Primjene matematike u informatici

Ivanišević, Željka

Sveučilište Jurja Dobrile u Puli

Fakultet informatike

ŽELJKA IVANIŠEVIĆ

PRIMJENE MATEMATIKE U INFORMATICI

Završni rad

Pula, rujan, 2020. godine

Sveučilište Jurja Dobrile u Puli

Fakultet informatike

ŽELJKA IVANIŠEVIĆ

PRIMJENE MATEMATIKE U INFORMATICI

Završni rad

JMBAG: 0114027781, izvanredni student

Studijski smjer: Informatika

Predmet: Matematika za informatičare 1

Znanstveno područje: Društvene znanosti

Znanstveno polje: Informacijske i komunikacijske znanosti

Znanstvena grana: Informacijski sustavi i informatologija

Mentor: izv. prof. dr. sc. Danijela Rabar

Pula, rujan, 2020. godine

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani _____, kandidat za prvostupnika _____ovime izjavljujem da je ovaj Završni rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

_____

U Puli, _____, _____ godine

IZJAVA o korištenju autorskog djela

Ja, _____ dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj završni rad pod nazivom

_____

_____

_____ koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama. Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, _____

Potpis

_____

Sadržaj

1. Uvod

Matematika i informatika su dvije različite znanosti, no ipak imaju nekih poveznica. Matematika se bavi raznim temama kao što su teorija brojeva, algebra, geometrija, matematička analiza itd. Razvoj matematike započeo je 3 tisuće godina prije nove ere kada su Babilonci i Egipćani započeli koristiti matematiku za izračun poreza, gradnju i astronomiju. Matematičke vještine bile su potrebne i drugim narodima pa su je i oni koristili. Tako su matematiku počeli koristiti i Stari Grci, Rimljani, Kinezi, a kasnije i svi drugi narodi jer bez nje ne bi bili u mogućnosti izgraditi cestu, vodovod ili bilo što drugo. Matematika se danas ne veže samo za tehničke znanosti, nego je i ostalim znanostima kao što su medicina, ekonomija, informatika omogućila razvoj.

O informatici se počinje govoriti tek u prošlom stoljeću te kao znanost se još uvijek smatra mladom. Ona se bavi informacijama, postupcima i načinima obrade podataka te primjenom računala. Riječ „informatik“ prvi je počeo koristiti Karl Steinbuch, a značila je automatsku obradu podataka. Dobio ju je spojivši dvije riječi, a to su informacija i automatika. Iako je jako mlada, informatika se jako brzo razvija i postaje dio svakodnevnice. Zahvaljujući razvoju informatike informacije se brže prenose, preciznije obrađuju i lakše skladište.

U današnjem dobu matematika i informatika mnogo surađuju. Kao i u razdoblju prije nove ere, pa tako i danas, za bilo kakvu gradnju potrebna je matematika. Međutim uporabom informatike, sve informacije potrebne za gradnju neke građevine brže će se obraditi i gradnja će biti brža. Ovakvom suradnjom matematike i informatike svakodnevni život je brži i dinamičniji, nego prije.

Matematikom se u znanosti pokušava složeniji problem prikazati na jednostavniji način. Za tu svrhu koriste se matematički modeli koji prikazuju proces na razumljiviji način te se olakšava računanje, istraživanje i predviđanje.

Kao i u drugim područjima, tako i u informatici matematika se koristi za jednostavniji prikaz problema. Takav jednostavniji problem pokušava se riješiti te se potom njegovo rješenje preslika u realni svijet.

Iako se matematika na više mjesta primjenjuje u informatici, u ovom radu bit će opisano samo nekoliko primjena.

U drugom poglavlju navedeni su primjeri matrica i matričnih transformacija u digitalnoj obradi fotografije. Opisano je kako se boje fotografije mogu predočiti pomoću matrica te osnovne obrade fotografije.

U trećem poglavlju objašnjeni su algoritmi. Oni se danas češće vezuju uz programiranje, no izvorno potječu iz matematike.

U četvrtom poglavlju predstavljena je teorija grafova, objašnjeni su pojmovi šetnje, ciklusa, grafa i stabla te njihove primjene u informatici.

U petom poglavlju opisane su teorija informacija i kodiranja, a u šestom poglavlju navedeni su još neki dijelovi matematike koji su važni za informatiku, kao što je diskretna matematika i transformacije.

Zadnje, sedmo poglavlje je ujedno i zaključak ovoga rada.

2. Primjena matrica i matričnih transformacija u digitalnoj obradi fotografije

Obrada fotografije podrazumijeva izradu bilo kakvih promjena na fotografiji u svrhu poboljšanja njezinih vizualnih informacija. Nekada su se fotografije ponajviše obrađivale analogno, upotrebom raznih optičkih uređaja. Međutim, napretkom tehnologije i računala, fotografije se sve više počinju obrađivati na digitalan način koji je jeftiniji, brži, precizniji i pouzdaniji način obrade fotografije. Takva obrada vrši se pomoću raznih programa kao što su GIMP, Adobe Photoshop ili Paint.NET. Digitalna fotografija prikazuje dvodimenzionalnu fotografiju koja se sastoji od konačnog skupa piksela. Oni su najmanji grafički element fotografije te mogu prikazivati samo jednu boju. Broj piksela određuje kvalitetu fotografije, veličinu datoteke u memoriji te stvarnu veličinu fotografije.

Dvodimenzionalna fotografija može se prikazati pomoću matrice. „Matrica je pravokutna tablica sačinjena od nekoliko redaka i stupaca ispunjenih njezinim elementima. Ti su elementi obično brojevi, najčešće realni, no ponekad i kompleksni. Elementi mogu biti i drugi objekti, poput funkcija, vektora, diferencijalnih operatora, pa čak i samih matrica.“ (Elezović, 1995., 1. str.).

Primjer matrice A:

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

Matrica A ima tri retka i dva stupca. Ako se želi prikazati samo jedan element neke matrice koriste se indeksi s time da se prvo navede redak pa zatim stupac, na primjer u prvom retku i drugom stupcu nalazi se broj 2 te to ovako zapisuje $a_{12} = 2$.

„Opći oblik matrice je

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & & & \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$$

Tako je a_{11} element matrice A koji leži u prvom retku i prvom stupcu.“ (Elezović, 1995., 2. str.).

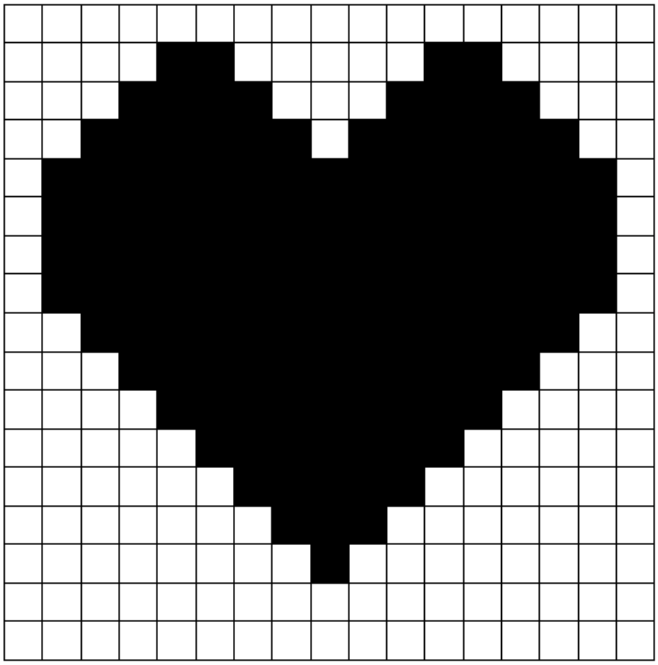
2.1 Binarna fotografija

Na fotografiji svaki element matrice predstavlja jedan piksel. Najjednostavniji oblik digitalne fotografije je binarna slika koja se sastoji od samo dvije boje, crne i bijele. Tada elementi matrice imaju vrijednosti 0 ili 1 (slika 1), gdje 0 predstavlja crnu, a 1 bijelu boju. Ovaj oblik slike najčešće se koristi za pohranjivanje teksta ili arhitektonskih planova.

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	0	0	1	1	1	1	1	0	0	1	1	1	1
1	1	1	0	0	0	0	1	1	1	0	0	0	0	1	1	1
1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1
1	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1
1	1	1	1	1	0	0	0	0	0	0	0	1	1	1	1	1
1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1
1	1	1	1	1	1	1	0	0	0	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Slika 1: Primjer binarne slike gdje su boje prikazane sa 1-bijela i 0-crna, izradio autor prema Jurković, 9. stranica prezentacije, dostupno na: https://www.fer.unizg.hr/_download/repository/Primjena_matricnih_transformacija_u_obradi_slike_Jurkovic.pdf [preuzeto: 1.7.2020.]

Kad se jedinice zamjene s bijelom bojom, a nule s crnom, fotografija izgleda kao na idućoj slici.



Slika 2: Primjer binarne slike koja se sastoji od bijele i crne boje, izradio autor prema Jurković, 9. stranica prezentacije, dostupno na: https://www.fer.unizg.hr/_download/repository/Primjena_matricnih_transformacija_u_obradi_slike_Jurkovic.pdf [preuzeto: 1.7.2020.]

