Dizajn i analiza algoritama

Lekcija 5

leto 2019/2020

Prof. dr Branimir M. Trenkić

Asimptotsko vreme izvršavanja

- ***Dosadašnji postupak*** za određivanje vremena izvršavanja algoritma:
  1. Bazira se na analizi koja se sastoji od ***prebrojavanja svih*** jediničnih ***instrukcija***
  2. Primenjiv u slučaju ***jednostavnijih algoritama***
  3. Zbog visokog nivoa "detaljisanja" - kod složenih algoritama ***sprečava da se stekne prava slika o brzini izvršavanja***
- ***Rešenje*** je u ***pojednostavljenju analize*** – na taj način možemo steći pravi uvid u složenost algoritma

Asimptotsko vreme izvršavanja

- ***Pravci pojednostavljenja*** analize
  – ***Ne interesuje nas*** apsolutno ***tačno vreme izvršavanja*** nekog algoritma
  – ***Interesuje nas*** samo ***koliko brzo raste*** vreme izvršavanja algoritma ***povećanjem*** veličine ***ulaza*** algoritma
- Na ovaj način se dobija ***asimptotsko vreme izvršavanja algoritma*** – vreme izvršavanja samo za veliki broj ulaznih podataka

Asimptotsko vreme izvršavanja

- ***Prilikom analize*** ***dva*** algoritama možemo dobiti vrlo ***komplikovane funkcije*** za njihova vremena izvršavanja:

$$T_1(n) = 10n^3 + n^2 + 40n + 80$$

$$T_2(n) = 17nlogn - 23n - 10$$

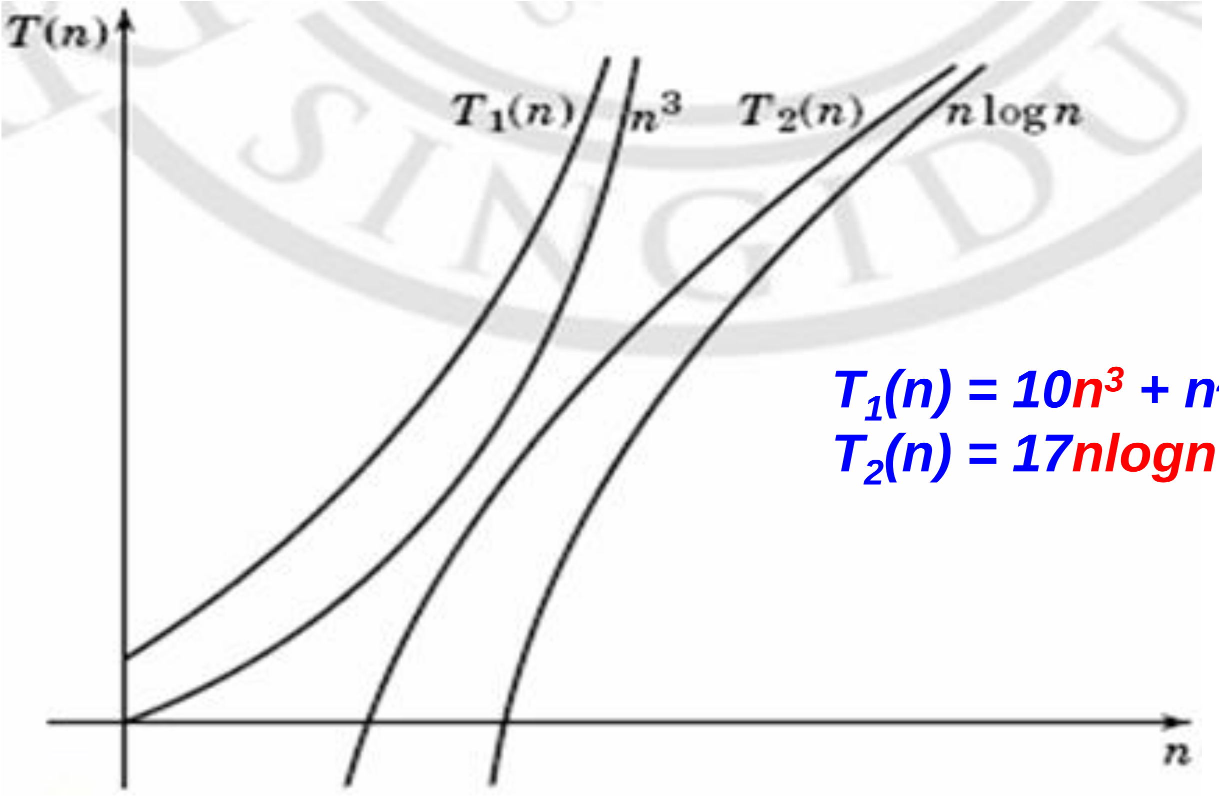
- Da li nam ***složenost*** ovih izraza ***zamagljuje sliku*** o odgovarajućim algoritmima?
- Na primer, ***koji je*** **od ova dva algoritma** ***brži***?

