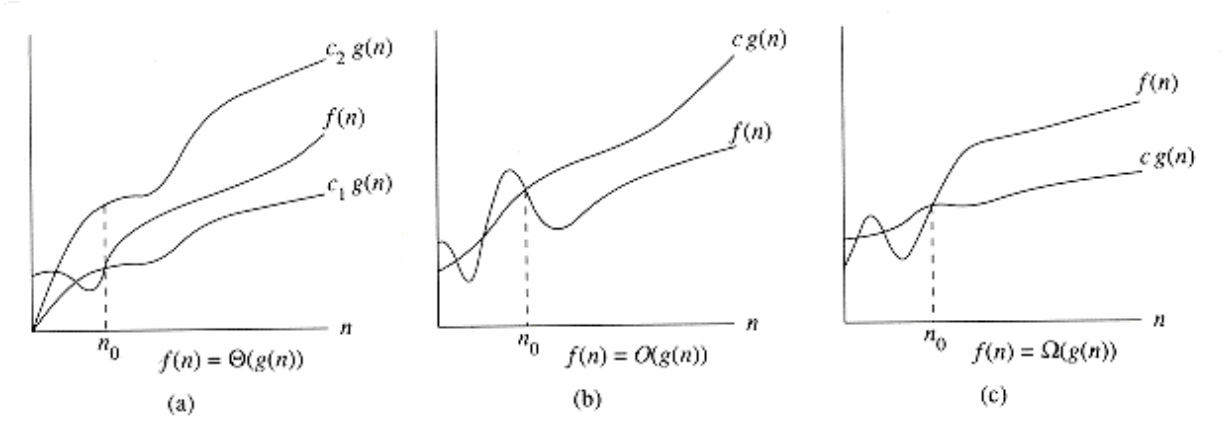
За много алгоритми определянето на точния вид на T(n) е сложна задача. Затова се изследва поведението на T(n), или т.нар. асимптотична сложност, при която се оценява скоростта на нарастване на времето за работа на алгоритъма при големи стойности на n ($n \to \infty$). Използват се следните класове асимптотична сложност:

T(n) = O(f(n)), ако ∃ c>0 n_0>0, такива, че T(n) ≤ cf(n) за ∀ n≥ n_0. В този случай казваме, че f(n) е ***горна асимптотична граница*** на T(n). Времето за решение на дадена задача T(n) расте със скорост по-малка или равна на скоростта на растеж на функцията f(n).

T(n) = Ω(f(n)), ако ∃ c>0 n_0>0, такива, че T(n) ≥ cf(n) за ∀ n≥ n_0. В този случай казваме, че f(n) е ***долна асимптотична граница*** на T(n). Временната сложност T(n) расте със скорост по-голяма или равна на скоростта на растеж на функцията f(n).

T(n) = Θ(f(n)), ако T(n) = O(f(n)) и T(n) = Ω(f(n)). В този случай функциите T(n) и f(n) растат с една и съща скорост.

Графично представяне на асимптотичните сложности е дадено в [19]:



Фигура 4.1. Графично представяне на Θ, Ω и O

4.2.5. Оценка на временната сложност на някои известни алгоритми

В [19] са дадени полезни правила за оценка на временната сложност на програмни фрагменти, съдържащи различни оператори.

Елементарен оператор

Сложността на елементарен оператор е константа, т.е. O(1).

Елементарен оператор е този, който се извършва за константно време, независещо от обема на обработваната информация. Елементарни операции в общия случай са присвояването, събирането, умножението и др. Когато обаче работим със стоцифрени числа, едва ли е редно да приемем умножението за елементарна операция. Не е редно да приемаме за елементарна операция тригонометричните функции (синус, косинус), експонентата, логаритъмът и др., които се пресмятат с редове. Обръщението към такава функция предизвиква цикъл.

Последователност от оператори

Временната сложност на последователност от оператори се определя от сложността на асимптотично по-бавния от тях. Формално, ако операторът s_1 със сложност F_1 е следван от оператора s_2 със сложност F_2, можем да запишем:

$$T(s_1) \in O(F_1), T(s_2) \in O(F_2) \Rightarrow T_{(s_1;\ s_2)} \in \max(O(F_1), O(F_2))$$

Композиция (влагане) на оператори

При влагане на оператор в областта на действие на друг оператор сложността се пресмята като произведение от сложностите им, т.е.

$$T(s_1) \in O(F_1),\ T(s_2) \in O(F_2) \Rightarrow T_{(s_1\{s_2\})} \in O(F_1.F_2))$$

if-конструкции

```
if (p)  then  s1
        else  s2;
```

Ако сложностите на p, s_1 и s_2 са O(P), $O(F_1)$, $O(F_2)$, то сложността на показания фрагмент е max(O(P), $O(F_1)$, $O(F_2)$), т.е. сложността на най-бързо нарастващата функция измежду P, F_2 и F_3.

$T(p) \in O(P)$, $T(s_1) \in O(F_1)$, $T(s_2) \in O(F_2) \Rightarrow$ T(if (p) s_1; else s_2) $\in$ max(O(P), $O(F_1)$, $O(F_2)$)

По подобен начин се оценява сложността на оператор за многовариантен избор.

Цикли

```
unsigned fact = 1;
      for (unsigned i = 1; i <= n; i++)
fact *= i;
```

Можем да считаме, че тялото на цикъла отнема константно време *c*, независещо от *n*. Сложността на оператора за цикъл for е O(*n*). Тогава по правилото за композицията за сложността на целия цикъл получаваме O(*c*.*n*), т.е. O(*n*). Тук трябва да прибавим и сложността на началната инициализация преди цикъла (която има сложност *O*(1)), при което по правилото за последователността получаваме: O(1+*n*). В крайна сметка сложността се оказва *O*(*n*).

Вложени цикли

```
unsigned sum = 0;
for (int i = 0; i < n; i++)
  for (int j = 0; j < n; j++)
    sum++;
```

Сложността при два или повече вложени цикъла с независими управляващи променливи може да се изведе лесно. В случая на два вложени цикъла от фрагмента по-горе тя е $f = n.O(g)$, където g е сложността на вътрешния цикъл, $g = O(n) \Rightarrow f = O(n.n) = O(n^2)$.

```
unsigned sum = 0;
for (int i=0; i<n-1; i++)
  for (int j=i; j<n; j++)
    sum++;
```

Въпреки че в този пример sum++ ще се изпълни $\frac{n.(n-1)}{2}$ пъти, сложността на фрагмента ще бъде $O(n^2)$, (ако приемем, че sum++ е със сложност $O(1)$). Сложността на следващия фрагмент е $O(n^2)$.

```
unsigned sum = 0;
for (int i = 0; i < n*n; i++)
   sum++;
```

4.3. ПОЛИНОМИАЛНИ И ЕКСПОНЕНЦИАЛНИ АЛГОРИТМИ. КЛАСОВЕ НА СЛОЖНОСТ

В зависимост от оценката на временната им сложност, алгоритмите се разделят на два класа: полиномиални и експоненциални.

Даден алгоритъм се нарича ***полиномиален*** (или *с полиномиална временна сложност*), ако временната му сложност T(n) = O(p(n)), където p(n) е полином от размера на задачата n. В

противен случай алгоритъмът се нарича ***експоненциален***, или *с експоненциална временна сложност.*

Временната сложност на полиномиалните алгоритми се оценява с O(n), O(n^2), O(n^3)… Този клас, означаван чрез P, е доста обширен, за много от изброителните задачи съществуват полиномиални методи за решение. От голямо значение за практическата реализация е степента на полиномиалната функция.

Временната сложност на експоненциалните алгоритми се оценява с O(2^n), O(3^n), O(n^n), O(n!) и т.н. Те принадлежат на класа NP (*NP е съкращение от non-deterministic polynomial time, т.е. полиномиално проверими задачи*). Дадена задача принадлежи към класа NP, ако е възможно с полиномиална сложност да се провери дали даден кандидат за решение действително е решение, без да се интересуваме колко време ще отнеме намирането на този кандидат. Така за NP задачите е характерно, че ако веднъж успеем да "налучкаме" правилното решение, то лесно можем да го проверим.

Описаният в достъпната литература практически опит в решението на многоразмерни реални задачи показва, че двата класа силно се различават по трудоемкост. Това различие се проявява особено ясно при теоретична оценка на времето за решение на задачи с голяма размерност. Следващата таблица показва скоростта на нарастване на времето за решение при различна временна сложност и входен размер. Пресмятанията са извършени при предположение, че разполагаме със свръхскоростна ЕИМ, извършваща 100 милиона операции в секунда.

Таблица 4.1

Оценка на времето за решение при различна временна сложност на алгоритмите и различен входен размер на задачите

Временна сложност / ***Размер***	***Полиномиална***			***Експоненциална***			
	$O(n^2)$	$O(n^5)$	$O(n^{10})$	$O(2^n)$	$O(3^n)$	$O(n!)$	$O(n^n)$
10	0,000001 сек	0,001 сек	1,667 мин	0,00001 сек	0,00059 сек	0,036 сек	1,67 мин
20	0,000004 сек	0,032 сек	1,185 дни	0,01 сек	35 сек	7,71 века	3,33E+08 века
30	0,000009 сек	0,243 сек	68 дни	11 сек	24 дни	8,41E+14 века	6,53E+26 века
50	0,000025 сек	3,125 сек	31 год	130 дни	2,28E+06 века	9,64E+46 века	2,82E+67 века
100	0,000100 сек	1,667 мин	317,098 века	4,02E+12 века	1,63E+30 века	2,96E+140 века	3,17E+182 века
500	0,002500 сек	3,617 дни	3,E+09 века	1,04E+133 века	1,15E+221 века	∞	∞

От таблицата се вижда, че увеличението на входния размер на задачата влияе по различен начин върху времето за решение при полиномиалните и експоненциални алгоритми. Поради значително по-бавното нарастване на времето при полиномиалните алгоритми, те се приемат за ефективни, практически използваеми, „добри” алгоритми. Бързото нарастване на времето при експоненциалните ги прави практически неприложими при голям размер на задачите, т.е. „лоши” алгоритми. Липсата на полиномиални алгоритми за решение на дадена задача я прави алгоритмично нерешима в приложен смисъл.

Предварителната оценка на временната сложност да даден алгоритъм е от съществено значение за практическата му реализация. Обикновено за проверка на правилната работа на разработваните програми се подбират тестови данни с малък обем, поради което дори неефективните експоненциални алгоритми могат да дадат резултат. Но ако такава програма се използва в реални условия при голям размер на входа, ще проличи нейната практическа неизползваемост, тъй като не може да даде резултат за приемливо време.

При изследване на изчислителната сложност на алгоритмите трябва да се има предвид, че. временната сложност е *теоретична оценка* на времето за работа на алгоритмите в най-лошия случай. Практиката показва, че в реални ситуации много алгоритми с експоненциална сложност работят много по-бързо от теоретично оценените време и могат да се решат задачи с по-голяма размерност от теоретично прогнозираната.

Съществената разлика между полиномиалните и експоненциалните алгоритми се състои в следното. Анализът на дискретните задачи показва, че те могат да бъдат решени чрез изчерпващо търсене в множеството от потенциални решения, което включва две стъпки:

- Пораждане на решения без пропуски и повторения.

- Проверка за допустимост.

Типични примери са задачата за търговския пътник, задачата за раницата и много други, наричани изброителни, при които решението се търси между краен в теоретично отношение брой варианти. На практика обаче броят на вариантите, а съответно и времето за решение растат експоненциално при увеличение на размера. При големи размерности такива задачи не могат да бъдат

решени за обозримо време, т.е. те са практически неразрешими. В това се състои и основният проблем на дискретната математика: съществува ли принципна възможност за намиране на решение без изчерпване на експоненциален брой варианти, т.е. елиминиране на изчерпващото търсене. На този въпрос досега не е намерен задоволителен отговор.

4.3.1. Сводимост на задачите в класа P

Казваме, че задачата Π_1 е ***сводима*** към задачата Π_2, ако е възможно да се намери алгоритъм за решението на Π_1, използващ полиномиален брой обръщения към функция, решаваща Π_2, и всички изчисления извън тези обръщения са с полиномиална сложност.

Сводимостта формално означава следното: ако $\Pi_1, \Pi_2 \in$ NP, то Π_1 е сводима към Π_2, ако методът за решение на Π_2 може да бъде преобразуван в метод за решение на Π_1. Сводимостта е полиномиална, ако такова преобразуване може да се извърши за полиномиално време.

Тогава, ако една задача Π_2 е от класа P и Π_1 е сводима към Π_2, то следва, че Π_1 също е от класа P.

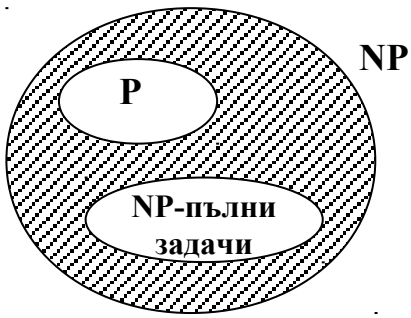
4.3.2. Класове на сложност NP и P

Класовете NP и P играят основна роля в количествената теория на алгоритмите и дават възможност за оценка на тяхната ефективност и практическа реализируемост.

Класът NP включва всички изброителни задачи, които могат да бъдат решени чрез метода на изчерпващото търсене, а ***класът P*** съдържа изброителните задачи с полиномиална временна сложност.

Част от задачите в NP служат като еталон на сложност и се наричат универсални или NP пълни. Към тях могат полиномиално да се сведат всички задачи от класа NP. Една NP задача A се нарича NP пълна, ако всяка друга задача B от класа NP е сводима към A.

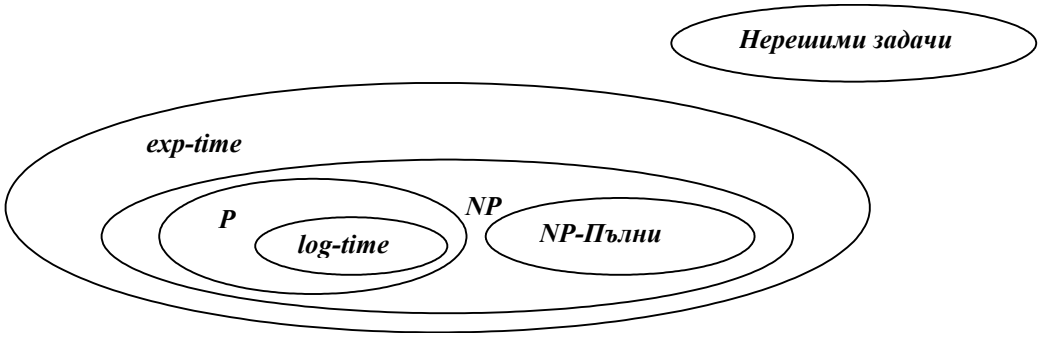
Въпросът за съотношението между класовете NP и P има голямо теоретично и практическо значение. Графичното им представяне е дадено на фиг.(4.2).



Фигура 4.2. Класът NP

Очевидно $P \subseteq NP$. Съвпадението на двата класа P = NP означава елиминация на изчерпващото търсене и насочване на усилията към разработка на ефективни полиномиални алгоритми (които в момента липсват, но може би откриването им е въпрос на време). Ако $P \subset NP$, трябва да приемем наличието на „трудноресими" задачи, за които не съществуват ефективни алгоритми. Ако дадена задача $П \in NP\backslash P$, то тя е „трудноресима". Американският математически институт Clay предлага 1 000 000 $ награда за отговор на въпроса P = NP?

По-прецизна картина за предполагаемото съотношение на класовете на сложност е дадена на следната фигура [19]:



Фигура 4.3. Класове на сложност

P задачи: Това са задачи, за които съществува решение с полиномиална сложност. Такива са много известни алгоритми за търсене, за сортиране, за обработка на масиви, за изчисляване на различни математически функции и др.

NP задачи: Полиномиално проверими задачи. Дадена задача принадлежи към класа NP, ако е възможно с полиномиална сложност да се провери дали даден кандидат за решение действително е решение на задачата. Следният прост пример илюстрира добре смисъла на класа NP задачи: Дадено е естествено число n. Да се провери дали n може да се разложи във вида a.b. Ако това е възможно, да се намерят a и b. Ако предполагаме, че a* и b* са решение на задачата, то лесно можем да проверим дали n= a*. b*, В противен случай трябва да използваме метода на изчерпващото търсене.

NP пълни задачи: Универсални, или NP пълни задачи, към които полиномиално се свеждат всички задачи от класа NP. Някои задачи от този клас ще бъдат разгледани по-късно.

Exp-time задачи: Това са задачи, които могат да се решат с експоненциална сложност. Този клас съдържа предишните три, но не е всеобхватен — съществуват задачи, за които най-добрите алгоритми са с по-висока от експоненциалната сложност.

Space задачи. Това са задачи, при които критична е паметта, която използват (като функция от размера на входните данни). При тях не се интересуваме от сложността на алгоритъма за решаването им. Този клас може да се разпадне до няколко подкласа: P-space (задачи за които е необходима полиномиална памет), Exp-space (експоненциална памет) и др.

Пример за такъв тип задачи е играта на шах [19]. При разработването на програми за игра на шах, редицата от ходове се

представя чрез дърво. Пред играча, започващ играта, има 20 възможни хода, за всеки от тях съществуват множество различни ответни ходове и т.н. Програмите обхождат дървото на състоянията и оценяват последствията от евентуалното извършване на всеки възможен ход.

Задачата за перфектно играещата шах-програма е от клас *space* — запазването на всички възможни конфигурации и най-добрият ход за всяка от тях, изисква константна памет, въпреки че тази константа е огромна.

От друга страна, височината на шах дървото не е голяма — рядко се играе по-дълга от 150 хода партия. Така, ако правим претърсване на това дърво при всеки ход, отпада проблемът, свързан с недостига на памет, но се появява друг - неприемливо (експоненциално) време за изчисление. Поради тази причина шах програмите реално извършват претърсване в по-малка дълбочина и въз основа на евристики избират възможно най-добрия ход.

В началото шахматните мачове човек-компютър привличат вниманието с оспорвания си характер. През 1997 *Deep Blue*, компютър на IBM, в официален мач побеждава за пръв път световния шампион по шах Гари Каспаров. Разработката на мощни компютри и интелигентни програми предопределя изхода на играта в полза на компютрите.

Нерешими задачи: Съществуват алгоритмични задачи, за които може да се докаже, че не могат да бъдат решени, независимо от това, с колко време и памет разполагаме.

Такива са повечето задачи, отнасящи се до поведението на дадена програма. Нерешима е задачата за намиране на алгоритъм, който може винаги и без грешка да определи дали дадена програма се намира в безкраен цикъл, или — не.

В [19] като пример е дадена следната програма, която проверява дали съществува решение на уравнението $a^n+b^n=c^n$, за a, b, c, n цели положителни числа и n>2.

```
void main() {
  int x=3;
    for (;;) {
      for(int a=1; a<=x; a++)
        for(int b=1; b<=x; b++)
          for(int c=1; c<=x; c++)
            for(int i=3; i<=x; i++)
              if (pow(a,i)+pow(b,i)==pow(c,i)) exit;
            x++;
    }
}
```

Голямата теорема на Ферма, формулирана през 1637 г. и доказана през 1994 г. от английския математик Ендрю Уайлс гласи, че такава четворка естествени числа не съществува. Ако този факт не е известен, за разработчика ще бъде трудно да прецени дали горната програма ще намери решение и ще завърши, или — не. Съществуват и други подобни задачи, за които съответен теоретичен резултат не е известен и не се знае дали съответстващите програми ще зациклят, или — не.

4.4. ХАРАКТЕРИСТИКА НА КЛАСА NP. ТЕОРЕМА НА КУК

4.4.1. Теорема на Кук

Основната заслуга на Кук (S. A Cook) е, че поставя началото на развитие на теорията на NP пълните задачи през 1971 г. в труда „Сложност на процедурите за доказателство на теореми” [3].

Той доказва, че класът на NP пълните задачи не е празен, т.е. съществува поне една NP пълна задача, към която са полиномиално сводими всички задачи от NP. Тази задача от булевата логика се нарича задача за удовлетворимост и се формулира (не строго) по следния начин:

Задача за удовлетворимост:

Дадено е множество от булеви променливи U = {u_1, u_2, … u_n}. Нека D_1, D_2, …D_m са дизюнкции над U от вида: $u_{к1} \vee u_{к2} \vee .... \vee u_{кр}$ и C е конюнкция на D_1, D_2, …D_m, C = $D_1 \wedge D_2 \wedge \ldots \wedge D_m$. Пита се: съществува ли такъв набор от стойности на променливите, за който C получава стойност „истина”.

Теорема 4.1 (*Теорема на Кук*). Задачата за удовлетворимост е NP пълна задача.

