Kompleksnost algoritama – Primeri

1. O-notacija

Definicija. Neka su f i g aritmetičke funkcije, tada je f(n) = O(g(n))) kada $(n \to \infty)$ ako postoje brojevi c i n_0 takvi da je $f(n) \le c \cdot g(n)$ za svako $n > n_0$.

- Ako finkcija f(n) pripada O(g(n)), onda to znači da je funkcija f(n) reda funkcije g(n) ili funkcija g je asimptotska gornja granica funkcije f.
- Primer. $n^2, 3n^2, n^2 - 5000, 5n^2 + 3n, \ldots \in O(n^2)$

Definicija. Neka su f i g aritmetičke funkcije, tada je $f(n) = \Omega(g(n))$ kada $(n \to \infty)$ ako postoje brojevi c i n_0 takvi da je $f(n) \ge c \cdot g(n)$ za svako $n > n_0$

- Ako finkcija f(n) pripada Ω(g(n)), onda to znači da je brzina rasta funkcije f(n) proporcionalna brzini rasta funkcije g(n) ili veća.

Defnicija. Funkcije f i g rastu istom brzinom, u oznaci $f(n) = \Theta(g(n))$ kada $(n \to \infty)$, ako postoje brojevi c_1, c_2 i n_0 takvi da je $c_1 \cdot f(n) < g(n) < c_2 \cdot f(n)$ za svako $n > n_0$, tj. ako je f(n) = O(g(n))) i g(n) = O(f(n))).

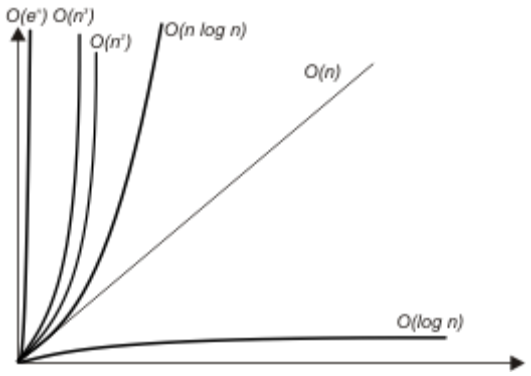
- Primer. $(n+1)^2 = \Theta(3n^2)$, $\left(1 + \frac{3}{n}\right)^n = \Theta(1)$

Definicija. Neka su f i g aritmetičke funkcije, tada je $f(n) = o(g(n))$ kada $(n \to \infty)$ ako postoji $\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$, tj. ako za svako c postoji n0 takvo da je $f(n) < c \cdot g(n)$ za svako $n > n_o$

- Ako finkcija f(n) pripada o(g(n)), onda to znači da f(n) raste brzinom koja je sigurno manja od brzine rasta funkcije g(n) kada n raste.
- Primer. $n^2 = o(n^5)$, $\sin(n) = o(n)$, ...

Definicija. Neka su f i g aritmetičke funkcije, tada je $f(n) \sim g(n)$ kada $(n \to \infty)$ ako postoji $\lim_{n \to \infty} \frac{f(n)}{g(n)} = 1$, tj. ako za svaku konstantu c postoji n_0 takvo da je $\left| \frac{f(n)}{g(n)} - 1 \right| < c$ za svako $n > n_0$

- Primer. $n^2 + n \sim n^2$, $\sin\frac{1}{n} \sim \frac{1}{n}$



Odnos brzina rasta funkcija

2. Suma elemenata niza

```
Suma(A, n)
    suma = 0
    for i = 1, n
        suma = suma + A(i)
```

- Na početku su inicijalizovane 2 promenljive (***i, suma***) – 2 instrukcije
- U svakom od koraku unutar petlje se izvrše 2 dodele (***i, suma***) – 2***n*** instrukcija
- Ukupan broj instrukcija je:

$$F(n) = 2 + 2n = O(n)$$

3. Suma svih podnizova koji počinju od prve pozicije

```
SumaSub(A, n)
    for i = 1, n
        suma = A(1)
        for j = 2, i
            suma = suma + A(j)
        write(suma)
```

