UNIVERZITET METROPOLITAN BEOGRAD

CS215 - DISKRETNE STRUKTURE

NIZOVI, SUME I ALGORITMI

Lekcija 06

PRIRUČNIK ZA STUDENTE

CS215 - DISKRETNE STRUKTURE

Lekcija 06

NIZOVI, SUME I ALGORITMI

UVOD

Fokus ove lekcije jesu algoritmi i njihova konstrukcija

U ovoj lekciji biće obradene sledeće teme:

- Nizovi
- Rekurzivno definisane relacije
- Faktorijel broja
- Fibonačijev niz
- Numerički algoritmi

Biće definisan pojam algoritma i nekih važnih osnovnih pojmova. Student će biti upoznat sa načinom računanja vrednosti polinoma u nekoj zadatoj tački, određivanja NZD i NZS za dva broja.

Poglavlje 1

NIZOVI

DEFINICIJA NIZA

Nizovi i indeksirane klase skupova su specijani tipovi funkcija. Oni se često pojavljuju u algoritmima

Niz je diskretna struktura koja se koristi za predstavljanje uređene liste. Na primer, 1, 2, 3, 5, 8 je niz sa pet članova, a 1, 3, 9, 27, 81, . . . , 3^n, . . . je beskonačan niz.

Koristimo notaciju {a $_n$} da bismo opisali niz. (Napomena da "a $_n$" predstavlja pojedinačni član niza {a $_n$}. Važno je napomenuti da notacija {a $_n$} za niz može da bude slična notaciji za skup, ali će kontekst uvek jasno ukazati da li se radi o skupu ili nizu. Takođe, iako smo koristili slovo "a" u notaciji za niz, za razmatrani niz se može koristiti druga slova ili izrazi. Izbor slova "a" je proizvoljan.) Nizove opisujemo tako što nabrajamo članove niza u redosledu rastućih indeksa.
Nizovi mogu biti jednodimenzionalni, višedimenzionalni i ugnježdeni (nizovi niza). Oznaka an se koristi da označi sliku celog broja n. Niz se obično označava sa

$$a_1, a_2, \ldots.$$

$$ili$$

$$(a_n)n \in \mathbb{N}$$

$$ili$$

$$a_n$$

Nekad se umesto N za domen niza uzima skup {0, 1, 2, . . . } ne-negativnih celih brojeva. U tom slučaju kažemo da n počinje od 0, a ne od 1, kao u slučaju N. Konačan niz nad skupom A je funkcija iz {1, 2, . . . , m} u A, i obično se obeležava sa

$$a_1, a_2, \ldots a_m$$

$$ili$$

$$(a_1, a_2, \ldots, a_m)$$

Takav konačan niz se zove i m- torka ili lista.

PRIMER

Primer poznatih nizova

Primer
1. Poznati nizovi

$$1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \ldots . \; i \, 1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \ldots .$$

se mogu formalno definisati sa

$$a_n = \frac{1}{n} \text{ za sve } n = 0, 1, \ldots \quad \text{i} \quad b_n = 2^{-n}, \text{ za sve } n = 1, 2, 3...$$

2. Važan niz 1, −1, 1, −1, . . . se može formalno definisati sa

$$a_n = (-1)^{n+1}$$

za n = 1, 2, . . . ili

$$b_n = (-1)^n$$

za n = 0, 1, . . .

ARITMETIČKA PROGRESIJA

Aritmetička progresija je niz oblika a, a + d, a + 2d, a + 3d, . . .

Definicija
Aritmetička progresija je niz oblika
a, a + d, a + 2d, a + 3d, . . .

to jest, niz počinje sa a, a svaki
sledeći element se dobija iz prethodnog,
dodavanjem broja d.

Primer

Posmatrajmo aritmetičku progresiju formiranu od:

a = 5, d = 3, to jest, 5, 8, 11, 14, . . .

a = 2, d = 5, to jest, 2, 7, 12, 17, . . .

a = 1, d = 0, to jest, 1, 1, 1, 1, ...

Opšta aritmetička progresija se može rekurzivno definisati na sledeći način

$a_1 = a$ i $a_{k+1} = a_k + d$ za $k \geq 1$

pri čemu je njeno rešenje

$a_n = a + (n - 1)d$

opštija forma

$a_n = a_m + (n - m)d$

GEOMETRIJSKA PROGRESIJA

Geometrijska progresija je niz oblika $a, ar, ar^2, ar^3, \ldots$

Definicija
Geometrijska progresija je niz oblika
$a, ar, ar^2, ar^3 \ldots$
to jest, niz počinje sa a, a svaki sledeći element se dobija iz prethodnog, množenjem sa r.

Opšta geometrijska progresija se može rekurzivno definisati na sledeći način

$a_1 = a$ i $a_{k+1} = ra_k$ za $k \geq 1$

pri čemu je njeno rešenje

$a_n = ar^n$

Primer

Posmatrajmo geometrijsku progresiju formiranu od

a = 1, r = 3,

to jest, 1, 3, 9, 27, . . .

a = 5, r = 2,

to jest, 5, 10, 20, 40, . . .

a = 1, r = 1 ,

to jest, 1, 1 , 1 , 1 , ...

ODREĐIVANJE ČLANOVA NIZA

Čest problem u diskretnoj matematici je da se pronađe formula ili uopšteno pravilo za određivanje članova niza.

Čest problem u diskretnoj matematici je da se pronađe formula ili uopšteno pravilo za određivanje članova niza. Nekada su samo pojedini članova niza poznati, a cilj je da se odredi sekvenca, odnosno ceo niz. Iako početni članovi niza ne određuju celu sekvencu (na kraju, postoji beskonačno mnogo nizova koji počinju sa konačnim brojem početnih članova), ipak kada znamo prve članove, oni nam mogu pomoći da odredimo celu sekvencu članova, odnosno ceo niz.

Zadatak

Pronađite iskaz koji definiše niz čiji su prvih pet članova
a) 1, 1/2, 1/4, 1/8, 1/16
b) 1, 3, 5, 7, 9
c) 1, -1, 1, -1, 1.