2.2 Okretanje fotografije

Transponiranjem matrice koja predstavlja fotografiju postiže se okretanje fotografije. Transponiranje matrice vrši se na način da se svi elementi prvog retka neke matrice napišu na mjesto prvoga stupca transponirane matrice. Isti postupak ponavlja se i za sve ostale retke polazne matrice. Poredak elemenata u retku se ne smije mijenjati. „Ako je A kvadratna matrica, tad se transponirana matrica dobiva tako da se elementi matrice A zrcale s obzirom na njezinu dijagonalu. Transponiranu matricu

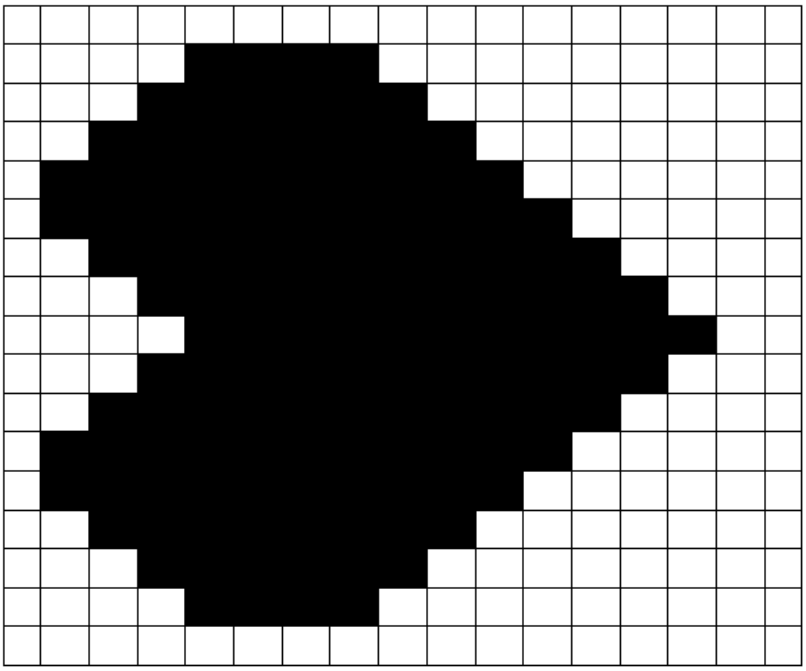
označavamo simbolom A^T." (Elezović, 1995., 4. str.). Matrica koja je bila tipa $m \times n$ transponiranjem postaje tipa $n \times m$.

Primjer transponiranja matrice:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \rightarrow A^T = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	0	0	0	0	1	1	1	1	1	1	1	1	1
1	1	1	0	0	0	0	0	0	1	1	1	1	1	1	1	1
1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1
1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1
1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
1	1	1	0	0	0	0	0	0	1	1	1	1	1	1	1	1
1	1	1	1	0	0	0	0	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Slika 3: Transponirana matrica koja predstavlja okrenutu binarnu fotografiju iz ranijeg primjera, izradio autor



Slika 4: Primjer okrenute binarne fotografije, izradio autor

2.3 Fotografije u sivim tonovima

Svaki pojedini element matrice definira intenzitet odgovarajućeg piksela. Taj intenzitet može biti u rasponu vrijednosti od 0 što predstavlja crnu pa do 255 što je bijela boja. Sveukupno postoji 256 (2^8) drugačijih nijansi sive boje. Da bi se piksel mogao prikazati potreban je 1 bajt (8 bita). Slike u sivim tonovima najčešće se koriste u medicini.

2.4 Izmjena svjetline

Izmjenom svjetline fotografije, ona postaje tamnija ili svjetlija. Ta izmjena može se dobiti množenjem ili dijeljenjem matrice koja predstavlja fotografiju skalarom. Množenjem matrice skalarom slika postaje svjetlija, a dijeljenjem tamnija. Množenje matrica skalarom vrši se na način da se svaki element iz matrice pomnoži skalarom. Na taj način svaki piksel u fotografiji postaje svjetlije nijanse.

Primjer množenja matrice skalarom:

$$3 * \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} = \begin{bmatrix} 3 & 6 & 9 \\ 12 & 15 & 18 \\ 21 & 24 & 27 \end{bmatrix}$$

2.5 Fotografije u boji

Za prikaz jedne fotografije u boji više nije dovoljna jedna matrica, nego je potrebno koristiti tri različite matrice koje predstavljaju intenzitete crvene, zelene i plave boje. Tada se piksel sastoji od tri podpiksela koji prikazuje različite intenzitete tih boja. Zbroj tih triju matrica daje nijansu boje koju čovjek vidi na fotografiji. „Zbrajanje matrica definirano je na način $(A + B)_{ij} := (A)_{ij} + (B)_{ij}$." (Elezović, 1995., 6. str.). Što znači da se element u prvom retku i prvom stupcu jedne matrice zbraja sa elementom u prvom retku i prvom stupcu druge matrice te se isti postupak odvija sa svim ostalim elementima. „Da bi zbroj matrica bio definiran, matrice A i B moraju biti istoga tipa. Rezultat zbrajanja je opet matrica, istoga tipa (m, n)." (Elezović, 1995., 6. str.).

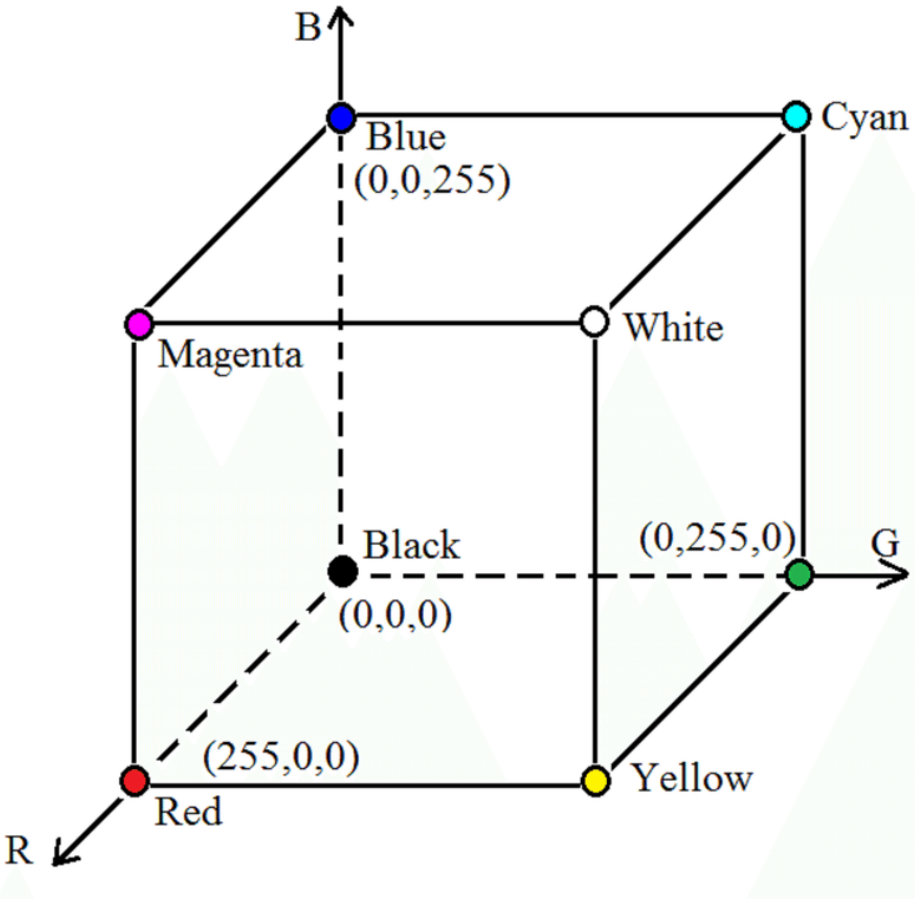
Primjer zbrajanja matrica:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} + \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} + \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 3 & 6 \\ 9 & 12 \end{bmatrix}$$

Ovakvo prikazivanje fotografije još naziva i RGB modelom. Ime mu potječe od početnih slova triju boja (Red-crvena, Blue-plava, Green-zelena) čiji zbroj različitih nijansi daje boju koja se vidi na fotografiji. U tom slučaju svaki element unutar svake matrice bit će u rasponu vrijednosti od 0 do 255. Iz ovoga dobivamo da postoji 16777216 (256^3) mogućih nijansi boja. Kod RGB modela za prikaz jednog piksela potrebno je 24 ($3 \cdot 8$) bita. Ovo je i razlog zašto fotografije u boji zauzimaju više memorije od sivih fotografija.

RGB model boja definiran je i pomoću Kartezijeva koordinatnog sustava koji se još naziva pravokutnim ili pravokutnim Kartezijevim sustavom. U prostoru je prikazan pomoću tri okomita pravca (x, y, z) koji se sijeku u ishodištu O. Kod RGB modela u

ishodištu će se smjestiti crna boja, a u vrhu najudaljenijem od ishodišta je bijela boja. U vrhovima najbližim ishodištu su crvena, plava i zelena boja, a u vrhovima najbližim vrhu koji predstavlja bijelu boju su žuta, magenta i cijan boja. Prostorna dijagonala koja povezuje ishodište i najudaljeniji vrh od ishodišta predstavlja razinu sivog.



Slika 5: Prikaz RGB modela boja pomoću Kartezijeva koordinatnog sustava, izvor: https://www.researchgate.net/figure/Cartesian-coordinate-system-of-RGB-color-space_fig1_322688596 [preuzeto 1.7.2020.]

3. Algoritmi

„Algoritam je striktno definirani postupak koji daje rješenje nekog problema. Iako se pojam algoritam koristi intenzivno u posljednje vrijeme, odnosno od pojave računala, on je nastao mnogo ranije, prvenstveno u matematici. Prvi značajni algoritam, koji se i danas koristi, nastao je još u 3. st. prije naše ere. Taj je algoritam poznat kao Euklidov algoritam jer ga je prvi opisao grčki matematičar Euklid u svojoj knjizi „Elementi“. Sam naziv dolazi od imena iranskog matematičara Abu Ja'far Mohammed ibn Musa al-Khowarizmi koji je 825. godine napisao knjigu pod nazivom „Pravila restoracije i redukcije.“ (Divjak, Lovrenčić, 2005., 125. str.).

