ALGORITMI I STRUKTURE PODATAKA

II čas

1 Analiza algoritama

1.1 Oznake *O, Ω, Θ, o*

Neka su *f* i *g* nenegativne funkcije f,g:$\mathbb{N}\to\mathbb{R}$. Kažemo da je ***g(n) = O(f(n))*** ako postoje pozitivne konstante *c* i *N* tako da $\forall\ n \geq N$ važi: $g(n) \leq c \cdot f(n)$.

Takođe važi:
$O(f(n)) + O(g(n)) = O(f(n) + g(n))$
$O(f(n)) \cdot O(g(n)) = O(f(n) \cdot g(n))$

Oznaka O(f(n)) se u stvari odnosi na klasu funkcija, a jednakost g(n)=O(f(n)) je oznaka za inkluziju g(n)∈O(f(n)).

Primer: $5n^2 + 8 = O(n^2)$
Jer je $5n^2 + 8 <= 6n^2$ za n>=3

Ovim dobijamo da možemo zanemarivati multiplikativne konstante, tj**. O(3n+2) pišemo kao O(n)**

Dalje, osnova logaritma nije bitna, jer je $\log_a n = \log_a (b^{\log_b n}) = \log_b n \underbrace{\log_a b}_{const}$

Uočimo i da O(1) je oznaka za klasu ograničenih funkcija.

Kako svaka eksponencijalna funkcija sa osnovom > 1 raste brže od svakog polinoma, onda za svaku monotono rastuću funkciju f(n) i brojeve a>1, c>0 važi: $f(n)^c=O(a^{f(n)})$

Specijalno za f(n)=n, dobija se: $n^c=O(a^n)$, odnosno stepenovanjem, dobija se: $(\log_a n)^c = O(n)$

Kažemo da je nenegativna funkcija $f(n)$ asimptotska donja granica nenegativne funkcije $g(n)$ i pišemo ***g(n) = Ω(f(n))*** ako postoje pozitivne konstante *c* i *N* tako da $\forall\ n \geq N$ važi: $g(n) \geq c \cdot f(n)$.

Ako za funkcije $f(n)$ i $g(n)$ istovremeno važi: $f(n) = O(g(n))$ i $f(n) = \Omega(g(n))$ onda pišemo ***f(n) =Θ(g(n)).***

1. Dokazati da važi: **$T(n)=2n^2+n-1 = \Theta(n^2)$**
DOKAZ: Potrebno je dokazati da postoje **c_1, c_2, N**, $\forall\ n \geq N$ važi: $c_1n^2 <= 2n^2+n-1 <= c_2n^2$, to jest

$$c_1 <= 2+\frac{1}{n}-\frac{1}{n^2} <= c_2$$

Poslednja nejednakost važi za $c_1=2$, $c_2=3$, N=1

Slično, važi i da:
$T(n)=2n+3 = \Theta(n)$
$T(n)=10+3\log n = \Theta(\log n)$
$T(n)=2^n + n^3 -100 = \Theta(2^n)$

2. Dokazati da važi: $17n \log n - 23n - 10 = \Theta(n \log n)$
DOKAZ: Potrebno je dokazati da postoje **c_1, c_2** tako da: $c_1 n \log n <= 17n \log n - 23n - 10 <= c_2 n \log n$, to jest deljenjem sa n log n

$$c_1 <= 17 - \frac{23}{\log n} - \frac{10}{n \log n} <= c_2$$

Poslednja nejednakost važi za $c_1=4$, $c_2=17$, N=4 (proverite neposredno)

Ako za funkcije $f(n)$ i $g(n)$ važi: $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$, pišemo: $f(n) = o(g(n))$.

3. Odrediti vreme izvršavanja sledećeg programskog fragmenta. Rešenje prikazati polinomijalnim izrazom i u O notaciji.

```
i=0;  k=1;
while (k<=n)
{
   k=2*k;
   i=i+1;
}
```

Rešenje: T(n)=1+1+log n + 2*log n + 2*log n =2 + 5*log n = O(log n)
Zašto? Odredite broj izvršenih operacija u svakoj liniji prethodnog programskog fragmenta.

