Dizajn i analiza algoritama

Lekcija 6

leto 2019/2020

Prof. dr Branimir M. Trenkić

Asimptotska notacija

- ***Asimptotska notacija*** omogućava upravo da se na ***precizan način*** uvede ***relativni poredak*** za funkcije
- Notacija ima ***četiri oblika***:
  - ***O-zapis*** (veliko o)
  - ***Ω-zapis*** (veliko omega)
  - ***Θ-zapis*** (veliko teta)
  - ***o-zapis*** (malo o)
- U praksi se najčešće koristi ***O-zapis***

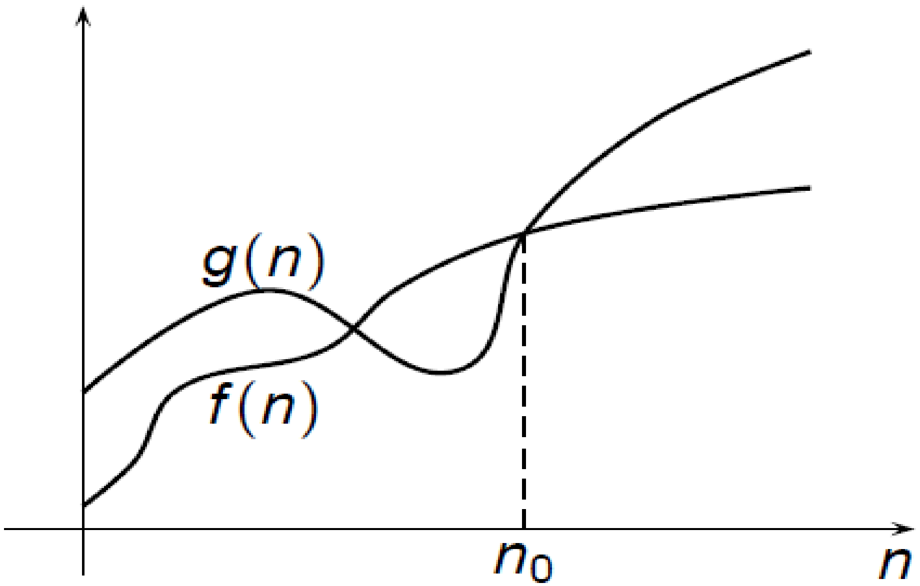
Asimptotska notacija

Asimptotska notacija	**Intuitivno značenje**
$f(n) = O(g(n))$	$f \leq g$
$f(n) = \Omega(g(n))$	$f \geq g$
$f(n) = \Theta(g(n))$	$f \approx g$
$f(n) = o(g(n))$	$f \ll g$

O-zapis

- ***Definicija.*** Za dve nenegativne funkcije ***f*** i ***g***

$$f(n) = O(g(n)) \text{ ako } \exists c, n_0, \forall n \geq n_0, f(n) \leq c \cdot g(n)$$



O-zapis označava da funkcija ***g(n)*** predstavlja ***asimptotsku gornju granicu*** za funkciju ***f(n)***

O-zapis

- ***Interpretacija*** u kontekstu vremena izvršavanja algoritma
- Neka funkcija ***T(n)*** predstavlja – ***vreme izvršavanja nekog algoritma***
- Neka je $g(n) = n^2$
- Ukoliko uspemo ***da pokažemo*** da je $T(n) = O(g(n))$ - pokazali smo zapravo (zanemarujući konstante) da je ***vreme izvršavanja algoritma*** sigurno ***ograničeno kvadratnom funkcijom***
- O-zapisom se dobija samo ***gornja granica vremena izvršavanja***

O-zapis

- Kako pokazati da je ***T(n) = O(g(n))***?

ili, uopšteno

- ***Kako naći dominirajuću funkciju*** (***g(n)***) za dato ***T(n)***?

Dva načina:

A. Možemo ***koristiti formalnu definiciju O-zapisa***
   – Postupak ubrzati korišćenjem ***opštih osobina O-zapisa***

B. Primena ***pravila limesa***

O-zapis

- ***Korišćenje formalne definicije O-zapisa***

- Dakle, ***treba naći*** konkretne vrednosti za $c > 0$ i n_0 takve da zadovoljavaju relaciju:

$$T(n) \leq cg(n) \text{ za svako } n \geq n_0$$

O-zapis

- Kako pokazati da je ***T(n) = O(g(n))***?