Asimptotsko vreme izvršavanja

- ***Rešenje 1***:
- Možemo uvek tačno ***nacrtati uporedo funkcije*** $T_1(n)$ i $T_2(n)$ i ***analizirati grafikon***
- ***Crtanje*** i izračunavanje složenih matematičkih funkcija – ***nije jednostavno***!
- ***Rešenje 2***:
- ***Pojednostavljenje*** – tražimo samo ***red veličine funkcije*** vremena izvršavanja algoritma ***za velike vrednosti ulaza***
  - Za ***velike vrednosti n*** $T_1(n)$ i n^3 ($T_2(n)$ i ***nlogn***) imaju **približno jednake vrednosti**!

Asimptotsko vreme izvršavanja

- Kažemo da je funkcija ***$T_1(n)$*** ***<u>reda veličine</u>*** ***n^3*** i da je funkcija ***$T_2(n)$*** ***reda veličine*** ***nlogn***



$T_1(n) = 10n^3 + n^2 + 40n + 80$

$T_2(n) = 17nlogn – 23n - 10$

Asimptotsko vreme izvršavanja

- Drugim rečima, ***za vreme izvršavanja*** nekog algoritma ***uzimamo jednostavnu funkciju*** koja ***za velike vrednosti*** ulaza ***najbolje <u>aproksimira</u>*** ***<u>tačnu funkciju</u>*** vremena izvršavanja tog algoritma
- To vreme izvršavanja – ***<u>asimptotsko vreme izvršavanja</u>***
- Predstavlja ***<u>meru brzine rasta</u>*** tačnog vremena izvršavanja algoritma sa povećanjem broja ulaznih podataka

Asimptotsko vreme izvršavanja

- ***Opravdanost pojednostavljenja***:

1. ***Brzina*** algoritma je naročito ***bitna*** kada je broj ulaznih podataka ***velik***
   - ***Svaki algoritam*** je obično ***efikasan*** za ***male ulazne podatke***
2. Ovo pojednostavljenje je ***matematički korektno***
   - ***Vrednost funkcije*** vremena izvršavanja (za velike vrednosti n) je preovlađujuće ***određena dominantnim članom***
3. Jednostavna funkcija se uzima ***bez konstanti***

Asimptotsko vreme izvršavanja

- $\boxed{\text{Asimptotsko vreme izvršavanja se ne određuje tako što se prvo odredi tačno vreme izvršavanja}}$
  - pa se onda dobijena ***funkcija uprošćava*** (recimo, uzimanjem dominantnog člana)
- ***To smo upravo hteli da izbegnemo!***

Pojednostavljena analiza

- ***Analiza je slična ranijom*** – rukovodimo se ***tabelom vremenske složenosti*** osnovnih algoritamskih konstrukcija
- Ali,