Američki znanstvenik Donald Ervin Knuth (1938.) zbog velikih doprinosa u razvijanju i sistematiziranju matematičkih tehnika za analizu složenijih računalnih algoritama smatra se 'ocem algoritama'. Opisao je pet svojstava koja algoritam mora imati da bi se smatrao algoritmom. Ta svojstva se po njemu zovu Knuthova svojstva.

Knuthova svojstva:

1. „Konačnost – Algoritam mora završiti nakon izvršenih konačno mnogo koraka. Također, algoritam mora biti opisan pomoću konačnog broja operacija, a i svaka operacija mora biti konačne duljine.
2. Definitnost – Svaki korak algoritma mora sadržavati nedvosmislene, rigorozno definirane operacije.
3. Ulaz – Svaki algoritam mora imati 0 ili više ulaza.
4. Izlaz – Svaki algoritam mora imati 1 ili više izlaza.
5. Efektivnost – Algoritam mora biti efektivan, tj. mora biti takav da se može izvesti samo uz pomoć olovke i papira u konačno vrijeme.“ (Divjak, Lovrenčić, 2005., 125. str.).

Za algoritam je potrebno utvrditi da li je ispravan, to jest formalno dokazati da je algoritam korektan. Ono što je potrebno napraviti je dokazati da će za određeni ulaz koji zadovoljava preduvjete algoritam vratiti rješenje nakon konačnog broja koraka, a vraćeno rješenje mora biti ono što se od algoritma traži da izračuna.

Pod pojmom algoritam smatra se i postupak koji se koristi za rješavanje masovnog problema. Masovnim problemom smatra se problem za koji treba naći rješenje te taj problem ima parametre koji su na početku rješavanja problema neodređeni.

„Pri definiciji masovnog problema potrebno je zadati:

1. generalni opis svih parametara
2. ograničenja koja rješenje masovnog problema mora zadovoljavati“ (Divjak, Lovrenčić, 2005., 126. str.).

Instanca problema dobiva se posije specificiranja svih njegovih parametara. Algoritam rješava neki masovni problem tek onda kada izbacuje rješenje za svaku instancu tog masovnog problema.

Algoritam se može zapisati u prirodnom govornom jeziku, pseudojeziku ili programskom jeziku.

3.1 Euklidov algoritam

Euklidov algoritam smatra se prvim značajnijim algoritmom te je ime dobio po svom autoru Euklidu. Taj algoritam računa najveću zajedničku mjeru dva prirodna broja. Najveća zajednička mjera označava se velikim slovom M, a to je najveći cijeli broj kojim se mogu podijeliti svi elementi nekoga skupa.

Primjer: Najveća zajednička mjera skupa brojeva 21, 28, 42 i 70 jest 7. Ne postoji cijeli broj veći od 7, a da se svi brojevi iz navedenog skupa mogu podijeliti njime bez ostatka.

Euklidov algoritam ima pet jednostavnih koraka. Kao ulazne parametre potrebno je unijeti dva prirodna broja a i b. Kao izlaz algoritam nam vraća najveću zajedničku mjeru $M\ (a, b)$ brojeva a i b. U prvom koraku provjerava se da li je broj a manji od b i, ako je manji, a dobiva vrijednost b te b dobiva vrijednost a. U suprotnom, ta zamjena se ne odvija. U drugom koraku r dobiva vrijednost cjelobrojnog ostatka dijeljenja brojeva a i b ($r = a\ mod\ b$). U trećem koraku a dobiva vrijednost b te b dobiva

vrijednost r. Potom se u četvrtom koraku provjerava da li je r različit od nule i, ako je, potrebno se vratiti na drugi korak. U suprotnom, prelazi se na sljedeći korak. Zadnji peti korak vraća vrijednost parametra a, što je ujedno i najveća zajednička mjera unesenih brojeva.

„Ulaz: Dva prirodna broja a i b.

Izlaz: Najveća zajednička mjera $M\ (a, b)$ brojeva a i b.

1. Ako je $a < b$ zamijeni a i b.
2. $r := a\ mod\ b$.
3. $a := b,\ b := r$.
4. Ako je $r \neq 0$ idi na 2.
5. Vrati a.“ (Divjak, Lovrenčić, 2005., 126. str.).

3.2 Složenost algoritama

Za isti problem može se napraviti više različitih algoritama koji će ga riješiti pa se dolazi do problema u kojem treba odlučiti koji će se algoritam koristiti. Određuje se mjera kakvoće algoritama te se na temelju te mjere odabire algoritam koji je bolji. Cilj je utvrditi kojem algoritmu treba manje resursa da riješi problem. Kod rješavanja problema algoritmi troše resurse vremena i prostora. Kod izračuna potrošnje vremena potrebno je saznati vrijeme utrošeno za izvršavanje algoritma te se ta mjera naziva vremenska složenost algoritama. Kod izračuna potrošnje prostora potrebno je saznati koliki prostor je zauzet za pohranjivanje ulaznih, izlaznih i međurezultata. Ova mjera naziva se prostorna složenost algoritama. Mjera vremenske složenosti algoritama češće se koristi za ocjenjivanje kakvoće algoritma.

Izračun složenosti algoritama radi se jer se želi poboljšati već postojeći algoritam ili pronaći najbolji algoritam za rješavanje nekog problema.

Postoje dvije vrste analize algoritama: a priori i a posteriori. Kod a priori analize procjenjuje se vrijeme izvođenja algoritama koje ne ovisi o upotrijebljenim računalima, programskim jezicima ili prevoditeljima. U ovom slučaju vrijeme izvođenja je vrijednost neke funkcije. Kod a posteriori analize dobiva se realno vrijeme koje je potrošeno za izvođenje algoritama na računalu. Rezultat mjerenja su statistički podaci koji se dobivaju mjerenjem na računalu.

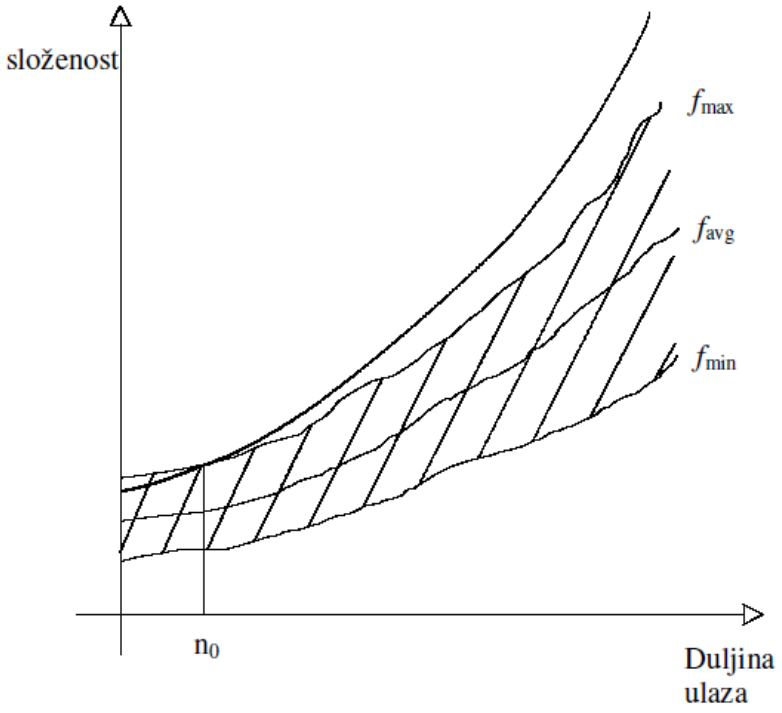
3.2.1 Vremenska složenost

Vrijeme koje algoritam utroši da dođe do rješenja mnogo ovisi o brzini računala na kojem se izvršava algoritam. Ako se želi usporediti vrijeme trajanja izvođenja dva ili više algoritama, dobivena mjera bit će dobra za usporedbu samo ako su se algoritmi izvodili na istom računalu. No, takva dobivena mjera nije korisna. Kod vremenske složenosti potrebno je izbjeći mjeru koja ovisi o računalu, odnosno brzini računala na kojem se izvodi algoritam, nego ta mjera mora ovisiti o samom algoritmu. „Stoga se vremenska složenost ne izražava vremenom potrebnim da algoritam izračuna rezultat, već brojem elementarnih operacija koje algoritam mora izvesti da bi izračunao rješenje.

Sljedeći problem je što broj operacija koje izvodi algoritam nije fiksan, već ovisi o veličini ulaza. Dakle, instance problema različitih dimenzija zahtijevat će različit broj elementarnih operacija za obradu. Međutim, masovni problem može imati i različite instance iste duljine, čija vremenska složenost ne mora biti jednaka. Dakle, složenost algoritma nije funkcija veličine ulaza.“ (Divjak, Lovrenčić, 2005., 131. str.).

Iz toga se može zaključiti da ocjena složenosti algoritma ovisi o instancama problema. Promatraju se tri moguće ocjene složenosti: u najgorem, u najboljem i u prosječnom slučaju. Najčešće se izračunava i koristi složenost najgoreg slučaja. To je instanca problema kod koje algoritam za njezino izvršavanje potroši najviše vremena. Sljedeća složenost koja se može izračunati, no nije od velike koristi jest složenost najboljeg slučaja. Kod te instance problema algoritam potroši najmanje vremena za njezino izvođenje. Ocjena složenosti koja je najsloženija za izračunati,

no i najbolje opisuje algoritam je ocjena složenosti prosječnog slučaja koja prikazuje kako se u prosjeku algoritam ponaša. Složenosti najgoreg i najboljeg slučaja odnose se na samo jednu instancu problema. U složenosti prosječnog slučaja uračunavaju se sve instance problema određene duljine ulaza, pa se potom svakoj instanci dodjeljuje vjerojatnost pojavljivanja. Složenost algoritma može se prikazati kao funkcija složenosti koja ovisi o duljini ulaza, rastom duljine ulaza raste i složenost algoritma.



