SLOŽENOST ALGORITAMA

def.
Algoritam je tačno definisana računarska procedura koja pretpostavlja neke vrijednosti kao ulaz i proizvodi neke vrijednosti kao izlaz.
Algoritam je dakle precizna i nedvosmislena specifikacija niza koraka koje se mogu mehanički (automatski) izvršiti.
Algoritam za neki definisani problem daje niz instrukcija kojima se korak po korak transformiva ulaz problema u njegov izlaz tako da bude zadovoljen traženi (predefinisani) ulazno-izlazni odnos.
Ispravan (korektan) algoritam završava rad ispravnim izlazom tj. algoritam rješava problem.

Analiza algoritama mjeri upotrebu računarskih resursa nekog algoritma.
Računarski resursi (koji se inače trebaju minimizivati) su: ⓐ vrijeme izvršavanja algoritma
ⓑ memorijska zauzetost algoritma
s tim u vezi se govori o vremenskoj i memorijskoj složenosti algoritma.

Nap.
Neki algoritam kome trebaju godine da bi završio svoj rad ili koji koristi nekoliko (stotina) GB radne memorije nije navočito koristan (efikasan) iako može biti potpuno ispravan!
U praktičnim slučajevima analize algoritama gotovo isključivo se proučava vremenska složenost algoritama tj. se da efikasnost algoritama jedino zavisi o njegovom vremenu izvršavanja.

Vrijeme koje je potrebno za izvršavanje algoritma zavisi od količine ulaznih podataka (veličine) koje on mora da obradi.

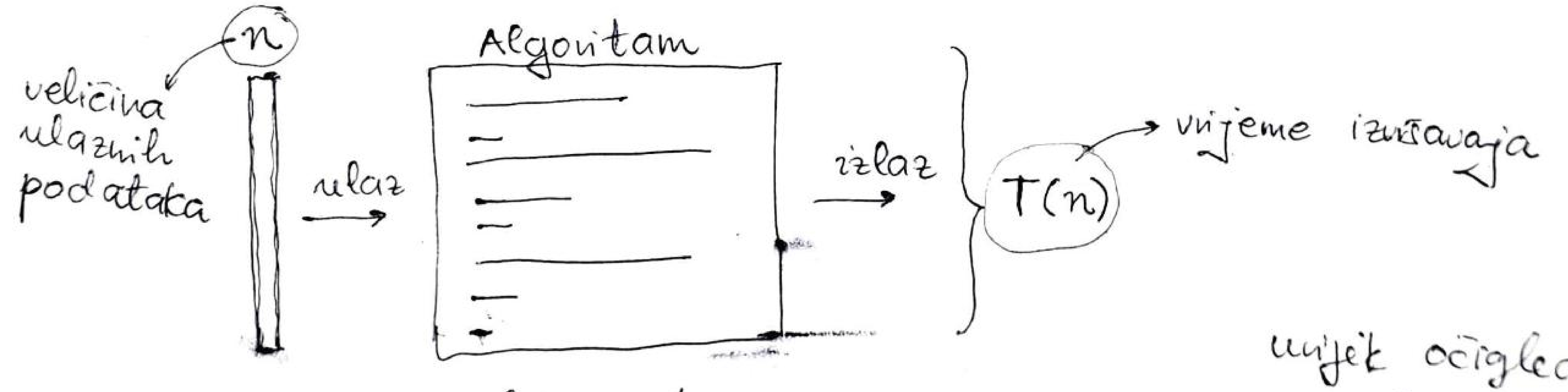
Primjer:

Algoritam za sortiranje 10000 brojeva zahtijeva puno više vremena od tog istog algoritma za sortiranje samo 10 brojeva!

Dakle, vrijeme izvršavanja nekog algoritma je direktna funkcija veličine ulaznih podataka.

Algoritmi za različite probleme mogu imati različite vidove veličine ulaza.

Npr.
- za problem sortiranja niza brojeva, veličina ulaza je broj članova niza
- za problem množenja dva broja, veličina ulaza je zbir broja cifara prvog i drugog broja
- za problem grafa mreže, veličina ulaza je broj čvorova i broj grana grafa itd.



slika 1.

uvijek očigledna!

Dakle, za svaki Algoritam se definiše veličina ulaza koja se predstavlja nenegativnim cijelim brojem n, a zatim se vrijeme izvršavanja algoritma predstavi preko odgovarajuće funkcije $T(n)$, slika 1.