Primer: za *bubble-sort* $T(n) = n^2 - n + 1$

$$n^2 - n + 1 \le n^2 + 1 \le n^2 + n^2 = 2n^2$$

$$\Rightarrow \quad n^2 - n + 1 \le 2n^2$$

$$\Rightarrow \quad T(n) = O(n^2), n_0 = 1, c = 2$$

O-zapis

Ili ***uopšteno***:

Tvrdnja:

Izrazi manjeg reda u ***polinomskom zbiru*** nisu važni

ili,

Ako se funkcija vremena izvršavanja algoritma može predstaviti u obliku polinomskog zbira,

$$T(n) = a_k n^k + a_{k-1} n^{k-1} + \ldots\ldots + a_1 n + a_0$$

tada je,

$$T(n) = O(n^k)$$

O-zapis

<u>Dokaz</u>:

Neka je $n_0 = 1$ i c je ***zbir*** svih ***pozitivnih*** koeficijenata $a_0, a_1, \ldots\ldots, a_k$ za svako $i \le k$ i $n \ge 1$

$$T(n) = \sum_{i=0}^{k} a_i n^i = \sum_{a_i>0} a_i n^i + \sum_{a_i<0} a_i n^i \le \sum_{a_i>0} a_i n^i \le \sum_{a_i>0} a_i n^k = n^k \sum_{a_i>0} a_i = cn^k$$

$$\Rightarrow$$

$$T(n) = O\left(n^k\right)$$

Svako n^i zamenimo sa n^k
n^k – najveći stepen

O-zapis

- ***Praktični postupak* dobijanja dominirajuće funkcije** može se znatno ubrzati ukoliko se iskoriste ***opšte osobine O-zapisa*:**

1. ***Dominantni term je najvažniji***

- Ako je ***T(n) = f(n) + g(n)*** i ***g(n) = O(f(n))***

$\Longrightarrow$ ***T(n) = O(f(n))***

f(n) je dominirajuća u odnosu na g(n)

Na primer: $n^2 - n + 1 = O(n^2)$

O-zapis

<u>2. Konstantni faktori nisu važni</u>

- Dakle, važi ***T(n) = O(cT(n))*** za svaku pozitivnu konstantu ***c > 0***

Na primer: $2n^2 = O(n^2)$

O-zapis

- Takođe važe i ***sledeća svojstva***:
  - Pravilo zbira:
    $O\big(f(n) + g(n)\big) = O(f(n)) + O(g(n))$
  - Pravilo proizvoda:
    $O\big(f(n) \cdot g(n)\big) = O(f(n)) \cdot O(g(n))$
  - Pravilo tranzitivnosti:
    $T(n) = O(f(n)), f(n) = O(g(n)) \Rightarrow$
    $T(n) = O(g(n))$

O-zapis - Primeri

Primer:

$$T(n) = 3n + 17\log n$$

$$T(n) = O(n) + O(\log n) \qquad \log(n) = O(n)$$

$$= O(n)$$

1. Zanemarivanje konstanti
2. Pravilo o dominantnom članu

Primer:

$$T(n) = 2n + 6n^2$$

$$T(n) = O(n) + O(n^2) \qquad n = O(n^2)$$

$$= O(n^2)$$

1. Zanemarivanje konstanti
2. Pravilo o dominantnom članu

O-zapis - Primeri

Ili u opštem slučaju:

$$T(n) = T_1(n) + T_2(n),$$

$$T_1(n) = O(f(n)),\ T_2(n) = O(g(n)) \text{ i } g(n) = O(f(n))$$

T(n) = ?

Postoje konstante n1, n2, n3, c1, c2, c3

(i) ako $n \geq n_1$, tada $T_1(n) \leq c_1 f(n)$;

(ii) ako $n \geq n_2$, tada $T_2(n) \leq c_2 g(n)$;

(iii) ako $n \geq n_3$, tada $g(n) \leq c_3 f(n)$.

O-zapis - Primeri

Neka je $n_0 = \max\{n_1, n_2, n_3\}$

(i), (ii) i (iii) važi za $n \geq n_0$.

$$\begin{aligned}
T(n) &= T_1(n) + T_2(n) \\
&\leq c_1 f(n) + c_2 g(n) \\
&\leq c_1 f(n) + c_2 c_3 f(n) \\
&= (c_1 + c_2 c_3) f(n) \\
&= c f(n).
\end{aligned}$$

$$T(n) = O(f(n)).$$

Ostale notacije - Definicije

- **Veliko-Omega**

$$f(n) = \Omega(g(n)) \text{ ako } \exists c, n_0, \forall n \geq n_0, f(n) \geq c \cdot g(n)$$

- **Veliko-Teta**

$$f(n) = \Theta(g(n)) \text{ ako } f(n) = O(g(n)), f(n) = \Omega(g(n))$$

- **Malo-o**

$$f(n) = o(g(n)) \text{ ako } \lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$$

Asimptotska notacija

- ***Ne mora*** se uvek ***polaziti*** ***od formalne definicije*** kako bi se pokazalo da data funkcija zadovoljava neku ***asimptotsku granicu***
- Postoji ***drugi način*** – pomoću graničnih vrednosti