Graf 1: Složenost algoritama o ovisnosti o duljini ulaza

Izvor: Divjak, Lovrenčić, 2005., 131. str.

Međutim, objašnjene funkcije nisu analitičke. One ponekad i mogu biti analitičke, ali onda su vrlo složene za prikazati. „Analitička je ona funkcija koja se može prikazati formulom izgrađenom pomoću aritmetičkih operacija i transcendentnih funkcija.“ (Divjak, Lovrenčić, 2005., 132. str.).

U potpunosti točno prikazati složenost algoritama nije ni potrebno. Složenost algoritama dovoljno je samo aproksimativno prikazati zato što je teško utvrditi broj svih operacija. Najviše koristi ima od prikaza rasta složenosti algoritama u ovisnosti o porastu duljine ulaza.

Zbog tog jednostavnijeg prikaza i lakšeg razumijevanja, funkcija složenosti prikazuje se, aproksimira jednostavnijom funkcijom u kojoj se uvodi asimptotska ocjena rasta funkcije. U grafu 1 najviša, odnosno jedina analitička funkcija je ona koja predstavlja složenost algoritama te će se ta funkcija najviše koristiti u daljnjim obradama.

Pojednostavljeni prikaz funkcije složenosti radi se na način da aproksimativna funkcija mora što bolje prikazati asimptotski rast funkcije složenosti. Aproksimativna funkcija nije potpuno točna za male duljine ulaznih podataka, no to nije ni važno jer problemi sa složenošću nastaju kod većih duljina ulaznih podataka, a ondje aproksimativna funkcija dovoljno dobro opisuje stvarnu složenost algoritma. Dakle, aproksimativna funkcija daleko je jednostavnija od stvarne funkcije složenosti, a dovoljno točna da bi razumjeli složenost algoritma.

3.3 Pretraživanje i sortiranje

U današnjem radu s algoritmima česta je potreba za sortiranjem i pretraživanjem raznih nizova, odnosno dokumenata. Pretraživanjem se želi pronaći točno određeni element u nizu, a sortiranjem se želi dohvatiti sve elemente, ali oni trebaju biti poredani po nekom kriteriju. „Općenito, problem (uzlaznog) sortiranja glasi ovako. Zadana je lista duljine n, označeno je s $(a_1, a_2, \dots, a_n)$. Vrijednosti elemenata liste mogu se uspoređivati totalnim uređajem $\leq$. Te vrijednosti treba permutirati tako da nakon permutiranja vrijedi $a_1 \leq a_2 \leq \cdots \leq a_n$. Algoritam za sortiranje je bilo koji algoritam koji rješava problem sortiranja.“ (Manger, 2014., 141. str.). Kako se ove radnje puno koriste postoje razni načini za pretraživanje i sortiranje.

3.3.1 Algoritam slijednog pretraživanja

Najjednostavniji oblik pretraživanja je slijedno pretraživanje gdje „algoritam redom uspoređuje elemente liste sa zadanom vrijednošću. Ako je neki od elemenata jednak traženom, algoritam staje i odgovara da je našao traženu vrijednost. U suprotnom, algoritam prolazi kroz cijelu listu do kraja i odgovara da nije pronašao zadanu vrijednost.“ (Divjak, Lovrenčić, 2005., 139. str.).

Algoritam za ulazne vrijednosti očekuje vrijednost x koju mora pronaći te niz od a_1 do a_n koji pretražuje. Kao izlaz mora izbaciti potvrdu da je pronašao element u nizu ili obavijestiti da se traženi element ne nalazi u nizu koji mu je zadan da pretražuje.

1. „$i := 1$.
2. ako je $i > n$ onda odgovori *ne* i stani.
3. ako je $a_i = x$ odgovori *da* i stani.
4. $i := i + 1$.
5. idi na 2.“ (Divjak, Lovrenčić, 2005., 139. str.).

U prvom koraku definira se element i te mu se daje vrijednost 1. U idućem koraku provjerava se da li je vrijednost i veća od vrijednosti n koja predstavlja broj elemenata u nizu. Ako je i veći od n ispisuje se odgovor da u algoritmu nema tražene vrijednosti te se algoritam prestaje izvoditi. U trećem koraku provjerava se da li je vrijednost a_i jednaka vrijednosti x. Ako je, algoritam javlja da je pronašao traženu vrijednosti te prestaje s izvođenjem. Predzadnji, četvrti korak vrijednost i uvećava za 1, a zadnji korak vodi na korak 2.

Algoritam će se najmanje izvoditi jednom te niti neće proći kroz sve korake, odnosno stati će na koraku 3. To je u slučaju kad se traženi element nalazi na prvom mjestu. U najgorem slučaju algoritam će se izvoditi $n + 1$ put, s time da će u zadnjem prolazu kroz algoritam stati na koraku 2. Ovo se događa kad u nizu elemenata nema tražene vrijednosti.

3.3.2 Binarno pretraživanje

Binarno pretraživanje rješava se metodom podijeli pa vladaj (eng. divide and conquer) što znači da se jedan složeniji problem svodi na jedan ili više jednostavnijih problema istoga tipa, no manjih dimenzija. Rješenja manjih problema spajaju se u rješenje složenog problema.

Binarno pretraživanje može se koristiti jedino kada postoji sortirana lista elemenata od najmanjeg prema najvećem. Inače ovo pretraživanje nije upotrebljivo.

Lista elemenata mora biti već sortirana jer na taj način se omogućuje da se pretražuje manji dio liste, a ne cijela lista. Stoga, pretraživanje započinje na način da se traženi element usporedi s elementom u sredini liste. U slučaju da lista ima paran broj elemenata, onda se uspoređuje sa elementom koji je približno u sredini. Ako taj element ima vrijednost koja se traži tada pretraživanje završava. U suprotnom slučaju, potrebno je utvrditi da li taj element u sredini liste ima veću ili manju vrijednost od traženog elementa. U slučaju da je taj traženi element veći od elementa u sredini liste, tada se zanemaruje prva polovica liste, a u slučaju da je traženi element manji od elementa u sredini liste, tada se zanemaruje druga polovica liste. Potom se uzima dio liste za koji se smatra da bi se u njemu mogao nalaziti traženi element te se na tom dijelu ponovno provodi isti postupak pretraživanja. Taj postupak potrebno je provoditi dok se ne pronađe traženi element ili dok se ne dođe do liste s jednim članom gdje se provjerava da li taj element ima istu vrijednost kao traženi element. Ako nema, znači da se traženi element niti ne nalazi u listi elemenata.

Kao ulaz u algoritam binarnog pretraživanja $binsearch\ (f, l, x, a)$ potrebno je odrediti 2 kursora f i l, jedan na početku, a drugi na kraju liste, traženu vrijednost x te listu elemenata $a_1, \dots, a_n$. Kao izlaz algoritam vraća *da* ako se tražena vrijednost nalazi u listi ili *ne* ako te vrijednosti nema u listi.

1. „$p := \left\lfloor \frac{f+l}{2} \right\rfloor$.
2. Ako je $a_p = x$ vrati *da* i stani.
3. Ako je $a_p > x$ i ako je $p > f$ onda pozovi $binsearch\ (f, p-1, x, a)$.
4. Ako je $a_p < x$ i ako je $p < f$ onda pozovi $binsearch\ (p+1, l, x, a)$.
5. Vrati *ne* i stani.“ (Divjak, Lovrenčić, 2005., 141. str.).

3.3.3 Sortiranje izborom

Niz se može sortirati na više načina, a jedan od jednostavnijih načina je sortiranje izborom. Algoritam prvo pronalazi najmanji element u nizu te kad ga pronađe element na prvom mjestu i element s najmanjom vrijednošću zamijene mjesta. Na ovaj način na prvo mjestu u nizu je doveden element s najmanjom vrijednošću te se ono više ne uspoređuje s drugim brojevima u nizu. Isti postupak ponavlja se za sve ostale elemente u nizu osim za posljednjeg jer on se nema ni sa čime više uspoređivati. Stoga, u svakom prolazu algoritma kroz listu, dio koji se treba sortirati smanjuje se za jedan, što znači da će algoritam listu koja je dugačka n elemenata sortirati za $n-1$ prolaza.

Za ulaz algoritmu je potrebno navesti listu od n elemenata $(a_1, \dots, a_n)$. Kao izlaz algoritam vraća sortiranu listu s istim elementima koji su unijeti, no oni su poredani od najmanjeg ka najvećem.

1. „Za $i := 1, \dots, n-1$ radi korake 2-5.
2. $min := i$.
3. Za $j := 2, \dots, n$ radi korake 4.
4. Ako je $(a_j < a_i)$ $min := j$.
5. $swap\ (a_i, a_{min})$.“ (Divjak, Lovrenčić, 2005., 146. str.).