Rešenje

a) Možemo uvideti da je svaki imenilac deljiv sa 2. Ovaj niz možemo opisati kao $a_n = 1/2^n$, za $n = 0, 1, 2, \ldots$. Ovaj niz predstavlja geometrijsku progresiju sa $a = 1$ i $r = 1/2$.

b) Primećujemo da svaki član dobijamo dodavanjem broja 2 prethodnom članu niza. Ovaj niz možemo opisati kao $a_n = 2n + 1$ za $n = 0, 1, 2, \ldots$. Ovaj niz predstavlja aritmetičku progresiju sa $a = 1$ i $d = 2$.
c) U ovom nizu članovi se naizmenično menjaju u vrednosti 1 i -1. 2. Ovaj niz možemo opisati kao
$a_n = (-1)^n$, za $n = 0, 1, 2, \ldots$. Ovaj niz predstavlja geometrijsku progresiju sa $a = 1$ i $r = -1$.

PRIMERI

Odrediti sve članove niza na osnovu prvih 10 članova

Zadatak
Ako su prvih 10 članova niza 5, 11, 17, 23, 29, 35, 41, 47, 53, 59, koji su ostali članovi ovog niza?
Rešenje
Primetimo da se svaki sledeći član ovog niza dobija dodavanjem broja 6 prethodnom članu. Samim tim, n-ti član ovog niza se može dobiti tako što se počne sa brojem 5, koji je prvi član u ovom nizu i doda se broj 6 $n-1$ puta. Tako dobijamo da je n-ti član $5+6(n-1)=6n-1$. Ovo predstavlja aritmetičku progresiju sa a = 5 i d = 6.
Zadatak
Ako su prvih 10 članova niza 1, 7, 25, 79, 241, 727, 2185, 6559, 19681, 59047, koji su ostali članovi ovog niza?
Rešenje
Pristupamo rešavanju ovog problema tako što prvo razmatramo razliku između susednih članova niza. Ovim možemo uvideti da ne postoji nikakva šema. Kada podelimo susedne termine, možemo uvideti da iako nije konstanta, da je vrednost uvek blizu broja 3. Tako da postoji velika verovatnoća da će u opštoj formuli ovog niza biti 3^n . Upoređivanjem članova niza sa vrednostima koji se dobijaju sa 3^n možemo uvideti da se n-ti član razlikuje za 2. Tako primećujemo da $a_n=3^n-2$ za $1\leq n\leq 10$, pa možemo pretpostaviti da ova formula važi za sve n.

Poglavlje 2

REKURZIVNE FUNKCIJE I NIZOVI

DEFINICIJA REKURZIVNE FUNKCIJE

Funkcija je rekurzivno definisana ako se definicija poziva na samu sebe

Rekurzivno definisane funkcije i nizovi se često javljaju u matematici i kompjuterskoj nauci.

Definicija
Funkcija je rekurzivno definisana ako se definicija poziva na samu sebe.

Da rekurzivna definicija ne bi bila kružna (beskonačna), definicija funkcije mora imati sledeće dve osobine:

1. Moraju postojati neki argumenti, koji se zovu bazne vrednosti za koje se definicija funkcije ne poziva na samu sebe
2. Svaki put kad se funkcija poziva na samu sebe, njen argument mora biti bliži baznoj vrednosti.

Mnogo metoda je razvijeno za rešavanje rekurzivnih funkcija. Ovde ćemo uvesti neposrednu metodu poznatu kao *iteracija* kroz nekoliko primera.
Neka je niz a_n opisan rekurentnom relacijom $a_n = a_{n-1} + 3$ za $n = 1, 2, 3, \dots$, znamo da je $a_1 = 2$. Odrediti a_2, a_3, a_4 .
Ako započnemo sa početnom vrednošću $a_1 = 2$, lako možemo odrediti ostale članove niza, dok se ne do]e do opšteg člana. Tada je
$a_2 = 2 + 3$
$a_3 = (2 + 3) + 3 = 2 + 3 \cdot 2$
$a_4 = (2 + 2 \cdot 3) + 3 = 2 + 3 \cdot 3$
$\dots$
$a_n = a_{n-1} + 3 = (2 + 3 \cdot (n - 2)) + 3 = 2 + 3(n - 1).$

Takođe se uspešno može primeniti i obrnuti način, tako da se počne sa članom a_n i da se nastavi naniže, dok se ne stigne do početne vrednosti $a_1 = 2$. To podrazumeva sledeće korake
$a_n = a_{n-1} + 3$
$= (a_{n-2} + 3) + 3 = a_{n-2} + 3 \cdot 2$
$= (a_{n-3} + 3) + 3 \cdot 2 = a_{n-3} + 3 \cdot 3$
$\dots$
$= a_2 + 3(n - 2) = (a_1 + 3) + 3(n - 2) = 2 + 3(n - 1).$

2.1 FAKTORIJEL

DEFINICIJA I PRIMER FAKTORIJELA

Funkcija faktorijel je definisana za sve n ∈ N_0

Proizvod pozitivnih celih brojeva od 1 do n uključujući i njih se zove n faktorijel, i označava sa n!, to jest,
$n! = 1 \cdot 2 \cdot 3 \ldots (n-2)(n-1)n$

Takođe je zgodno definisati 0! = 1, pa je onda funkcija ! definisana za sve nenegativne cele brojeve. Saglasno sa tim, dajemo sledeću rekurzivne definiciju funkcije faktorijel

Definicija

Funkcija faktorijel je definisana za sve n ∈ N_0

0! = 1 i n! = n (n − 1)! za svako n ≥ 1

Primetimo da je funkcija n! rekurzivna, pošto se poziva na samu sebe korišćenjem (n − 1)!

1. Vrednost n! je eksplicitno data za n = 0; pa je 0 bazna vrednost.
2. Vrednost n! za proizvoljno n se definiše preko manjih vrednosti argumenta n, koji su bliži baznoj vrednosti 0.

Tako ova definicija nije kružna, pa je funkcija dobro definisana.