Задачата за удовлетворимост на булева функция е в известен смисъл "първата" NP пълна задача. Тя често се използва за доказателство на теоремата на Кук, като чрез свеждане към нея се доказва NP пълнотата на много други задачи.

Важни следствия от теоремата на Кук са:

- Ако съществува полиномиален алгоритъм за решение на задачата за удовлетворимост, то всички задачи от NP са полиномиално разрешими.

- Обратно, ако някоя задача от NP е „трудноpешаема”, то такава е и задачата за удовлетворимост.
- Всички въпроси, свързани с анализ на изчислителната сложност на алгоритмите, се свеждат до един: “Трудноpешаеми ли са задачите от класа NP?”

Работата на Кук дава тласък на изследванията в тази област. Доказва се NP пълнотата на много известни задачи. В [4] са описани над 300 такива задачи, разпределени в 12 категории (теория на графите, мрежи, множества, съхранение и търсене на данни, математическо програмиране, алгебра, теория на числата, логика и др.). Ако само за една от тях бъде доказано, че може, или че не може да бъде решена с полиномиален алгоритъм, то това ще следва и за всички останали.

Доказателството на NP пълнотата на дадена задача Π се извършва по следната схема:

1. Доказва се, че $\Pi \in$ NP.
2. Избор на известна NP пълна задача Π’.
3. Свеждане на Π’ към Π.
4. Доказателство, че сводимостта е полиномиална.

4.4.2. Известни NP пълни задачи

Най-често при доказателства на NP пълнота за сводимост се използват седем основни NP пълни задачи, известни в литературата със следните имена: изпълнимост (удовлетворимост), 3-изпълнимост, тримерно съчетание, върхово покритие, разбиване, гамилтонов цикъл, клика. Някои от тези задачи ще разгледаме по-късно, тъй като за описанието им е необходима специална терминология от теория на графите.

Лесно се формулират някои известни NP пълни задачи, за описанието на които не е необходима специална терминология. Такива са следните задачи:

Задача за търговския пътник

Дадени са множество C от m града, $C = \{c_1, c_2, \ldots, c_m\}$, разстоянията между всеки два града $d(c_i, c_j) > 0$ и цяло положително число B. Съществува ли маршрут с дължина $\leq B$, преминаващ през всички градове?

Задача за раницата

Дадено е крайно множество от предмети U. За всеки предмет $u \in U$ е зададен размер $s(u) \in Z^+$ и цена $v(u) \in Z^+$. Освен това са зададени целите положителни числа B и K. Съществува ли подмножество $U' \subseteq U$, такова, че:

$$\sum_{u \in U'} s(u) \leq B \quad \text{и} \quad \sum_{u \in U'} v(u) \leq K?$$

Квадратични сравнения

Дадени са цели положителни числа a, b и c. Съществува ли цяло положително число $x < c$, такова, че $x^2 \equiv a \pmod b$?

Квадратни диофантови уравнения

Дадени са цели положителни числа a, b и c. Съществуват ли цели положителни числа x, y < c, такива, че $ax^2 + by = c$?

Обобщена игра Instant Insanity

Дадени са крайни множества от цветове C и кубчета Q, за които $|C| = |Q|$. Стените на всички кубчета от Q са оцветени с цветове от C. Възможно ли е всички кубчета да се наредят в стълб едно върху друго така, че на всяка стена на получения стълб всеки цвят от C да се среща точно един път?

Съставно число (открита!)

Дадено е цяло положително число N. Съществуват ли цели числа m > 1 и n > 1, такива, че N = m.n?

Заключение:

Теорията на NP пълните задачи ни предоставя математическия апарат, необходим за изследване на изчислителната сложност на компютърните алгоритми. Въпросите за оценка на времето за работа и ефективността на алгоритмите имат съществено значение за практическата им реализация.

ЗАДАЧИ ЗА САМОПОДГОТОВКА

1. Да се анализират класовете на сложност P и NP, тяхното взаимоотношение, следствията при P = NP и P $\subset$ NP.
2. Полиномиален или експоненциален е алгоритъмът с временна сложност $O(n^{\log n})$?.
3. Да се оцени временната сложност на известни алгоритми за сортировка на масиви.
4. Да се оцени временната сложност на задачата за раницата, задачата за разбиване на кодове, задачата за търговския пътник.
5. Да се оцени времето за работа на алгоритми с временна сложност $O(n)$, $O(n^2)$, $O(n^3)$, $O(2^n)$, $O(3^n)$, $O(n^n)$, $O(n!)$.

   Упътване: използвайте таблица на Excel и получете стойностите на функциите при стойности на аргумента n = 10, 20, 30, 50, 100, 500.

5. При доказателства на NP пълнота за сводимост се използват седем основни NP пълни задачи: изпълнимост, 3-изпълнимост, тримерно съчетание, върхово покритие, разбиване, гамилтонов цикъл, клика. Защо тези задачи са основни?

глава

5

ДИСКРЕТНИ СТРУКТУРИ. ГРАФИ, ДЪРВЕТА, МРЕЖИ

- **Дискретни структури. Графи**
- **Представяне на графи**
- **Стратегии за обхождане на граф**
- **Търсене на път в граф**
- **Двуделни графи. Задача за назначенията**

5.1. ДИСКРЕТНИ СТРУКТУРИ. ГРАФИ

5.1.1. Теория на графите. Същност и предназначение

Теория на графите е раздел от дискретната математика, чийто обект за изучаване са графите. От всички абстрактни математически структури, графите са предпочитан формален модел за описание на реални процеси, обекти и системи.

Повишеният интерес към тях е свързан с развитието на компютърната техника, която предоставя техническа база за решение на сложни задачи, които удобно и нагледно се представят чрез графови модели, а за решението им се използват алгоритми върху графи. От академична дисциплина теория на графите се превръща в инструмент за формализация на широк кръг задачи от най-различни научни области. Освен в традиционните математически направления като топология, теория на групите и теория на вероятностите, графите намират голямо приложение в теоретичната кибернетика, в теория на автоматите, изследване на операциите, математическо моделиране, теория на кодирането, теория на игрите, както и във физиката, биологията, историята, електротехниката, химията, икономиката, социологията, лингвистиката и др.

5.1.2. Неориентирани графи. Ориентирани графи. Основни понятия

Графът (*Graph, undirected graph*) ***G = (V, E)*** е математически обект, който се състои от множество от върхове V и множество от ребра E. Върховете и ребрата се наричат ***елементи на графа***. Абстрактната структура, образувана от съвкупността от еднородни

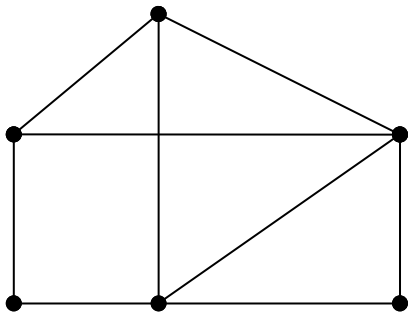
(обикновено) обекти – върхове, заедно със свързващите ги еднородни обекти - ребра, поражда топологическата структура, наречена граф. Множеството от върхове V и множеството от ребра E на графа G се означават чрез V(G) и E(G) съответно. Ако $|V(G)| = n$ и $|E(G)| = m$, то графът G се нарича (n, m) граф.

Връх *(Vertex)* - фундаментално понятие от теорията на графите, елемент от множеството V, съответстващ на обект от някаква фиксирана природа. Други названия - възел, точка.

Ребро *(Edge)* - фундаментално понятие от теорията на графите, елемент от множеството E. Всяко ребро $e \in E$, което свързва два върха $v_1, v_2 \in V$, може да се зададе като ненаредена двойка от върхове $e = (v_1, v_2.) = (v_2, v_1)$. Множеството E от ребра на графа е подмножество на декартовата степен на множеството от върхове :

$$E \subseteq V \times V = V^2$$

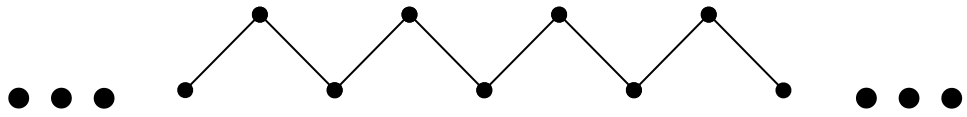
При графично представяне върховете се изобразяват чрез точки, а ребрата – чрез отсечки, които ги съединяват.



Фиг. 5.1. Графично представяне на граф с 6 върха и 9 ребра.

Ако $V = \{v_1, v_2 \ldots v_n\}$ и $E = \{e_1, e_2 \ldots e_m\}$ са крайни множества, то G се нарича ***краен*** граф, в противен случай G се нарича ***безкраен***

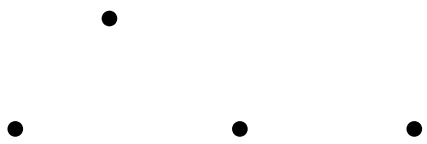
(Infinite graph). Безкрайният граф съдържа безкрайно множество върхове или безкрайно множество ребра.



Фиг. 5.2. Безкраен граф

Графът G е от ***ред n*** (Order of a graph), ако броят на върховете му е равен на n.

Граф, съдържащ един връх, се нарича ***тривиален*** граф (*Trivial graph*). Граф, не съдържащ ребра се нарича ***празен*** (*Empty graph*).

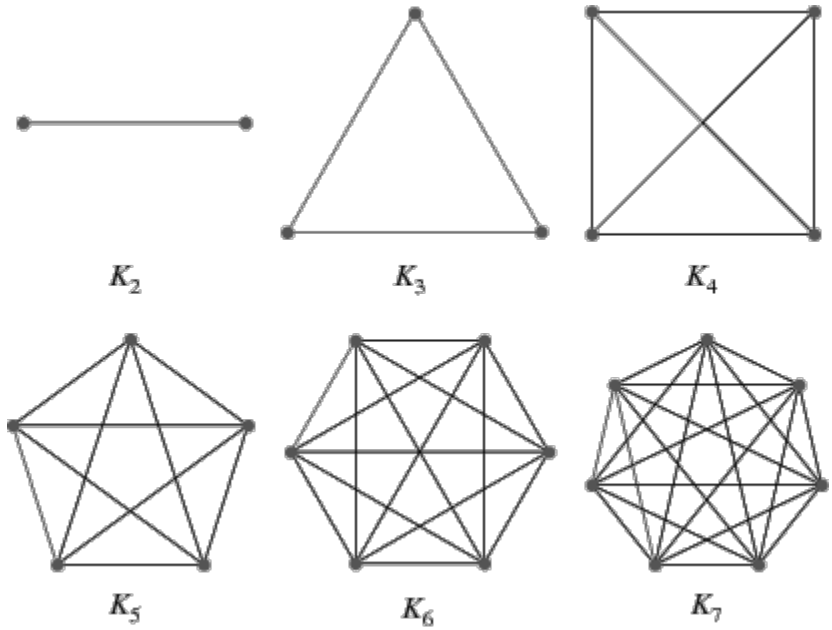


Фиг. 5.3. Празен граф

Граф, за който всяка двойка върхове е съединена с ребро, се нарича ***пълен*** граф *(Complete graph).* Пълен граф с n върха се означава чрез K_n. Броят на ребрата на пълен граф с n върха е n(n-1)/2 (фиг. 5.4).

Всяко ребро свързва два върха, които се наричат ***крайни*** *(Terminal vertex of an edge)*.

Тъй като графите се състоят от две множества (на върховете и на ребрата), то операциите обединение и сечение на множества пораждат съответни операции върху графи.



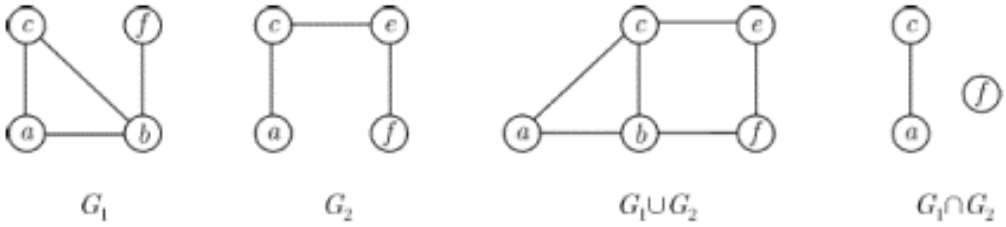
Фиг. 5. 4. Пълен граф с 2, 3, 4, 5, 7 и 8 върха

Обединение на графите $G_1(V_1, E_1)$ и $G_2(V_2, E_2)$ е графът:

$G(V,E) = G_1 \cup G_2$, за който $V = V_1 \cup V_2$ и $E = E_1 \cup E_2$.

Сечение на графите $G_1(V_1, E_1)$ и $G_2(V_2, E_2)$ е графът:

$G = G_1 \cap G_2$, за който $V = V_1 \cap V_2$ и $E = E_1 \cap E_2$).

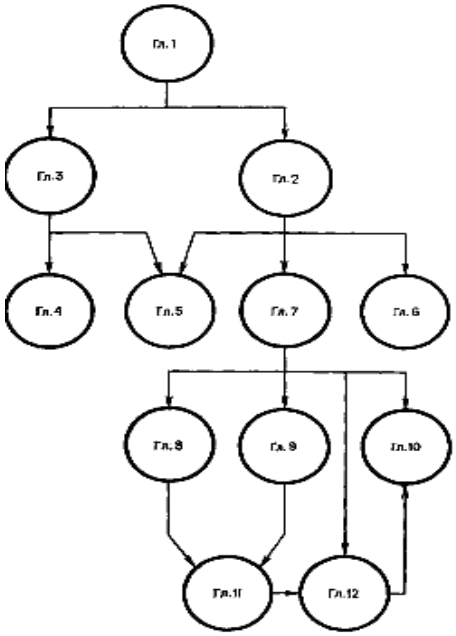


Фиг. 5.5. Обединение и сечение на графи

Ориентиран граф *(Directed graph, орграф)* G се задава чрез множествата (V, A), където V е множество от върхове, а $A \subseteq V \times V$ - множество от дъги (ориентирани ребра).

Дъгата (Arc) е фундаментално понятие в теорията на графите и представлява ***ориентирано ребро*** на графа, т.е. ребро, което има посока. Определя се като наредена двойка (v_1, v_2.)

върхове. Върхът v_1 е ***начален*** (*Start vertex*), а v_2 е ***краен*** (*Finish vertex*) за дъгата (v_1, v_2). Графически дъгите се изобразяват чрез стрелки, определящи посоката на преход от един връх в друг. Дъгата (v_1, v_2) е насочена от върха v_1 - начало на дъгата, към върха v_2 – край на дъгата. Тя е ***входяща*** за върха v_2 и ***изходяща*** за върха v_1.



Граф на връзките в книгата
Н. Кристофидес. Теория графов
(алгоритмический подход). Москва,
"Мир", 1978

Фиг. 5.6. Ориентиран граф.

Орграф, за който всяка двойка върхове е съединена с дъга, се нарича ***пълен*** орграф *(Complete directed graph).*

Графите се използват за представяне на схеми на пътищата, газопроводи, електрически мрежи, самолетни линии, станции на метрото, химически и генетични структури, комуникационни мрежи, административни и социални структури и др.

Инцидентност

Ако реброто е съединява върховете v_1 и v_2 казваме, че реброто е и върховете v1, v2 са ***инцидентни. Инцидентността*** *(Incidenty)* е отношение между реброто (дъгата) и неговите крайни върхове, т.е. реброто е = (v_1, v_2) е инцидентно на върховете v_1 и v_2 и

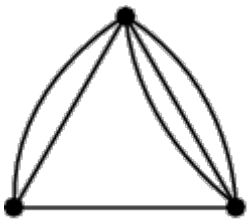
върховете v_1, v_2 са инцидентни на реброто e = (v_1,v_2). Връх, който няма инцидентни ребра (или дъги), се нарича ***изолиран*** *(Isolated vertex)*.

Върховете v_1 и v_2, съединени с реброто e се наричат ***съседни*** *(Adjacent vertices, joined vertices).* При това, v_1 е съседен на v_2, тогава и само тогава, когато (v_1, v_2) $\in$ E. Съседни се наричат и ребрата *(Adjacent edges)* или дъгите *(Adjacent arcs)*, имащи общ връх.

Равнинен граф *(Planar graph)* - граф, който може да бъде изобразен върху равнината така, че никои две ребра да нямат общи точки, освен инцидентния им връх.

Две ребра, свързващи една и съща двойка върхове се наричат ***паралелни*** или ***кратни*** *(Multiple edges).* В ориентиран граф дъгите, които имат общ начален и краен връх, се наричат ***кратни дъги (****Multiple arcs).*

Мултиграф *(Multigraph)* – неориентиран граф с кратни ребра. Задава се чрез G(V, E), където V е непразно множество от върхове на графа, а E - семейство от подмножества на множеството V × V.



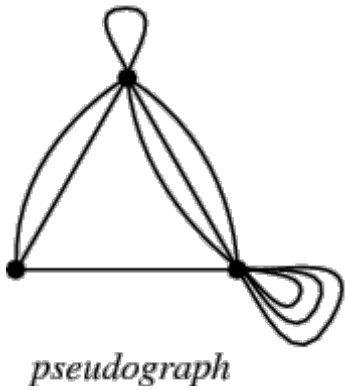
multigraph

Броят на ребрата, съединяващи два дадени върха в мултиграф се нарича ***кратност*** на реброто *(Multiplicity of edge).*

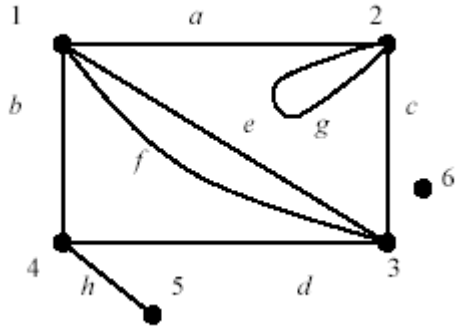
Орграф, в който се допускат кратни дъги, се нарича ***ориентиран мултиграф*** *(Directed multigraph)*.

Ребра или дъги от вида (v, v), v $\in$ V(G), свързващи връх сам със себе си, се наричат ***примки*** *(Loop)*. Обикновено примките се разглеждат като елементи на псевдографите, но се допускат и в ориентираните графи, разглеждани като модел на система.

Псевдограф *(Pseudograph)* – неориентиран граф, в който се допускат и кратни ребра, и кратни примки.



Пример 5.1. Графът на следващата фигура:



Фиг. 5.7. Псевдограф с кратни ребра и примка.

съдържа 6 върха, означени с цифрите 1, 2, 3, 4, 5, 6 и осем ребра, означени с буквите a, b, c, d, e, f, g, h, т.е.:

V = {1, 2, 3, 4, 5, 6}

E = { a, b, c, d, e, f, g, h }.

Реброто a свързва върховете 1 и 2; ребрата e и f са кратни, тъй като свързват върховете 1 и 4; реброто g е примка и свързва върха 2 със себе си, върхът 1 е инцидентен на ребрата a, b, e, f; реброто c е инцидентно на върховете 2 и 3; върхът 6 е изолиран, върхът 5 е висящ и т.н…

Степен на връх или валентност (*Degree of a vertex, valency of a vertex)* в граф се нарича броя на инцидентните на върха ребра. Използва се означението deg(v). В орграф – броя на инцидентните дъги. Изолираните върхове имат степен 0 (ако нямат примки). Връх с примка има степен две.

Лема за ръкостисканията *(Handshake's lemma):*

Ако G = (V, E) е неориентиран граф и V = {v_1, v_2… v_n}, сумата от степените на всички върхове на графа е четно число, равно на удвоения брой на ребрата:

$$\sum_{i=1}^{n} d(v_i) = 2|E| ,$$

където с d(v_i) сме означили степента на върха v_i.

Наименованието на лемата е свързано с аналогията при ръкостисканията, чийто брой е двойно по-голям от броя на участващите хора. В сила е за мулти и псевдографи. Върху нея се базира и доказателството на следната теорема:

Теорема 5.1. Във всеки неориентиран граф броят на върховете с нечетна степен е четен.

Доказателството с следното: Нека S е сумата от степените на върховете с четна степен, а T - сумата от степените на върховете с нечетна степен. Очевидно S е четно, тъй като сумата на произволен брой четни числа е четна. Съгласно лемата за ръкостисканията, сумата:

S + T = 2| E |

E също четна, което е изпълнено само при условие, че T е четно число. T е сума от нечетни числа и може да бъде четно само ако броят им е четен.

Теорема 5.2. Във всеки граф с n ≥ 2 върха, съществуват поне два с еднаква степен.

