

Analiza efikasnosti algoritma

Merenje vremena izvršavanje algoritma

- Broj ulaznih podataka - n
 - npr. sortiranje niza od n članova
- Traži se vreme izvršavanja u zavisnosti od veličine ulaza, odnosno kako se vreme izvršavanja algoritma povećava sa povećavanjem broja ulaznih podataka
- Manja veličina ulaza daće i manje vreme izvršavanja algoritma
- Veća veličina ulaza daje veće vreme izvršavanja algoritma
- Povećava se veličina ulaza, vreme izvršavanja se takođe povećava
- Ako se ulaz u algoritam duplira
 - vreme izvršavanja se može duplirati
 - vreme izvršavanja se može uvećati 4 puta
 - vreme izvršavanja se može uvećati 100 puta

Merenje vremena izvršavanje algoritma

- Ekperimentalna metoda
 - Implementacija algoritma u programskom jeziku
 - Izvršavanje algoritama korišćenjem različitih ulaza
 - Beleženje vremena izvršavanja algoritma
 - Zavisi od softvera i hardvera
 - Konačan broj ulaza u algoritam
- Analitička metoda
 - Analiza vremena izvršavanja algoritma bazirana na veličini ulaza
 - Asimptotska analiza
 - Nezavisna od softvera i hardvera
 - Razmatra sve moguće ulaze u algoritam

Klasa Stopwatch

- Biblioteka System.Diagnostics
- Konstruktor: Stopwatch t = new Stopwatch();
- Metoda Start() startuje merenje vremenskog intervala
- Metoda Stop() zaustavlja merenje vremenskog intervala
- Metoda Reset() služi za reetovanje tajmera
- Svojstvo Elapsed vraća TimeSpan strukturu koja sadrži proteklo vreme
- Svojstvo ElapsedMilliseconds vraća broj milisekundi (ne koristiti ako je vreme kraće od ms)

```
Stopwatch t = new Stopwatch();  
t.Start();  
TestMetoda(1000);  
t.Stop();  
Console.WriteLine(t.ElapsedMilliseconds);
```

Primer merjenja vremena izvršavanja algoritma -1

```
static int[] KreirajNiz(int n)
{
    Random rnd = new Random();
    int[] x = new int[n];
    for (int i = 0; i < n; i++)
    {
        x[i] = rnd.Next(1,101); // od 1 do 100
    }
    return x;
}
```

```
static void PisiNiz(int[] x)
{
    for (int i = 0; i < x.Length; i++)
    {
        Console.Write(x[i] + "\t");
    }
    Console.WriteLine();
}
```

Primer merjenja vremena izvršavanja algoritma -2

```
static void Main(string[] args)
{
    // menjaj velicinu ulaza n =10, 100, 1000
    int[] x = KreirajNiz(1000);

    Stopwatch t = new Stopwatch();

    double[] vremena = new double[10];
    for (int i = 0; i < 10; i++)
    {
        // merim vreme 10 puta zbog preciznijeg rezultata
        t.Start();
        PisiNiz(x);
        t.Stop();
        Linija(100);
        vremena[i] = t.ElapsedMilliseconds;
        t.Reset();
    }

    double ukupnoVreme = 0;
    foreach (double vr in vremena)
    {
        ukupnoVreme += vr;
    }

    Console.WriteLine($"Proteklo vreme: { ukupnoVreme/10} ms");
    Console.ReadLine();
}
```

Rezultati merenja vremena izvršavanja metode PisiNiz()

Broj ulaznih parametara n	10	100	1000	10000
Vreme izvršavanja [ms]	0.1	10.3	117.2	1360.6

Napomena: Pri testiranju ne pokretati aplikaciju u debug modu nego sa CTRL + F5

Analitičko određivanje vremena izvršavanja algoritma

- Analizira se rad algoritma korak po korak i izvodi se vreme njegovog izvršavanja
- Ne dobija se apsolutno precizna ocena vremena izvršavanja
- Ocena je dovoljno dobra da se stekne opšta slika o efikasnosti algoritma
- Vreme izvršavanja algoritma je funkcija veličine ulaznih podataka
- Za n ulaznih podataka definiše se funkcija $T(n)$ koja daje broj vremenskih jedinica koliko traje izvršavanje algoritma

Jedinična instrukcija

- Jedinična instrukcija algoritma je osnovna računaska operacija čije je vreme izvršavanja konstantno
- Pretpostavlja se da se sve osnovne instrukcije algoritma izvršavaju za jednu vremensku jedinicu
- Apsolutna vrednost vremenske jedinice nije bitna (ms, μ s, ...)
- Tipične jedinične instrukcije algoritama su:
 - dodela vrednosti promenljivoj
 - poređenje vrednosti dve promenljive
 - aritmetičke operacije
 - logičke operacije
 - ulazno/izlazne operacije