Primer

Imamo

0! = 1, 1! = 1 x 0! = 1, 2! = 2 · 1! = 2 · 1 = 2,

3! = 3 · 2! = 3 · 2 = 6, 4! = 4 · 3! = 4 · 6 = 24,

5! = 5 · 4! = 5 · 24 = 120, 6! = 6 · 5! = 6 · 120 = 720

2.2 FIBONAČIJEV NIZ

DEFINCIJA I PRIMER FIBONAČIJEVOG NIZA

Za n=0,1 F(n) = n, a za n ≥ 2 F (n) = F (n-2) + F (n-1)

Definicija

Fibonačijev niz, u oznaci je definisan sa

a) $F_n = n$ za $n = 0, 1$

b) $F_n = F_{n-2} + F_{n-1}$ za $n \geq 2$

Primetimo da je definicija Fibonačijevoj niza rekurzivna, pošto se poziva na samu sebe, koristeći F_{n-1} i F_{n-2}.

Dalje:

Bazne vrednosti su 0 i 1.

Vrednost F_n je definisana preko manjih vrednosti argumenta n, koje su bliže baznim vrednostima.

Tako definicija nije kružna, a Fibonačijev niz je dobro definisan.

Primer

$F_0 = 0, F_1 = 1, F_2 = F_0 + F_1 = 0 + 1 = 1,$

$F_3 = F_1 + F_2 = 1 + 1 = 2, F_4 = F_2 + F_3 = 1 + 2 = 3,$

$F_5 = F_3 + F_4 = 2 + 3 = 5, F_6 = F_4 + F_5 = 3 + 5 = 8,$

$F_7 = F_5 + F_6 = 5 + 8 = 13, F_8 = F_6 + F_7 = 8 + 13 = 21,$

$F_9 = F_7 + F_8 = 13 + 21 = 34, F_{10} = F_8 + F_9 = 21 + 34 = 55, \ldots$

Poglavlje 3

SUME

DEFINICIJA SUME

Indeks sumiranja je definisan u opsegu sa donjom granicom m i gornjom granicom n

Simbol koji se koristi za sumiranje je $\sum$.

Neka je a_1, a_2, . . . neki niz (realnih brojeva), tada pišemo
$\sum_{k=m}^{n} a_k = a_m + a_{m+1} + \ldots + a_n \; za \; sve \; m, n \in \mathbb{N} \; gde \; n \geq m$
i koristimo konvenciju
$\sum_{k=m}^{n} a_k = 0 \; ako \; jem > n.$

Slovo k u gornjem izrazu se zove indeks sumiranja**.** Indeks sumiranja je definisan u opsegu sa donjom granicom m i gornjom granicom n.

Uobičajeni aritmetički zakoni se primenjuju kod sume. Na primer, kada su a i b realni brojevi tada imamo

$\sum_{j=1}^{n}(ax_j + by_j) = a\sum_{j=1}^{n} x_j + b\sum_{j=1}^{n} y_j$

gde su $x_n \in \mathbb{R}$ i $y_n \in \mathbb{R}$

Suma je linearni operator, što znači da se konstante mogu izvući ispred sume.

PRIMERI

Primeri iskazivanja suma

Primer

$$\sum_{j=1}^{n} a_j b_j = a_1 b_1 + a_2 b_2 + \ldots + a_n b_n$$

$$\sum_{k=0}^{n} a_k b_{n-k} = a_0 b_n + a_1 b_{n-1} + \ldots + a_n b_0$$

$$\sum_{k=2}^{5} k^2 = 2^2 + 3^2 + 4^2 + 5^2 = 4 + 9 + 16 + 25 = 54$$

$$\sum_{k=1}^{n} k = 1 + 2 + \ldots + n = \frac{n(n+1)}{2}$$

Primer

Iskažite sumu prvih 100 elemenata niza a_n , tako da $a_n = \frac{1}{n}$ za $n \in \mathbb{N}$

Rešenje

$$\sum_{j=1}^{100} \frac{1}{j}$$

Primer

Koja je vrednost $\sum_{k=4}^{8}(-1)^k$

Rešenje:

$$\sum_{k=4}^{8}(-1)^k = (-1)^4 + (-1)^5 + (-1)^6 + (-1)^7 + (-1)^8 = 1$$

UGNJEŽDENE SUME I NAČIN IZRAČUNAVANJA SUME

U narednim primerima biće objašnjen pojam i način izračunavanja sume niza

Primer ugnježdene ili duple sume je sledeći

$$\sum_{i=1}^{4}\sum_{j=1}^{3} ij = \sum_{i=1}^{4}(1 + 2i + 3i) = \sum_{i=1}^{4} 6i = 6 + 12 + 18 + 24.$$

Takođe se može iskoristiti notacija sume da se doda vrednost funkciji, ili elemenata indeksiranog skupa, gde se indeks nalazi u opsegu svih vrednosti skupa.

Primer

Koja je vrednost sledećeg iskaza

$$\sum_{s \in \{0,2,4\}} s$$

Rešenje

$$\sum_{s \in \{0,2,4\}} s = 0 + 2 + 4 = 6$$

Prozivod suma je jednak sumama proizvoda. Važi i obrnuto

$$\sum_{i=n}^{m} x_i \sum_{j=n'}^{m'} y_j = \sum_{i=n}^{m}\sum_{j=n'}^{m'} x_i y_j$$

VIDEO - PRIMER

Određivanje vrednosti suma

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Poglavlje 4

ALGORITMI

ŠTA SU ALGORITMI?