1. ***Zanemaruje se*** uticaj algoritamskih konstrukcija čije je vreme izvršavanja - ***konstantno***
2. ***Zanemaruju se*** ***ne-dominantni članovi*** dobijeni u izrazu (koji ne utiču na red veličine)
3. Jednostavna funkcija se uzima ***bez konstanti***

Pojednostavljena analiza – Primer1

- Inicijalizacija matrice ***a*** u ***jediničnu matricu***
- Algoritamski fragment:

T(n) = v.i. petlje(1) + v.i. petlje(4)

Petlja(1):
Broj iteracija ***spoljne*** – ***n***
Broj iteracija ***unutrašnje*** – ***n***
Vreme izvršavanja tela – ***konstantno***
Zanemarujemo ga! (za veliko n)

Petlja(4):
Broj iteracija – ***n***
Vreme izvršavanja tela – ***konstantno***
Zanemarujemo ga!

```
1 for i = 1 to n do
2     for j = 1 to n do
3         a[i,j] = 0;
4 for i = 1 to n do
5     a[i,i] = 1;
```

Dakle, ukupno vreme ***T(n) je reda*** $n \cdot n + n = n^2 + n$
Izraz možemo dalje uprostiti izvlačenjem dominantnog člana
Zaključujemo – ***T(n) je*** ***reda veličine*** n^2

Pojednostavljena analiza – Primer2

- Inicijalizacija matrice ***a*** u ***jediničnu matricu***
- Algoritamski fragment:

```
for i = 1 to n do
   for j = 1 to n do
      if (i == j) then
         a[i,j] = 1;
      else
         a[i,j] = 0;
```

Pojednostavljena analiza – Primer2

- Inicijalizacija matrice a u jediničnu matricu
- Algoritamski fragment:

```
1 for i = 1 to n do
2     for j = 1 to n do
3         if (i == j) then
4             a[i,j] = 1;
5         else
6             a[i,j] = 0;
```

T(n) = v.i. petlje(1)

Petlja(1):

Broj iteracija ***spoljne*** – ***n***

Broj iteracija ***unutrašnje*** – ***n***

Vreme izvršavanja if naredbe (3-6) – ***konstantno***

Zanemarujemo ga! (za veliko n)

Zaključujemo – ***T(n) je reda veličine*** n^2

$$T(n) \sim n^2$$

Pretraga sortiranog niza

- Ako su dati niz ***a*** od ***n*** brojeva ***sortiranih u rastućem redosledu*** $a_1 \le a_2 \le \cdots \le a_n$ i neki broj ***x***, treba ***odrediti da li se broj x nalazi u nizu a***

- ***Ukoliko je to slučaj***, ***rezultat*** treba da bude ***indeks k*** (prvog) elementa a_k u nizu ***a*** koji je jednak traženom broju ***x***
- ***U suprotnom slučaju***, ***rezultat*** treba da bude ***0***

Binarna pretraga sortiranog niza

- Posmatramo ***specijalan slučaj*** opšteg ***problema pretrage*** (***neuređenog niza*** brojeva)
- ***Specifičnost*** – dati ***niz brojeva a*** je ***sortiran*** u rastućem redosledu
- Ovu specifičnost možemo ***iskoristiti da pretragu*** niza znatno ***ubrzamo***
- Postupak koji primenjujemo i ovom slučaju naziva se ***binarna pretraga***
- Sličan je postupku ***traženja neke osobe u telefonskom imeniku***