Доказателството се базира на принципа на Дирихле за чекмеджетата. В грах с n ≥ 2 върха минималната степен на връх е 0, а максималната n - 1. Допускаме, че всички върхове {1, 2, 3, … n} имат различни степени - 0, 1, 2, ... n-1. Но ако има връх със степен 0 (не свързан с никой от останалите върхове), не може да има връх от степен n-1 (свързан чрез ребра с всички останали). Съгласно принципа на Дирихле, поне два върха имат еднаква степен.

Полустепен на входа на връх (*Indegree*) в орграф се нарича броя на дъгите, влизащи в даден връх.

Полустепен на изхода на връх (*Outdegree*) в орграф се нарича броя на дъгите, излизащи от даден връх.

Теорема 5.3. В ориентиран граф с m дъги сумата от полустепените на входа и полустепените на изхода на всички върхове е равна на броя на дъгите m.

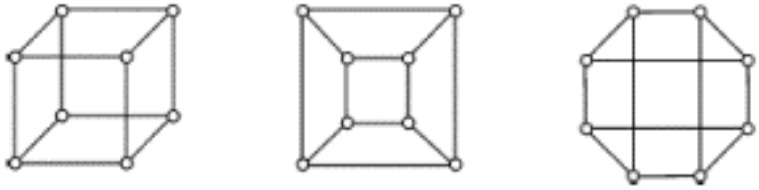
В неориентиран граф ***висящ връх****,* наричан още ***лист,*** *(Terminal (pendant) vertex)* се нарича върхът със степен 1. В орграф висящ е върхът с полустепен на входа, равна на 1 и полустепен на изхода, равна на 0.

Висящо ребро *(Terminal pendant edge)* - ребро, един от крайните върхове на което има степен 1.

Степен на ребро *(Degree of an edge)* – степен на реброто (v_1, v_2) се задава чрез двойката (s_1, s_2), където s_1 - степен на върха v_1, а s_2 - степен на върха v_2.

Степен на графа или ***максимална степен*** *(Degree of a graph)* – най-голямата степен на върховете на графа.

Регулярен граф (*Regular graph*) - граф, за който степените на всички върхове са равни. Регулярен граф от степен 0 е празен граф. Регулярен граф от степен n представлява n-мерен куб.



Фиг. 5.8. Регулярен граф от трета степен

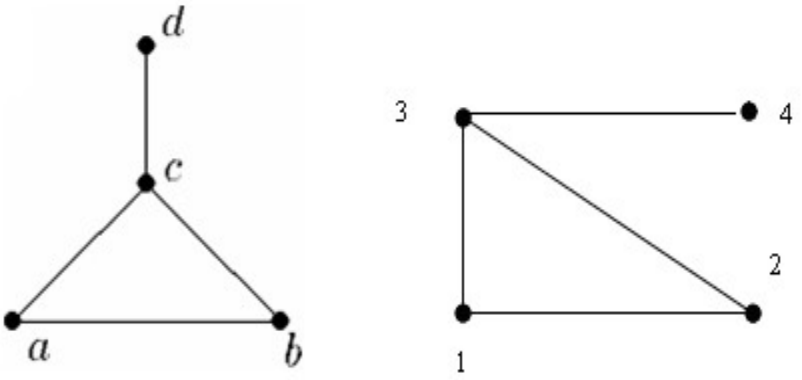
Претеглен граф *(Weighted graph)* - граф (орграф), в който на всяко ребро (дъга) е съпоставено тегло. Понякога се задава във вида (G,w), където *w* - теглова функция, дефинирана върху множеството от ребра на графа, w: E $\rightarrow$ R. Теглата се използват за задаване на дължината на пътя между два върха, времето за преход от един връх в друг, цена на билета, разход на гориво и други числови характеристики на реалните задачи. При графично изобразяване на графи, теглата се изписват над съответните им ребра (дъги).

В много случаи, при изследване на структурата на графите, имената или номерата на върховете нямат значение. Такива графи, които се получават чрез преименуване на върховете, е удобно да се смятат еднакви. За такива графи се отнася и понятието изоморфизъм.

Изоморфизъм между графи *(Graph isomorphism)* се нарича взаимно еднозначното съответствие (биекция) V(G) $\rightarrow$ V(H) между множеството от върхове на графа G и множеството от върхове на графа H, което запазва отношението на съседство между върховете (образите на съседни върхове са съседни и обратно). Отношението на изоморфизъм между графи е отношение на еквивалентност и

притежава свойствата симетричност, транзитивност и рефлексивност.

Два графа се наричат ***изоморфни*** (G ≅ H), *(Isomorphic graphs)*,- ако между тях може да се установи изоморфизъм. Задачата за проверка на изоморфността на два графа е NP-пълна задача. Следните два графа са изоморфни:



Фиг. 5.9. Изоморфни графи

За графите от фиг. 7 един възможен изоморфизъм е изображението f, зададено таблично по следния начин:

x	a	b	c	d
f(x)	1	2	3	4

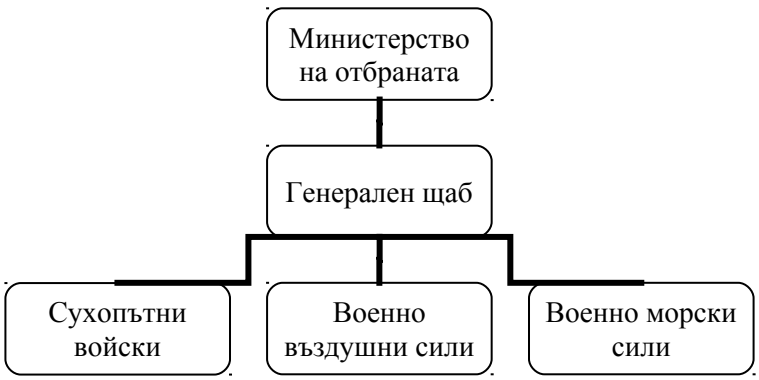
Изоморфните графи са еквивалентни, различават се само по имената (етикетите) на върховете. Много практически задачи като информационно търсене, определяне структурата на химически съединения и др. се свеждат до установяване на изоморфизъм между графи.

5.1.3. Дървета. Основни понятия

Дърветата (*Tree*) са специален вид графи с йерархична структура на върховете, които се използват за представяне на

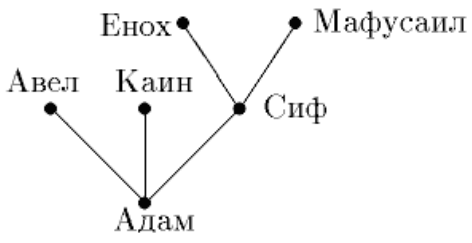
реални обекти, в които структурните свойства се изразяват във функционалната връзка 1:n, т.е. всеки връх е свързан с n върха, които от своя страна могат да бъдат свързани с n върха и т.н. Ребрата (дъгите) на графа служат за представяне на връзките между върховете. Ако ребрата са ориентирани (дъги), дървото се нарича ***ориентирано дърво***.

Реални йерархични обекти с дървовидна структура са административните системи, родословното дърво, в което наследниците на всеки връх са неговите родители и др. Като предпочитана абстрактна информационна структура, дърветата се използват в операционните системи, компилаторите, системите за управление на бази данни, алгоритмите и много други компютърни приложения.



Фиг. 5.10. Йерархичната структура на Българската армия, представена чрез дърво

Терминологията, използвана за описание на дърветата е заимствана от ботаниката (дърво, лист, корен), или от генеалогията (връх - наследник, връх - предшественик, родител ...).



Дървото с крайно (***крайно дърво***) или безкрайно (***безкрайно дърво***) множество от върхове, се дефинира по следния начин:

1. Съществува точно един връх, който се нарича корен на дървото. Той няма предшественици, но може да има върхове - наследници. Връзката на корена с неговите наследници се задава чрез ребро (дъга).

2. Всички останали върхове имат точно един връх - предшественик и евентуално върхове - наследници.

3. Върховете, които нямат наследници, се наричат листа.

Дървото се нарича празно и се бележи с T0, ако множеството от върхове е празно. Ако съдържа единствен връх, дървото се нарича изродено и се бележи чрез T1.

Рекурсивната природа на тези йерархични структури се вижда от следното рекурсивно определение за дърво, дадено от американския математик Доналд Кнут (*Donald Ervin Knuth*, в преведената на много езици „The Art of Computer Programming"), наричан «баща» на алгоритмите, заради приносите си към изследването и систематизирането на алгоритмите и тяхната изчислителна сложност.

Дървото T е множество от върхове, което е или празното множество, или:

- Съдържа единствен елемент t, наречен корен на дървото, който обикновено се означава с *root* (T).

- Останалите елементи (с изключение на корена) са разпределени в $m \geq 0$ непресичащи се множества T_1, T_2, ..., T_m, всяко от които е дърво. Множествата T_1, T_2, ..., T_m се наричат ***поддървета*** на T. Корените t_1 на T_1, t_2 на T_2, ..., t_m на T_m се наричат наследници на t.

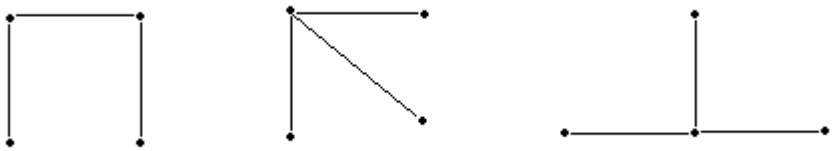
Всеки връх на T е корен на някакво поддърво. Броят на поддърветата с корен върха t се нарича ***степен на върха*** t. Ако степента на върха t е 0, той се нарича лист.

Височина на връх *(Height of vertex*) се задава чрез максималния брой ребра (дъги), свързващи дадения връх с листата.

Височина на дърво *(Height of tree)* - височина на корена на дървото (ориентираното дърво).

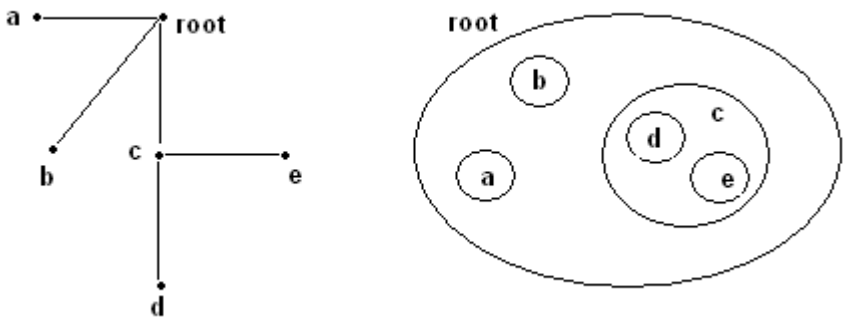
Дърветата могат да бъдат представени по следните начини:

- като *ацикличен свързан граф* (термините свързан и цикличен граф ще бъдат въведени в (5.4.1)) с избран връх за корен;



Фиг. 5.11. Представяне на дърво чрез ацикличен свързан граф

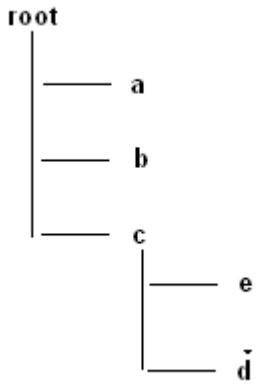
- като *вложени множества*, при което всеки връх, представен чрез множество, обхваща наследниците си;



Фиг. 5.12. Представяне на дърво чрез вложени множества

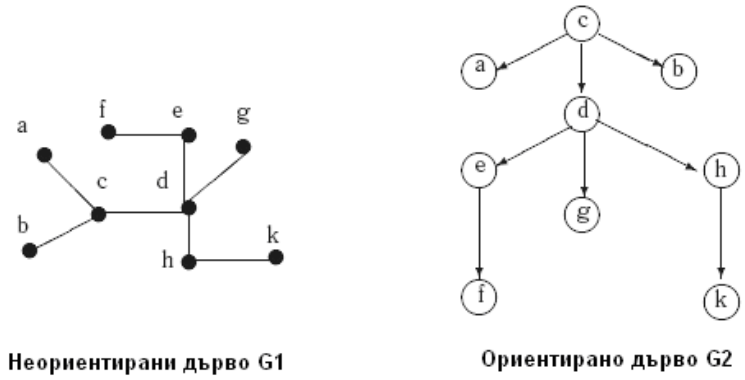
- чрез *вложени скоби* (root ((a) (b) (c((d) (e)))), което намира приложение в компилаторите за представяне на математически изрази;

- чрез *стъпаловидно отместване* в дясно за графично представяне на дървовидната структура на организация на папките (директориите) в операционните системи или съдържанието на текстови документи:



Фиг. 5.13. Представяне на дърво чрез стъпаловидно отместване

На следващата фигура са показани примери за неориентирано и ориентирано дърво. Дървото G_2 е получено от G_1 чрез избор на корен - върха с и ориентация на всички ребра в направление от корена към листата.



Фиг. 5.14. Неориентирано и ориентирано дърво

Броят на поддърветата на всеки връх може да бъде различен, в съответствие с което различаваме ***двоични***, ***троични***, ***n-арни*** (*n -ary tree*) дървета. Ако степента на всеки връх от дървото е по-малка или равна на две, то дървото се нарича ***двоично или бинарно дърво***. Тъй като поддърветата на всеки връх t са най-много две, те се наричат ***ляво поддърво*** и ***дясно поддърво*** на t, а корените на лявото и дясното поддърво се наричат ***ляв*** и ***десен наследник*** на t. Регулярни са дърветата в които степените на върховете съвпадат.

Фиг. 5.16. Регулярни дървета

Дърво на решенията (*Decision tree*) - дърво (бинарно дърво), в което всеки връх, който не е терминален, представлява решение. В зависимост от избора, приет във всеки връх, управлението се предава на някой от неговите наследници (например, на най-левия наследник, или на левия (десен) наследник

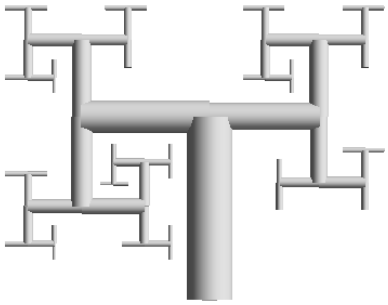
в бинарните дървета). В резултат на редицата последователни избори, започвайки от корена, получаваме точно определен лист от дървото. Дърво на решенията е предпочитана структура за представяне на пространството на състоянията при организация на изчерпващо търсене.

Дърво на Фибоначи (*Fibonacci tree*), имащо пряка връзка със задачата на Фибоначи е двоично дърво, което се дефинира по следния начин:

- *празното дърво* е дърво на Фибоначи F_0;

- *изроденото дърво* е дърво на Фибоначи F_1;

- F_i е дърво на Фибоначи с ляво поддърво F_{i-2} и дясно поддърво F_{i-1}.

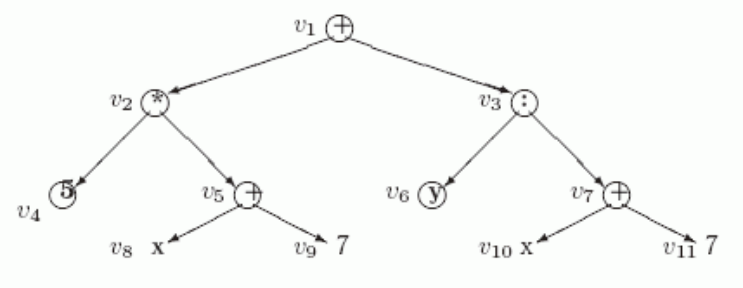


Фиг. 5.17. Фрактал „Дърво на Фибоначи"(Fibonacci Tree)

АВЛ - дърво (*AVL-Tree, балансирано по височина дърво*) е бинарно дърво, в което разликата от височините на лявото и дясно поддърво е не по-голяма от единица. Наречени са така в чест на Г.М. Аделсон-Велски и Е.М. Ландис, които ги въвеждат за пръв път (1962).

Едно от приложенията на дърветата е за представяне на математически изрази. Терминалните възли представят операндите, а останалите - операциите. Йерархията на върховете

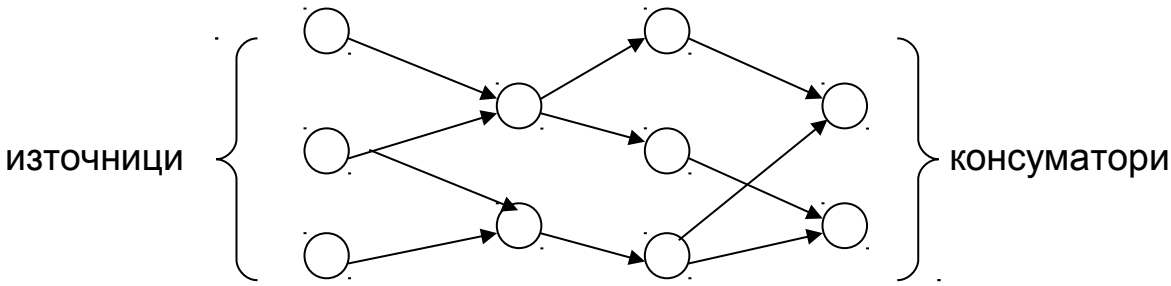
следва приоритета на операциите. Например изразът 5(x + 7) + y:(x + 7) може да бъде представен чрез следното дърво:



Фиг. 5.18. Представяне на математически израз чрез дърво

5.1.4. Транспортни мрежи

Мрежа (*Net, network*) - ориентиран граф G(V, E), в който се допускат примки и кратни дъги и който се използва като модел на система, процес и др. Част от възлите в мрежата са маркирани като входни (източници) и изходни (консуматори). Дъгите са ориентирани от входните върхове към изходните. Често под мрежа се разбира транспортна мрежа. Мрежа с три източника и два стока е показана на следващата фигура:



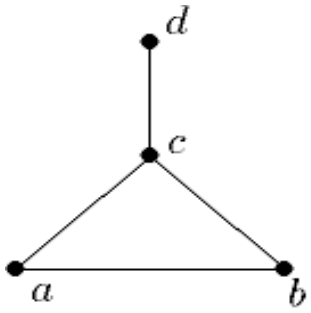
Транспортна мрежа (*Transportation network*) – ориентиран граф G(V, E), в който са фиксирани два върха - вход и изход на мрежата и на всяка дъга е определена пропускателна способност F: E $\rightarrow$ N. Едно от приложенията на транспортните мрежи е в мрежовото планиране и управление, което ще разгледаме в гл. (6).

5.2. ПРЕДСТАВЯНЕ НА ГРАФИ

Представянето на графи *(Graph representation)* в оперативната памет на компютрите при разработване на програми предполага съхранение на цялата информация за структурата на графа. Изборът на един или друг начин за представяне на графи зависи от конкретната задача и влияе съществено върху ефективността на използваните алгоритми. За целта се използват матрици на съседство, матрици на инцидентност, списъци на съседство, списъци на ребрата и др.

Матрица на съседство *(Adjacency matrix, vertex incidence matrix)* – квадратна (0, 1) матрица A(G) с размер $n \times n$ (n – брой на върховете в графа G), с елементи a_{ij}, определени по следния начин: $a_{ij} = 1$, ако върховете v_i и v_j са съседни, т.е. съединени с дъгата или реброто (v_i, v_j) и 0 в противен случай. За неориентиран граф матрицата на съседство е симетрична матрица с нули на главния диагонал. Съхраняването само на елементите над главния диагонал ще позволи да се спести памет, което може да бъде съществено при големи матрици, но ще увеличи времето за работа на алгоритъма. Временната сложност на алгоритми за обработка на графи, представени чрез матрица на съседство е $O(n^2)$. При представяне на мулти и псевдографи матрицата се модифицира, като (i,j)-я елемент е равен на броя ребра, съединяващи върховете v_i и v_j. Примките се броят като две ребра. Матрицата на съседство представя графа (орграф, мултиграф, псевдограф) с точност до изоморфизъм (изоморфните графи имат еднакво представяне).

На следващата фигура е дадено представяне на граф чрез матрица на съседство:

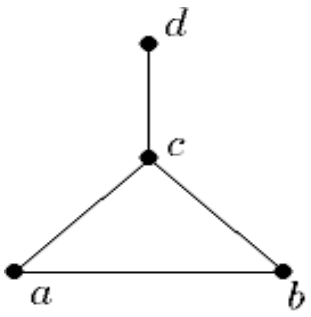


A(G) =

	a	***b***	***c***	***d***
a	0	1	1	0
b	1	0	1	0
c	1	1	0	1
d	0	0	1	0

Фиг. 5.19. Представяне на граф чрез матрица на съседство

Матрица на инцидентност *(vertex-edge incidence matrix)* – правоъгълна (0,1)-матрица I(G) с размер n $\times$ m, където n – брой на върховете, а m – брой на ребрата (дъгите) на графа G. В неориентиран граф елементът a_{ij} е равен на 1, ако върхът v_i е инцидентен на реброто e_j, и равен на 0 в противен случай. В ориентиран граф елементът a_{ij} е равен на 1, ако върхът v_i е начало на дъгата e_j, равен на -1, ако v_i е край на дъгата e_j и равен на 0 в противен случай. Матрицата на инцидентност задава графа с точност до изоморфизъм. Временната сложност на алгоритми за обработка на графи, представени чрез матрица на инцидентност е O(n.m).