Algoritam je kompletna konačna lista koraka potrebnih za izvršavanje nekog zadatka ili izračunavanja. Koraci u algoritmu mogu biti opšti opisi, bez mnogo detalja, ili vrlo precizni

Kao primer algoritama navodimo sledeće:

- Euklidov algoritam koji se koristi za određivanje najvećeg zajedničkog delioca
- Kulinarski recept

Osnovna svojstva:

- Ulaz – svaka ulazna vrednost algoritma se uzima iz unapred zadatog skupa
- Izlaz – svaki algoritam pokušava da nađe rešenje za neki problem, pa je samim tim taj problem rešiv ili ne. Ukoliko je zadati problem rešiv onda dobijamo odgovor na taj problem, odnosno rešenje.
- Konačnost– algoritam treba da se završi nakon određenog broja koraka
- Definisanost i nedvosmislenost - svaki korak u algoritmu treba da je jasno definisan i da ne bude dvosmislen.
- Efikasnost – svaki korak u algoritmu treba da je jasan i razumljiv za čoveka.
- Generalnost - trebalo bi biti moguće da se algoritam primeni za sve probleme koji su dati u traženoj formi, ne samo za određeni skup ulaznih vrednosti.

Definicija

Algoritam predstavlja skup konačnog broja instrukcija koji se koristi da bi se nešto izračunalo ili rešio problem.

PRIMER ALGORITMA

Opišite algoritam za pronalaženje maksimalnog broja u konačnom nizu brojeva

Opišite algoritam zapronalaženje maksimalnog broja u konačnom nizu brojeva.

Iako je problem pronalaženja najvećeg člana u nizu trivijalan problem, daje dobru osnovu za ilustraciju koncepta algoritma. Takođe, postoje mnoge situacije kada je zaista potrebno da pronađemo najveću vrednost i nizu nekih brojeva. Na primer, univerzitet možda treba da pronađe najveći rezultat na nekom ispitu, gde je polagalo stotine studenata. Procedura za rešavanje ovog problema se može definisati na više načina. Jedan od načina je da se jednostavno govornim jezikom opiše koje korake treba primeniti.

Primer rešenja:

1. Postaviti privremeni maksimum da bude jednak prvom broju u nizu (privremeni "maksimum" će biti najveći broj koji je ispitan u bilo kom momentu procedure ispitivanja)

2. Uporediti sledeći broj u nizu sa privremenim maksimumom. Ako je taj broj veći onda on postaje privremeni maksimum.

3. Ponoviti prethodni korak, ako ima još brojeva u nizu.

4. Stati kada više nema brojeva u nizu. U ovom koraku vrednost privremenog maksimuma predstavlja najveći broj niza.

Algoritam takođe možemo opisati koristeći računarski jezik. Međutim, ovo često dovodi do slabo razumljivog algoritma. Umesto korišćenja specifičnog progamskog jezika da opišemo algoritam, odnosno njegove korake, može se koristiti pseudo kod. Pseudi kod ustvari predstavlja međukorak između govornog jezika i programskog jezika. Prethodno opisani algoritam možemo ovako napisati u pseudo kodu

procedure max(a_1, a_2, . . . , a_n: integers)

max := a_1

for 1 := 2 **to** n

if max < a_i **then** max := a_i

{max je najveci broj}

Poglavlje 5

NUMERIČKI ALGORITMI

NUMERIČKI METODI

Tri numerička metoda za nalaženje nula realnih funkcija na realnom intervalu su bisekcioni metod, metod sečice i Njutnov metod

Brzi razvoj numeričke matematike je bio pospešen brzim razvojem nauke i tehnike, a posebno razvojem u računarstvu. Numerička matematika omogućava rešavanje kompleksnih problema korišćenjem računara. Kako računari imaju mogućnost da obave veliki broj operacija uz automatizovani proces računanja, to pruža velike mogućnosti numeričkoj matematici. Na taj način niz matematičkih problema koji se klasičnim matematičkim metodima ne mogu uvek tačno rešiti ili bi njihovo rešavanje bilo necelishodno, efikasno se rešavaju korišćenjem aparata numeričke matematike.
Glavni zadatak numeričke matematike je konstrukcija i analiza metoda(algoritama) i formiranje odgovarajućeg numeričkog softvera. U okviru numeričke matematike postoje posebne oblasti - teorija aproksimacija i teorija optimizacija.
Mnogi numerički algoritmi su primene specijalnih, rekurzivno definisanih nizova funkcija. Koristimo ih, na primer, za nalaženje nula funkcije, ili korena jednačine. Ovde izučavamo 3 numerička metoda za nalaženje nula realnih funkcija na realnom intervalu

- Bisekcioni metod
- Metod sečice
- Njutnov metod

Ovde pretpostavljamo da $f : I = [a, b] \to \mathbb{R}$ je neprekidna na I . Umesto rešavanja problema
$f(x) = 0 (x \in I)$
posmatramo ekvivalentan problem koji je povezan sa prvim
$x = \phi(x) (x \in I)$
Gde je ϕ pogodno izabrana funkcija koja je takodje neprekidna na intervalu I .

Numerički algoritam koji koristi konstruisanje iterativnog niza za aproksimaciju rešenja zove se iterativni metod ili iterativni proces. Konvergencija iterativnog niza prirodno zavisi od osobina funkcije ϕ i izbora početne vrednosti x_0. Ovi problemi se detaljno izučavaju u numeričkoj matematici.
Ako zahtevamo da prethodno definisana funkcija ϕ zadovoljava uslov
$\phi(I) \subset I$
onda možemo definisati iterativni niz
$x_{k+1} = \phi(x_k) (k = 0, 1, \dots)$
sa inicijalnom (početnom) vrednošću $x_0 \in I$. Ako niz $(x_k) (k = 0, \dots .)$ konvergira ka, recimo,
$x_k \to \xi (k \to \infty)$
onda je ξ rešenje jednačine.

5.1 BISEKCIONI METOD

OPIS BISEKCIONOG METODA

Funkcija f ima bar jednu nulu na intervalu (a, b) po teoremi srednje vrednosti za neprekidne funkcije

Bisekcioni metod je prilično jednostavan, ali efikasan iteracioni metod.