4. Primjena teorije grafova

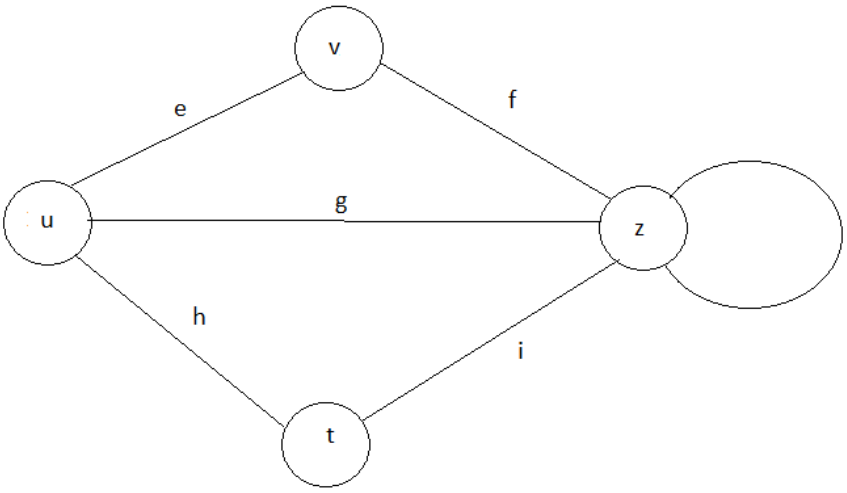
Teoriju grafova utemeljio je matematičar Leonhard Euler u 18. stoljeću. Riješio je tada kompliciran problem Königsberških mostova. U tom gradu bila su dva rukavca rijeke koja su grad dijelila na 4 dijela (3 kopnena dijela i 1 otok) te su ti dijelovi bili povezani sa 7 mostova. Problem koji je trebao riješiti je napraviti šetnju gradom koja započinje i završava u istoj točki te se točno jednom prolazi preko svih mostova. Ovaj problem prikazao je kao model koji se danas smatra grafom. 4 dijela grada koje treba povezati označio je vrhovima, a mostove bridovima. Problem je riješio tako da bi krenuo iz bilo kojeg vrha i prošao kroz svaki brid točno jednom.

4.1 Grafovi, šetnje i ciklusi

„Teorija grafova je matematička disciplina koja proučava zakonitosti na grafovima. Neusmjereni graf *(*eng. undirected graph) G je par (V, E), pri čemu je V skup vrhova (eng. vertices) grafa, a $E \subseteq V \times V$ skup neuređenih parova elemenata iz V, koji čini skup bridova (eng. edges) grafa G." (Divjak, Lovrenčić, 2005., 193. str.).

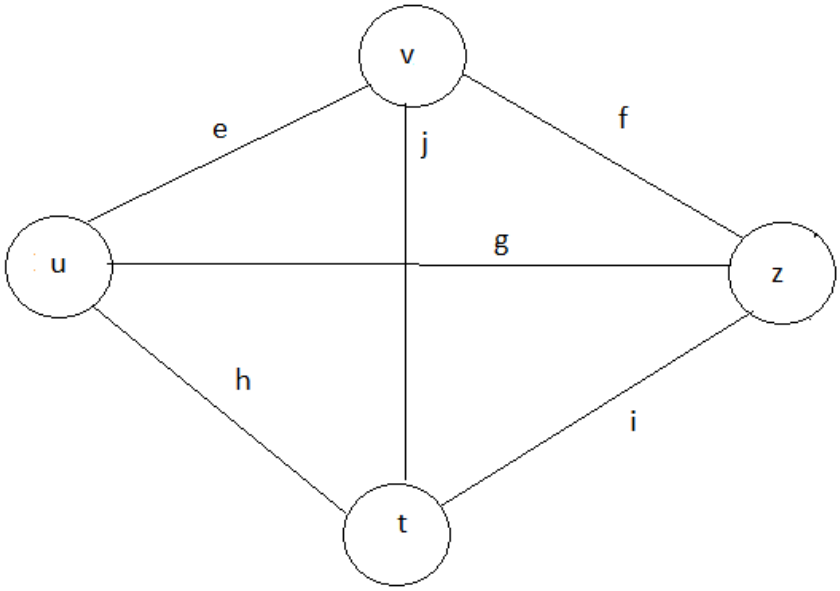
„Za par vrhova u i v kažemo da su susjedni (eng. adjacent) ako postoji brid e koji ih povezuje. Pri tome kažemo da je brid e incidentan (eng. incident) vrhovima u i v. Bridovi e i f su susjedni ako imaju zajednički barem jedan vrh.

Graf G je planaran (ravninski) graf ako se grafički može predočiti tako da se bridovi sijeku samo u vrhovima. Brid koji je incidentan samo s jednim jedinim vrhom se zove petlja. Graf G je konačan ako su oba skupa V i E konačni." (Divjak, Lovrenčić, 2005., 194. str.).



Slika 6: Prikaz grafa, izradio autor

„Graf je jednostavan ako nema petlje i ako ne postoje dva brida koji spajaju isti par vrhova. Jednostavni graf, u kojem je svaki par vrhova spojen jednim bridom, zove se potpuni graf (eng. complete graph).“ (Divjak, Lovrenčić, 2005., 194. str.).



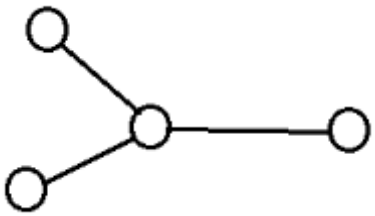
Slika 7: Prikaz potpunog grafa, izradio autor

Sam graf kao prikaz modela nekog problema često nije dovoljan za njegovo rješavanje. Za rješavanje je potrebno razumjeti putove unutar grafa.

„Šetnja u grafu G je netrivijalni konačan niz, čiji su članovi naizmjenice vrhovi i bridovi. Vrhovi su na početku i kraju šetnje. Vrhovi, koji nisu na početku ili na kraju, zovu se unutarnji vrhovi. Duljina šetnje je broj bridova u šetnji. Ako su svi vrhovi u šetnji međusobno različiti, šetnju zovemo put. Šetnja je zatvorena ako joj se podudaraju početak i kraj. Zatvorenu šetnju kod koje su početak i unutrašnji vrhovi različiti, zovemo ciklus.“ (Divjak, Lovrenčić, 2005., 200. str.).

4.2 Stabla

„Stablo (eng. tree) je povezani graf T koji ne sadrži niti jedan ciklus. Vrh se naziva listom (eng. leaf) stabla $T = (V, E)$ ako ima samo jednog susjeda. Primijetimo da stabla s najmanje dva vrha imaju najmanje dva lista.“ (Divjak, Lovrenčić, 2005., 215. str.). Stabla prvi put spominje Arthur Cayley u 19. stoljeću, a koristio ih je za prikaz kemijskih spojeva.



Slika 8: Stablo s 4 vrha, izradio autor

Postoji više vrsta stabala, a najosnovnije je uređeno stablo. „Uređeno stablo (engl. ordered tree) T je neprazni konačni skup podataka istog tipa koje zovemo čvorovi.

Pritom:

- postoji jedan istaknuti čvor r koji se zove korijen od T;
- ostali čvorovi grade konačni niz $(T_1, T_2, \dots, T_k)$ od 0, 1 ili više disjunktnih (manjih) uređenih stabala.“ (Manger, 2014., 53. str.).

„List je čvor bez djece. Unutrašnji čvor je čvor koji nije list. Djeca istog čvora su braća.“ (Manger, 2014., 57. str.).

4.3 Primjena u praksi

Razni stvarni problemi mogu se prikazati pomoću modela grafa te se rješavanje tih problema svodi na pronalaženje putova u grafovima. Često se pomoću njih prikazuju fizičke mreže, kao što su ceste, željeznice, planinarski putovi, vodovodi, plinovodi, dalekovodi itd. Grafovima je moguće prikazati zadatke u projektu, obaveze zaposlenika u poduzeću, hranidbeni lanac, karte u geografiji, socijalne i društvene odnose te hijerarhiju u sustavu itd.

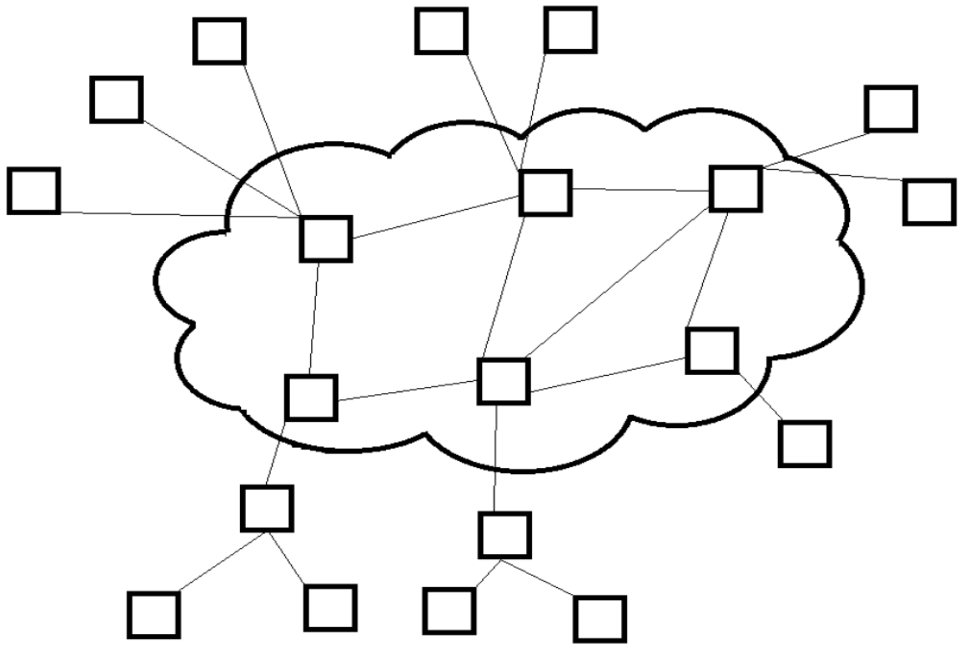
4.3.1 Prikaz računalnih mreža

U informatici grafovi se koriste za prikaz računalnih mreža. „Recimo ovako: računalna mreža je sustav koji omogućuje prijenos informacijskih sadržaja između dvaju ili više samostalnih računala.“ (Radovan, 2010., 15. str.).

„Mnogi procesi u računalnim sustavima odvijaju se prema metodi klijent-server (client-server). Klijent je sustav koji šalje neki zahtjev serveru, a server je sustav koji izvršava taj zahtjev; pritom, server obično predaje klijentu rezultat istraživanja

njegova zahtjeva, ali ne mora uvijek biti tako. Međutim, u situaciji kada neki klijent traži od servera koji upravlja bazom podataka, da izvrši upis nekih sadržaja (podataka) u bazu, onda se rezultat izvršenja tog zahtjeva ne šalje klijentu; server tada klijentu obično šalje obavijest o izvršenju tražene operacije.“ (Radovan, 2010., 10. str.).