Binarna pretraga sortiranog niza

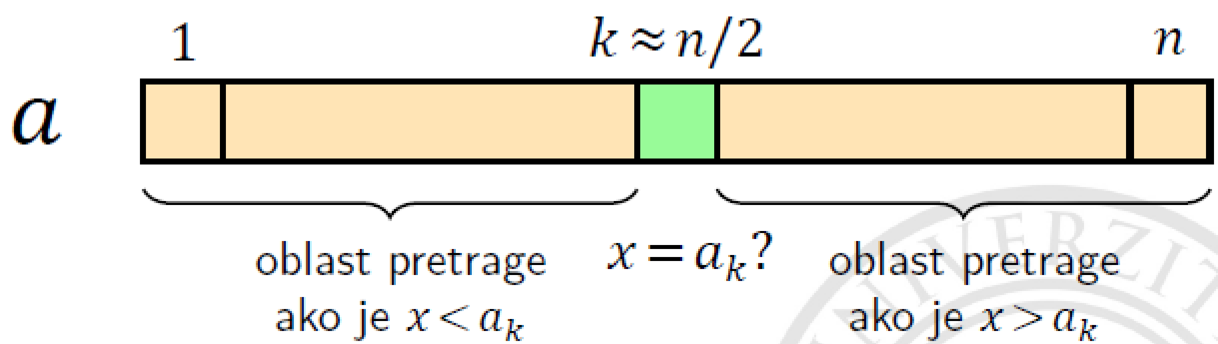
- ***Početni korak*** u algoritmu – pravilno ***inicijalizovati*** ***oblast pretrage*** niza
  - Oblast pretrage je ograničena
    - s donje strane – indeksom ***i***
    - s gornje strane – indeksom ***j***
- Na početku, ***oblast pretrage*** se proteže na ***ceo niz***

Binarna pretraga sortiranog niza

- ***Iterativni postupak***
- ***Izlazni uslov***: Sve dok donji indeks ne premaši gornji, (***i ≤ j***)
- ***Izračunavamo*** ***indeks k*** ***srednjeg elementa*** oblasti pretrage

Binarna pretraga sortiranog niza

- ***Ispituje se*** taj ***srednji element*** a_k
- Ako je $a_k \neq x$ – pretraga se ***nastavlja***
  a) Ako je $a_k > x$ – pretraga se nastavlja u ***prvoj polovini oblasti pretrage***
  b) Ako je $a_k < x$ – pretraga se nastavlja u ***drugoj polovini oblasti pretrage***
- Ako je $a_k = x$ – pretraga se prekida i vraća se rezultat ***k***



Binarna pretraga sortiranog niza

- ***Algoritam*** u pseudo kodu:

```
// Ulaz:  sortiran niz a, njegov broj elemenata n, broj x
// Izlaz: k takvo da x = a_k, ili 0 ako x nije u nizu a
algorithm bin-search(a, n, x)

   i = 1; j = n;      // donji i gornji indeks oblasti pretrage
   while (i <= j) do
      k = (i + j)/2; // indeks srednjeg elementa
      if (x < a[k]) then
         j = k - 1;  // x se možda nalazi u prvoj polovini
      else if (x > a[k]) then
         i = k + 1;  // x se možda nalazi u drugoj polovini
      else
         return k;   // x je nađen
   return 0;         // x nije nađen
```

Binarna pretraga sortiranog niza

- Analiza (***asimptotskog***) ***vremena izvršavanja algoritma***
- Dovoljno je ***samo odrediti*** ***vreme izvršavanja while*** petlje
- ***Ostali delovi*** algoritma se izvršavaju u ***konstantnom*** vremenu
- ***Vreme izvršavanja*** while petlje je proporcionalno ***broju iteracija*** te petlje
- ***Zaključak***: Treba ***naći broj iteracija*** while petlje u najgorem slučaju

Binarna pretraga sortiranog niza

- ***Najgori slučaj*** – vrednost ***x*** se ***ne nalazi u nizu a***

  Koliki je najveći broj iteracija while petlje u tom slučaju?