- Na početku je inicijalizovano ***i*** – 1 instrukcija
- Spoljašnja petlja se izvršava ***n*** puta, a pri svakom koraku se izvrši štampanje, inicijalizacija za promenljive ***suma*** i ***j***, uvećanje vrednosti za ***i*** – 4***n*** instrukcija
- Unutrašnja petlja se izvršava ***n***-1 puta. Svako izvršavanje ima ***i***-1 korak, a u svakom koraku se dodeljuju vrednosti za promenljive ***suma*** i ***j*** – 2(***i***-1) instrukcija
- Ukupan broj instrukcija:

$$\begin{aligned} F(n) &= 1 + 4n + \sum_{i=2}^{n} 2(i-1) \\ &= 1 + 4n + 2(1 + 2 + \cdots + n - 1) \\ &= 1 + 4n + n(n-1) \\ &= O(n) + O(n^2) \\ &= O(n^2) \end{aligned}$$

4. Suma poslednjih 5 elemenata podnizova koji počinju od prve pozicije

```
SumaPosl5(A, n)
   for i = 5, n
      suma = A(i-4)
      for j = i-3, i
         suma = suma + A(j)
      write(suma)
```

- Za svaku vrednost promenljive ***i*** unutrašnja petlja se izvrši 4 puta
- U svakoj iteraciji spoljašnje petlje ima 11 dodela vrednosti (za ***i***, ***j***, ***suma***) i ovaj broj operacija ne zavisi od dužine niza
- Ukupan broj instrukcija:

$$F(n) = 1 + 11 \cdot (n - 4) = O(n)$$

5. Binarno pretraživanje niza

```
BinSearch(a, n, key)
   lo = 1
   hi = n
   while (lo <= hi)
      mid = (lo + hi) / 2
      if (key < A(mid)) hi = mid - 1
      else if (key > A(mid)) lo = mid + 1
      else return mid   // Trazeni element niza je sa indeksom mid
   return 0             // Trazeni element nije pronadjen
```

- Ako je tražena vrednost u sredini niza, petlja će biti izvršena samo jedanput
- U prvom prolazu algoritam razmatra niz dužine ***n***, u narednom prolazu niz dužine $\frac{n}{2}$, u sledećem prolazu $\frac{n}{2^2}$, i tako dalje sve do dužine 1.
- Dobijen je niz vrednosti $n, \frac{n}{2}, \frac{n}{2^2}, \dots, \frac{n}{2^m}$, gde poslednja vrednost mora da bude 1, pa je $m = \frac{\log n}{\log 2}$.
- U najgorem slučaju, da ***key*** nije element niza, biće utvrđeno za $\log n$ iteracija.
- Maksimalan broj iteracija:

$$F(n) = O(\log n)$$

6. „Brzo“ množenje dve kvadratne matrice $n \times n$

$$\begin{pmatrix} c_{11} & \cdots & c_{1n} \\ \vdots & \ddots & \vdots \\ c_{n1} & \cdots & c_{nn} \end{pmatrix} = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \cdot \begin{pmatrix} b_{11} & \cdots & b_{1n} \\ \vdots & \ddots & \vdots \\ b_{n1} & \cdots & b_{nn} \end{pmatrix}$$

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} \qquad (i, j = 1,2, \ldots n)$$

```
MatrProd(A, B, n)
   for i = 1, n
      for j = 1, n
         C(i,j) = 0
         for k = 1, n
            C(i,j) = C(i,j) + A(i,k)*B(k,j)
```

- Pri izvršavanju treće petlje se obavi ***n*** množenja i ***n***-1 sabiranje.
- Treća petlja se izvrši n^2puta, što je ukupno n^3množenja i $n^3 - n^2$sabiranja