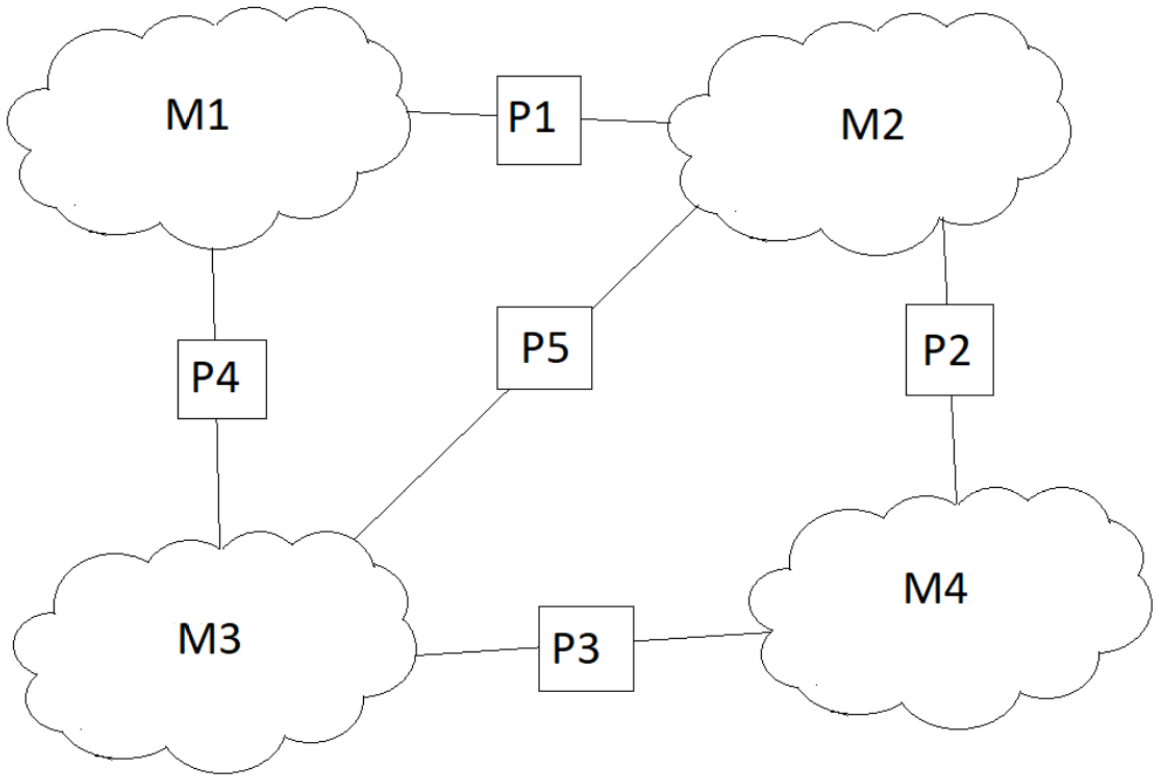
Da bi se razumjelo kako izgleda neka računalna mreža, lakše ju je prikazati pomoću grafa, nego opisivanjem riječima. Ona računala koja su uključena u neku mrežu nazivaju se čvorovima. Postoje dvije vrste čvorova, domaćini i prijenosnici. Razlikuju se u vrsti posla kojeg rade. Prijenosnici su čvorovi čija je uloga prijenos podataka u mreži. Domaćini (hosts) su računala na kojima rade serveri na koje se vezuju klijenti, odnosno korisnici sa svojim računalima. Korisnici koriste mrežne usluge koje pružaju serveri. Na slici se vidi prikazana osnovna struktura jednog mrežnog sustava. U njoj se nalaze i navedeni čvorovi.



Slika 9: Osnovna struktura mrežnog sustava, izradio autor prema Radovanu, 2010., 16. str.

Simbol oblaka se uobičajeno koristi za prikaz mreže. To može biti globalna mreža, a može biti i mala lokalna mreža. Čvorovi koji se nalaze unutar oblaka, odnosno mreže su prijenosnici, a oni izvan su domaćini. Na slici crte koje spajaju kvadrate smatraju se vezama. „Sustav čvorova (prijenosnika) i veza unutar oblaka na slici, ostvaruje prijenos sadržaja u mreži. Čvorovi izvan oblaka (domaćini) koriste taj prijenosni sustav, odnosno usluge koje on pruža. Na tim čvorovima rade razni serveri koji realiziraju razne mrežne usluge; na te čvorove (i servere) vezuju se korisnici sa svojim računalima i na taj način koriste usluge koje ti serveri pružaju.“ (Radovan, 2010., 20. str.).

Na slici je prikazan prijenos podataka pomoću prijenosnika unutar jedne mreže, no prijenosnici mogu povezivati i više različitih mreža u jednu sastavljenu mrežu. Na taj način omogućuje se komunikacija između korisnika koji pripadaju različitim mrežama.

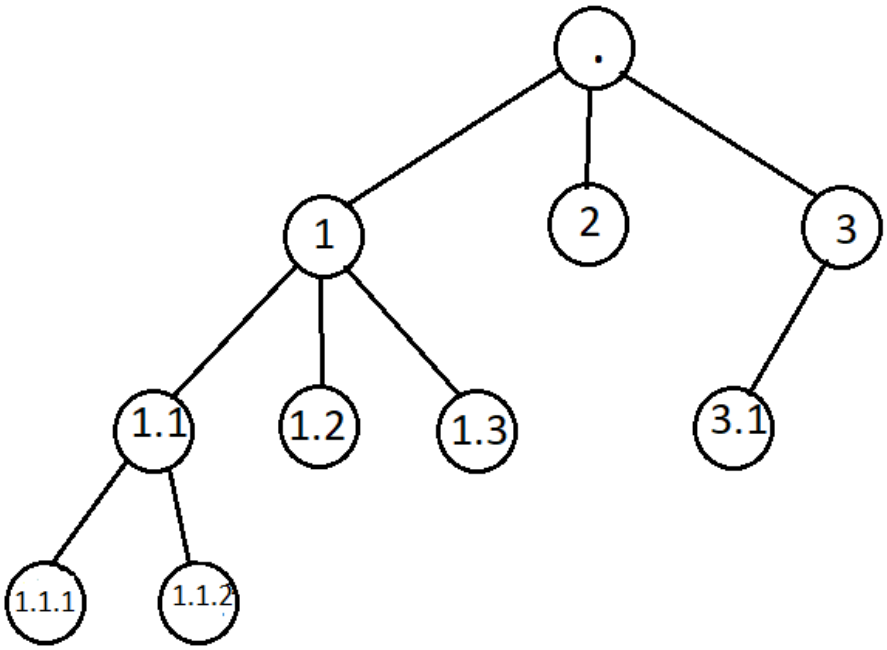


Slika 10: Povezivanje više mreža, izradio autor prema Radovanu, 2010., 21. str.

Internetwork je naziv za računalnu mrežu koja se sastoji od više različitih računalnih mreža, a kraći naziv je internet. Najveća, najstarija i najpoznatija mreža takve vrste je Internet.

4.3.2 Uređeno stablo

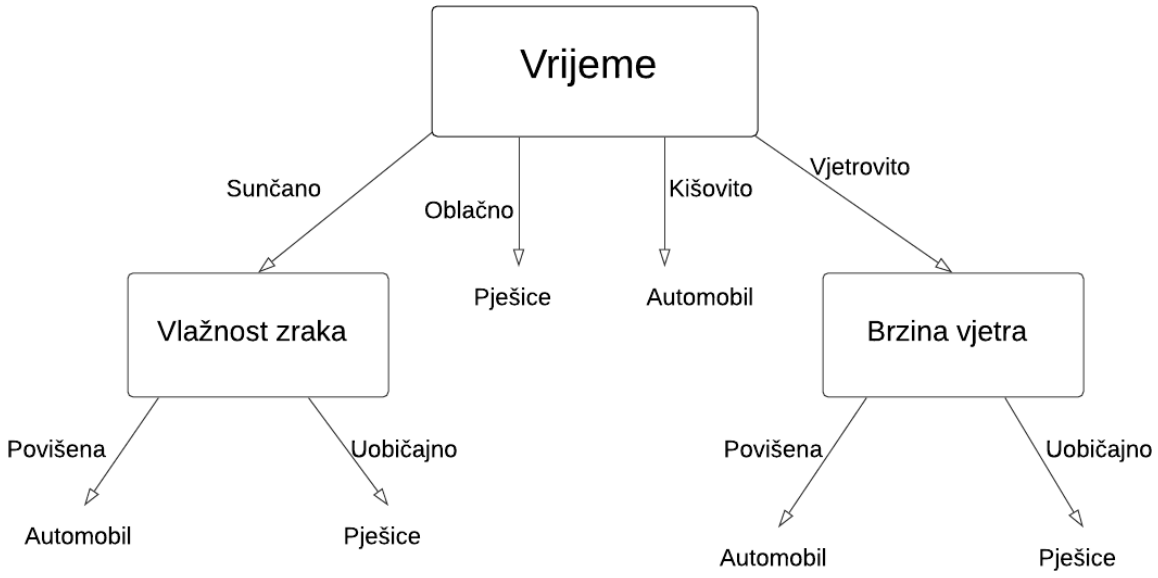
U informatici uređena stabla se koriste za spremanje podataka te se među čvorovima želi naglasiti hijerarhijski odnos, odnosno odnos „roditelj-dijete“. Za podređene podatke unutar uređenog stabla važan je redoslijed, važno je koji od njih je prvi, a koji drugi i tako redom. Korijen je jedini čvor bez roditelja, a sva djeca imaju po jednog roditelja. Na slici je prikazan sadržaj neke knjige uređen kao stablo.