Posmatrajmo neprekidnu funkciju f : I → R i izaberimo a0, b0 ∈ I takve da

f(a0)f(b0) < 0.
Tada funkcija f ima bar jednu nulu na intervalu (a, b) po teoremi srednje vrednosti za neprekidne funkcije. Definišimo sada niz intervala na sledeći način
I $_k$ = [a $_k$, b $_k$] takvih da [a $_{k+1}$, b $_{k+1}$] ⊂ [a $_k$, b $_k$]
za k = 0, 1, . . .
Neka je
$m_k = \frac{1}{2}(a_k + b_k)(k = 0, 1, \dots)$srednja tačka k − tog intervala I_k .
Ako je $f(m_k) = 0$ onda smo našli nulu i završavamo algoritam. Ako nije, stavljamo

$$[a_{k+1}, b_{k+1} = \begin{cases} [a_k, m_k] & akof(m_k)f(a_k) < 0 \\ [m_k, b_k] & akof(m_k)f(b_k) < 0 \end{cases}$$

Iterativni metod
$x_k = m_k(k = 0, 1, \dots)$
definiše bisekcioni metod .

Iterativni metod

x$_k$ = m$_k$ (k = 0, 1, . . .)

definiše bisekcioni metod.

Konvergencija bisekcionog modela je vrlo spora. Bez obzira na strukturu funkcije f u proseku treba 3.3 koraka iteracije da bi se dostigao faktor 10^{-1} u aproksimaciji. Kako je f neprekidna, po konstrukciji algoritma, postoji nula funkcije f na svakom intervalu I$_k$, pa bisekcioni metod konvergira u svakom slučaju.

5.2 METOD SEČICE

OPIS METODE SEČICE

Svaki element x_{k+1} iterativnog niza predstavlja presek x-ose i prave (sečice) koja spaja prethodne dve tačke

Izaberimo x_0, $x_1 \in I = [a, b]$ takve da $f(x_0)\, f(x_1) < 0$.

Svaki element x_{k+1} iterativnog niza predstavlja presek x-ose i prave (sečice) koja spaja predhodne dve tačke

$P_k = (x_k, f(x_k))$ i $P_{k-1} = (x_{k-1}, f(x_{k-1}))$

Prava linija kroz ove dve tačke je opisana jednačinom

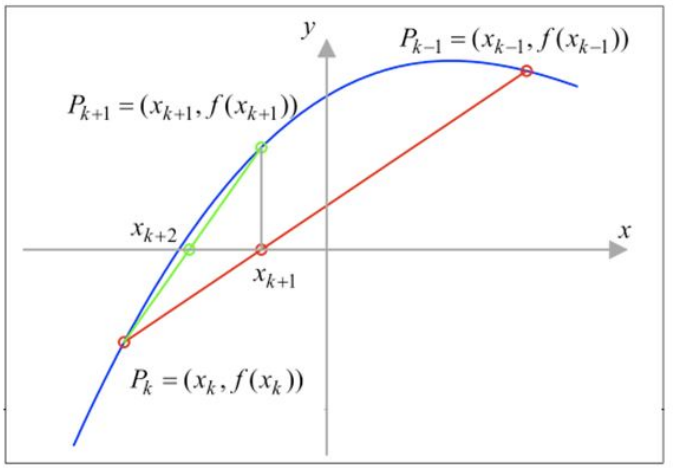
$$(f(x_k) - (f(x_{k-1}))\frac{x - x_k}{x_k - x_{k-1}} + f(x_k)$$

a njen presek, x_{k+1}, sa x-osom se izračunava za y = 0, znači

$$x_{k+1} - x_k = -f(x_k)\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})}$$

Tako je metod sečice definisan iterativnim nizom

$$x_{k+1} = x_k - f(x_k)\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})}\,(k = 1, 2, \ldots\, za f(x_k) \neq f(x_{k-1})$$



Slika 5.1.1 Dve iteracije metoda sečice [Izvor: Autor]

5.3 NJUTNOV METOD

DEFINICIJA NJUTNOVOG METODA

Njutnov metod se dobija zamenom količnika u definiciji iterativnog niza metoda sečice sa nagibom tangente

Njutnov metod se dobija zamenom količnika u definiciji iterativnog niza metoda sečice sa nagibom tangente T_k grafa funkcije f u tački x_k , f '(x_k). Izaberemo neku početnu vrednost

$x_0 \in I = [a, b]$.

Svaki element x_{k+1} iterativnog niza predstavlja presek x-ose sa tangentom grafa funkcije f u predhodnoj tački $P_k = (x_k, f(x_k))$. Tako je Njutnov metod definisan iterativnim nizom

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \, za \, k = 0, 1, \ldots \, i \, f'(x_k) \neq 0$$

Primer

Posmatrajmo funkciju

f : (0, ∞) → (0, ∞)

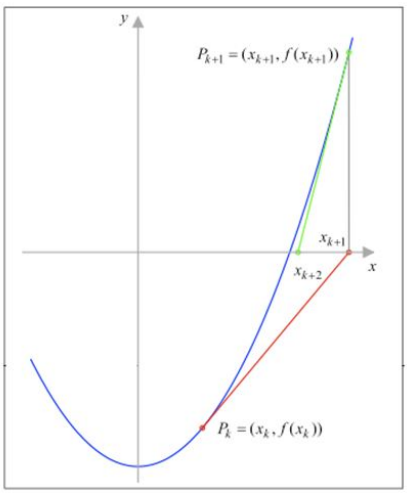
definisanu sa

$f(x) = x^2 - a$

gde je a > 0 konstantno. Tada je f ‘(x) = 2x i iterativni niz se svodi na

$$x_{k+1} = x_k - \frac{x_k^2 - a}{2x_k} = \frac{x_k^2 + a}{2x_k}$$

što je rekurzivna formula.



Slika 5.2.1 Dve iteracije Njutnovog metoda [Izvor: Autor]

Poglavlje 6

SLOŽENOST ALGORITAMA

BIG O NOTACIJA

Big O se obično koristi za analizu kompleksnosti najgoreg slučaja algoritma.