Teorema (***Pravilo limesa***):

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = c > 0 \Rightarrow f(n) = \Theta(g(n))$$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = c \geq 0 \Rightarrow f(n) = O(g(n))$$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} \neq 0 \Rightarrow f(n) = \Omega(g(n))$$

Asimptotska notacija

- ***Pravilo limesa***
- Skoro uvek ***lakše primeniti*** nego ***formalnu definiciju***
- Može se primenjivati ***u skoro svim*** praktičnim ***primerima*** funkcija vremena izvršavanja
- Izuzeci su ***funkcije koje nemaju graničnu vrednost*** (na primer, $f(n) = n^{\sin n}$)

NZD problem

- Ovaj problem smo ***već rešavali*** na nivou ***dizajna algoritma***
- Došli smo do ***algoritma*** (mada ***neefikasnog***) koji rešava ovaj problem – ***gcd(x, y)***
- Ostali smo dužni ***analize vremena izvršavanja ovog algoritma*** – kako bi ga uporedili sa efikasnijim algoritmom (***Euklidov algoritam***)

NZD problem

<u>Analiza vremena izvršavanja gcd algoritma</u>

- <u>**Proporcionalno broju iteracija while petlje**</u>
- Ostele naredbe za **konstantno** vreme
- **Broj iteracija?**
  - ***Zavisi*** od ulaznih brojeva ***x i y***
  - <u>Najbolji slučaj</u>: ***x = y***, ***broj iteracija = 0***
  - Ako je x ≠ y, ***broj iteracija = min(x,y) – 1***
  - <u>Najgori slučaj</u>: ***x i y se razlikuju za 1***

```
// Ulaz:  pozitivni celi brojevi x i y
// Izlaz: nzd(x,y)
algorithm gcd(x, y)

  d = min{x,y};
  while ((x % d != 0) || (y % d != 0)) do
    d = d - 1;

  return d;
```

- ***Zaključak***: vreme izvršavanja algoritma ***gcd*** je ***<u>proporcionalno</u> (linearno)*** manjem od dva broja

NZD problem

- ***Ključno pitanje***: Da li to znači da je ovaj algoritam ***brz***?
- Odgovor je ***negativan!***
- Ova kontroverza proističe iz ***ključnog pitanja za analizu*** aritmetičkih algoritama:

  Šta je veličina ulaza za ovaj algoritam?
- Kako smo do sada shvatali veličinu ulaza nekog algoritma – odgovor je 2!
- ***Zaključak***: ***pogrešno shvatanje*** veličine ulaza za algoritme ovoga tipa

NZD problem

- Činjenica koju ***ne smemo zanemariti***:
- Ako algoritam ***gcd*** primenimo na brojeve x i y koji imaju ***preko 200 cifara*** – ***ne možemo očekivati*** da se ***osnovne aritmetičke operacije*** nad njima mogu izvršiti za ***jednu vremensku jedinicu***

NZD problem

- U ***gcd*** algoritmu to su operacije:
  - ***Dodele***
  - ***Računanja po modulu***
  - ***Oduzimanja***

```
// Ulaz:  pozitivni celi brojevi x i y
// Izlaz: nzd(x,y)
algorithm gcd(x, y)

   d = min{x,y};
   while ((x % d != 0) || (y % d != 0)) do
      d = d - 1;

   return d;
```

- ***Vreme izvršavanja algoritama*** koji realizuju ove operacije ***zavisi od broja cifara*** operanada

NZD problem

- Možemo zaključiti:
- ***Prava mera veličine ulaza*** aritmetičkih algoritama ***jednaka je zbiru broja cifara*** (tačnije, ***broja bitova*** u binarnom zapisu)
- Sam broj ulaznih parametara nije dobra mera
  - Broj ulaznih podataka je konstantan (2) – što navodi na zaključak da je vreme izvršavanja isto bez obzira koje brojeve naveli kao ulazne
  - To je pogrešan zaključak – jer smo ***zanemarili složenost operacija nad velikim brojevima***

NZD problem

- Primenimo ovaj pristup u ***analizi gcd algoritma***
- n_1 – broj bitova binarnog zapisa broja x
- n_2 – broj bitova binarnog zapisa broja y
- Mera veličine ulaza - $n = n_1 + n_2$

- Sada, zavisno od ove prave mere veličine ulaza za algoritam gcd – ***možemo oceniti*** njegovo ***vreme izvršavanja*** u najgorem slučaju

NZD problem

- Račun baziramo na pravilu:

broj bitova binarnog zapisa svakog pozitivnog celog broja a = $\lfloor \log a \rfloor + 1$