Broj operacija u liniji 1 je 2. Zašto?
Broj operacija u liniji 2 je log n. Zašto? Koliko puta se ponavlja while ciklus i poređenje k<=n?
Pomoć: Uslov za izlazak iz ciklusa je k>n.
Kako se menja promenljiva k? Na početku k=1.
Nakon prvog ponavljanja cilkusa k=2*1=2
Nakon drugog ponavljanja cilkusa k=2*2=4
Nakon m-tog ponavljanja cilkusa $k=2*2^{m-1}=2^m$

Uslov za izlazak iz ciklusa je $k=2^m>n$,
odnosno nakon m-tog koraka kada se dobije vrednost $2^m>n$, tj. kada postane $m > \log_2 n$

Koliki je broj operacija u linijama 4,5?

4. **(Za vežbu)** Odrediti vreme izvršavanja sledećeg programskog fragmenta. Rešenje prikazati polinomijalnim izrazom i u O notaciji.

```
for (i=1, j=0; i<=n; i<<1) j++;
```

5. Odredite vremensku slozenost sledeceg fragmenata programskog kôda.
Broj koraka prikazati u obliku polinomijalnog izraza i O notaciji.

```
if (x == 0)
  for (i=0; i<=n-1; i++)
     for (j=0; j<n; j++) A[i][ j] = 0;
else for (i=0;i<=n-1;i++) A[i][ i] = 1;
```

Princip matematičke indukcije

6. Dokazati:

(a) $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$

(b) $\sum_{i=0}^{n} q^i = \frac{q^{n+1} - 1}{q - 1}$

(Za vežbu)

(c) $\sum_{i=1}^{n} i \cdot 2^i = (n-1)2^{n+1} + 2$

(d) $\sum_{i=1}^{n} i \cdot 2^{n-i} = 2^{n+1} - n - 2$

(e) Dokazati da važi: $\lceil \log n \rceil = O(n)$

Podsećanje: Koji rezultat vraćaju funkcije floor(x), ceil(x) u programskom jeziku C, ako x=4.7?

Neka $x \in \mathbb{R}$. Važi da: **$x-1 < \lfloor x \rfloor <= x <= \lceil x \rceil < x+1$**

Na primer, $\lfloor 7.51 \rfloor = 7$, $\lceil 7.51 \rceil = 8$. Koliko je $\lfloor 4 \rfloor$? Koliko je $\lceil 4 \rceil$?

Neka $n \in \mathbb{N}$. Važi da: **$\lfloor n/2 \rfloor + \lceil n/2 \rceil = n$**

1.2 Rekurentne relacije

7. Problem Pn sa parametrom **n** (n je prirodan broj) algoritam A rešava za T(n) vremenskih jedinica. Ako se zna da za svaki prirodan broj **n** važi T(n+2)=3T(n+1)-2T(n), gde je T(1)=3, T(2)=5, odrediti koliko vremenskih jedinica utroši algoritam A za ulaznu vrednost **n**

Rešenje Karakteristična j-na je: $t^2-3t+2=0 <=> (t-1)(t-2)=0 => t_1=1, t_2=2$.
Tada je rešenje oblika $T(n)=C_1*t_1^n+C_2*t_2^n$
Iz početnih uslova: za n=1: $3=T(1)=C_1*1+C_2*2$

Za n=2: $5=T(2)=C_1*1+C_2*2^2 => C_1=1, C_2=1$

Zato: $T(n)=2^n+1$

8. Problem Pn sa parametrom **n** (n je prirodan broj) algoritam A rešava za T(n) vremenskih jedinica. Ako se zna da za svaki prirodan broj **n** važi T(n+2)=4T(n+1)-4T(n), gde je T(1)=6, T(2)=20, odrediti koliko vremenskih jedinica utroši algoritam A za ulaznu vrednost **n**

Rešenje Zbog dvostrukog korena se ovaj zadatak razlikuje od prethodnog.

Karakteristična j-na je: $t^2-4t+4=0 <=> (t-2)^2=0$

Tada se rešenje traži u obliku $T(n)=C1*t_1^n+C_2*n*t_1^n$

Dakle za $t_1=2$ rešenje je oblika $T(n)=C_1\ 2^n+C_2\ n\ 2^n$

Iz početnih uslova: za n=1: $6=T(1)=C_1*2+C_2*2$

Za n=2: $20=T(2)=C_1*4+C_2*8 => C_1=1, C_2=2$

Zato $T(n)=2^n+2n*2^n$

9. Rešiti diferencnu jednačinu: $T(n) = 5T(n-1) - 8T(n-2) + 4T(n-3)$, $n \geq 4$
za koju važi $T(1) = 2$, $T(2) = 4$, $T(3) = 12$.