U računarstvu, kada su u pitanju algoritmi, često se postavljaju dva pitanja:

1. Koliko vremena je potrebno algoritmu da se izvrši?

2. Koliko memorije je potrebno algoritmu za njegovo izvođenje?

Big O notacija je standardan način na koji matematičari i informatičari opisuju koliko vremena i memorije je potrebno algoritmu da se izvrši. Big O se obično koristi za analizu kompleksnosti najgoreg slučaja algoritma. Na primer, ako je n veličina ulaznih podataka, Big O zapravo samo brine o tome šta se dešava kada veličina ulaznih podataka n postane proizvoljno velika, a ne toliko kada je ulaz mali. Matematički, želimo da govorimo o kompleksnosti u asimptotskom smislu, kada je n proizvoljno velik. U ovom asimptotskom smislu za veliko n, možemo zanemariti konstante.

Veličina kompleksnosti ulaza sortirana od najmanje do najveće:

- Konstantna složenost: O(1)
- Logaritamska složenost: O(log(n))
- Linearna složenost: O(n)
- Kvadratna složenost:: $O(n^2)$
- Kubna složenost: : $O(n^3)$

Algoritmi čija je složenost O(1) karakterišu se time da vreme izvršavanja ostaje konstantno bez obzira na veličinu ulaznih podataka. Drugim rečima, za rešavanje određenog problema potrebno je samo jednom pristupiti jednom elementu, bez obzira na to koliko tih elemenata ulazi u igru.

Primer

```
zadati_niz = [1, 2, 3, 4, 10]

function pronadjiElemt(zadati_niz){

console.log(zadati_niz[2]);

}
```

U gornjem primeru, funkcija jednostavno prikazuje element niza u konzoli. Koliko operacija ova funkcija zahteva ako se veličina niza poveća na 1.000 ili recimo 100.000? Pošto samo izvlačimo pojedinačan element iz niza, bez obzira na to koliko je veličina niza, vreme koje je potrebno ostaje konstantno ili O(1).

LINEARNA SLOŽENOST

Linearnu složenost sa Big O notacijom obeležavamo kao O(n).

Linearnu složenost sa Big O notacijom obeležavamo kao O(n).

Razmotrimo algoritam koji prikazuje deo funkcije za izračunavanje srednje vrednosti niza. Ukupan broj operacija koje će se izvršiti tokom ovog algoritma iznosi 2n + 2. Ovo se događa jer ćemo n puta povećati vrednost promenljive i, n puta povećati sumu za vrednost a[i], izvršiti jednu operaciju za inicijalizaciju promenljive suma na 0 i jednu operaciju za računanje srednje vrednosti. Ako zanemarimo konstante, možemo zaključiti da je ukupna složenost ovog algoritma n operacija. Ovaj faktor složenosti će se označavati kao O(n). U ovom slučaju, kažemo da je algoritam linearne složenosti.

```
suma = 0;

for i = 1 to n

suma = suma + a[i];

avg = suma / n;
```

Primer

```
let zadatiNiz = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

function sumirajNiz(zadatiNiz) {

let totalZbir = 0;

for (let i = 0; i < zadatiNiz.length; i++) {

totalZbir += zadatiNiz[i];

}

return totalZbir;

}
```

Ovaj JavaScript kod ima za cilj da sumira sve elemente u nizu zadati_niz. Da bismo razumeli složenost ovog algoritma, treba da razmotrimo kako se ponaša sa povećanjem veličine ulaznih podataka. U datom kodu, imamo niz od 10 elemenata. Funkcija sumirajNiz koristi petlju for da prolazi kroz niz i sabira sve elemente. Kako se broj elemenata u nizu povećava, broj operacija koje su potrebne za izračunavanje se povećava linearno. To znači da ako

imamo n elemenata u nizu, moramo izvršiti n operacija (u ovom slučaju, petlju koja se izvršava n puta) da bismo dobili željeni rezultat. Ova vrsta povećanja broja operacija sa povećanjem veličine ulaznih podataka karakteristična je za algoritme čija je složenost O(n), ili se u terminima Velike O notacije kaže da ovi algoritmi zahtevaju "linearno vreme" da reše problem. To znači da efikasnost ovog algoritma raste proporcionalno sa veličinom ulaznih podataka, odnosno brojem elemenata u nizu.

KVADRATNA SLOŽENOST

Kvadratna složenost znači da broj operacija koje se izvode raste u odnosu na kvadrat ulazne veličine

Složenost O(n^2) znači da broj operacija koje se izvode raste u odnosu na kvadrat ulazne veličine.

U slučaju O(n^2), broj operacija koje se izvode raste proporcionalno kvadratu ulazne veličine. Ovo je često slučaj s algoritmima koji uključuju ugnježđene petlje ili iteracije.

Da bismo ilustrovali to na primeru, razmotrimo kako želimo da pronađemo zbir svakog para elemenata u datom nizu zadatiNiz, koji ima 10 elemenata.

```
let zadatiNiz = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
let pairSum=0;

for (let i = 0; i < zadatiNiz.length; i++) {
  for (let j = i + 1; j < zadatiNiz.length; j++) {
    pairSum = zadatiNiz[i] + zadatiNiz[j];
    console.log("Suma svakog para elemenata u zadatiNiz:"+ pairSum);
  }
}
```

U gornjem primeru, za svaku iteraciju spoljnje petlje (i), izvršavamo n operacija unutar unutrašnje petlje (j), i spoljnu petlju izvršavamo n puta jer se obrađuje za svaku vrednost i. To znači da ukupan broj operacija iznosi i * j operacija, ili, kako koristimo isti ulaz, n * n operacija, što se zapisuje kao n^2. Ovo pokazuje kvadratnu složenost, jer broj operacija eksponencijalno raste sa povećanjem broja elemenata u nizu. Ovaj tip složenosti često se javlja kod algoritama koji uključuju dva ugnježđena prolaza kroz podatke ili dve ugnježđene iteracije.

LOGARITAMSKA SLOŽENOST

Algoritmi logaritamske složenosti spadaju u grupu veoma efikasnih algoritama čije vreme izvršenja raste znatno sporije od rasta ulaznih podataka

Algoritamska složenost se koristi kako bismo analizirali brzinu rasta broja operacija koje algoritam izvršava u zavisnosti od veličine ulaznih podataka. Algoritmi logaritamske složenosti spadaju u grupu veoma efikasnih algoritama čije vreme izvršenja raste znatno sporije od rasta ulaznih podataka.