Vreme izvršavanja ugnježenih petlji

```
for (int i = 0; i < n; i++) // izvrsava se n puta
{
    for (int j = 0; j < n; j++) // izvrsava se n puta
    {
        if (i > j) // jedinica instrukcija
        {
            x[i, j] = 1; // jedinica instrukcija
        }
    }
}
```

$$T(n) = n \cdot n \cdot (1 + 1) = 2n^2$$

Primer 1 – indeks najmanjeg elementa niza

n - broj članova niza x.Length

```
static int MinimalniClan(int[] x)
{
  1  int jmin = 0; // indeks minimalnog elementa
  2  int xmin = x[0];
  3  int i = 1;
  4  while (i < x.Length)
  {
  5      if (x[i] < xmin)
  {
  6          xmin = x[i];
  7          jmin = i;
  }
  8      i++;
  }
  9  return jmin;
}
```

Red	1	2	3	4	5 6 7	8	9
Vreme izvršavanja	1	1	1	n-1	3	1	1

$$\begin{aligned} T(n) &= 1 + 1 + 1 + (n-1)(3 + 1) + 1 \\ &= 4 + 4(n-1) \\ T(n) &= 4n \end{aligned}$$

Zadatak 1 – odrediti vreme izvršavanja algoritma

```
1 if (x == 0)
{
2   for (int i = 0; i < n; i++)
    {
3     a[i] = i;
    }
}
```

Red	1	2	3
Vreme izvršavanja	1	n	1

$$T(n) = 1 + n \cdot 1$$

$$T(n) = 1 + n$$

Zadatak 2 – odrediti vreme izvršavanja algoritma

```
1 int i = 1;
2 do
  {
3     a[i] = b[i] + c[i];
4     i++;
  }
5 while (i < n);
```

Red	1	2 5	3	4
Vreme izvršavanja	1	n-1	1	1

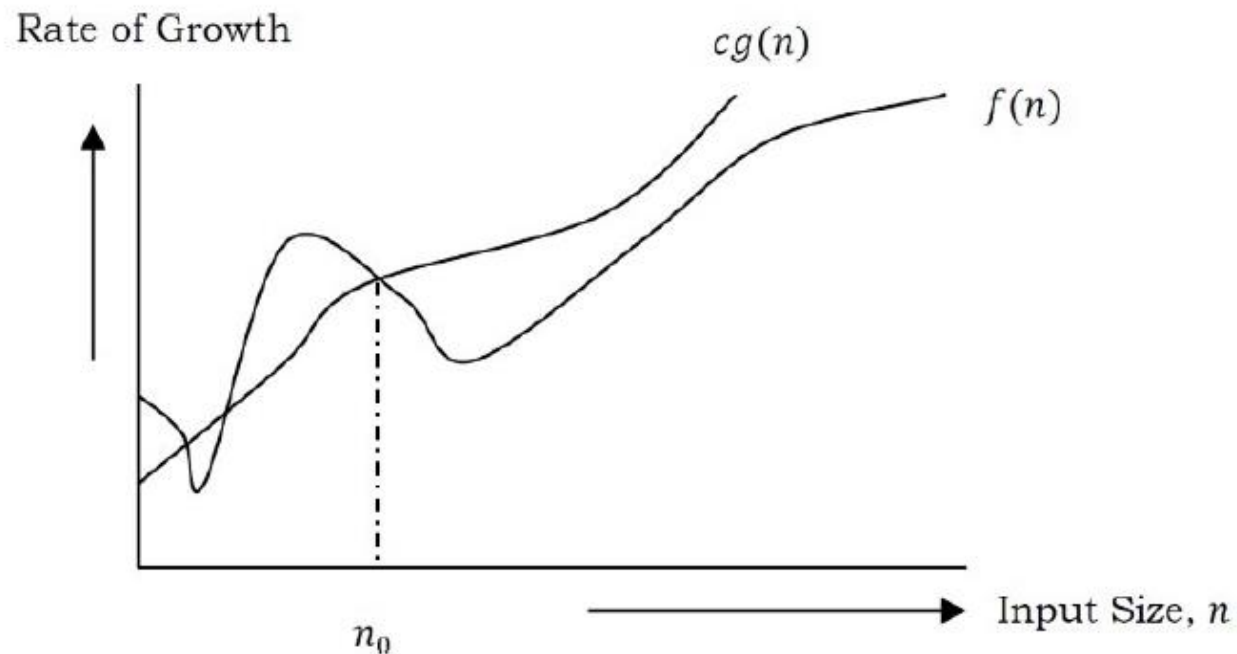
$$T(n) = 1 + (n-1)(1 + 1)$$

$$T(n) = 2n - 1$$

O-zapis

- O-zapis se koristi za precizno definisanje pojma da je neka funkcija manja od druge
- Za dve nenegativne funkcije $f, g: N \rightarrow R^+$ kažemo da je $f(n) = O(g(n))$ ako postoje pozitivne konstante c i n_0 tako da je $f(n) \leq c * g(n)$ za svako $n > n_0$
- Odnosno kažemo da funkcija $g(n)$ predstavlja asimptotsku granicu za funkciju $f(n)$