Za svaki $n \geq 0$; $n \in \mathbb{Z}$, vrijednost $T(n)$ daje broj vremenskih jedinica koliko traje izvršavanje algoritma.

Pp. se da se sve osnovne operacije izvršavaju za jednaku (jediničnu/jednu) vremensku jedinicu.

Osnovne operacije:

- dodjela vrijednosti promjenljivoj ($x = 2$)
- aritmetičke operacije ($+, -, /, *$)
- poređenje dvije promjenljive ($>, \geq, <, \leq$)
- logičke operacije ($\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$)
- ulazne / izlazne operacije (read, return)

Za analizu vremenske složenosti algoritama tj. vremena izvršavanja je važan ukupan (zbirni) broj osnovnih operacija koje se izvršavaju a ne njihovo tačno ukupno vrijeme izvršavanja!

U praksi se funkcija vremena izvršavanja $T(n)$ dobije tako da se prosto prebroje (saberu) osnovne operacije koje se izvršavaju u algoritmu i to u najgorem slučaju!

Dakle, funkcija vremena izvršavanja u najgorem slučaju - $T(n)$ je maximalno vrijeme izvršavanja tog algoritma za veličinu ulaza n.

Ova funkcija $T(n)$ se još zove i vremenska složenost u najgorem slučaju.

Praktično se ovi termini kratko zapisuju samo kao vrijeme izvršavanja tj. vremenska složenost.

$T(n)$ → vrijeme izvršavanja u najgorem slučaju
$T(n)$ → vremenska složenost u najgorem slučaju
$T(n)$ → vrijeme izvršavanja
$T(n)$ → vremenska složenost

Dakle, najgori slučaj izvršavanja algoritma je onaj slučaj u kojem se izvršava najveći broj osnovnih operacija. 4

Primjer:

Imamo dva algoritma A1 i A2:

A1:	A2:
$n = 5$	read učitaj (n);
repeat ponavljaj	repeat ponavljaj
read učitaj (m);	read učitaj (m);
$n = n - 1$;	$n = n - 1$;
until sve dok ne bude $(m=0) \vee (n=0)$;	until sve dok ne bude $(m=0) \vee (n=0)$;
U ovom fragmentu algoritma imamo 5 iteracija petlje u najgorem slučaju	U ovom fragmentu alg. imamo n iteracija iste petlje u najgorem slučaju

Vrijeme izvršavanja složenijih konstrukcija se može izvesti na osnovu vremena izvršavanja njihovih sastavnih dijelova. U sljedećoj tabeli su data vremena izvršavanja osnovnih algoritamskih konstrukcija gdje uopšteno: T_X je vrijeme izvršavanja (u najgorem slučaju) bloka navedbi X.

Konstrukcija	Vrijeme izvršavanja
sekvenca navedbi S: P; Q	$T_S = T_P + T_Q$
uslovna navedba S: ~~ako~~ if (uslov) ~~tada~~ then P; ~~inače~~ else Q;	$T_S = \max\{T_P, T_Q\}$
For petlja S: for $i = 1$ ~~do~~ n ~~radi~~ do P;	$T_S = n \cdot T_P$

While / Repeat petlja S:
while (uslov) do
P;
repeat
P;
until (uslov);
	$T_S = m \cdot T_P$
	gdje je m – broj iteracija while/repeat petlje u najgorem slučaju

Primjeri : (zbir n prvih brojeva!) → $1, 2, \dots, n$

$$S = 1+2+3+\dots+n = \frac{n(n+1)}{2};$$

A1:

$S = n \cdot (n+1)/2$; — bila bi 5 da piše i return S

(return S)

$T(n) = 1+1+1+1 = 4 = \text{const.}$ ($T(n) = 5$ ako postoji return S)

1 - operacija sabiranja $n+1$

1 - operacija množenja $n \cdot (n+1)$

1 - operacija dijeljenja $n \cdot (n+1)/2$

1 - operacija dodjele vrijednosti

A2:

$S = 0;$ → 1

for $i = 1 : n$ do — u for se po def. ne broji dodjela!!!