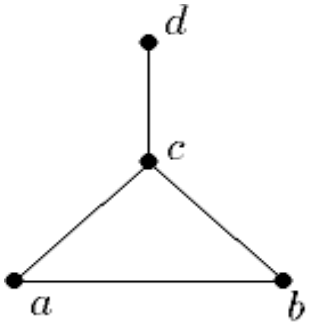


I(G) =

	(a,b)	*(a,c)*	*(b,c)*	*(c,d)*
a	1	1	0	0
b	1	0	1	0
c	0	1	1	1
d	0	0	0	1

Фиг. 5.20. Представяне на граф чрез матрица на инцидентност

Матрица на Киркхоф *(Kirchhoff's matrix)* - квадратна (0,1)-матрица B(G) с размер n $\times$ n, n – брой на върховете в G. Елементът b_{ij} е равен на -1, ако върховете v_i и v_j са съседни, равен е на степента deg(v_i) на върха v_i при i = j и е равен на 0 в останалите случаи. Сумата от елементите във всеки ред и всеки стълб е равна на 0.



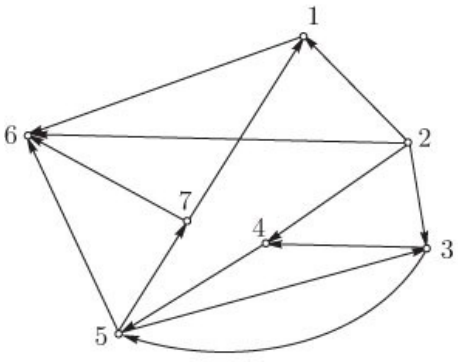
B(G) =

	a	***b***	***c***	***d***
a	2	-1	-1	0
b	-1	2	-1	0
c	-1	-1	3	-1
d	0	0	-1	1

Фиг. 5.21. Представяне на граф чрез матрица на Киркхов

Матрица на теглата *(Weight matrix)* - на матрицата на съседство за претеглен граф. Представлява квадратна матрица с размер n $\times$ n (n – брой на върховете на графа). Елементът a_{ij} е равен на теглото на реброто (дъгата) (v_i, v_j) ако такова ребро съществува. В противен случай a_{ij} се полага равен на нула или безкрайност в зависимост от решаваната задача.

Списък на съседство *(Adjacency list)* – за всеки един от n–те върха на графа се задава списък на съседните му върхове. Списъчните структури в програмирането са удобен начин за представяне на данни, поради възможността за извършване на модификации. Следващият орграф може да бъде представен чрез списък на съседство по следния начин:



Adj(v)

1: 6
2: 1, 3, 4, 6
3: 4, 5
4: 5
5: 3, 6, 7
6:
7: 1, 6

Фиг. 5.22. Представяне на граф чрез списък на съседство

Списък на ребрата. Всяко ребро се представя чрез двойка инцидентни върхове. За целта се използват два едномерни масива с брой на елементи |E|. Първият съдържа началните, а втория – крайните върхове на ребрата. Този начин за предствяне е ефикасен, ако графът е силно разреден (има много върхове и малко ребра). Лесно се представят примки и кратни ребра (дъги). Графът от фиг. 11 има следното представяне:

A = (1, 2, 2, 2, 2, 3, 3, 4, 5, 5, 5, 7, 7)

B = (6, 1, 3, 4, 6, 4, 5, 5, 3, 6, 7, 1, 6)

5.3. СТРАТЕГИИ ЗА ОБХОЖДАНЕ НА ГРАФИ

Много алгоритми върху графи предполагат обхождане на всички върхове на даден граф. Обхождането може да се осъществи по два начина:

Обхождане в дълбочина (*Depth First Search* или *DFS;* други наименования - *търсене с връщане, бектрекинг*) – метод за систематично обхождане на всички върхове на даден граф, достижими от даден начален връх. Движението се осъществява в две посоки:

- напред - от текущия връх по реброто напред към все още непосетен връх винаги, когато това е възможно и

- назад - връщане от текущия връх по реброто назад към вече посетения връх, ако движението напред е невъзможно, поради достигане до връх без наследник или до връх, през който вече сме минали.

При обхождане в дълбочина за текущия връх s се избира и посещава един съсед v_i, а останалите съседи на s ще бъдат посетени едва след като са обходени всички съседи на v_i. Изборът

на съседен връх за посещение зависи от избраната схема и може да бъде най-левия наследник, или най-близкия и др.

Обхождането в дълбочина винаги завършва след краен брой стъпки в началния връх, от който е започнало обхождането. Ако графът е ориентиран, движението напред по дъгите се извършва в съответствие с тяхната ориентация.

Обхождане в ширина (*Breadth (Width) First Search* или *BFS*) - метод за систематично обхождане на всички върхове на даден граф, при който върховете се обхождат по реда на нарастване на разстоянието (в брой върхове) от началния връх. С други думи, първо се посещават всички непосредствени съседи на даден връх, след което техните съседи и т.н.. Обхождането следва схемата:

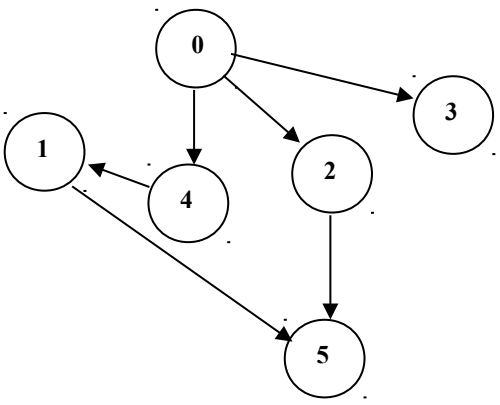
- на началния връх s присвояваме етикет 0;

- на всички съседи на началния връх присвояваме етикет 1;

- за всеки връх с етикет 1 определяме съседните, все още не номерирани върхове, на които присвояваме етикет 2 и т.н.

В процеса на обхождане върховете на графа се номерират. При това между началния връх с номер 0 и върха с номер i има точно i – 1 върха и i ребра.

Последователността от върхове при примерно обхождане в дълбочина и в ширина е следната:



DFS (в дълбочина): 0 4 1 5 2 3

BFS (в ширина): 0 4 2 3 1 5

Фиг. 5.23. Обхождане на граф в дълбочина и в ширина

5.4. ТЪРСЕНЕ НА ПЪТ В ГРАФ

5.4.1. Пътища в граф. Основни понятия

Маршрут *(Sequence)* в неориентиран граф е редица от върхове и ребра:

$v_0, e_1, v_1, e_2, \ldots e_{n-1}, v_{n-1}, e_n, v_n, e_i = (v_{i-1}, v_i), 1 \leq i \leq n.$

Даденият маршрут започва от началния връх v_0, до крайния връх v_n с вътрешни върхове $v_1, v_2, \ldots v_{n-1}$. Може да се зададе чрез изреждане само на върховете $v_0, v_1, \ldots v_{n-1}, v_n$ или само на ребрата $e_1, e_2, \ldots e_{n-1}, e_n$., Маршрутът се нарича:

- ***краен***, ако броят на влизащите в него ребра е краен и безкраен в противен случай;

- ***отворен***, ако крайните върхове са различни и затворен в противен случай;

- ***с дължина n****,* ако съдържа n ребра;

- ***обхващащ***, ако съдържа всички върхове.

Път *(Path)* - в орграф е е редица от върхове и дъги:

$S = (v_0, e_1, v_1, e_2, \ldots e_{n-1}, v_{n-1}, e_n, v_n),$

такава, че дъгата $e_i = (v_{i-1}, v_i), 1 \leq i \leq n$ и нито един връх не се среща два пъти. Пътят започва от началния връх v_0, до крайния връх v_n с вътрешни върхове $v_1, v_2, \ldots v_{n-1}$.

Верига *(Chain, trail)* - в неориентиран граф е маршрут, всички ребра на който са различни. В ориентиран граф е редица от

върхове на графа v_1, v_2, … v_k, в която съседните върхове v_i, v_{i+1} определят дъгата (v_i, v_{i+1}) или дъгата (v_{i+1} v_i).

Цикълът *(Loop, Circuit, Cycle)* е затворена верига, краищата на която съвпадат. Цикълът е прост *(Simple cycle)*, ако нито един връх не се среща два пъти.

Контур *(Cycle)* - затворен път в орграф. Контурът е прост *(Simple cycle)*, ако нито един връх не се среща два пъти.

Свързани върхове *(Connected verticies)* са тези, между които съществува проста верига.

Графът G се нарича ***свързан*** (*Connected graph*), ако всяка двойка върхове v_i и v_j са свързани. В противен случай графът е несвързан. В един несвързан граф може да има свързани компоненти, т.е. подграфи, които са свързани. В съответствие с тази дефиниция, *дървото е свързан граф без цикли*. Следващата теорема дава връзката между графи и дървета.

Теорема 5.4. Граф с n върха и m ребра е дърво тогава и само тогава, когато са изпълнени две от следните три условия:

1. графът е свързан;
2. няма цикли;
3. m = n-1.

Дължина на ребро (дъга) *(Length of an arc)* в претеглен граф – тегло на реброто (дъгата), разглеждано като разстояние между върховете, в непретеглен граф се приема за единица.

Дължина на верига *(Length of a chain)* – в непретеглен граф е равна на броя на участващите ребра (дъги), а в претеглен граф – сумата от дължините на ребрата (дъгите).

Дължина на път *(Length of a phat)* – в непретеглен орграф е равна на броя на участващите дъги, а в претеглен орграф – сумата от дължините на дъгите.

Дължина на цикъла (контура) *(Length of a cycle)* – в непретеглен граф е равна на броя на участващите ребра (дъги), а в претеглен граф – сумата от дължините на ребрата (дъгите).

Разстояние между върховете *(Distance between two vertieces)* – дължината на най – късата верига (в орграф – най – късия път), съединяваща дадените върхове. Ако такава верига (път) не съществува, разстоянието се полага равно на ∞.

Най - кратък път *(Shortest path)* между два фиксирани върха е пътя с минимална дължина, съединяващ тези върхове.

Път в дърво T е редица от върхове t_1, t_2, t_3 ..., t_k, в която върхът t_1 е корен на T, а върхът t_i е наследник на t_{i-1} (за i=2, ..., k). Числото k–1 се нарича ***дължина*** на пътя. В претеглено дърво, дължината на пътя е сума от дължините на ребрата (дъгите). Може да се докаже, че между всеки два върха на дървото T съществува единствен път (в частност, от корена t до всеки връх $t_i \in$ T има единствен път). ***Ниво*** на върха t $\in$ T се нарича дължината на пътя от корена на дървото T до върха t .

5.4.2. Екстремални пътища в граф

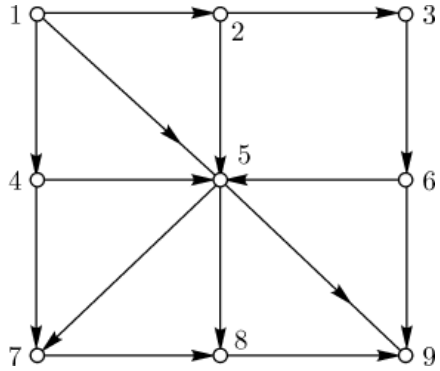
Интересен клас задачи са свързани с търсене на път, отговарящ на някакви предварително зададени критерии, например най-къс път, най-дълъг път, път между два фиксирани върха и др. Разглежданите графи могат да претеглени или непретеглени ориентирани или неориентирани графи.

Един от известните алгоритми е свързан с построяване на *дърво на решенията* (*дърво на възможните пътища*). Приложим е както за неориентирани, така и за ориентирани графи. Дължината на пътя се определя от броя на участващите в него ребра (дъги).

Може да се използва за намиране на броя на възможните пътища, за търсене на път с минимална или максимална дължима.

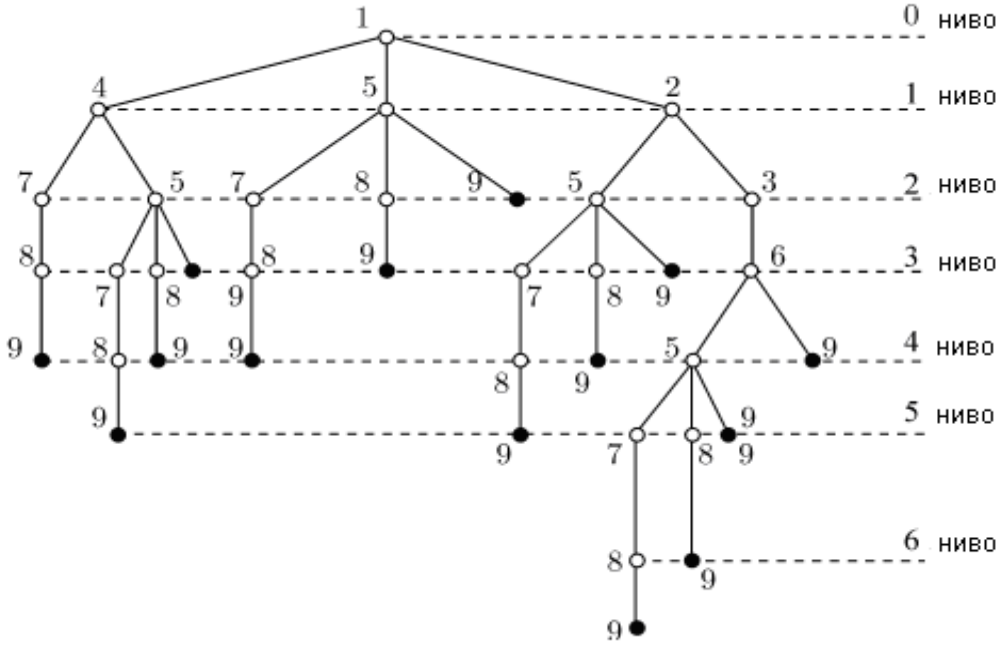
Дървото на решенията се построява чрез обхождане на графа по широчина. За демонстрация на алгоритъма ще разгледаме графа на фиг. 5.24.

Търсим най-краткия път от връх 1 до връх 9, в който всеки връх участва не по-вече от един път. Движението се осъществява само по посока на стрелките. На графа съпоставяме дърво, построено в съответствие със стратегията за обхождане на граф в ширина по следния начин: за корен на дървото определяме върха 1.



Фиг. 5.24 Търсене на екстремален път в ориентиран (неориентиран) граф от връх 1 до връх 9

От връх 1, следвайки дъгите, можем да попаднем във върховете 4, 5 и 2, които приемаме за наследници. Следваме същата логика за определяне на наследниците на 4, 5 и 2 връх и т.н. до достигане на връх 9, в който пътя приключва. Получаваме следното дърво:



Фиг. 5.25. Дърво на възможните пътища от връх 1 до връх 9

Търсенето на път следва стратегията за обхождане в дълбочина, но не в самия граф, а в дървото на възможните пътища. Броят на възможните пътища съвпада с броя на терминалните възли (листа) на дървото и е 14. Нивата на листата определят дължината на съответния път. Най-късият път 1, 5, 9 е с дължина 2, а най-дългия 1, 2, 3, 6, 5, 7, 8, 9 е с дължина 7.

Алгоритъмът може да се модифицира и за претеглени графи. В такъв случай, след построяване на дървото на решенията, намираме дължината на всички пътища от корена до листата, сравняваме дължините им и избираме минималния (максималния).

5.4.3. Хамилтонови графи. Задача за търговския пътник.

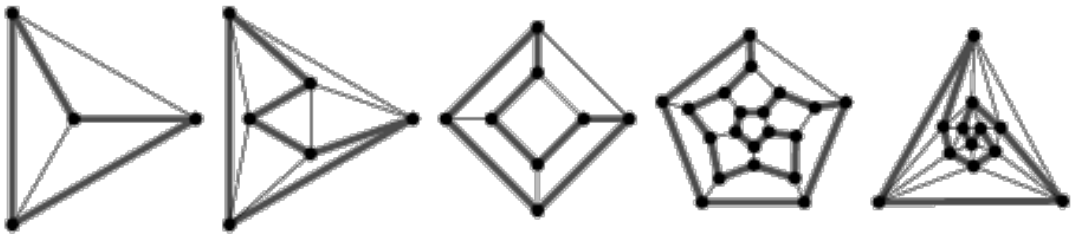
Този вид графи са наречени на името на ирландския математик Уилям Хамилтон, който през 1859 година предлага занимателната задача „Околосветско пътешествие”, в която се изисква да се обходят всички върхове на додекаедър по такъв

начин, че през всеки връх да се мине точно един път. Додекаедърът е многостен с 20 върха и 30 ребра, стените на който са правилни петоъгълници.

Неориентирани графи

Хамилтонова верига *(Hamiltonian chain)* – верига в граф, минаваща през всеки връх точно един път.

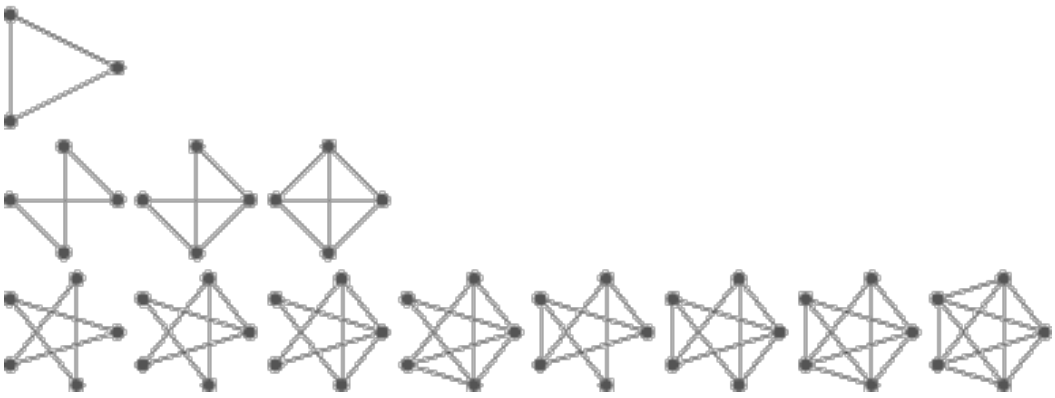
Хамилтонов цикъл *(Hamiltonian circuit)* - цикъл в граф, минаващ през всеки връх точно един път.



Фиг. 5.27. Хамилтонов цикъл в граф (с плътни линии)

Хамилтонов граф *(Hamiltonian graph)* - граф, съдържащ Хамилтонов цикъл или Хамилтонова верига, съединяваща зададена двойка върхове.

Примери за хамилтонови графи са пълните графи, графите на правилните многостени и др.



Фиг. 5.28. Хамилтонови графи с 3, 4 и 5 върха

Ориентирани графи

Хамилтонов път *(Hamiltonian path)* - път в орграф, минаващ през всеки връх точно един път.

Хамилтонов цикъл *(Hamiltonian cycle)* - контур, минаващ през всеки връх на орграфа точно един път

Хамилтонов орграф *(Hamiltonian directed graph)* - орграф, съдържащ Хамилтонов контур или Хамилтонов път между зададена двойка върхове.

До момента са известни само достатъчни условия за хамилтоновост на граф и орграф.

Теорема 5.5 (Теорема на Дирак) *(G.A.Dirac, 1952*). Ако в граф с n ≥ 3 върха за всеки връх v се изпълнява неравенството

deg (v) ≥ $\frac{n}{2}$,

то G е хамилтонов граф.

От теоремата следва, че ако даден граф с n ≥ 3 върха е пълен, то той е хамилтонов. Действително, за всеки връх v на пълен граф е изпълнено:

deg (v) = (n - 1) ≥ $\frac{n}{2}$

С този вид графи са свързани две класически задачи: проверка, дали даден граф е Хамилтонов и задачата за търговския пътник. И двете задачи са с експоненциална временна сложност.

Задача за търговския пътник (*Traveling salesman problem*)

Търговски пътник трябва да посети всеки от дадени n града по един път, тръгвайки от един начален град и връщайки се в него. Ако са известни разстоянията между всеки два града, да се намери най-краткия път.

Така формулирана, задачата е оптимизационна, тъй като от всички възможни пътища търсим този с минимална дължина. Математическата постановка е свързана с намиране на хамилтонов цикъл (или контур) в пълен претеглен граф с минимално тегло. В зависимост от смисъла на теглата, съпоставени на ребрата (дъгите), можем да търсим път с минимална дължина, с минимален разход на гориво, с най-ниска цена и др.