Logaritam je matematička operacija koja predstavlja suprotnost stepenovanju. U računarstvu se obično koristi logaritam sa osnovom 2, što predstavlja kontinuirano deljenje na pola. Logaritamski algoritmi su efikasni jer nakon svakog koraka eliminišu polovinu preostalih podataka.

Klasičan primer algoritma logaritamske složenosti je binarna pretraga, koja u svakom koraku deli sortiran niz na pola i eliminiše jednu polovinu, znatno smanjujući broj preostalih podataka za obradu. Primer iz stvarnog sveta koji ilustruje ovu efikasnost je traženje reči u rečniku. Logaritam od 1.000.000 (milion) iznosi 19,9, što znači da u ogromnom rečniku od milion reči možemo pronaći bilo koju reč u najviše 20 koraka, što je izuzetno brzo. Algoritmi korenske složenosti, koji takođe spadaju u podlinijske algoritme, su retki, ali takođe pružaju brzu i efikasnu obradu podataka.

Poglavlje 7

VEŽBA

ZADATAK 1

Određivanje elemenata niza.

Predviđeno vreme trajanja: 10 minuta

ZADATAK:

Nastavi dati niz ako je prvih 10 elemenata niza: 1, 2, 2, 3, 3, 3, 4, 4, 4, 4.

REŠENjE:

1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6...

U ovom nizu broj 1 se pojavljuje jedan put, broj 2 dva puta, broj 3 tri puta, broj 4 četiri puta. Pravilo za ovaj niz jeste da se broj n pojavljuje tačno n puta. Tako bi narednih pet članova ovog niza predstavljalo broj 5, a narednih šest elemenata broj 6 i tako dalje redom.

ZADATAK 2

Pronalaženje elemenata niza.

Predviđeno vreme trajanja: 10 minuta

ZADATAK:
Nastavi dati niz ako je prvih 10 elemenata niza: 1, 3, 4, 7, 11, 18, 29, 47, 76, 123.

REŠENJE:

1, 3, 4, 7, 11, 18, 29, 47, 76, 123, 199, 322, 521... Počevši od trećeg elementa niza, možemo primetiti da svaki element ovog niza predstavlja zbir prethodna dva člana. Shodno tome je 4 = 1 + 3, 7 = 3 + 4, 11 = 4 + 7 i tako dalje redom.

Prema tome, ako je L_n n-ti član niza onda je niz određen relacijom

$L_n = L_{n-1} + L_{n-2}$ sa početnim članovima niza $L_1 = 1$ i $L_2 = 3$.

ZADATAK 3

Određivanje prvih 10 članova niza.

Predviđeno vreme trajanja: 15 minuta

ZADATAK:

Odredi prvih 10 članova sledećih nizova:

a) n^2

b) n^3

c) n^4

d) 2^n

e) 3^n

f) n!

g) f_n

REŠENJE:

a) 1, 4, 9, 16, 25, 36, 49, 64, 81, 100...

b) 1, 8, 27, 64, 125, 216, 343, 512, 729, 1000...

c) 1, 16, 81, 256, 625, 1296, 2401, 4096, 6561, 10000...

d) 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024...

e) 3, 9, 27, 81, 243, 729, 2187, 6561, 19683, 59049...

f) 1, 2, 6, 24, 120, 720, 5040, 40320, 362880, 3628800...
g) 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89...

ZADATAK 4

Pronaći formulu za an.

Predviđeno vreme trajanja: 20 minuta

ZADATAK:

Pronaći jednostavnu formulu za a_n ako je prvih 10 članova niza: 1, 7, 25, 79, 241, 727, 2185, 6559, 19681, 59047.

REŠENJE:

Da bismo rešili ovaj problem, počinjemo tako što posmatramo razliku uzastopnih članova, ali ne primećujemo neki obrazac. Kada formiramo odnos uzastopnih članova da bismo videli da li je svaki član množilac prethodnog člana, primećujemo da ovaj odnos, iako nije konstantan, jeste blizu broja 3. Stoga je razumno pretpostaviti da su članovi ovog niza generisani

formulom koja uključuje 3^n. Upoređujući ove članove sa odgovarajućim članovima niza $\{3^n\}$, primećujemo da je n-ti član za 2 manji od odgovarajuće potencije broja 3. Dakle, opšte rešenje za zadati niz je

$a_n = 3^n - 2$

ZADATAK 5

Primer sumiranja niza.

Predviđeno vreme trajanja: 15 minuta

ZADATAK:
Odrediti vrednost sledeće sume:

$$\sum_{k=4}^{8}(-1)^k$$

REŠENJE:

$$\sum_{k=4}^{8}(-1)^k = (-1)^4 + (-1)^5 + (-1)^6 + (-1)^7 + (-1)^8 = 1 + (-1) + 1 + (-1) + 1 = 1$$

ZADATAK 6

Određivanje sume niza.

Predviđeno vreme trajanja: 15 minuta

ZADATAK:
Odrediti vrednost sledeće sume:

$$\sum_{k=50}^{100} k^2$$

REŠENJE:

$$\sum_{k=1}^{100} k^2 = \sum_{k=1}^{49} k^2 + \sum_{k=50}^{100} k^2$$

$$\sum_{k=50}^{100} k^2 = \sum_{k=1}^{100} k^2 - \sum_{k=1}^{49} k^2$$

$$Formula: \sum_{k=1}^{n} k^2 = n(n+1)(2n+1)/6$$

$$\sum_{k=50}^{100} k^2 = \frac{100 \cdot 101 \cdot 201}{6} - \frac{49 \cdot 50 \cdot 99}{6} = 338350 - 40425 = 297925$$

ZADATAK 7

Primeri sumiranja niza.