  $S \underset{1}{=} S \underset{1}{+} i;$

(end for)

(return S)

u sklopu for petlje imamo $1+1 = 2$ operacije

↓ sabiranje

↓ dodjela vrijednosti

$i=1: \quad i=2: \quad \dots \quad i=n$

$$\underbrace{2\,op. + 2\,op. + \dots + 2\,op.}_{n-puta} = 2n$$

$T(n) = 2n + 1;$ ($T(n) = 2n+2$ ako postoji i return S)

↳ linearna fja!

Primjer : (srednja vrijednost članova niza $a_1, a_2, \dots, a_n$)

$suma = 0;$ 1

for $i = 1 : n$

  $suma \underset{1}{=} suma \underset{1}{+} a(i);$

(end for)

$average \underset{1}{=} suma / n;$ 1

Van for petlje:

$1 + 1 + 1$ → average = suma/n;

↓ suma = 0; → suma/n

unutar for petlje:

$(1 + 1 = 2) \cdot n$ → radi for petlje!

↓ suma + a(i) → suma = suma + a(i)

=> fja vrem. složenosti: $T(n) = 2 \cdot n + 1 + 1 + 1 = 2n + 3;$

↳ lin. fja!

Primjer: (anuliranje svih članova matice $(a_{ij})_{m\times n}$) 6.

$$\text{for } i = 1 : n \quad \text{nezavisni!}$$
$$\quad \text{for } j = 1 : n$$
$$\qquad \text{if } (i < j) \text{ then} \quad \text{Ne broji se!}$$
$$\qquad\quad a(i,j) = 0; \quad 1$$
$$\qquad \text{(end if)}$$
$$\quad \text{(end for)}$$
$$\text{(end for)}$$

$i < 1$

$a(i,j) = 0$

tijelo unutrašnje petlje ima $1 + \cancel{1} = 1$ op

unutrašnja petlja ima : $1 \cdot n$

vanjska petlja:

$i = 1 \;;\; i = 2 \;;\; \ldots \;;\; i = n$

$$\underbrace{2n + 2n + \ldots + 2n}_{n-\text{puta}} = 2n \cdot n = 2n^2$$

$$\Rightarrow \quad T(n) = 2 \cdot n \cdot n = 2n^2; \quad \text{(kvadratna f-ja!)}$$

$$\text{for } i = 1 : n \quad \text{zavisni!}$$
$$\quad \text{for } j = i : n$$
$$\qquad a(i,j) = 0; \quad 1$$
$$\quad \text{(end for)}$$
$$\text{(end for)}$$

$i = 1$:	$i = 2$	$i = 3$		$i = n$
$j = 1 : n$	$j = 2 : n$	$j = 3 : n$		$j = n : n$
$1 \cdot n$ +	$1 \cdot (n-1)$ +	$1 \cdot (n-2)$ +	$\ldots$ +	1

$$\Rightarrow \quad T(n) = \cancel{\frac{n(n-1)}{}}\; n + (n-1) + (n-2) + \ldots + 1 = \frac{n(n+1)}{2} =$$

$$= \frac{1}{2}n^2 + \frac{1}{2}n \quad \text{(kvadr. f-ja!)}$$

DZ.

Odrediti fje vremenske složenosti $T(n)$ za sljedeće algoritamske fragmente:

ⓐ

```
for i = 1 : n-1          → ne računati ove operacije u sklopu for petlji!
    for j = (i+1) : n
        a(i,j) = 1;
    end for
end for
```

ⓑ

```
for i = 1 : n
    for j = 1 : n
        if i < j then
            pom = a(i);
            a(i) = a(j);
            a(j) = pom;
        end if
    end for
end for
```

ⓒ (nalaženje max. elementa niza)

```
m = a(1);
j = 1;
i = 2;
while i ≤ n do          → ne računa se op. u sklopu petlje!
    if m < a(i) then
        m = a(i);
        j = i;
        i = i + 1;
    end if
end while
return j;
```

ⓓ

```
j = 0;
i = 1;
while i ≤ n do
    if a(i) ≥ 1 then
        j = i;
        i = i + 1;
    end if
end while
return j;
```

ⓔ

```
if x = 0 then
    for i = 1 : n
        a(i) = i;
    end for
end if
```

ⓕ

```
i = 1;
repeat
    a(i) = b(i) + 1;
    i = i + 1;
until i = n          → ne broji se!
```

ⓖ

```
if x = 0 then
    for i = 1 : n
        for j = 1 : n
            a(i,j) = 0;
        end
    end
else
    for i = 1 : n
        a(i,i) = 1
    end
end
```