Известни са и други формулировки на задачата, свързани с проверка за съществуване на път, започващ от даден град и завършващ в него или съществуване на път с дължина $\leq$ L, където L е предварително зададено положително число.

Задачата за търговския пътник е NP-пълна задача. За решението и не съществуват полиномиални алгоритми, единствения метод за решение е изчерпващото търсене в пространството на допустимите решения с временна сложност O(n!). При голям брой градове задачата е практически неразрешима, затова се използват евристики за намаляване на броя на анализираните маршрути (метод на клоните и границите) или приближени методи, които дават близко до оптималното решение като алчни алгоритми (метод на най-близкия съсед), генетични алгоритми, вероятностни методи и др. Задачата е еталон за сложност и има много голямо теоретично и практическо значение. Именно тя се използва при анализ на различни евристични подходи за съкращаване на изчерпващото търсене.

При ***изчерпващо търсене*** номерираме подлежащите на обхождане градове последователно {1, 2, 3, … n}. Всеки маршрут, започващ и завършващ в град 1, може да се разглежда като пермутация на числата 2, 3, ...n. Задачата за генериране на всички маршрути е еквивалентна на задачата за получаване на всички (n-

1)! на брой пермутации на числата от 2 до n. Алгоритъмът за решение на задачата за търговския пътник чрез изчерпващо търсене включва следните стъпки:

1. Полагаме дължината на минималния път $S_{MIN} = \infty$, инициализираме търсения път $P_{MIN} =$ “”; номера на поредната пермутация I = 0;
2. I = I + 1; генерираме I-тата пермутация; пресмятаме дължината на текущия път S_I;
3. Ako $S_{MIN} > S_I$, полагаме $S_{MIN} = S_I$; запомняме I - тата пермутация в P_{MIN};
4. Ако I = (n-1)! – край на алгоритъма; в противен случай се връщаме към стъпка 2.

След приключване на алгоритъма S_{MIN} ще съдържа дължината на минималния път, а P_{MIN} – реда на обхождане на градовете.

При ***метода на клоните и границите*** (*Horowitz, 1978*) на всяка стъпка се поражда един от възможните варианти на обхождане, като се отчита оптималния до момента път, който се използва за сравнение със следващите кандидати. Ако добавянето на нов, все още не посетен връх към текущото решение увеличава дължината на пътя над оптималната за момента, върхът заедно със всички възможни продължения се отсича. В резултат на отсичането на неперспективни решения броят на вариантите съществено се намалява, но въпреки това остава експоненциален ($1{,}26^n$). В най – лошия случай, когато оптималният път е последен от анализираните, сложността на метода ще съвпадне със сложността на изчерпващото търсене.

Алчните алгоритми следват стратегия, при която не се анализират всички възможни варианти. Решението се конструира

чрез последователно добавяне на следващ не посетен връх с най-малко тегло на реброто (дъгата). Методът е високо ефективен, тъй като на всяка стъпка се избира само един вариант за продължение. Въпреки че няма гаранция за оптималност на полученото решение, евристиката на „най-близкия съсед" е възможен компромис в случаите, когато изчерпващото търсене е практически неосъществимо.

Вероятностните алгоритми на всяка стъпка генерират по случаен начин един от възможните пътища и пресмятат дължината му, която се сравнява с оптималната до момента. Ако текущо генерираният път има по-малка дължина, то се съхранява този път и неговата дължина. Многократното повторение на тези действия ще доведе до получаване на път с минимална дължина от всички генерирани.

5.4.4. Ойлерови графи. Задача за кьонингсбергските мостове

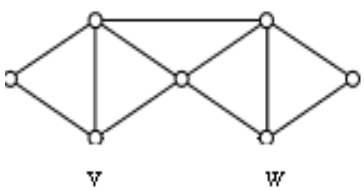
Този специален вид графи са свързани с името на техния пръв изследовател Леонард Ойлер.

Неориентирани графи

Ойлерова верига *(Eulerian chain)* – верига, минаваща през всяко ребро точно един път.

Теорема 5.6. Необходимо и достатъчно условие за съществуване на Ойлерова верига в неориентиран граф е наличието на точно два върха с нечетни степени.

Графът на рисунката съдържа точно два върха v и w от нечетна степен. За него може да се построи Ойлерова верига, започваща от



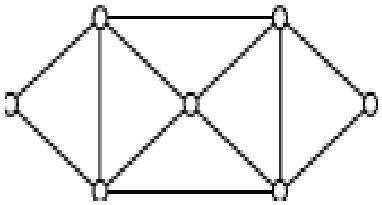
върха v и завършваща във върха w, която минава през всяко ребро на графа точно един път.

Ойлеров цикъл *(Eulerian circuit)* - цикъл, минаващ през всички ребра на графа точно един път.

Ойлеров граф **(***Eulerian graph***)** - свързан граф, съдържащ Ойлеров цикъл.

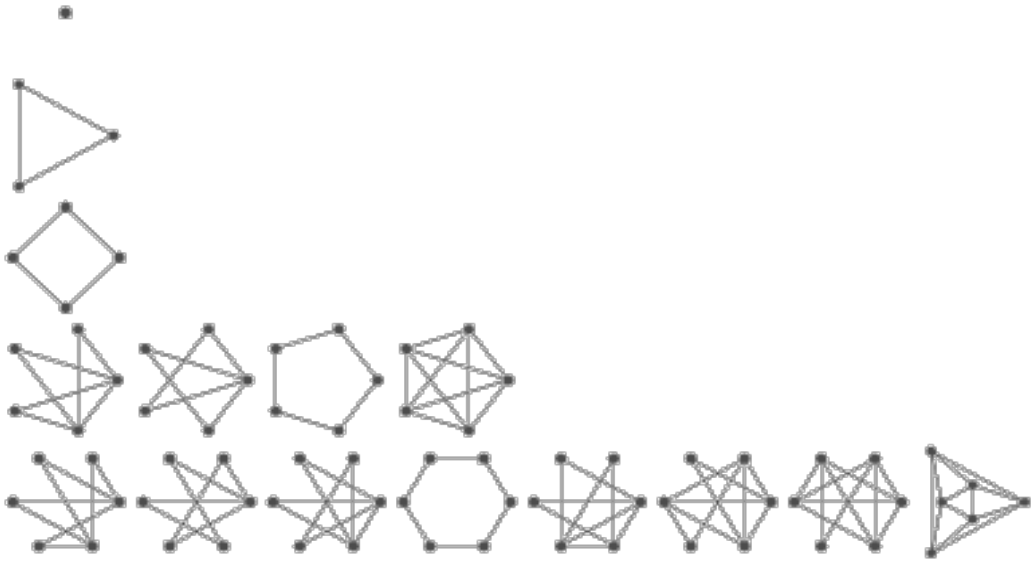
Необходимо и достатъчно условие даден граф да бъде Ойлеров се дава от следната теорема:

Теорема 5.7 (Теорема на Ойлер) *(L.Euler, 1736).* Непразен свързан граф е Ойлеров тогава и само тогава, когато не съдържа върхове от нечетна степен.



Фиг. 5.29. Ойлеров граф

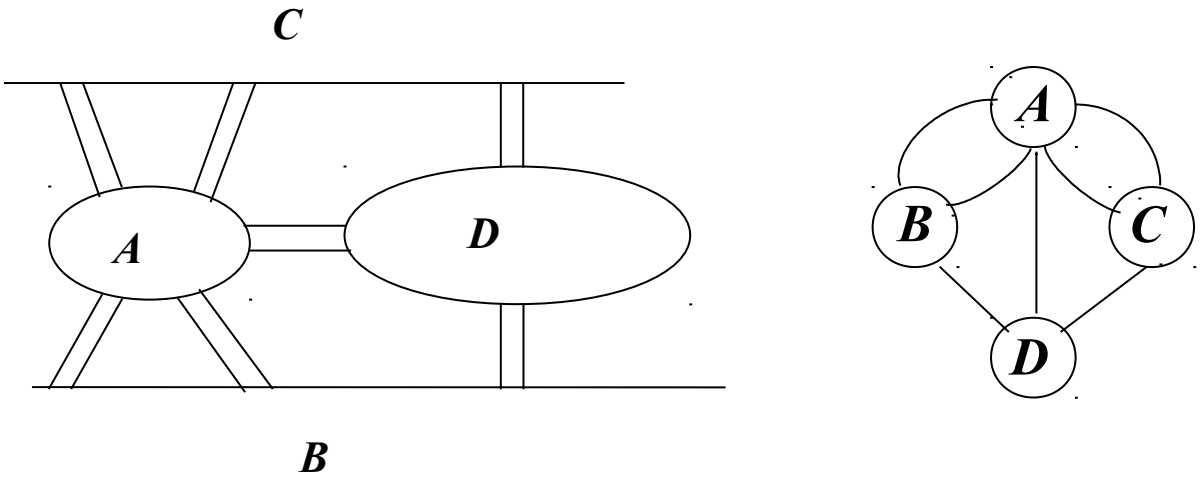
Следователно, ако в даден граф всички върхове са с четна степен, то той е Ойлеров и обратно. Такива графи могат да бъдат нарисувани без да се вдига молива от листа и без повторение на линиите. Следните графи са Ойлерови графи, тъй като всички върхове са от четна степен:



Фиг. 5.30. Ойлерови графи с 1, 3, 4, 5 и 6 върха

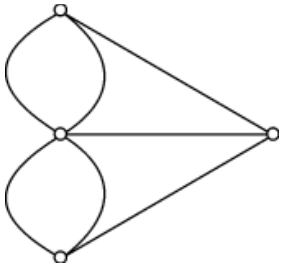
Задача за Кьонингсбергските мостове *(Koenigsberg's bridges problem)*

Свързана е с обхождане на всичките седем моста на Кьонигсберг по такъв начин, че през всеки мост да се мине точно един път. Градът Кьонигсберг, където през XVIII век е живял Ойлер, е имал седем моста на река Прегел, построени по следния начин:



Фиг. 5.31. Мостовете на река Прегел в Кьонингсберс

Формално задачата се свежда до търсене на Ойлеров цикъл в мултиграфа:



Задачата е поставена и решена от Ойлер в следния общ вид: при какви условия свързан граф съдържа цикъл, минаващ през всяко ребро точно един път?. Решението следва от теоремата на Ойлер: цикъл не съществува, тъй като върховете на графа имат нечетни степени.

Връзка с Ойлеровите графи има следната оптимизационна задача:

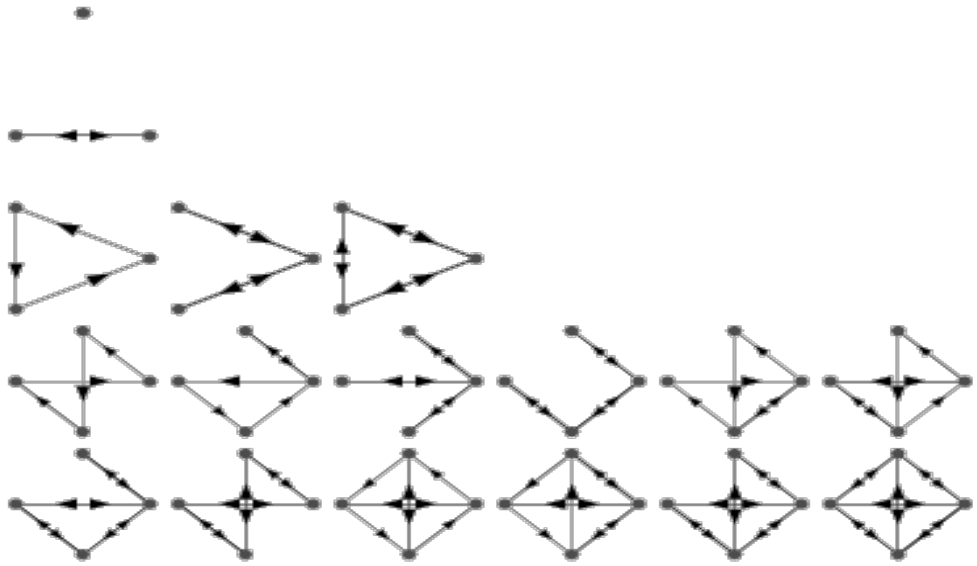
Задача за китайския пощальон *(Chinese postman's problem)* – в претеглен граф да се намери цикъл, минаващ през всяко ребро поне по един път, за който сумата от теглата на ребрата да бъде минимална. Дължината на всяко ребро се включва в сумата толкова пъти, колкото реброто се среща в цикъла. Ако графът е Ойлеров, всеки Ойлеров цикъл е решение на задачата.

Ориентирани графи

Ойлеров цикъл *(Eulerian cycle, eulerian trail)* - контур, минаващ през всяка дъга на орграфа в точно един път.

Ойлеров орграф *(Eulerian digraph)* - орграф, който притежава Ойлеров цикъл.

Теорема 5.8. Даден орграф е Ойлеров тогава и само тогава, когато полустепента на входа на всеки връх е равна на полустепента на изхода.



Фиг. 5.32. Ойлерови орграфи с 1, 2, 3, и 4 върха

При търсене на Ойлеров цикъл в граф може да се приложи следния алгоритъм [19]:

Започваме обхождане на графа от произволен връх *i*. Намираме инцидентно с него ребро *(i, j)* и го маркираме като посетено. Продължаваме с върха *j*: за него намираме непосетено ребро *(j,k),* маркираме го като посетено и преминаваме в *k*. Продължавайки по този начин, в даден момент ще се озовем в началния връх *i* и ще затворим цикъла. Ако всички ребра на графа вече са маркирани, то този цикъл е Ойлеров. Ако са останали непосетени ребра, то намираме връх *x*, принадлежащ на току-що намерения цикъл и инцидентен с поне едно непосетено ребро. От това, че всеки връх трябва да бъде от четна степен, следва, че *x* е инцидентен с четен брой (т.е. поне две) непосетени ребра . От *x* започваме да изграждаме цикъл по вече описания начин, докато отново се върнем в него. Получаваме втори цикъл, който обединяваме с първия (общата им точка ще бъде върхът *x*). Така,

след краен брой стъпки, всички ребра ще се включат в един общ цикъл - търсеният Ойлеров цикъл.

В случай, че търсим Ойлеров път, можем да съединим върховете от нечетна степен с ребро (съществуват точно два такива върха) и да намерим Ойлеров цикъл по описания алгоритъм. След отстраняване на добавеното ребро получаваме търсения Ойлеров път.

5.5. ДВУДЕЛНИ ГРАФИ. ЗАДАЧА ЗА НАЗНАЧЕНИЯТА

Двуделните графи се използват за моделиране на отношения между обекти от два типа например, отношението между продукти и готови изделия: „продуктът x се използва в производството на изделието y″; между работници и професии: „работникът x владее професията y″ и др.

5.5.1. Задача за сватбите. Теорема на Хол

За илюстрация на двуделните графи често се използва следната задача.

Задача за сватбите.

Дадени са две крайни множества: на юношите и на девойките. Всеки юноша познава няколко девойки. При какви условия можем да оженим юношите така, че всеки юноша да се ожени за позната девойка?

Пример 5.1. Нека юношите са четири $\{b_1, b_2, b_3, b_4\}$, а девойките пет $\{g_1, g_2, g_3, g_4, g_5\}$. Отношението «юношата x познава девойката y» задаваме таблично:

Възможно решение е:

Юнош а	***Девойки, с които се познава***		
b_1	g_1	g_4	g_5
b_2	g_1		
b_3	g_2	g_3	g_4
b_4	g_2	g_4	

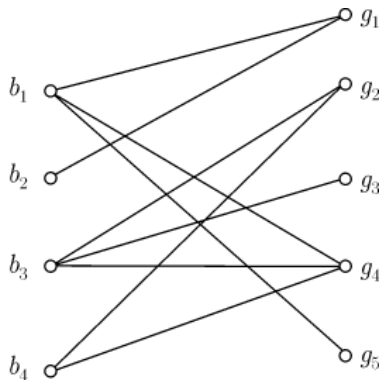
$b_1 \rightarrow g_5$

$b_2 \rightarrow g_1$

$b_3 \rightarrow g_2$

$b_4 \rightarrow g_4$

Графично можем да представим юношите и девойките чрез върхове, а отношението на познанство - чрез ребра по начина, показан на следващата фигура. Същият граф ще получим, ако разглеждаме 4 работника, които се конкурират за 5 работни места. Такива графи се наричат ***двуделни.***



Фиг. 5.33. Граф, представящ отношението „познанство" между 4 юноши и 5 девойки

Условията за съществуване на решение на задачата за сватбите дава следващата теорема, доказана от Филип Хол през 1936 г.

Теорема 5.9 (Теорема на Хол за сватбите). Задачата за сватбите има решение тогава и само тогава, когато всеки к юноша познават поне k девойки за всяко $0 \leq k \leq n$, където n е броят на юношите.

Доказателство:

Необходимост: Предполагаме, че задачата има решение. Ще докажем, че условието на теоремата е вярно.

Нека допуснем, че условието на теоремата не е изпълнено, т.е съществуват k юноши, които могат да бъдат оженени най-много за k-1 девойки. От принципа на Дирихле за чекмеджетата следва, че поне един от тези k юноши не може да бъде оженен за нито една девойка, следователно и всичките n юноши не могат да бъдат оженени. Но тогава задачата няма решение, което противоречи на допускането, че задачата има решение. Следователно допускането не е вярно и условието на теоремата е изпълнено.

Достатъчност: Предполагаме, че условието на теоремата е вярно. Ще докажем, че задачата има решение.

Доказателството ще извършим чрез индукция по n.

Нека n = 1. Тогава трябва да оженим точно един юноша. Но условието " всеки к юноша познават поне k девойки за всяко k" е еквивалентно на условието "единственият юноша познава поне една девойка", за която можем да го оженим и задачата има решение.

Допускаме, че задачата има решение, ако броят на юношите е по-малък от n.

Ще докажем, че задачата има решение и ако броят на юношите е точно n. Нека първо считаме, че всеки k юноши познават поне k+1 девойки (1<=k<n). Тогава, ако оженим един от юношите за позната девойка, ще останат още n-1 юноши. Но от n-те юноши

всеки к познават поне k+1 девойки (1<=k<=n) и понеже един вече е оженен, то сега всеки k юноши познават поне k девойки. Така получихме група от n-1 юноши, всеки k от които познават поне k девойки и от индукционното предположение задачата има решение (1<=k<n). Нека сега съществуват k юноши, които познават точно k девойки. По индукционното предположение за тези k юноши задачата има решение. Остават още n-k юноши. Ако допуснем, че съществуват h юноши (1<=h<=n-k), които могат да бъдат оженени за най-много h-1 девойки, то тези юноши, заедно с предните к образуват група от k+h юноши, които могат да бъдат оженени за най-много k+h-1 девойки (k+h<=n), което е в противоречие с условието. Следователно всеки h юноши познават поне h девойки (h<=n-k) и от индукционното предположение следва, че задачата има решение и за (n-k)-те юноши. Съгласно метода на пълната математическа индукция, задачата има решение за всичките n юноши (n-k<n).

Теоремата на Хол има теоретично значение, поради трудността за практическа проверка на формулирания в нея критерий.

5.5.2. Двуделни графи. Паросъчетания

Двуделен граф (*Bipartite graph*) – граф G(V, E) за който множеството от върхове V може да се представи като обединение на две непресичащи се множества V_1 и V_2,

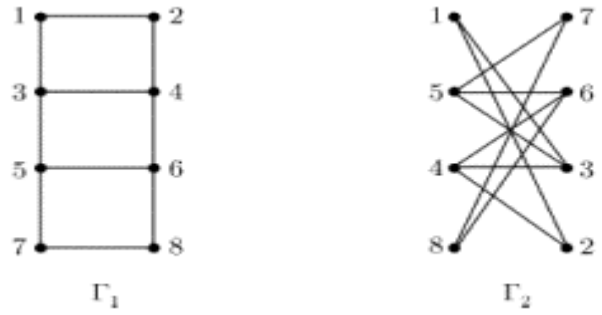
$V1 \cup V2 = V$ и

$V1 \cap V2 = \varnothing$), такива, че:

$(V1 \times V1) \cap E = \varnothing$ и

$(V2 \times V2) \cap E = \varnothing$,

т.е. между върховете на V_1 и между върховете на V_2 няма ребра.