Rešenje: $T(n)=4 + (-2+n)2^n$

10. Rešiti rekurentnu jednačinu: $T(n) = 3T(n-1) + 2$, $n \geq 2$ za koju važi $T(1) = 1$.

Rešenje: $T(n) = 2*3^{n-1}-1$ Zašto?

Naime,
$T(n) = 3T(n-1) + 2$
$T(n-1) = 3T(n-2) + 2$

==========
Oduzimanjem sledi: T(n)-T(n-1)=3T(n-1)-3T(n-2)
Tj. dobija se homogena j-na: T(n) -4T(n-1)+3T(n-2)=0, $T(1) = 1$

Jednačine zasnovane na dekompoziciji

U slučaju kada se obrada ulaza veličine **n** svodi na **a** obrada ulaza veličine $(\frac{n}{b})$ i potrebno je izvršiti još $c \cdot n^k$ koraka,

jednačina izgleda ovako: $T(n) = aT(\frac{n}{b}) + c \cdot n^k$ gde *a, b, c, k >=0, b !=0* i dato je T(1).

Master teorema: Rešenje diferencne jednačine: $T(n) = aT(\frac{n}{b}) + c \cdot n^k$ gde $a, b, c, k >= 0, b \neq 0$ je:

$$T(n) = \begin{cases} O(n^{\log_b a}), a > b^k \\ O(n^k \log n), a = b^k \\ O(n^k), a < b^k \end{cases}$$

Primer: T(1)=1, T(n)= 2 T(n/2) + O(n)
Kako je a=2, b=2, a=b (tj. k=1), onda je po master teoremi: T(n) = O (n log n)

11. Odrediti asimptotsko ponašanje rešenja $T(n)$ diferencne jednačine:

$$T(n) = T(\frac{n}{2}) + \sqrt{n}, \quad n \geq 2$$

za koju važi $T(1) = 1$. Može se smatrati da n uzima samo vrednosti jednake stepenima dvojke.

Rešenje: a=1, b=2, k=$\frac{1}{2}$ => a<b^k => po master teoremi T(n)=O(n^k)=O($\sqrt{n}$)

12. Neka je data rekurentna jednačina za vreme izvršavanja T(n) algoritma BINARNA_PRETRAGA:
T(1)=1,
T(n)=T(⌊n/2⌋) + 1, n>1
Dokažite indukcijom po n da je za svako n>=1 ispunjeno T(n)=1+⌊log n⌋

Rešenje:
Baza: osnovni slučaj n=1 je zadovoljen, jer je T(1)=1+⌊log 1⌋=1+0=1
Induktivna hipoteza (IH): Pretpostavimo da je rekurentna jednacina tacna za +⌊n/2⌋.
Dokazimo da je tacna za n>1.

Dakle, po IH vazi da T(⌊n/2⌋)=1+⌊log ⌊n/2⌋⌋

Onda je: T(n)= T(⌊n/2⌋) + 1 =
(po IH) = 1+⌊log ⌊n/2⌋⌋ +1
= 1+⌊log ⌊n/2⌋ +1⌋
= 1+⌊log ⌊n/2⌋ +log 2⌋
= 1+⌊log 2*⌊n/2⌋ ⌋ (***)

Ako je n paran broj, onda vazi da je 2*⌊n/2⌋=⌊2*n/2⌋, te se neposrednim izvodjenjem u (***) dobija da je tada
T(n)=1+⌊log n⌋

Ako je n neparan broj, onda vazi da je $2*\lfloor n/2 \rfloor = \lfloor 2*n/2 \rfloor - 1$, te se neposrednim izvodjenjem u (***) dobija da je tada $T(n)=1+\lfloor \log(n-1) \rfloor$. Kako vazi da $\lfloor \log(n) \rfloor \neq \lfloor \log(n-1) \rfloor$ samo kada je broj n jednak stepenu broja 2, onda za neparno n vazi da $\lfloor \log(n) \rfloor = \lfloor \log(n-1) \rfloor$.
Odatle sledi da za neparno n vazi: $T(n)=1+\lfloor \log(n-1) \rfloor = 1+\lfloor \log(n) \rfloor$ QED