Slika 11: Sadržaj knjige prikazan kao stablo, izradio autor prema Mangeru, 2014., 55. str.

Stablima se može prikazati i način odlučivanja, te se takva stabla zovu stabla odlučivanja. Pomoću prikaza jednostavnih upita, dolazi se do rješenja koje na prvu nije bilo uočljivo. Ovakva stabla se često kreiraju i u ekonomiji ili medicini gdje se s obzirom na stanje u kojem se nalazi treba postupati po smjeru koje stablo navodi. Prednost korištenja stabla odlučivanja jest da ih je lako razumjeti, čak i za nestručne osobe. Algoritmi koji grade stablo odlučivanja su ID3, CART itd.

Na slici je prikazano stablo odlučivanja za primjer odabira korištenja automobila ili odlaska pješice do određenog cilja.



Slika 12: Stablo odlučivanja, izradio autor

Kako u realnom svijetu ima puno ovakvih odnosa koji se mogu prikazati pomoću uređenog stabla javila se potreba za njihovim pohranjivanjem u računala.

Matematički pojam stabla prvo se morao pretvoriti u apstraktni tip podataka. Potom ga je bilo potrebno implementirati te definirati elementarne operacije nad njima i definirati oznake svakog čvora. Tako definirani apstraktni tip podataka može se koristiti za spremanje raznih podataka u kojima se želi naglasiti hijerarhijski odnos.

5. Teorije kodiranja i informacija

Teorija informacija i teorija kodiranja grane su matematike koje se primjenjuju u komunikaciji.

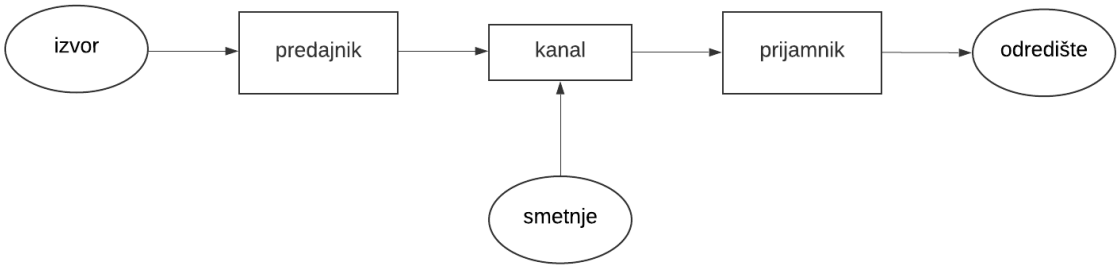
5.1 Teorija informacija

Teorija informacije bavi se poteškoćama u komunikaciji, odnosno problemima u prijenosima informacija s jednog mjesta na drugo.

„Općenito, prijenos informacije želimo izvršiti:

- što brže,
- što točnije,
- uz što manje utrošene energije,
- usprkos neizbježnim smetnjama u sustavu prijenosa,
- te ponekad uz prikrivanje i zaštitu od zlouporabe.“ (Pandžić i dr., 2007., 3. str.).

Cilj je da poruka stigne od izvora do odredišta nepromijenjena. Poslana poruka sastoji se od niza simbola abecede, a abeceda je konačan skup simbola. Na slici je prikazan opći model komunikacijskog sustava koji se sastoji od izvora, predajnika, kanala, smetnji, prijamnika i odredišta. Njega je 1948. godine predstavio Claude E. Shannon. Na izvoru nastaje poruka koja preko predajnika odlazi u komunikacijski kanal. Ondje se mogu dogoditi razne namjerno ili slučajno izazvane smetnje. Preko kanala poruka stiže do prijamnika koji poruku daje odredištu. U predajniku se poruka kodira, a u prijamniku dekodira.



Slika 13: Komunikacijski sustav, izradio autor prema Pandžić i dr., 2007., 3. str.

Kako bi se dobiveni sadržaj poruke mogao promatrati na sustavan način, treba uvesti nekakvu mjeru za količinu informacija koju primatelj poruke dobije. Uvodi se jedan bit kao osnovna jedinica informacije, odnosno količina informacije potrebna za rješavanje elementarne neodređenosti. Elementarna neodređenost smatra se jedan izbor između dvije jednako vjerojatne mogućnosti. Primjer za takvu neodređenost bilo bi bacanje novčića gdje može pasti pismo ili glava. Ako je pala glava, dobili smo informaciju o tome što je palo, odnosno primili jedan bit informacije. Isto i u slučaju da je palo pismo. Problem kod određivanja količine informacije javlja se kad postoji više različitih događaja koji mogu biti ishod, a i vjerojatnost pojavljivanja svakog od njih je drugačija. Tada je potrebna entropija za izračun količine informacije. Entropija se smatra mjerom neodređenosti informacije.

„Diskretna slučajna varijabla je varijabla odabrana iz skupa od n mogućih vrijednosti, gdje je za svaku vrijednost poznata vjerojatnost odabira. Entropija diskretne slučajne varijable je definirana kao:

$H(X) = -\sum_{i=1}^{n} p\,(x_i) log_2\, p(x_i)$ [bit/simbol],

gdje je X diskretna slučajna varijabla koja poprima vrijednosti iz skupa $\{x_1, \dots, x_i, \dots, x_n\}$, a $p(x_i)$ su vjerojatnosti pojavljivanja vrijednosti x_i, $1 \leq i \leq n$.“ (Pandžić i dr., 2007., 16. str.).

U prvom slučaju bacanja novčića gdje postoje dva ishoda s vjerojatnošću 50% entropija je $H(p_1, p_2) = -(0{,}5 \cdot log_2 0{,}5 + 0{,}5 \cdot log_2 0{,}5 = 1$ bit.

U drugom slučaju bacanja novčića kod kog je vjerojatnost prvog ishoda 10%, a drugog 90% entropija je $H(p_1, p_2) = -(0{,}1 \cdot log_2 0{,}1 + 0{,}9 \cdot log_2 0{,}9) = 0{,}47$ bita.

U trećem slučaju bacanja novčića kod kog je vjerojatnost samo jedan ishod (uvijek padne samo pismo) entropija je $H(p_1) = -(1 \cdot log_2 1) = 0$ bita. Ne postoji nikakva neodređenost jer se zna da će uvijek pasti pismo.

Vjerojatnosti pojavljivanja nekog simbola su od 0 do 1, odnosno od 0% do 100%, stoga entropija ne može biti negativni broj. Najmanja moguća entropija je 0, a događa se onda kad se u potpunosti zna koji je ishod (primjer bacanja novčića kada uvijek padne pismo). Najveća moguća entropija postiže se kada su vjerojatnosti pojave simbola jednako raspoređene. U slučaju s novčićem kad je 50% vjerojatnost pojavljivanja glave, a 50% pojavljivanja pisma, entropija je 1. Kada je vjerojatnost pojavljivanja nekog od simbola drugačija od 50%, entropija se smanjuje.

Osim prijenosom informacija, teorija informacija bavi se i kompresijom podataka bez gubitaka (teksta, govora, glazbe, videa). Bez ove mogućnosti, mnoge današnje tehnologije kao što su razgovori putem mobitela, pristup internetu ili prijam satelitske televizije ne ni bilo toliko upotrebljive kao što jest.

5.2 Teorija kodiranja

„Kodiranje je postupak dodjeljivanja kodnih riječi (kodova) simbolima poruke. Svaka se kodna riječ sastoji od jednog ili više simbola iz abecede. Dakle, kodiranjem se poruka (niz) simbola pretvara u niz kodnih riječi.“ (Pandžić i dr., 2007., 22. str.). Kodiranje se radi zato što je takvu poruku sigurnije poslati te lakše zaštititi ili pohraniti. U komunikacijskom sustavu kodiranje se odvija unutar predajnika, a dekodiranje unutar prijamnika.

Postoji više vrsta kodiranja, a neke od njih su:

- kompresija, gdje je kodirana poruka kraća od poruke nastale na izvoru,
- zaštitno kodiranje, gdje se poruci daju svojstva za otkrivanje i ispravljanje pogrešaka nastalih zbog smetnji u prijenosu poruke,
- kriptografija, gdje kodirana poruka ima sigurnosne karakteristike.

Kada se vrši kodiranje gdje se želi napraviti kompresija podataka, mora se odrediti granica do koje se može sažimati podatke bez gubitaka. Tada je ponovno potrebno koristiti entropiju.

U tablici su prikazani podaci potrebni za pojašnjenje primjera kodiranja. Neki izvor može se sastojati od ovih simbola abecede $X = \{A, B, C, D\}$. Uz simbol navedena je i vjerojatnost pojavljivanja simbola, kodna riječ za taj simbol te duljina kodne riječi.

Simbol (x_i)	Vjerojatnost pojavljivanja ($p(x_i) = p_i$)	Kodna riječ (C_i)	Duljina kodne riječi (l_i)
A	1/2	0	1
B	1/4	10	2
C	1/8	110	3
D	1/8	111	3

Tablica 1: Primjer kodiranja riječi, izradio autor prema Pandžić i dr., 2007., 22. str.

Neki niz simbola iz zadanog skupa kodiranjem se pretvara u kodnu riječ. Niz simbola ABCDA kodiranjem postaje 0101101110.

Uvrštavanjem vjerojatnosti pojavljivanja u formulu za entropiju dobiva se da je entropija 1,75.

$$H(X) = -(0{,}5 \cdot log_2 0{,}5 + 0{,}25 \cdot log_2 0{,}25 + 0{,}125 \cdot log_2 0{,}125 + 0{,}125 \cdot log_2 0{,}125) = 1{,}75$$

Prosječna duljina kodne riječi za neki niz slova dobiva se kao zbroj umnožaka vjerojatnosti pojavljivanja pojedinog simbola i duljine kodne riječi.