- Koristimo ***postupak*** koji ***se bazira*** na uspostavljanju ***veze između*** ***dužine oblasti pretrage*** i ***rednog broja iteracije***
- ***Početna*** dužina oblasti pretrage ***= n***
- ***Ukupan broj*** iteracija while petlje ***= m***

Binarna pretraga sortiranog niza

- na početku ***1. iteracije*** – ***dužina*** oblasti pretrage iznosi ***n***;
- na početku ***2. iteracije*** – ***dužina*** oblasti pretrage iznosi otprilike $n/2$ = $n/2^1$;
- na početku ***3. iteracije*** – dužina oblasti pretrage iznosi otprilike $(n/2)/2$ = n/4 = $n/2^2$;
- i tako dalje........
- na početku poslednje ***m-te iteracije*** – ***dužina oblasti pretrage*** iznosi otprilike $n/2^{m-1}$;

Binarna pretraga sortiranog niza

- Znamo još i to da ***dužina oblast pretrage*** na ***početku poslednje*** m-te iteracije ***iznosi 1 element***
- Prema tome,

$$\Rightarrow \quad \frac{n}{2^{m-1}} = 1$$

$$\Rightarrow \quad n = 2^{m-1}$$

logaritam sa osnovom 2

$$\Rightarrow \quad m = \log n + 1$$

$$\Rightarrow \quad T(n) \approx m \approx \log n$$

Tipične funkcije vremena izvršavanja

- ***Standardne funkcije*** kojima se najčešće ***opisuje vreme izvršavanja*** algoritma
- Navedene su redom ***odozgo na dole*** po rastućim brzinama rasta za velike vrednosti ulaza

Funkcija	**Neformalno ime**
1	konstantna funkcija
$\log n$	logaritamska funkcija
n	linearna funkcija
$n \log n$	linearno-logaritamska funkcija
n^2	kvadratna funkcija
n^3	kubna funkcija
2^n	eksponencijalna funkcija

Asimptotsko vreme izvršavanja - Primer

- Neka je dato ***T(n)*** za ***četiri*** algoritama

  A_1, A_2, A_3 i A_4

$$A_1 : \quad T_1(n) = 2n^2 + n - 1$$
$$A_2 : \quad T_2(n) = 2n + 3$$
$$A_3 : \quad T_3(n) = 10 + 3\log n$$
$$A_4 : \quad T_4(n) = 2^n + n^3 - 100$$

Asimptotsko vreme izvršavanja - Primer

- Interesuje nas ***T(n)*** za ***<u>velike</u>*** vrednosti ***n***

$\Longrightarrow$

- Vršimo ***asimptotsku analizu*** vremena izvršavanja algoritama:

$$T_1(n) = 2n^2 + n - 1 \qquad \approx n^2$$

$$T_2(n) = 2n + 3 \qquad \approx n$$

$$T_3(n) = 10 + 3\log n \qquad \approx \log n$$

$$T_4(n) = 2^n + n^3 - 100 \qquad \approx 2^n$$

Asimptotsko vreme izvršavanja - Primer

- Ako sada pretpostavimo da se nad svakim elementom ulaza izvršava samo ***jedna elementarna operacija*** u trajanju ≈ $1\mu s$

n	$\log n$	n	n^2	2^n
10	$1\ \mu s$	$10\ \mu s$	$100\ \mu s$	$1\ ms$
100	$2\ \mu s$	$100\ \mu s$	$10\ ms$	$2^{70}\ g$
1000	$3\ \mu s$	$1\ ms$	$1\ s$	$2^{970}\ g$

Asimptotska notacija

- Iz prethodnog primera ***zaključujemo***:
  - Da ***algoritme*** ***uporeďujemo*** prema njihovim ***funkcijama asimptotskog*** vremena izvršavanja
  - Što je takva ***funkcija "manja"*** to je ***algoritam brži*** (i obrnuto)
- Obično je ***jednostavno utvrditi*** da li je neka funkcija "manja" od druge (tabela tipičnih funkcija)
- Za ***složene slučajeve*** je potrebno imati ***precizniju definiciju*** – šta znači tvrdnja da je neka funkcija "manja" od druge?