7. „Brzo“ množenje dve kvadratne matrice $n \times n$

- Pri množenju matrica 2×2 potrebno je izvršiti sledećih 11 koraka

$$\begin{aligned} Q_1 &= (a_{12} - a_{22}) \cdot (b_{21} - b_{22}) \\ Q_2 &= (a_{11} + a_{22}) \cdot (b_{11} + b_{22}) \\ Q_3 &= (a_{11} - a_{21}) \cdot (b_{11} + b_{12}) \\ Q_4 &= (a_{11} + a_{12}) \cdot b_{22} \\ Q_5 &= a_{11} \cdot (b_{12} - b_{22}) \\ Q_6 &= a_{22} \cdot (b_{21} - b_{11}) \\ Q_7 &= (a_{21} + a_{22}) \cdot b_{11} \end{aligned}$$

$$\begin{aligned} c_{11} &= Q_1 + Q_2 - Q_4 + Q_6 \\ c_{12} &= Q_4 + Q_5 \\ c_{21} &= Q_6 + Q_7 \\ c_{22} &= Q_2 - Q_3 + Q_5 - Q_7 \end{aligned}$$

- Ovaj algoritam se može primeniti i na matrice *n*×*n*. Neka je $n = 2^k$ i neka su date dve *n*×*n* matrice *A* i *B*. Matrice *A* i *B* se podele na 4 matrice dimenzije $2^{k-1} \times 2^{k-1}$ i dobijen proizvod je podeljen

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \cdot \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} \qquad (*)$$

- Svako veliko slovo označava matricu $2^{k-1} \times 2^{k-1}$. Navedeni koraci se sada primenjuju na matrice i svaka opreacija sabiranja i množenja koja se pojavljuje u ovim koracima se sada odnosi na matrice, a ne na brojeve.
- Ukoliko dimenzije polaznih matrice ne mogu da se pretstave kao $2^k \times 2^k$, dodaju im se vrste i kolone nula dok im nova dimenzija ne postane stepen od 2.

```
MatrProd(A,B:matrix, N):matrix
    if N nije stepen od 2 then
        Proširi A i B kolonama i vrstama nula i promeni N
    if N=1 then MatrProd=A*B
    else
        Izdeli A i B prema (*)
        Q1=MatrProd(A11-A22, B21+B22, N/2)
        Q2=MatrProd(A12+A22, B11+B22, N/2)
          ... i ostale formule
        C22=Q2-Q3+Q5-Q7
```

- Ukupna broj množenja označimo sa *f(k)*, za matrice dimenzije $2^k \times 2^k$. Funkcija `MatrProd` poziva samu sebe 7 puta i u svakom pozivu ima *f(k-1)* množenja i *f(0)=1*, tj.

$$f(k) = 7f(k-1),\ f(0) = 1 \quad \Rightarrow \quad f(k) = 7^k,\ \ k \geq 0$$

- Dakle, za dve matrice $2^k \times 2^k$ potrebno je 7^k množenja. Kada *k* izrazimo preko *n* dobija se

$$7^k = 7^{\frac{\log n}{\log 2}} = n^{\frac{\log 7}{\log 2}} < n^{2.81}$$

  tj. Strassenov algoritam koristi samo $O\left(n^{2.81}\right)$ množenja.

- Ukupan broj sabiranja, pri množenju dve $2^k \times 2^k$ matrice, označimo sa *g(k)*. Funkcija `MatrProd` poziva samu sebe 7 puta i ima 18 poziva funkcije za sabiranje/oduzimanje. To znači da je

$$\begin{aligned} g(k) &= 7g(k-1) + 18 \cdot 4^{k-1} \qquad g(0) = 0 \\ &= 7g(k-1) + \frac{9}{2}4^k \end{aligned}$$

- Uvođenjem smene $g(k) = 7^k y_k$, $k \geq 0$, $y_0 = 0$ dobija se

$$\begin{aligned} y_k &= y_{k-1} + \frac{9}{2}\left(\frac{4}{7}\right)^k \\ &= \frac{9}{2}\sum_{j=1}^{k}\left(\frac{4}{7}\right)^j \\ &\leq \frac{9}{2}\sum_{j=1}^{\infty}\left(\frac{4}{7}\right)^j \qquad \left(\sum_{i=1}^{\infty} q^i = \frac{1}{1-q}\right) \\ &= \frac{21}{2} \end{aligned}$$

$$g(k) = 7^k y_k \leq 10.5 \cdot 7^k = O(7^k)$$

- Dakle, Stassenov algoritam koristi samo $O\left(n^{2.81}\right)$ sabiranja.