$L = \sum_{i=1}^{n} p_i \cdot l_i$ [bit/simbol]

U primjeru to je $L = 0{,}5 \cdot 1 + 0{,}25 \cdot 2 + 0{,}125 \cdot 3 + 0{,}125 \cdot 3 = 1{,}75$.

Za kodiranje informacije iz navedenog izvora koji se sastoji od niza slova A, B, C, D treba u prosjeku 1,75 bita po simbolu. Svaka kodna riječ koja nastane iz navedenog izvora ne može imati duljinu kodne riječi manju od 1,75 bita, a da bude jednoznačno određena.

„Općenito, može se dokazati da je nemoguće jednoznačno kodirati simbole s nekog izvora uz prosječnu duljinu koda manju od entropije izvora. Drugim riječima, entropija je granica kompresije bez gubitka. Ova temeljna činjenica ukazuje na praktičnu važnost entropije.“ (Pandžić i dr., 2007., 23. str.).

6. Ostale primjene matematike u informatici

Utjecaj matematike vidljiv je na mnogim mjestima u informatici te bi bilo teško navesti sve primjere. Istaknuta su još dva područja matematike koja imaju velik utjecaj u razvoju informatike.

6.1 Diskretna matematika

Potrebno je još spomenuti utjecaj diskretne matematike koja je grana matematike te se bavi skupovima (najčešće prirodnih i cijelih brojeva). Diskretna matematika bila je potrebna za razvoj računalne geometrije, a bez računalne geometrije ne bi bio moguć razvitak robotike i navigacije.

Osnovni pojam u diskretnoj matematici je skup, a odnosi se na kolekciju elemenata koja se promatra kao cjelina. Skup je zadan onda kada su poznati elementi toga skupa. Nakon što se skup zada dolazi se do relacija među skupovima te do operacija nad njima. Najvažnije relacije su relacija sadržavanja, jednakosti skupova te pravi podskup, a najvažnije operacije su unija, presjek i razlika skupova te komplement skupa.

Idući važni pojam za diskretnu matematiku je funkcija. „Funkcija *f* je preslikavanje (pridruživanje) elemenata između dva skupa (domene i kodomene), i to takvo da se svakom elementu prvog skupa (domene) pridružuje jedan i samo jedan element drugog skupa (kodomene).“ (Divjak, Lovrenčić, 2005., 69. str.). Praktični primjer primjene funkcija je pridruživanje svakom stanovniku Hrvatske njegovog jedinstvenog OIB-a.

Sljedeći je bitni pojam iz diskretne matematike, kojeg je potrebno navesti, diskretna teorija vjerojatnosti koja proučava vjerojatnosti događanja na konačnim skupovima događaja. Koristi se kod računanja prosječnog trajanja algoritama te kod izračuna sigurnosti sustava. Za sigurnost nekog sustava važno je znati vjerojatnost pojavljivanja nekakvog kvara te koji rizik taj kvar predstavlja za sustav. Na temelju tog

izračuna određuje se zaštita za sustav. Osnovni pojmovi koje treba razumjeti za izračun vjerojatnosti su permutacije, varijacije i kombinacije.

6.2 Transformacije

Razne transformacije se koriste kod kreiranja računalnih igara. Prvotno je potrebno razumjeti da se transformacije događaju u koordinatnom sustavu koji ima 2 osi. Međutim, za današnje igrice koje postaju sve realnije potrebno je razumjeti transformacije u koordinatnom sustavu s 3 osi. Važne transformacije su skaliranje, translacija, rotacija i zrcaljenje.

Skaliranje služi za rastezanje ili smanjivanje nekog objekta. Takvo što se dobiva na način da se svaka koordinata toga objekta pomnoži odgovarajućim koeficijentom. Kod računalnih igara koriste se realni, pozitivni koeficijenti, što bi značilo da ako se objekt skalira s 4 dobiva se objekt koji je četiri puta veći od prvotnog objekta.

Translacija se koristi kad se želi napraviti nekakav pomak. Sve točke nekog objekta pomiču se u određenom smjeru. Kod računalnih igara, translacija se koristi kada se želi objekt pomaknuti na novo mjesto na ekranu.

Rotacija je transformacija kod koje se objekt rotira oko osi, a kod zrcaljenja objekt će se simetrično preslikati preko ravnine.

Transformacije se koriste kako bi se moglo upravljati objektima unutar računalnih igara.

7. Zaključak

Matematika se koristi u mnogim znanostima te bez njezinih dostignuća ne bi bio moguć napredak raznih znanosti pa tako ni napredak informatike.

Matrice i matrične transformacije pomogle se u razvitku obrade digitalnih fotografija. Algoritmi su pojam koji se danas povezuje uz programiranje, no oni su nastali u Staroj Grčkoj te se smatraju dijelom matematike. Grafovi i stabla korisni su za jednostavniji prikaz raznih sustava. Grafom je lakše prikazati sustav računalne mreže, a stablima se može prikazati hijerarhijski odnos ili pak konstruirati stablo odlučivanja. Razvoj komunikacijske tehnologije ne bi bio moguć bez teorija kodiranja i informacija koje su izvorno grane matematike. Potrebno je spomenuti diskretnu matematiku koja pridonosi razvoju robotike i navigacije, te transformacije koje se koriste kod kreiranja računalnih igara.

Iako na prvu korist matematike nije uvijek uočljiva, dubljim promišljanjem o nekoj temi shvati se da matematika ima značajni utjecaj. Bilo bi gotovo nemoguće zamisliti današnji razvitak informatike i tehnologije bez uplitanja matematike u taj razvitak.

Literatura

Knjige:

1. Divjak B. i Lovrenčić A. (2005) Diskretna matematika s teorijom grafova. Varaždin, TIVA tiskara
2. Elezović N. (1995) Linearna algebra. Zagreb, Element, 1. izdanje
3. Manger R. (2014) Strukture podataka i algoritmi. Zagreb, Element, 1. izdanje
4. Pandžić I. S., Bažant A., Ilić Ž., Vrdoljak Z., Kos M. i Sinković V. (2007) Uvod u teoriju informacije i kodiranje. Zagreb, Element, 1. izdanje
5. Radovan M. (2010) Računalne mreže 1. Rijeka, Digital point tiskaka d.o.o.

Članci:

1. Ghosh, T., Fattah, S. A., & Wahid, K. A. (2018). CHOBS: Color histogram of block statistics for automatic bleeding detection in wireless capsule endoscopy video. *IEEE journal of translational engineering in health and medicine*, *6*, 1-12., dostupno na: https://www.researchgate.net/figure/Cartesian-coordinate-system-of-RGB-color-space_fig1_322688596 [preuzeto 1.7.2020.]

Internetski izvori:

1. Jurković I. Primjena matričnih transformacija u obradi slike. Dostupno: https://www.fer.unizg.hr/_download/repository/Primjena_matricnih_transformacija_u_obradi_slike_Jurkovic.pdf [preuzeto: 1.7.2020.]

Popis slika:

Popis grafova:

Popis tablica:

Sažetak

Cilj ovog rada je prikazati upotrebu matematike u razvoju informatike. Iako nije uvijek vidljiv doprinos matematike u razvoju raznih područja informatike, želi se naglasiti njezina važnost. Za razvoj raznih informacijskih tehnologija važno je razumijevanje matematičke logike.

U radu su navedeni samo neki od primjera primjene matematike u informatici. Matematika se upotrebljava u mnogo područja, pa nije bilo moguće sve navesti.

Prvo je opisana upotreba matrica i matričnih transformacija u obradi digitalne fotografije. Pojašnjen je prikaz crno-bijele, sive i fotografije u boji te zaokretanje i izmjena svjetline fotografije. Pomoću Kartezijeva sustava prikazan je RGB model.

Potom je naglašena važnost algoritama koji izvorno potječe iz matematike, no imaju veliku primjenu u informatici. Prikazana je složenost algoritama te algoritmi sortiranja i pretraživanja koji je često koriste.

Zatim slijedi primjena teorije grafova, opisani su grafovi, šetnje, ciklusi te stabla. Grafovi se mogu koristiti za prikaz računalne mreže, a stabla za prikaz i spremanje podataka u hijerarhijskom odnosu te za prikaz stabla odlučivanja.

Objašnjeno je korištenje teorija informacije i kodiranja za ostvarivanje komunikacije, a posljednje su navedene važnosti diskretne matematike i transformacija.

Ključne riječi: matematika, informatika, matrice, obrada fotografije, algoritmi, teorija grafova, računalne mreže, stabla, teorija informacija, teorija kodiranja, diskretna matematika, transformacije

Abstract

The aim of this paper is to show the use of mathematics in the development of information technologies. Although the contribution of mathematics in the development of various fields of informatics is not always recognized, its importance is emphasized. Understanding mathematical logic is important for the development of various information technologies.

This paper presents only some of the examples of the application of mathematics in informatics. Mathematics is used in many areas, so it is not possible to list them all.

The use of matrices and matrix transformations in digital photography processing is first described. The display of black and white, grey and colour photographs is explained, as well as the rotation and brightness change of the photograph. The RGB model is shown using the Cartesian system.

Next, the importance of algorithms that originally derived from mathematics, but have great application in computer science, is emphasized. The complexity of the algorithms and the sorting and search algorithms that often use it are presented.

Then follows the application of graph theory, and graphs, walks, cycles and trees are described. Graphs can be used to display a computer network, and trees to display and store data in a hierarchical relationship, and to display a decision tree.

The use of information and coding theories to achieve communication is explained, and finally the importance of discrete mathematics and transformations is stated.

Keywords: mathematics, informatics, matrix, photographic processing, algorithms, graph theory, computer networks, trees, information theory, coding theory, discrete mathematics, transformation