Фиг. 5.34. Примери за двуделни графи

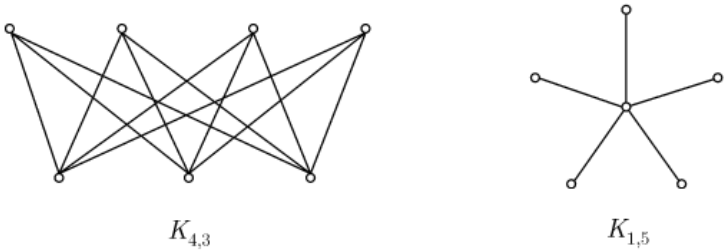
Двуделните графи се наричат също ***бихроматически***, тъй като всички върхове могат да се оцветят в два цвята по такъв начин, че съседните върхове да са оцветени в различни цветове.

Ако разлагането на V на две непресичащи се множества V_1 и V_2 не е зададено, възниква въпросът дали въобще съществува. При брой на върховете $n = |V|$, възможните разлагания на V на две непресичащи се множества е 2^{n-1}. Проверката на възможните варианти е с експоненциална сложност и при големи n е практически неосъществима. Следващата теорема дава *критерий за двуделност на граф*, на който се базират ефективни алгоритми за проверка.

Теорема 5.10. Следващите твърдения за графа G са равносилни:

1. G е двуделен граф.
2. В G няма цикли с нечетна дължина.
3. В G няма прости цикли с нечетна дължина.

Ако G е прост граф и всеки връх от V_1 е съединен чрез ребро с всеки връх от V_2, G се нарича пълен двуделен граф и се означава с $K_{m,n}$, където m и n са брой на върховете в V_1 и V_2 съответно. Пълните двуделни графи имат точно m + n върха и m.n ребра.



Фиг. 5.26. Пълни двуделни графи

Паросъчетание *(Matching)* M $\subseteq$ E в графа ще наричаме такова произволно подмножество от ребра на графа, в което никои два елемента нямат общ връх.

Максимално паросъчетание (*Max Matching*) – съчетание с максимален брой ребра.

Съвършено (или пълно) паросъчетание *(Perfect matching)* в двуделен граф G(V_1, V_2, E) с $|V_1| = |V_2|$ – паросъчетание, в което се включват всички върхове на V_1 (следователно и на V_2). Ако M е съвършено , то всеки връх на графа е инцидентен на някое ребро от M.

Задачата за сватбите е еквивалентна на задачата за намиране на максимално паросъчетание (*Match Problem*) в двуделен непретеглен граф G(V_1, V_2, E).

5.5.3. Задача за назначенията. Унгарски метод

Задачата за назначенията е оптимизационна задача върху претеглени двуделни графи и може да бъде формулирана по следния начин:

Задача за назначенията *(Assignment problem)*

Дадено е множество от n работника и m работни места. Известна е ефективността от назначението на всеки работник на всяко работно място. Търсим разпределение на работниците по работните места с максимална ефективност.

Формално: Даден е двуделен неориентиран претеглен граф $G(X_1,X_2,E)$. $|X_1| = |X_2| = n$, $E = X_1 \times X_2$. Да се намери пълно съчетание M, $|M| = n$, оптимално по зададен критерий.

Практически значими оптимизационни критерии са минимална или максимална сума от теглата на участващите в M ребра, или максимизация на най-малкото ребро.

Задачата за назначенията има пряка връзка със задачата за максималното паросъчетание. Съществуването на максимално паросъчетание гарантира съществуване на решение на задачата за назначенията. Ако максималното паросъчетание е единствено и съвършено, то се явява и решение на разпределителната задача. В противен случай, наличието на по-вече от едно максимални паросъчетания дава възможност за избор на оптимално назначение, съответстващо на избрания критерий за оптималност.

Ще разгледаме ***унгарския метод*** *(Hungarian method)*, който се използва за решение на задачата за максималното паросъчетание и задачата за назначенията. За целта ще преобразуваме задачата за намиране на пълно съчетание в двуделен граф в задача на линейното оптимиране. Възможността за това преобразуване показва, че за една и съща задача могат да се съставят различни математически модели, построени със средствата на различни математически теории.

Нека V_1 е множество от подлежащи на разпределение ресурси (работници, специалисти), а V_2 е множество от специалности (работни места). Въвеждаме коефициентите:

$$0 \leq c_{ij} \leq 1,\ i, j = 1, 2, ...n,$$

които за краткост ще наричаме коефициенти за пригодност на i-тия ресурс към j-тата специалност.

Ясно е, че колкото коефициентът c_{ij} е по-близък до 1, толкова ресурсът е по-пригоден за съответната специалност. При стойност на коефициента 1 ресурсът е абсолютно пригоден за специалността. Обратно, колкото c_{ij} е по близък до 0, толкова ресурсът е по-непригоден за специалността, а при стойност $c_{ij}=0$ имаме абсолютна непригодност, т.е. ресурсът не може да бъде разпределен на дадената специалност. По такъв начин коефициентите c_{ij} могат да служат като количествена мяра за ефективността на назначението на a_i-тия ресурс към s_j-тата специалност $a_i \rightarrow s_j$. Коефициентите c_{ij} образуват матрицата на пригодност C.

Ако x_{ij}, i, j = 1,2,…n са двоични променливи, приемащи стойност 1, ако i-тия ресурс е назначен на j-тата специалност и 0 в противен случай, то матрица от нули и единици, в която във всеки ред и всеки стълб има точно една 1, а останалите елементи са 0, ще ни дава конкретно назначение, качеството на което може да се определи, като се сумират коефициентите на пригодност от матрицата C, на които съответстват ненулеви елементи в матрицата X.

Ясно е, че за да е възможно разпределението, е необходимо броят на ресурсите да е равен на броя на специалностите. В противен случай ще останат свободни ресурси, или незаети специалности. В общия случай можем да си осигурим равенството чрез въвеждане на фиктивни ресурси или фиктивни специалности.

От казаното до тук следва, че задачата за назначенията може да се сведе до следната целочислена задача на линейното програмиране, известна като линейна задача за назначенията:

Линейна задача за назначенията *(Linear Assignment Problem*): Да се намери минималната (максималната) стойност на целевата функция

$$K_1 = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \to \min, i, j \in N \qquad (5.1)$$

при ограниченията:

$$\sum_{j=1}^{n} x_{ij} = 1, i \in N \qquad (5.2)$$

$$\sum_{i=1}^{n} x_i = 1, j \in N \qquad (5.3)$$

$$x_{ij} \in \{0,1\}, i \in N, j \in N \qquad (5.4)$$

В случая целевата функция K_1 дава критерий за качеството на решението, а нейната минимална (максимална) стойност ще определи най-доброто (оптималното) назначение. Ограниченията (5.2) и (5.3) осигуряват изискването всеки ресурс да бъде назначен на точно една специалност и обратно, а условието (5.4) - булевостта на променливите x_{ij}, $i,j \in N$. Силното изискване за целочисленост на променливите не е необходимо тъй като многостенът на решенията на задачата, определен от ограниченията (5.2), (5.3) притежава целочислени координати на върховете.

Могат да се разгледат и други, различни от (1), критерии за оптималност на решението, които са нелинейни, например:

Min-max Assignment Problem

$K_2 = \min(c_{ij}x_{ij}) \to \max$, $i,j \in N$

Balanced Assignment Problem

$K_3 = \max(c_{ij}x_{ij}) - \min(c_{ij}x_{ij}) \to \min$, $i,j \in N$

Критериите имат различна семантика. K_1 предполага търсене на решение, при което на всяка специалност се разпределя възможно най-подходящия ресурс, при K_2 максимизираме най-лошото назначение, а при K_3 минимизираме разликата между най-доброто и най-лошото назначение. Изборът на критерий за оптималност е експертна задача и зависи от естеството на реалната задача.

5.5.4. Алгоритъм на унгарския метод

Ще наричаме ***независими*** елементите на една матрица, ако никои два елемента не лежат на един и същи ред или стълб. Очевидно е, че решението на задачата за назначенията се свежда до намиране на такива n независими елемента на ценовата матрица C, чиято сума е минимална .

Решението на задачата за назначенията по унгарския метод включва следните стъпки:

1. От всеки стълб на матрицата на цените с размерност $n \times n$ се избира най-малкия елемент и се изважда от останалите елементи на стълба. В резултат на това всеки стълб на преобразуваната матрица ще съдържа поне една нула. Ако измежду тях има n независими, оптималното решение е намерено. Ако не е намерено се преминава към втория етап.

2. Намираме най-малкия елемент във всеки ред на преобразуваната матрица и го изваждаме от останалите елементи на реда. В получената редуцирана матрица на цените търсим n независими нули. Ако няма такива, преминаваме към третия етап. Ако има, получили сме оптималното решение

3. В редуцираната матрица зачеркваме редовете и стълбовете, съдържащи нули така, че броят на тези линии да бъде

минимален. Това се постига като първо зачеркваме ред или стълб, съдържащ най много нули, след това следващия по брой на нулите и т.н. Ако броят на линиите е n, намерили сме оптимално решение, чиито единици се намират точно там, където са нулите на матрицата. Ако броят на линиите е по-малък от n, преминаваме към следващата стъпка.

4. Измежду не зачертаните елементи избираме най-малкия и го изваждаме от всеки от тях, а го прибавяме към елементите, в които се пресичат линии. Проверяваме дали получената матрица съдържа n независими нули. Ако това не е така , отново повтаряме т.3, докато получим желаният резултат.

Сумата от елементите в първоначалната матрица, съответстващи на независимите нули, определя минималната стойност на назначението.

Забележки :

Ако в задачата за назначението се търси максимум на целевата функция, точка едно от алгоритъма се модифицира така: във всеки стълб се избира най-големият елемент и от него се изваждат всички елементи на стълба.

Ако някое назначение е недопустимо по някакви причини, съответната му стойност се полага равна на M.

Ако изходната матрица не е квадратна, се въвеждат необходимите фиктивни редове и стълбове, като на елементите им се приписват стойности, съобразени с условието на задачата.

Ако броят линии, необходими за зачеркване на нулевите елементи е равен на размерите на матрицата, то оптималното назначение е намерено.

Пример 5.2. Работата на алгоритъма ще демонстрираме със следната матрица :

2	10	15	0
10	18	20	9
15	24	26	10
12	25	27	8

Решение :

I. етап – Минималните елементи от съответните стълбове са съответно: 2, 10, 15, 0. Като ги извадим от елементите на съответните стълбове, получаваме следната матрица :

0	**0**	**0**	**0**
8	8	5	9
13	14	11	10
10	15	12	8

II. етап – Минималните елементи в съответните редове са съответно: 0, 5, 10, 8. Изваждаме ги от елементите на съответните редове и получаваме матрицата:

0	**0**	**0**	**0**
3	3	**0**	4
3	4	1	**0**
2	7	4	**0**

Очевидно е, че максималния брой независими нули в тази матрица е 3. Има няколко възможни комбинации. Следователно липсва пълно назначение, поради което преминаваме към третия етап.

III. етап – Зачеркваме редовете и стълбовете съдържащи нули с минимален брой линии в следната последователност: първи ред , трети стълб, четвърти стълб. Получаваме

~~**0**~~	~~**0**~~	~~**0**~~	~~**0**~~
3	3	**0**	4
3	4	1	**0**
2	7	4	**0**

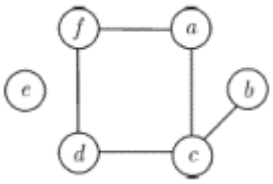
матрица, в която минималният измежду не зачертаните елементи в нея е 2.

0	0	2	2
1	1	0	4
1	2	1	0
0	5	4	0

IV. етап - Изваждаме 2 от останалите незачеркнати елементи и го прибавяме към двойно зачеркнатите a_{13} и a_{14} . Получаваме матрица, в която има четири независими нули (оградените). Сумата от съответните им елементи в началната матрица на цените 10+20+10+12=52 е най-малката цена на възможните назначения. Тя се получава като първият специалист се назначи на втората работа, вторият на третата, третият на четвъртата, а четвъртият на първата

5.6. ЗАДАЧИ ЗА САМОПОДГОТОВКА

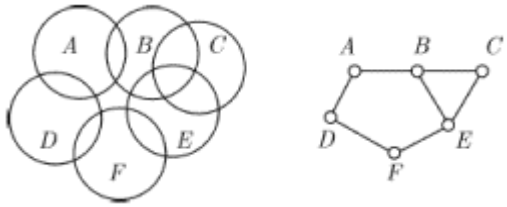
1. Задайте аналитично (чрез множество от върхове и множество от ребра) следния неориентиран граф:



2. Представете чрез дърво и ацикличен ориентиран граф следния математически израз:

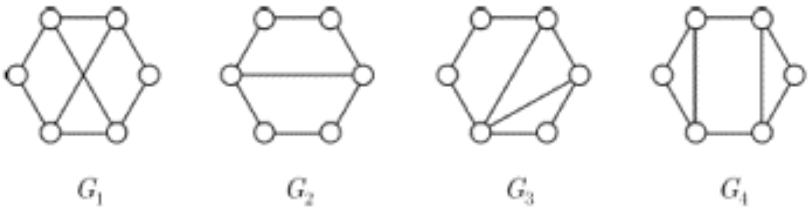
   (a + b)/(c + a* d) + ((c + a*d) - (a + b)*(c - d)

3. Може ли графът от следващата фигура да се използва за представяне на пресичащите се множества:



*На всяко семейство множества $\{S_1, S_2, \ldots S_n\}$ може да се съпостави граф на сеченията с n върха {1, 2, ... n}. Реброто (i, j) между двата върха i и j съществува само в този случай, когато $i \neq j$ и $S_i \cap S_j \neq \varnothing$.

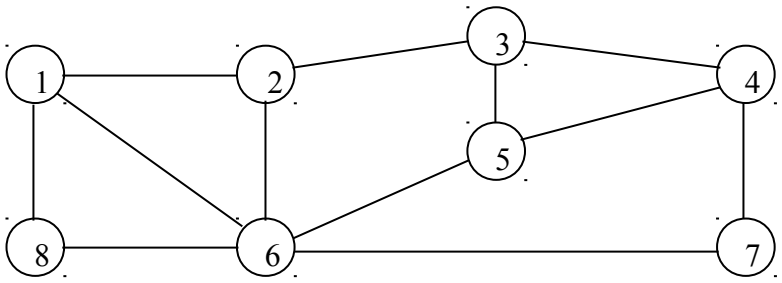
4. Изоморфни ли са следните графи:



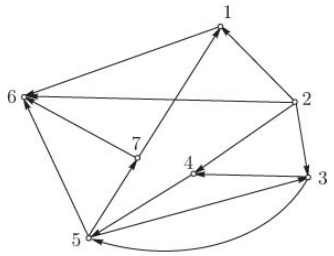
5. Нарисувайте всички неизоморфни неориентирани графи с четири върха.

6. Представете зададените графи чрез матрица на съседство.

а) неориентиран граф:



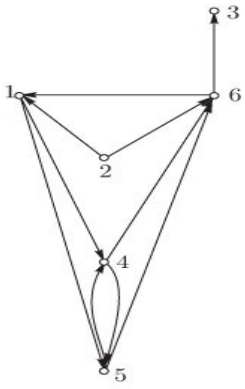
b) ориентиран граф:



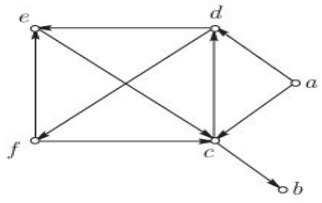
G

7. Каква е разликата между следните два графа:

8. Докажете, че G_1 и G_2 са изоморфни графи.



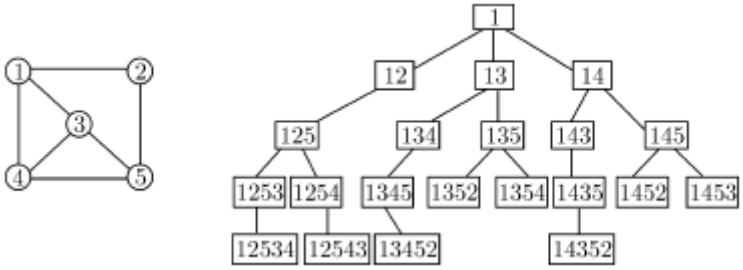
G_1



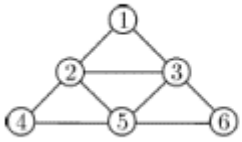
G_2

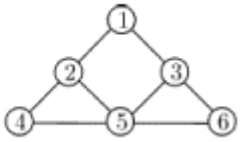
9. Постройте дърво за представяне на следната логическа формула: $((x \vee \neg y) \wedge \neg (z \rightarrow (x \wedge y))) \vee (\neg z \vee y)$.

10. Каква е връзката между графа и дървото на следващата фигура:

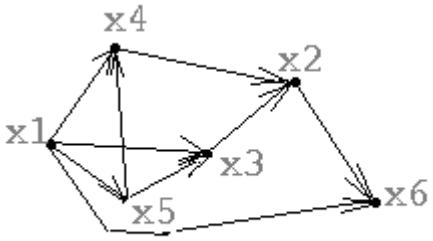


11. Съществува ли ойлеров път или ойлеров цикъл в следните графи:



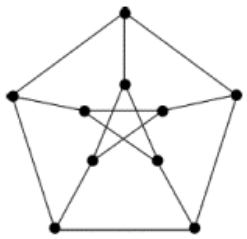


12. Решете задачата за търговския пътник като намерите маршрут на обхождане с минимална дължина. За целта използвайте изчерпващо търсене, метод на клоните и границите, метод на „най-близкия съсед". Анализирайте получените резултати, ако градовете и разстоянията между тях са зададени чрез следния граф и матрица на теглата:

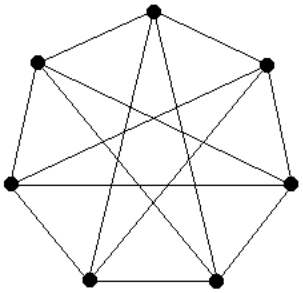


	X1	X2	X3	X4	X5	X6
X1	Б	Б	5	6	1	15
X2	Б	Б	Б	Б	Б	1
X3	Б	3	Б	Б	Б	Б
X4	Б	3	Б	Б	Б	Б
X5	Б	Б	3	1	Б	Б
X6	Б	Б	Б	Б	Б	Б

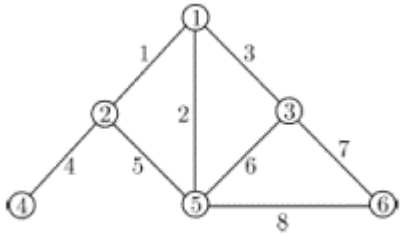
13. Регулярен ли е следващият граф, който се нарича граф на Петерсън?.



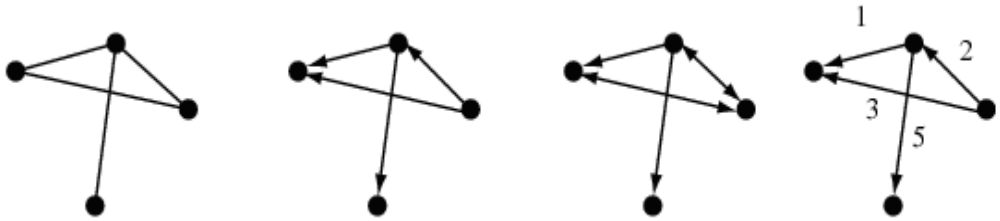
14. Ойлеров или Хамилтонов е следния граф?



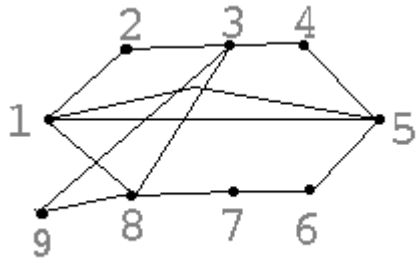
15. В какъв ред ще се обходят върховете на графа при обхождане в дълбочина и обхождане в широчина:



16. Какви видове графи са дадени на следната фигура?