8. Selection sort

```
Selectionsort(x, n)
    for i=1, n-1
        for j=i+1, n
            if x[j] < x[i] then swap(x[j], x[i])
```

- Ako se za jednicu mere kompleksnosti ovog algoritma uzme poređenje dva broja, tada je broj poređenja koja procedura pravi po jedno poređenje za svaku vrednost *j=i+1,...,n* u unutrašnjoj petlji, pa je ukupan broj poređenja

$$f_1(n)=\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}1=\sum_{i=1}^{n-1}n-i=(n-1)+(n-2)+\cdots+1=\frac{(n-1)n}{2}$$

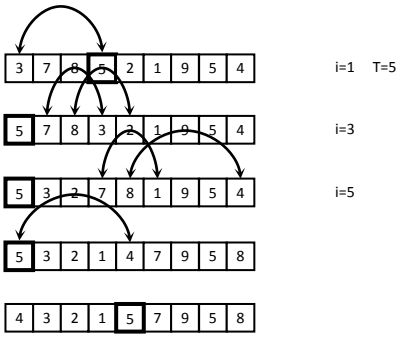
- Ovo znači da je broj poređenja $\Theta(n^2)$, nezavisno od ulaznog niza.

- Ako se za jedinicu mere uzme broj zamena, tada će u nekim slučajevima biti izvršeno $\frac{(n-1)n}{2}$ zamena, a drugim slučajevima neće biti zamena. U proseku sa svih *n*! mogućih poredaka, ako su svi ključevi različiti , biće izvršeno $\Theta(n^2)$ zamena.

9. Quick sort

- Algoritam je 1961. godine razvio C.A.R. Hoare. Ideja algoritma je “podeli i vladaj” strategija. Naime, u posmatranom nizu se uoči element *spliter* koji će nakon sortiranja niza ostati na istoj poziciji i svi elementi koji su manji od njega su levo od njega, a oni koji su veći su desno. Niz se deli na dva dela u odnosu na spliter i prvi i drugi deo niza se sortiraju nezavisno. Algortam je rekurzivan. Ukoliko spliter ne postoji u nizu (što je uglavnom slučaj) on se kreira.

```
Split(x, left, right, i)
    L = slučajan broj iz [left, right]
    swap(x[left],x[L])
    T = x[left]
    i = left
    for j = left+1, right
        if x[j]<T then i = i + 1
                       swap(x[i], x[j])
    swap(x[left], x[i])
```




```
Qsort(x, left, right)
    if right-left>1 then
        split(x, left, right,i)
```

```
        qsort(x, left, i-1)
        qsort(x, i+1, right)

Quicksort(x, n)
    qsort(x, 1, n)
```