Prema tome, broj bitova: ***$n_1 \approx \log x$*** i ***$n_2 \approx \log y$***

ili

$$x \approx 2^{n1} \quad \text{i} \quad y \approx 2^{n2}$$

U najgorem slučaju, ***$y = x - 1$*** $\Rightarrow$ ***$n_1 \approx n_2 \approx n/2$***

NZD problem

- Kako je ***vreme izvršavanja*** (***T(n)***) ***proporcionalno vrednosti manjeg*** od dva ulazna broja
  - Ako zanemarimo složenost izvršavanja aritmetičkih operacija

$$T(n) = \Omega(y) = \Omega\left(2^{n_2}\right) = \Omega\left(2^{n/2}\right)$$

- To pokazuje da se **algoritam izvršava za eksponencijalno vreme** i time ***beskoristan*** u slučaju velikih brojeva od 100-200 cifara (što je slučaj npr. u kriptografiji)

Euklidov algoritam

- ***Teorema*** (**Euklid**). Ako su ***x*** i ***y*** pozitivni celi brojevi takvi da je ***x*** $\geq$ ***y***, onda je

nzd(x, y) = nzd(y, x%y)

- ***Posledica***: ***svaki pozitivan ceo broj deli 0 bez ostatka***, t.j.

nzd(x, 0) = x

- ***NZD dva broja*** se lako može dobiti ***uzastopnim ponavljanjem jednakosti iz Euklidove teoreme***

Euklidov algoritam

- ***Primeri***

nzd(x, y) = nzd(y, x%y)

nzd(1055,15) = nzd(15,5) = nzd(5,0) = 5

nzd(400,24) = nzd(24,16) = nzd(16,8) = nzd(8,0) = 8

nzd(34,17) = nzd(17,0) = 17

Euklidov postupak – dati ***par brojeva*** se ***transformiše*** u niz parova

- ***Prvi broj*** u paru – ***jednak drugom*** u prethodnom paru
- ***Drugi broj*** u paru – ***jednak ostatku deljenja*** prvog sa drugim iz prethodnog para

$$(x_0, y_0) \to (x_1, y_1) \to (x_2, y_2) \to \cdots \to (x_m, y_m) \to (x_{m+1} = \mathrm{nzd}(x, y), y_{m+1} = 0)$$

Euklidov algoritam

- ***Algoritam*** (***iterativno*** rešenje)

```
// Ulaz:  celi brojevi x i y takvi da x ≥ y > 0
// Izlaz: nzd(x, y)
algorithm euclid1(x, y)

   while (y > 0) do
      z = x % y;
      x = y;
      y = z;

   return x;
```

$T(n) = O(n^3)$

Euklidov algoritam

- *Analiza vremena izvršavanja Euklidovog algoritma*
- ***Lema.*** Ako je ***x ≥ y > 0***, onda je ***x%y < x/2***

$$(x_0, y_0) \to (x_1, y_1) \to (x_2, y_2) \to \cdots \to (x_m, y_m) \to (x_{m+1} = \mathrm{nzd}(x, y), y_{m+1} = 0)$$

- ***(x_i,y_i)*** – par brojeva (promenljive x i y u ***while*** petlji) ***nakon i-te iteracije***
- ***m*** - ***ukupan broj iteracija while petlje***

Euklidov algoritam

- *Analiza vremena izvršavanja Euklidovog algoritma*
- Nakon ***prve iteracije***, $x_1 = y_0$ i $y_1 = x_0\%y_0$, na osnovu Leme:

$$x_1 \cdot y_1 = y_0 \cdot (x_0\%y_0) < x_0 \cdot y_0/2$$

- Nakon ***druge iteracije***, $x_2 = y_1$ i $y_2 = x_1\%y_1$, na osnovu Leme:

$$x_2 \cdot y_2 = y_1 \cdot (x_1\%y_1) < x_1 \cdot y_1/2 < x_0 \cdot y_0/2^2$$

$$x_m \cdot y_m = y_{m-1} \cdot (x_{m-1}\%y_{m-1}) < x_{m-1} \cdot y_{m-1}/2 < x_0 \cdot y_0/2^m$$

Euklidov algoritam

- *Analiza vremena izvršavanja Euklidovog algoritma*
- Pored toga, $x_m \cdot y_m \geq 1$

  $x_m \cdot y_m = x_{m-1} \cdot (x_{m-1} \% y_{m-1}) < x_{m-1} \cdot y_{m-1}/2 < x_0 \cdot y_0 / 2^m$

Ako spojimo ove dve nejednakosti - $xy/2^m > 1$

$$m < \log xy = \log x + \log y < n_1 + n_2 = n$$

Broj iteracija: $m = O(n)$

Operacija izračunavanja ostatka („najskuplja operacija") $= O(n^2) \Longrightarrow T(n) = O(n)O(n^2) = O(n^3)$