Predviđeno vreme trajanja: 15 minuta

ZADATAK:
Odrediti vrednosti sledećih suma:
$a) \sum_{j=0}^{8} 3 \cdot 2^j$
$b) \sum_{j=1}^{8} 2^j$
$c) \sum_{j=2}^{8} (-3)^j$
$d) \sum_{j=0}^{8} 2 \cdot (-3)^j$

REŠENjE:
$a) \sum_{j=0}^{8} 3 \cdot 2^j = 3 \cdot 2^0 + 3 \cdot 2^1 + 3 \cdot 2^2 + 3 \cdot 2^3 + 3 \cdot 2^4 + 3 \cdot 2^5 + 3 \cdot 2^6 + 3 \cdot 2^7 + 3 \cdot 2^8 = 3 \cdot 1 + 3 \cdot 2 + 3 \cdot 4 + 3 \cdot 8 + 3 \cdot 16 + 3 \cdot 32 + 3 \cdot 64 + 3 \cdot 128 + 3 \cdot 256 = 3 + 6 + 12 + 24 + 48 + 96 + 192 + 384 + 768 = 1533$
$b) \sum_{j=1}^{8} 2^j = 2^1 + 2^2 + 2^3 + 2^4 + 2^5 + 2^6 + 2^7 + 2^8 = 2 + 4 + 8 + 16 + 32 + 64 + 128 + 256 = 510$

$c) \sum_{j=2}^{8} (-3)^j = (-3)^2 + (-3)^3 + (-3)^4 + (-3)^5 + (-3)^6 + (-3)^7 + (-3)^8 = 9 + (-27) + 81 + (-243) + 729 + (-2187) + 6561 = 4923$

$d) \sum_{j=0}^{8} 2 \cdot (-3)^j = 2 \cdot (-3)^0 + 2 \cdot (-3)^1 + 2 \cdot (-3)^2 + 2 \cdot (-3)^3 + 2 \cdot (-3)^4 + 2 \cdot (-3)^5 + 2 \cdot (-3)^6 + 2 \cdot (-3)^7 + 2 \cdot (-3)^8 = 2 \cdot 1 + 2 \cdot (-3) + 2 \cdot 9 + 2 \cdot (-27) + 2 \cdot 81 + 2 \cdot (-243) + 2 \cdot 729 + 2 \cdot (-2187) + 2 \cdot 6561 = 2 - 6 + 18 - 54 + 162 - 486 + 1458 - 4374 + 13122 = 9842$

ZADATAK 8

Određivanje suma nizova.

Predviđeno vreme trajanja: 15 minuta

ZADATAK:
Odrediti vrednosti sledećih suma:
$a) \sum_{i=1}^{2} \sum_{j=1}^{3} (i + j)$
$b) \sum_{i=0}^{2} \sum_{j=0}^{3} (2i + 3j)$
$c) \sum_{i=1}^{3} \sum_{j=0}^{2} ij$
$d) \sum_{i=0}^{2} \sum_{j=1}^{3} ij$

REŠENJE:

$a) \sum_{i=1}^{2} \sum_{j=1}^{3} (i+j) = \sum_{i=1}^{2} (3i+6) = (3 \cdot 1 + 6) + (3 \cdot 2 + 6) = (3+6) + (6+6) = 9 + 12 = 21$

$b) \sum_{i=0}^{2} \sum_{j=0}^{3} (2i+3j) = \sum_{i=0}^{2} (8i + (3 \cdot 0 + 3 \cdot 1 + 3 \cdot 2 + 3 \cdot 3) = \sum_{i=0}^{2} 8i + 18 = 78$

$c) \sum_{i=1}^{3} \sum_{j=0}^{2} ij = \sum_{i=1}^{3} 3i = 3 \cdot 1 + 3 \cdot 2 + 3 \cdot 3 = 3 + 6 + 9 = 18$

$d) \sum_{i=0}^{2} \sum_{j=1}^{3} ij = \sum_{i=0}^{2} 6i = 0 + 6 + 12 = 18$

ZADATAK 9

Opis Euklidovog metoda.

Predviđeno vreme trajanja: 10 minuta

ZADATAK:

(a) Korišćenjem Euklidovog algoritma naći najveći zajednički delioc d brojeva 137 i 50.

(b) Korišćenjem Euklidovog algoritma naći najveći zajednički delioc d brojeva 414 i 662.

REŠENjE:
a)
137=2*50+37

50=1*37+13

37=2*13+11

13=1*11+2

11=5*2+1

2=2*1+0
NZD(137, 50) = 1

b)

662=414*1+248

414=248*1+166

248=166*1+82

166=82*2+2

82=2*41+0

NZD(414, 662)=2

ZADATAK 10

Euklidov algoritam.

Predviđeno vreme trajanja: 5 minuta

ZADATAK:

Korišćenjem Euklidovog algoritma naći najveći zajednički delioc d brojeva 252 i 198.

REŠENjE:
252 = 1*198 + 54
198 = 3*54 + 36
54 = 1*36 + 18
36 = 2*18 + 0
NZD (252, 198) = 18

Poglavlje 8

Zadaci za samostalni rad

ZADACI

Zadaci za provežbavanje

Zadatak 1 - predviđeno vreme trajanja 10 minuta
Odrediti niz koji se sastoji iz sledećih elemenata. Pretpostavljajući da je vaša formula za niz tačna, odredite naredna tri elementa niza
1,0,2,0,4,0,8,0,16,0...

Zadatak 2 - predviđeno vreme trajanja 10 minuta

Korišćenjem Euklidovog algoritma naći najveći zajednički delioc d brojeva 196 i 154.

Zadatak 3 - predviđeno vreme trajanja 10 minuta

Korišćenjem Euklidovog algoritma naći najveći zajednički delioc d brojeva 845,175.

ZAKLJUČAK

U ovoj lekciji su prikazana osnovna svojstva algoritama.

Fokus predavanja je bio na nizovima, rekurzivno definisanim funkcijama, fibonačijevom nizu i faktorijelu broja, kao i numeričkim algoritmima.

Cilj predavanja je da se shvati princip i svrha algoritama.

Literatura

[1] Rosen, Kenneth H. "Discrete mathematics and its applications." AMC 10 (2007): 12.
[2] Epp, Susanna S. Discrete mathematics with applications. Cengage Learning, 2010.