- Ako je pri svakom izboru u podnizovima za splitra izabran najmanji (najveći) broj, tada se pri sortiranju podniza od *k* elemenata dobijaju nizovi dužine 0 i *k*-1. Ako polazni niz ima *n* elemenata i sa *L(n)* je označen broj poređenja, pri ovakvom izboru splitera, dobija se
$$L(n) = L(n-1) + n - 1, \quad n \geq 1, \quad L(0) = 0$$
gde je *L*(*n-1*) broj poređenja u dobijenom podnizu od *n-1* elemenata, a *n-1* broj poređenja pri postavljanju splitera. rešavanjem rekurentne veze, dobija se
$$L(n) = 1 + 2 + \cdots + (n-1) = \Theta(n^2)$$
- Dakle, u najgorem slučaju QuickSort ima isti broj poređenja kao SelectionSort.
- U opštem slučaju, pretpostavimo da su svi elementi niza različiti, što znači da ima *n!* mogućih ulaza. Sa F(n) označimo prosečan broj poređenja u QuickSort-u. F(n) sadrži dve komponente:
    - poređenje pri postavljanju splitera
    - poređenja u rekurentnim pozivima
- Broj poređenja pri postavljanu splitera u nizu od n elemenata je n-1.
- Neka je niz preuređen oko splitera i neka je spliter na poziciji i. Obziroma na način izbora splitera, svaka vrednost za i je jednako verovatna $\left(p = \frac{1}{n} \right)$ i rekurzivni pozivi se odnose na nizove dužine i-1 i n-i, sa prosečnim brojem poređenja F(i-1) i F(n-i). Ovo znači da je prosečan broj poređenja u nizu od n elemenata
$$F(n) = n - 1 + \frac{1}{n}\sum_{i=1}^{n}\left(F(i-1) + F(n-i)\right), \quad n \geq 1, \quad F(0) = 0$$
kako je
$$\sum_{i=1}^{n} F(n-i) = F(n-1) + F(n-2) + \cdots + F(0)$$
$$= \sum_{i=1}^{n} F(i-1)$$
dobija se
$$F(n) = n - 1 + \frac{2}{n}\sum_{i=1}^{n} F(i-1) \qquad /\cdot n$$
$$nF(n) = n(n-1) + 2\sum_{i=1}^{n} F(i-1)$$
- Zamenom n sa n-1 dobijamo:
$$(n-1)F(n-1) = (n-1)(n-2) + 2\sum_{i=1}^{n-1} F(i-1)$$
- Oduzimanjem prethodne dve jednakosti se dobija
$$nF(n) - (n-1)F(n-1) = n(n-1) - (n-1)(n-2) + 2F(n-1)$$

pa je

$$F(n)=\frac{n+1}{n}F(n-1)+2\frac{n-1}{n}$$

- Uvođenjem zamene $F(n)=\frac{n+1}{n}\cdot\frac{n}{n-1}\cdot\frac{n-1}{n-2}\cdots\frac{2}{1}y_n=(n+1)y_n$

$$(n+1)y_n=\frac{n+1}{n}ny_{n-1}+2\frac{n-1}{n}$$

$$y_n=y_{n-1}+2\frac{n-1}{n(n+1)}\ ,\quad n\geq 1\ ,\quad y_0=0$$

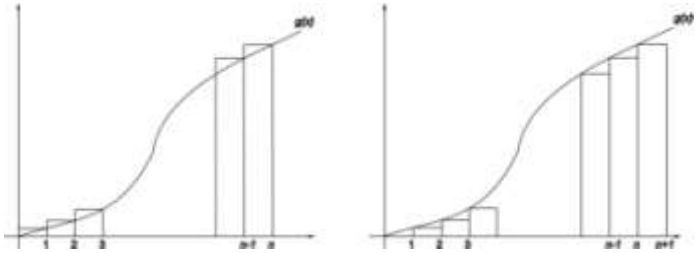
- Rešavanjem rekurentne formule dobija se

$$y_n=2\sum_{j=1}^{n}\frac{j-1}{j(j+1)}=2\sum_{j=1}^{n}\left(\frac{2}{j+1}-\frac{1}{j}\right)$$

$$=2\sum_{j=1}^{n}\frac{1}{j}-\frac{4n}{n+1}$$

- Konačno se dobija: $F(n)=2(n+1)\sum_{j=1}^{n}\frac{1}{j}-4n$

- Koristeći relaciju



$\int_0^n g(t)dt\leq\sum_{j=1}^{n}g(j)\leq\int_1^{n+1}g(t)dt$ za $g(t)=\frac{1}{t}$ dobija se

$$F(n)\sim 2n\ln n\quad (n\to\infty)$$