O-zapis



$$f(n) \leq c * g(n) \text{ za svako } n > n_0$$

$f(n)$ vreme izvršavanja algoritma u zavisnosti od broja ulaza n
Veliko O daje asimptotsku gornju granicu vremena izvršavanja algoritma
 $f(n) = O(g(n))$ // $g(n)$ je veliko O za $f(n)$

Tipične funkcije složenosti poredane po rastućim brzinama rada

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

Pronalaženje velikog O

- Ako je $f(n) = c$, $f(n) = O(1)$, veliko O za konstantnu funkciju je 1
- Ako je $f(n) = a_k n^k + a_{k-1} n^{k-1} + \dots + a_0$ onda je $f(n) = O(n^k)$
 - zadržava se najbrže rastuću član polinoma
- Koeficijenti se ignorišu $f(n) = k g(n)$ onda je $f(n) = O(g(n))$
- Baza logaritma je nebitna: $f(n) = 8 \log_2 n$, onda je $f(n) = O(\log n)$
- Ako je $f(n) = O(2^n)$ tj. veliko O za funkciju eksponencijalno, kažemo da je vreme izvršavanja algoritma eksponencijalno
 - Ne postoji polinom koji bi ga mogao ograničiti
 - Algoritmi koji zahtevaju eksponencijalno vreme izvršavanja mogu biti nerešivi u realnom vremenu

Primeri nalaženja gornje granice

$$f(n) \leq c * g(n) \text{ za svako } n > n_0$$

$$f(n) = 3n + 8$$

$$3n + 8 \leq 4n, n \geq 8$$

$$f(n) = O(n), c = 4, n_0 = 8$$

$$f(n) = n^2 + 1 \leq 2n^2, n \geq 1$$

$$f(n) = O(n^2), c = 2, n_0 = 1$$

$$f(n) = 2n^3 - 2n^2 \leq 2n^3, \text{ za } n \geq 1$$

$$f(n) = O(n^3), c = 2, n_0 = 1$$

Primeri određivanja velikog O

$$f(n) = 45$$

$$O(1)$$

$$f(n) = 6n^3 + 27\log_2 n + 2n$$

$$O(n^3)$$

$$f(n) = 8\log_2 n + 7n + 6$$

$$O(n)$$

$$f(n) = n \log_{10} n + 5n + 81n^2$$

$$O(n^2)$$

$$f(n) = \log_2 n + n \log_{10} n$$

$$O(n \log n)$$

$$f(n) = 3n + 5n^2 + 7n^3 + 2^n$$

$$O(2^n)$$



Asimptotsko vreme izvršavanja algoritama

```
for (int i = 0; i < n; i++)  
{  
    for (int j = 0; j < n; j++)  
    {  
        a[i, j] = 0;  
    }  
}  
  
for (int i = 0; i < n; i++)  
{  
    a[i, i] = 1;  
}
```

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

za veliko n dominira kvadratni član

$$T(n) = O(n^2)$$

Asimptotsko vreme izvršavanja algoritama

```
for (int i = 0; i < n; i++)  
{  
    for (int j = 0; j < n; j++)  
    {  
        if (i == j)  
        {  
            a[i, j] = 1;  
        }  
        else  
        {  
            a[i, j] = 0;  
        }  
    }  
}
```

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

$$T(n) = O(n^2)$$

Pitanje 1

Eksperimentalna metoda vremena izvršavanja algoritma:

- a. Ne zavisi od hardvera i softvera na kome se radi testiranje
- b. Zavisi od hardvera i softvera na kome se radi testiranje
- c. Zavisi isključivo od broja ulaznih parametara

Odgovor: b

Pitanje 2

Jedinična instrukcija algoritma ima:

- a. Promenljivo vreme izvršavanja
- b. Konstantno vreme izvršavanja
- c. Vreme izvršavanja koje zavisi od tipa algoritma

Odgovor: b

Pitanje 3

Ko asimptotske analize algoritma izračunava se tačno vreme izvršavanja algoritma:

- a. Da
- b. Ne

Odgovor: b

Pitanje 4

Šta je veliko O za funkciju : $f(n) = 3n^2 + 5n + 18$

- a. $O(2^n)$
- b. $O(n)$
- c. $O(n^2)$

Odgovor: c

Pitanje 5

Vremenska kompleksnost algoritma koji se izvršava za isto vreme bez obzira na broj ulaznih parametara je:

- a. $O(1)$
- b. $O(n)$
- c. $O(\log n)$

Odgovor: a

Pitanje 6

Koja je vremenska kompleksnost sledećeg koda:

```
for (int i = 0; i < n; i++)  
{  
    a[i] = 0;  
}
```

- a. $O(n^2)$
- b. $O(n)$
- c. $O(1)$

Odgovor: b