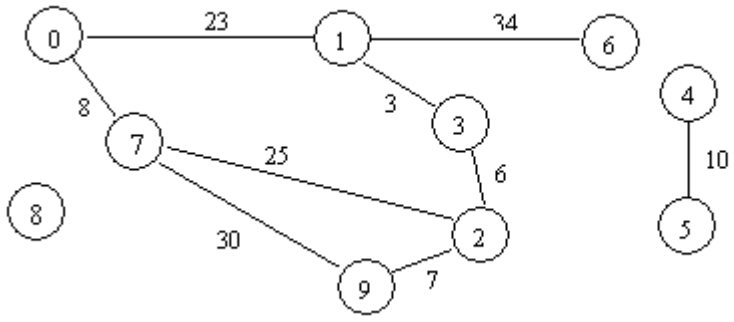


17. Намерете ойлеров цикъл в графа:



18. Опишете с необходимата терминология следните графи:

19. Приложете алгоритъма на Дейкстра за намиране на пътища с минимални дължини от върха 0 до всички останали върхове на графа:



1 – 23
2 – 32
3 – 26
4 – нямa път
5 – нямa път
6 – 57
7 – 8
8 – нямa път
9 – 38

20. Като използвате унгарския метод, намерете оптималното назначение на осем работника, които кадидатстват за осем работни места, ако на предварителните изпити са показали следните резултати:

2	5	2	1	4	6	6	3
3	3	4	3	0	3	0	5
1	6	3	5	0	2	3	6
6	4	3	4	6	3	0	0
2	3	1	1	3	3	3	0
6	5	2	4	6	4	4	3
4	6	6	1	4	1	3	5
1	5	2	6	5	6	1	0

21. Осем детайла могат да се изработят на осем различни машини. Като използвате унгарския метод, намерете такова разпределение на детайлите по машините, че сумарното време за изработка да бъде минимално. Времената в минути за изработка на детайлита са дадени в следната матрица:

12	35	17	15	24	16	31	35
23	29	14	39	15	0	30	55
61	30	33	52	61	27	0	6
41	0	53	43	0	33	12	14
0	39	19	10	13	36	20	0
67	59	28	4	56	0	24	3
19	11	41	71	0	32	40	61
31	57	0	18	35	51	44	48

глава

6

МРЕЖОВО ПЛАНИРАНЕ И УПРАВЛЕНИЕ

- **Същност и предназначение на метода**
- **Видове модели**
- **Основни понятия**
- **Метод на решение**
- **Временни характеристики на мрежовия модел**
- **Анализ на получените резултати.**
- **Графова интерпретация. Метод на критическия път**

6.1. СЪЩНОСТ И ПРЕДНАЗНАЧЕНИЕ НА МЕТОДА

Изследване на операциите е математическа дисциплина, която се занимава с изучаване, разработка и практическо приложение на методи за математически анализ на икономически ситуации на различни нива - от организация на работата на работното място до управление на крупни фирмени и корпоративни проекти. Развитието на това научно направление води до възникване на научни направления с огромен брой методи и алгоритми за решение на практически задачи в различни приложни области.

Мрежовото планиране и управление е едно от крупните постижения в научното управление и организация на труда. За първи път то се използва в САЩ в края на 50-те години, главно за военни цели - за планиране на конструирането и изработката на ракети. По-късно започва масово използване на мрежовото планиране за управление на крупномащабни проекти в различни области: в икономиката, в строителството, административно-управленската дейност и др.

Във военната област може да се използва за [9]:

- планиране на учения, проверки, конференции;
- планиране на бойна подготовка;
- моделиране и планиране на календарни планове;
- военно строителство, материално-техническо снабдяване и др.

Мрежовото планиране и управление е универсален и високоефективен метод за моделиране и изследване на операциите. Той е предназначен за количествена оценка на

целесъобразността от изпълнение на комплекс от операции, насочени към постигане на определен краен резултат. Основава се на моделирането на процеси с помощта на мрежови модели.

Всеки мащабен проект се реализира след изпълнение на определен брой етапи (операции). Някои от етапите могат да се извършват едновременно, а други само последователно.

Управлението на проекта предполага планиране на последователността на изпълнение на операциите във времето, синхронизация и отчет на времето за изпълнение на всяка операция с цел да се завърши задачата в най-кратки срокове. За целта се използват мрежов модел на проекта (използвана е терминологията от [9]).

Мрежовият модел (*график*) е удобен инструмент за анализ, оптимизация и управление на проекти. Той представлява нагледно графично изображение на комплекс от свързани помежду си операции, реализацията на които води до постигането на определена цел.

От математическа гледна точка, мрежовият модел представлява транспортна мрежа, в която са фиксирани два върха - начало и край на работата. Всеки връх на графа представя някакъв етап от изпълнението на проекта и е свързан с настъпване на някакво събитие. Дъгите съответстват на работите, а теглата им определят тяхната продължителност. Най-често се решават задачи, свързани с намирането на минималното време за завършването на проекта, т.е. пътят с най-голяма дължина между върха s (начало) и върха t (край). Този най-дълъг път се нарича ***критичен***, а етапите му определят необходимото време за реализация на проекта.

6.2. ВИДОВЕ МОДЕЛИ

Според начина за определяне продължителността на работите, мрежовите модели биват детерминирани и вероятности.

Детерминираните мрежови графици са тези, при които може сравнително точно да се определи продължителността на всяка операция. Използваните времена в мрежовия график са точни.

Вероятностните мрежови графици са тези, при които не може предварително да се определи продължителността на отделните работи. Използват се статистически оценки на времената за изпълнение на работите в проекта.

6.3. ОСНОВНИ ПОНЯТИЯ

Работа - може да има следните значения:

- ***действителна работа*** - означава дейност, трудов процес, за изпълнението на който са необходими време и ресурси;
- ***изчакване*** - процес, който изисква само време за неговото изпълнение но не и ресурси;
- ***фиктивна работа*** - не изисква нито време, нито ресурси. Използва се за показване на връзки между действителни работи. На фиктивните работи в мрежовия модел съответстват фиктивни дъги с време за изпълнение, равно на нула.

При разработване на мрежови модели действителните работи и изчакванията се изобразяват с плътни стрелки, а фиктивните - с пунктирани. Над стрелките се записва

наименованието на работата или нейния код, а под стрелките - времето, необходимо за нейното изпълнение.

Събитие - означава междинен или окончателен резултат от изпълнението на една или няколко работи, който позволява да започнат други работи. Те определят приключване на съответния етап и е добре да се изразяват чрез свършено време на глаголите, например „ изходният код на програмите е готов", „базата данни на проекта е готова", „модулите за тестване са подготвени" и т.н. Събитията се означават с кръг и се номерират последователно. Върховете на мрежовия модел представят настъпилите събития при реализирането на проекта. Събитието x е настъпило, ако всички операции, представени чрез дъги, влизащи във върха x са извършени. Всяка работа се ограничава от две събития - начално и крайно. ***Изходно събитие*** е това, от което започват, но в което не завършват работи. ***Завършващо събитие*** е това, в което завършват работи, без да започват други.

Път - означава всяка непрекъсната редица от работи, в която крайното събитие на дадена работа съвпада с началното на следващата. Дължината на пътя числено е равна на сумата от времената за изпълнение на работите.

Различаваме следните видове пътища:

- ***път, предшестващ дадено събитие*** - от изходното събитие до даденото;
- ***път, следващ даденото събитие*** - от даденото събитие до завършващото;
- ***пълен път*** - от изходното до завършващото събитие;
- ***критичен път*** - най-дългият от пълните пътища.

Работите, които принадлежат на критичния път, се наричат ***критични***. Продължителността на критичния път е най-важната

характеристика на мрежовия модел, понеже е равна на минималното време за изпълнение на целия процес.

6.4. МЕТОД НА РЕШЕНИЕ

Теоретична основа на мрежовите модели е ***методът на критическия път***. За практическата му реализация и избягване на необходимостта от използване на специална терминология от теория на графите е създаден алгоритъм, включващ последователност от действия, водещи до построяване на мрежов модел. Описанието на алгоритъма е съобразено с подготовката на хората, които се занимават с разработка на мрежови модели - ръководители на фирми, офис организатори, мениджъри, технически специалисти и др. и не изисква математическа подготовка. Основните стъпки на алгоритъма са:

6.4.1. Подготовка на необходимите данни за построяване на мрежов модел

За построяване на мрежовия модел се извършва подготовка на данните. Изготвя се списък на всички етапи, през които ще се премине по време на изпълнението на заданието.

Съставянето на плана на проекта е най-отговорната дейност в мрежовото планиране, от него зависи целият ход и крайният резултат. В заданието се посочва продължителността на етапите, мястото им в структурата на проекта и необходимите им ресурси. Данните се нанасят в таблица със следните колони:

№	Наименование на работата	Продължителност	Номер на непосредствено предшестващата работа	Ресурси

В таблицата се записват всички работи, които съставят моделирания процес. На всяка работа се присвоява номер. Мястото на всяка работа в мрежовия график се определя чрез задаване на предшестващите работи. Това са работите, които задължително трябва да са извършени, за да започне непосредствено след това изпълнението на дадената работа. Временната продължителност се дава в произволна, но еднаква за всички работи единица мярка, например в минути, дни, месеци и др., която е съобразена с продължителността на проекта.

6.4.2. Построяване на мрежовия модел

За илюстрация на метода ще използваме конкретен пример: разработването на календарен план за настъпление:

Таблица 6.1.

№	Наименование на работата	Продължи телност в минути	Номер на предшеств. работа
1	Проучване и изясняване на заданието	25	0
2	Отдаване на предварително разпореждане	20	1
3	Подготовка на доклади	90	1
4	Изслушване на доклади	50	3
5	Обявяване на решението	30	4
6	Рекогнисцировка	120	5
7	Доуточняване на заданието	30	5
8	Даване указания по взаимодействието	30	6
9	Оформяне на решението	300	8
10	Подготовка на войските за настъпление	800	2, 5
11	Материално-техническо осигуряване	800	2
12	Организиране боя на подчинените	360	2, 7
13	Докладване за готовност	10	9, 10, 11, 12

Първи етап: определяне на начални и крайни събития на работите.

На всяка работа определяме начални събития с номер, равен на номера на непосредствено предшестващите ги работи. Крайните събития номерираме с номера на работата. За да могат всички завършващи работи да се слеят в едно завършващо събитие, присвоява им се завършващо събитие с уникален номер. Обикновено ако работите са N, завършващото събитие е с номер N+1.

Описаните дейности за определянето на началните и крайните събития имат математическо обяснение. Липсата на цикли в мрежовия график дава възможност да се номерират събитията (върховете) с числата 1, 2, 3... Този процес се нарича топологическа сортировка на върховете.

Таблица 6.2.

№	*No. на началното събитие*	*No на крайното събитие*
1	0	1
2	1	2
3	1	3
4	3	4
5	4	5
6	5	6
7	5	7
8	6	8
9	8	9
10	2, 5	10
11	2	11
12	2, 7	12
13	9, 10, 11, 12	14

Втори етап: обединяване на събитията.

Някои работи имат повече от едно начално събитие. Такива са работите 10, 12, 13. Необходимо е всяка работа да има

единствено начално събитие. Това може да се постигне или чрез обединяване на събитията, или чрез въвеждане на допълнителни събития и фиктивни работи. Където не е възможно да се обединят събитията, въвеждат се фиктивни работи.

Обединяване на събитията е възможно, ако същите не са начални събития за други работи. Тогава те могат да се обединят в едно събитие с номер - номера на едно от тях. Обединяването се означава със зачертаване на номерата на излишните събития.

В разглеждания пример само работа 13 отговаря на условието за обединяване.

Трети етап: въвеждане на допълнителни събития и фиктивни работи.

Извършва се след обединяване на всички възможни събития, ако са останали още работи с повече от едно начално събитие.

Събитието, което е начално само на дадената работа (ако има такова), остава единствено начално събитие. Ако няма такова събитие, поставяме допълнително такова за начално. Останалите начални събития се свързват с първото посредством фиктивни работи. Всяка фиктивна работа получава следващия номер от реда на работите.

В нашия пример правилото за въвеждане на фиктивна работа се прилага за работа 12. Събитие 7 не е начално за никоя друга работа, затова става начално за работа 12. Събитие 2 се свързва със събитие 7 чрез фиктивна работа със следващ номер 14. При работа 10 , тъй като и двете начални събития са начални и за други работи, начално събитие става допълнителното събитие 15.

Таблица 6.3.

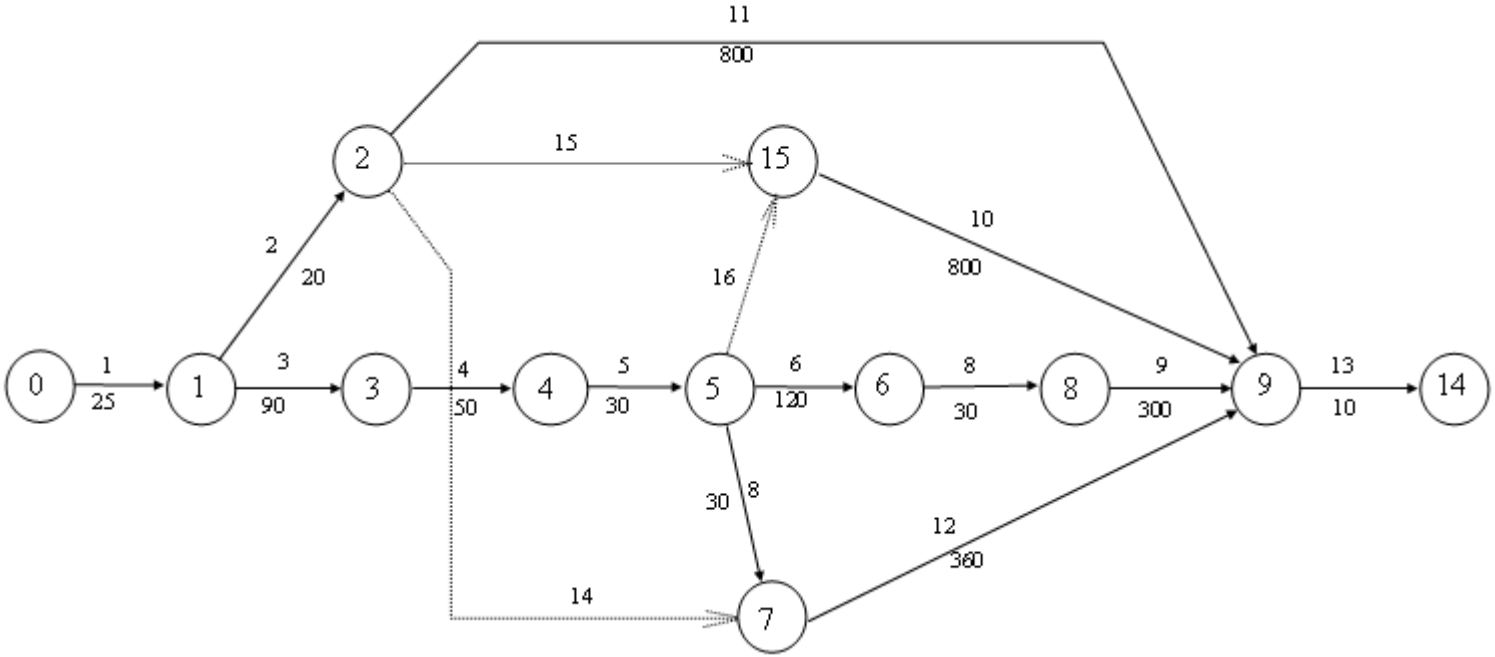
№	**No. на начално събитие**	**No. на крайно събитие**
1	0	1
2	1	2
3	1	3
4	3	4
5	4	5
6	5	6
7	5	7
8	6	8
9	8	9
10	2, 5 15	9
11	2	9
12	2, 7	9
13	9, 10, 11, 12	14
14	2	7
15	2	15
16	5	15

След тази дейност се чертае следната таблица, по която се съставя мрежовия модел. Определят се началния (***Нач.събитие***) и крайния връх (***Кр. събитие***) за всяка дъга на мрежата (***No на работа***).

Таблица 6. 4.

Нач.събитие	0	1	1	3	4	5	6	8	15	2	7	9	2	2	5
No на работа	1	2	3	4	5	6	8	9	10	11	12	13	14	15	16
Кр. Събитие	1	2	3	4	5	6	8	9	9	9	9	14	7	15	15

Мрежовият модел на задачата се представя чрез следната транспортна мрежа (фиг. 1):



Фиг. 6.1. Мрежов модел

6.5. ВРЕМЕННИ ХАРАКТЕРИСТИКИ НА МРЕЖОВИЯ МОДЕЛ

Основните характеристики на мрежовия график се определят чрез времената за настъпване на събитията t_i и продължителността на работите, включени в заданието. Те са:

- време за ранно настъпване на събитието t_P
- време за късно настъпване на събитието t_K
- продължителност на критичния път Ткр
- резерви от време.

За нагледност, тези параметри след изчисляване се нанасят в таблица. Възможно е нанасянето им директно в мрежовия график. За целта всички кръгчета на събития се разделят на четири сектора чрез два диагонала. В горния сектор се записва номера на събитието, в левия и десен сектор съответно времето за ранно и късно настъпване, в долния сектор- резерва от време на събитието.

Времето за ранното настъпване на дадено събитие е равно на дължината на максималния път, който го предшества.

$$t_P(i) = \max\{ L(0,i) \}$$

Ако са известни времената за ранно настъпване $t_P(i)$ за началните събития на всички работи (i,j), предшестващи даденото събите j, то ранното настъпване на събитието може да се определи по формулата:

$$t_P(i) = \max\{ t_P(i)+t(i,j) \}$$

Ранните срокове за настъпване на събитията се пресмятат от ляво на дясно, започвайки от началото. Срокът за завършване на

завършващото събитие е и срок за завършване на целия процес, т.е. това е продължителността на критичния път Ткр. В нашия пример :

$t_P(8) + t(8, 9) = 345 + 300 = 645$

$t_P(15) + t(15, 9)=195 + 800 = 995$

$t_P(2) + t(2, 9) =45 + 800 = 845$

$t_P(7) + t(7, 9) =225 + 360 = 585$

$t_P(9) = \max(645, 995, 845, 585)=995$

$Tкр= t_P(14) = t_P(9) + t(9,14) = 995 +10 = 1005$ min

Следващ етап е пресмятане на късните срокове за настъпване на събитията. ***Късният срок*** $\mathbf{t_K(i)}$ е равен на разликата между дължината на критичния път и дължината на максималния път, следващ събитието.

$t_K(i) = Tкр - \max\{L(i)\}$

Ако са известни късните настъпвания на крайните събития (j) на работите, следващи непосредствено събитието (i), тогава времето за късно настъпване на събитието (i) може да се изчисли по формулата:

$t_K(i) = \min\{ t_K(j) - t(i, j) \}$

Късните срокове се изчисляват от дясно наляво, започвайки със завършващото събитие.

$t_K(6) - t(5, 6) = 665 - 120 = 545$

$t_K(7) - t(5, 7) = 635 - 30 =605$

$t_K(15) - t(5, 15) = 195 - 0 = 195$

$t_K(9) = \min(545, 605, 195) = 195$

Следва изчисляване на резерва от време за събитията.

Резерв от време на дадено събитие се нарича разликата между най-късното и най-ранното време за настъпване на това събитие.

$R(j) = t_K(j) - t_P(j)$

Събитията, лежащи на критичния път, имат резерв нула. Данните за събитията се нанасят в следната таблица:

Таблица 6.5.

Събитие	**$t_P(j)$**	**$t_K(j)$**	**R(j)**
1	25	25	0
2	45	195	150
3	115	115	0
4	165	165	0
5	195	195	0
6	315	665	350
7	225	665	310
8	345	695	350
9	995	995	0

Следващ етап е изчисляване на резерва от време за работите.

Пълен резерв от време на дадена работа се нарича разликата между късният край и ранното начало на работата и продължителността на работата:

$Rп(i, j) = t_K(j) - t_P(i) - t_{ij}$

Изчислените параметри на мрежовия график се нанасят в следната таблица:

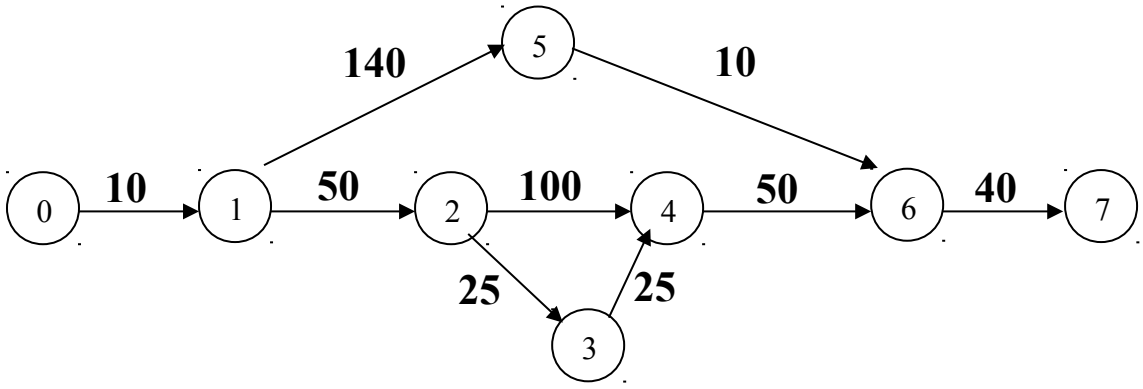
Таблица 6.6.

Работа	Продължителност	Ранно		Късно		Пълен резерв
		Начало	Край	Начало	Край	
1	25	0	25	0	25	0
2	20	25	45	25	195	150
3	90	25	115	25	115	0
4	50	115	165	115	165	0
5	30	165	195	165	195	0
6	120	195	315	195	665	350
7	30	195	225	195	635	410
8	30	315	345	665	695	350
9	30	345	995	695	995	350
10	800	195	995	195	995	0
11	800	45	995	195	995	150
12	360	225	995	635	995	410
13	10	995	1005	995	1005	0

Операция, на която пълният резерв от време е нула се нарича ***критична операция***. Това е операция при изпълнението, на която всяко закъснение води до закъснение на целия проект.

Тези операции, които не са критични, допускат закъснения, не превишаващи пълния резерв, без това да влияе върху пълния срок за изпълнение на проекта. Всяка операция от най-дългия път между началното и крайното събитие на мрежовия модел се явява критична.

Пример 6.1.



От мрежовия график се вижда, че за дейностите (1, 5) и (5, 6) пътят {0, 1, 5, 6, 7} е единствен, на който те лежат. Следователно резервът на този път е равен на пълния резерв от време за тези дейности.

R(1, 5) = R (5, 6) = Ткр - L = 250 - 200 = 50

По аналогичен начин за дейностите (2, 3) и (3, 4), лежащи на пътя {0, 1, 2, 3, 4, 6, 7} се получава пълен резерв от време

R(2, 3) = R(3, 4) = Ткр - L = 250 - 200 = 50

Следователно, всички четири некритични дейности имат един и същ пълен резерв от време, равен на 50.

6.6. АНАЛИЗ НА ПОЛУЧЕНИТЕ РЕЗУЛТАТИ. ОПТИМИЗИРАНЕ ПО ВРЕМЕ И ПО РЕСУРСИ

Главната цел на съставянето на мрежовия модел е да се определят възможностите на поставения проект и да се открият неговите недостатъци преди реализацията му, което ще даде възможност да се анализират и евентуално да се отстранят. Затова след съставянето му и изчисляване на неговите параметри се прави анализ на полученото решение, сравняват се резултатите със

заповядания срок и наличните ресурси, за да се постигне оптимален план.

6.6.1. Оптимизация по време

При анализа на мрежовия модел по време има три възможности за развитие:

1. Полученото време за изпълнение на операцията съвпада с установения срок или има минимални различия. В този случай планът е оптимален и може да се пристъпи към неговото изпълнение.

2. Полученото време е по-кратко от заповядания срок. Тогава има две възможности:

2.1. Дейността е почти самостоятелна и всяко минимизиране на срока е благоприятно. Тогава плана също е оптимален и се изпълнява;

2.2. Операцията е част от по-голяма и сроковете трябва да се спазват с точност. Налага се да се удължи критичния път на мрежовия график.

3. Полученото време е по-дълго от срок, с който разполагаме. Тогава трябва да се съкрати критичния път. Това налага големи промени в целия мрежов модел. Има няколко начина за съкращаване на критичния път:

3.1. чрез дейности, целящи съкращаване на времето за изпълнение на отделните работи, които лежат на критичния път;

3.2. чрез дейности, изменящи конструкцията на мрежовия график като паралелно изпълнение на операции, които преди това са били изпълнени последователно.

6.6.2. Оптимизация по ресурси

Анализът на мрежовия модел по ресурси се налага, за да се определят необходимите материално-технически средства за реализацията на проекта.

Съставя се линейна диаграма на дейността. Тя служи не само за оптимизиране на ресурсите, но и за непосредствено управление на целия процес.

Диаграмата се начертава върху координатната система, като всяка дейност от линейната диаграма се нанася чрез права.

За графическо представяне на плана за работа често се използва диаграмата на Гант (*Gantt),* показана на таблица (6.7)

Таблица 6.7.

Под линейната диаграма се начертава диаграмата на необходимите ресурси.

Оптимизирането по ресурси се осъществява по някои от следните начини:

- разместват се некритичните дейности за сметка на пълният им резерв от време, за да се компенсират ресурсите;

- проверява се дали има възможност да се намалят ресурсите за некритичните ситуации за сметка на удължаване на времето в рамките на техния резерв;
- търси се начин да се увеличи запасът от ресурси;
- разглежда се възможността да се прекрати изпълнението на някои некритични дейности за определен период от време;
- ако нито един от тези методи не е успешен, критичните операции се разместват, което удължава общият срок за изпълнение на проекта.

И като демонстрация на казаното, ето какви съвети към мениджърите за действие в критични за организацията ситуации са включени във вътрешния правилник на крупна западна компания (http://www.intuit.ru/courses.html):

- преразпределете работите, лежащи на критическия път така, че да се изпълняват от по-опитни членове на колектива;
- увеличете екипа с временни сътрудници;
- упростете изискванията към работите;
- включете средства за автоматизация на дейностите;
- организирайте допълнително техническо обучение на екипа;
- организирайте работата на смени;
- осигурете спокойствие на екипа, изключете възможността за прекъсване на работата им;
- препланирайте целия проект, като намалите броя на работите, лежащи на критическия път;

6.7. ГРАФОВА ИНТЕРПРЕТАЦИЯ. МЕТОД НА КРИТИЧЕСКИЯ ПЪТ

Задачата за управление проекти на мрежовото планиране се свежда до задача за намиране на най-дълъг път между два върха в граф. В общия случай задачата за търсене на най-дълъг път в цикличен граф е NP-пълна задача и единствен метод за решение е изчерпващо търсене. Липсата на цикли в ациклични графи дава възможност за реализация на по-ефективни решения.

За модел използваме претеглен ориентиран граф, в който върховете са етапите, а ребрата задачите. Търсеното минимално време ще бъде равно на дължината на максималният (измежду всички възможни) път в графа.

Алгоритъмът за решение [19] се основава на техниката ***динамично оптимиране*** и се състои в следното:

За даден граф G(V, E) с n върха, нека A [N] [N] е матрица на теглата на G. Теглата се определят от времето за изпълнение на операциите.

1. За получаване на максималните разстояния до всеки връх в графа използваме масив maxDist[ ], с начални стойности на елементите нула. В процеса на работа maxDist[i] ще бъде равно на дължината на максималния маршрут с край върха i.

2. Разглеждаме връх i $\in$ V, от който не излизат ребра. Такъв връх задължително съществува, тъй като графът е ацикличен. Премахваме този връх от графа и за всеки негов предшественик k присвояваме:

maxDist[k] = max {maxDist[k], maxDist[i]+A[k][i] }

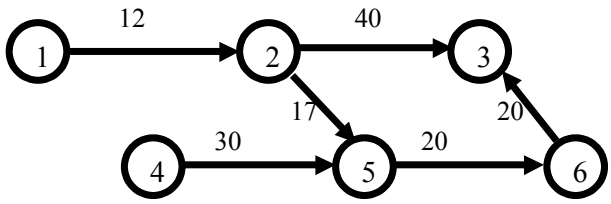
3. Повтаряме действията в стъпка (2) до пълно изчерпване на всички върхове в графа. Тогава максималната стойност на

maxDist[i], за i=1…n ще бъде дължината на търсеният максимален път в графа.

За съхранение на пътя е необходим втори масив.

Алгоритъмът има сложност O(m), а при представяне на графа с матрица на съседство — O(n2).

Примерна реализация на алгоритъма [19] е дадена на фигура (6.2). На фигурата може да се види реда, в който се отстраняват върховете от графа и как се променят стойностите на maxDist след съответните отстранявания.




```
1. Инициализираме maxDist[i]=0, за всяко i.
2. Изтриваме връх 3 =>
       maxDist[6] = max{ maxDist[6], maxDist[3]+20} = max{0, 20} =20;
       Аналогично, maxDist[2] = 40
3. Изтриваме връх 6 =>  maxDist[5] = 40;
4. Изтриваме връх 5 =>
       maxDist[4] = 70,  maxDist[2] = max{ 40, 40+17} = 57
5. Изтриваме връх 4
6. Изтриваме връх 2 =>  maxDist[1] = 69
7. Изтриваме връх 1

Най-дълъг път : 70 (maxDist[4])
```

Фиг.6.2. Алгоритъм за търсене на критичен път в граф.

Заключение:

Използването на метода е ефективно при разработка на мащабни проекти, включващи много дейности, които се нуждаят от съгласуване. Предимства на мрежовото планиране са:

- Използва се мрежов модел, който определя взаимовръзката между отделните етапи и дава нагледна представа за тяхната последователност;

- Определяне на продължителността на отделните операции и на проекта като цяло;
- Разпределяне на времето и ресурсите при планирането и реализацията на всеки етап;
- Извършва се моделиране и симулация на дадения процес, след което се отстраняват откритите слабости;
- Допуска компютърна реализация.

Поради големия интерес към мрежовото планиране, свързан с необходимостта от управление на големи проекти, много софтуерни фирми разработват програмни продукти за автоматизиране на построяването на мрежови модели. Български приложен продукт, използван в планирането и управлението е софтуерният пакет Team@Work - система, която изготвя бизнес планове, дефинира работни процеси и разпределя задачите в реално време.

Универсалността на метода го прави широко използван за оптимизиране на дейностите при решение на задачи в различни приложни области.

6.8. ЗАДАЧИ ЗА САМОПОДГОТОВКА

1. Основите на една сграда се състоят от 4 секции. Технологичните операции за изграждането на всяка секция са:

   - изкопаване на легло за основата;
   - монтаж на металната конструкция;
   - изливане на бетона.

Изкопните работи могат да се извършват само на една секция едновременно. Същото се отнася за изливането на бетон. Съставете мрежов модел.

2. Да се състави мрежов модел на програма за социологическо проучване. Дейностите, необходимото за тях време и зависимостите им са следните:

   - изработване на анкети - 7 дни;
   - обучение на сътрудници - 2 дни;
   - избор на анкетираните лица - 4 дни;
   - изпращане на анкети на тези лица - 3 дни;
   - анализ на получените резултати - 2 дни.

   Дейността 4 не може да започне преди дейността 3, която пък не може да започне преди дейността 2. Дейността 5 не може да започне преди дейността 4.

   Да се намери най-краткия срок за изпълнение на програмата. отг.: 12 дни

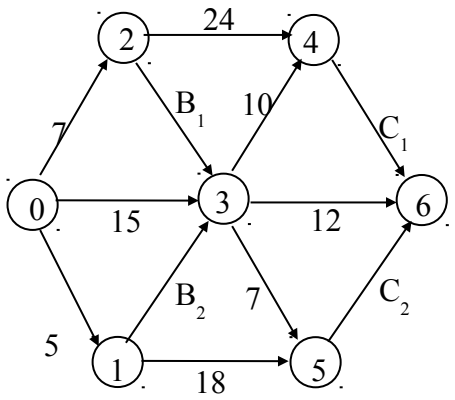
3. Намерете каква продължителност трябва да имат работите (2, 3), (1, 3), (4, 6), (5, 6), за да бъде изпълнена задачата във възможно най- кратък срок , като се спазят условията, наложени от технологични ограничения:

   $B_1+B_2=18$;

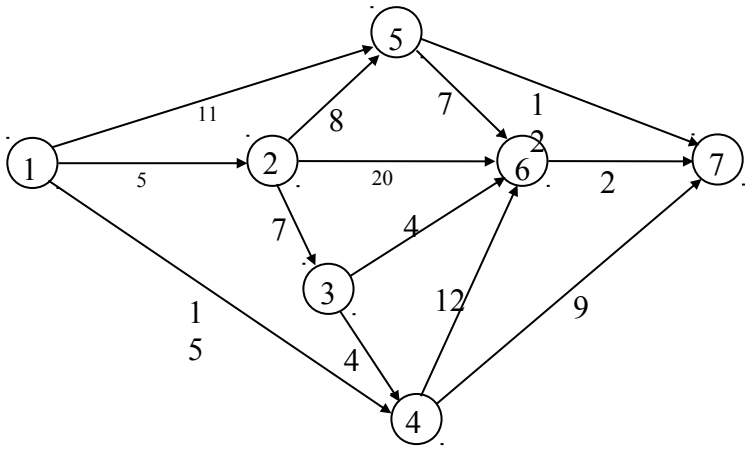
   $C_1+C_2=28$;

   отг.: $B_1=8$, $B_2=10$

   $C_1=10$, $C_2=18$.

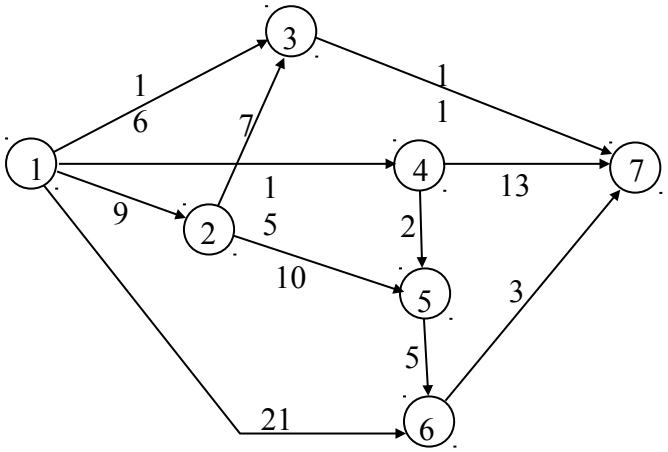


4. Определете дължината на критичния път в мрежовия план:



отг.: $T_{кр} = 30$.

5. Намерете максималният от резервите от време за настъпване на събитията от мрежовият план, даден в задача (4). отг.: max{ R(j) } = 5
6. Определете критичния път и всички параметри на следния мрежов модел:



отг.: $T_{кр}=28$.

7. Постройте мрежов модел, който включва операциите от A до L и изобразява следните отношения на наредба:

   1. A, B и C са начални операции, които могат да започнат едновременно.
   2. A и B предшестват D.
   3. B предшества E, F, H.
   4. F и C предшестват G.

5. C и H предшестват I и J.

6. C, D, F, J предшестват K.

7. K предшества L.

8. I, G, L са завършващи операции.

Продължителността на операциите е: t(A) = 3; t(B) = 2; t(C) = 2; t(D) = 5; t(E) = 0.5; t(F) = 0.5; t(G) = 4; t(H) = 0.5; t(I) = 10; t(J) = 10; t(K) = 3; t(L) = 2. Намерете дължината на критичния път.

отг.: $T_{кр} = 17.5$

8. Крупна софтуерна фирма получава заявка за разработване на програмна система за мрежово планиране и управление. Мениджърите на фирмата разработват план за работа, който включва следните дейности:

- анализ на задачата;
- оценка на необходимите ресурси (програмисти, проектанти, компютри);
- формализация и построяване на математически модел;
- избор на метод за решение
- разработка на алгоритъм;
- програмиране;
- разработка на тестващи модули;
- тестване на програмната система;
- анализ и отстраняване на грешките;
- подготовка на документация;
- оценка на цената на разработката;
- сформиране на екип;

- закупуване на оборудване;

Съставете мрежов модел на проекта, като за целта анализирайте последователността на работите, възможностите за паралелно изпълнение, направете прогноза за времето, необходимо за извършване на дейностите. Пресметнете временните характеристики на получения мрежов график.

9. В следната таблица е дадена продължителността в дни на работите, включени в даден проект. Известно е, че събитията A, B, C започват едновременно, а Z е край на проекта. Да се състави мрежов модел и да се определи времето за изпълнение на проекта.

Работа	***AD***	***BE***	***CE***	***CF***	***DG***	***EG***	***DH***	***FI***	***GZ***	***HZ***	***IZ***
Време	***6***	***9***	***8***	***8***	***6***	***4***	***5***	***3***	***5***	***6***	***5***

След включване на още една бригада, работата BE е съкратена с 1 ден. Ще повлияе ли това върху продължителността на проекта?

ЛИТЕРАТУРА

1. Алексеев В.Е.,Таланов В.А. Графы и алгоритмы, 2006. http://www.intuit.ru/department/algorithms/gaa/

2. Алтунин А.Е., Семухин М.В. Модели и алгоритмы принятия решений в нечетких условиях: Монография. Тюмень: Издательство Тюменского государственного университета, 2000.

3. Гэри М., Джонсон Д. Вычислителные машины и труднорешаемые задачи. Москва, Мир 1982

4. Гисин В.Б. Лекции по дискретной математике. Часть 1. М, МГУ, 2004, http://subscribe.ru/archive/lit.book.library.poleznosti/200601/26095541.html

5. Гисин В.Б. Лекции по дискретной математике. Часть 2. М, МГУ, 2004

6. Дехтярь М.Й. Основы дискретной математики, 2007. http://www.intuit.ru/department/ds/discrmath/

7. Дюбоа Д., Прад А. Теория возможностей. Москва, Радио и связь, 1990

8. Евстигнеев В.А., В.Н.Касьянов. Толковый словарь по теории графов. Для студентов Новосибирского Государственного Университета, http://pco.iis.nsk.su/grapp/index.html

9. Изследване на операциите. Теория, военни примери и задачи. Под редакцията на М. Узунов. София, ДВИ,1975

10. Казимиров Н.И. Лекции по аксиоматической теории множеств. Учебное пособие. Интернет ресурс (pdf), 2000

11. Костюкова Н.И. Графы и их применение, 2006. http://www.intuit.ru/department/algorithms/graphsuse/

12. Костюкова Н.И. Комбинаторные алгоритмы для программистов, 2006. http://www.intuit.ru/department/algorithms/algocombi/

13. Кофман А. Введение в прикладную комбинаторику. Москва, Наука, 1975

14. Кофман А. Введение в теорию нечетких множеств. Москва, Радио и связь, 1982.

15. Кристофидес Н. Теория графов. Алгоритмический подход. Москва, Наука, 1978

16. Липский В., Комбинаторика для программистов. Москва, Мир, 1988

17. Люгер, Джордж, Ф. Искусственный интеллект: стратегии и методы решения сложных проблем, 4-е издание. Пер. с англ. М.: Издательский дом 'Вильяме', 2003
http://filosof.historic.ru/books/item/f00/s00/z0000716/index.shtml

18. Мелехина Т.Л. Математика. Учебное пособие. http://www.studlib.net/down/r-24.html

19. Наков П., Добриков П. Програмиране=++Алгоритми; София, TopTeam Co, 2003

20. Нечеткие множества в системах управления. /Под ред. Ю.Н.Золотухина. Курс лекций в Новосибирском Государственном Университете, 2000, http//www.idisys.iae.nsk/.

21. Носов В.А. Специальные главы дискретной математики. М, Наука, 2002.

22. Носов В.А. Комбинаторика и теория графов. Учебное пособие, 2005. http://intsys.msu.ru/staff/vnosov/combgraph.zip.

23. Рэйнгольд Э., Нивергельт Ю., Део Н. Комбинаторные алгоритмы. Теория и практика. Москва, Мир, 1980

24. Сачков В. Н. Введение в комбинаторные методы дискретной математики. Москва, Наука, 1982

25. Свами М., Тхуласираман К. Графы, сети и алгоритмы. Москва, Мир 1984

26. Харари Ф. Теория графов. М, Мир, 1973

27. Черняк Д. Дискретная математика. Курс лекций для студентов МГУ им. М.В.Ломоносова, 2007, http://crow.academy.ru/dm/lectures.htm

28. Шень А.Х., Верещагин Н.К. Введение в теорию множеств. http://www.intuit.ru/department/ds/theorysets/

29. Яхъяева Г.Э. Основы теории нечетких множеств, 2006. http://www.intuit.ru/department/ds/fuzzysets/

30. An Annotated List of Selected NP-complete Problems http://www.csc.liv.ac.uk/~ped/teachadmin/COMP202/annotated_np.html

31. Beifang Chen. Course on Discrete Mathematics, The Hong Kong University of science and Technology, 2004, http://www.math.ust.hk/~mabfchen/Math132/.

32. http://www.internet-academy.org.ge/CC2001/, Рекомендации по преподаванию информатики в университетах, русский перевод.

33. www.computer.org/education/cc2001/, Computing Curricula Series, Final Report, IEEE Computer Society, Association for Computing Machinery.

34. http://www.computer.org/portal/cms_docs_ieeecs/ieeecs/education/cc2001/CC2005-March06Final.pdf, Computing Curricula 2005: The Overview Report covering undergraduate degree programs in Computer Engineering, Computer Science, Information Systems, Information Technology, Software Engineering, The Association for Computing Machinery (ACM), The Association for Information Systems (AIS), The Computer Society (IEEE-CS).

35. http://www.wolfram.com/index.html Wolfram Demonstrations Project

ДИСКРЕТНИ СТРУКТУРИ

ISBN 978 - 954 - 9681 - 32 - 1

Автор: Маргарита Владимирова Василева

Рецензенти: доц. д-р Начо Борисов Василев
доц. д-р инж. Станимир Стоянов Станев

Националност българска
Първо издание
Формат 60x90/16
Печатни коли
Издателски коли
Тираж 150