PROBLEM SORTIRANJA

Problem:

Ulaz: niz (realnih) brojeva $\langle a_1, a_2, \ldots, a_n \rangle$

Izlaz: permutacija niza $\langle a_1, a_2, \ldots, a_n \rangle$:

$\langle a_{\pi(1)}, a_{\pi(2)}, \ldots, a_{\pi(n)} \rangle$ t.d.

$$a_{\pi(1)} \leq a_{\pi(2)} \leq \cdots \leq a_{\pi(n)}$$

Primjer: Zadan je niz brojeva $\langle 8, 2, 4, 9, 3, 6 \rangle$

Ulaz: $\langle 8, 2, 4, 9, 3, 6 \rangle$

Izlaz: $\langle 2, 3, 4, 6, 8, 9 \rangle$

$\langle 8, 2, 4, 9, 3, 6 \rangle$

$1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6$

$2 \quad 5 \quad 3 \quad 6 \quad 1 \quad 4$

$\langle 2, 3, 4, 6, 8, 9 \rangle$

$\pi(1) = 2 \qquad \pi(4) = 6$

$\pi(2) = 5 \qquad \pi(5) = 1$

$\pi(3) = 3 \qquad \pi(6) = 4$

- tijekom predavanja razmatramo dva pristupa, tj. dva algoritma za rješavanje problema sortiranja:
  - * INSERTION-SORT
  - * MERGE-SORT

INSERTION SORT ALGORITAM

- ideja: - slaganje karata u ruci
  - svakoj karti želimo pronaći odgovarajuću poziciju

Primjer:

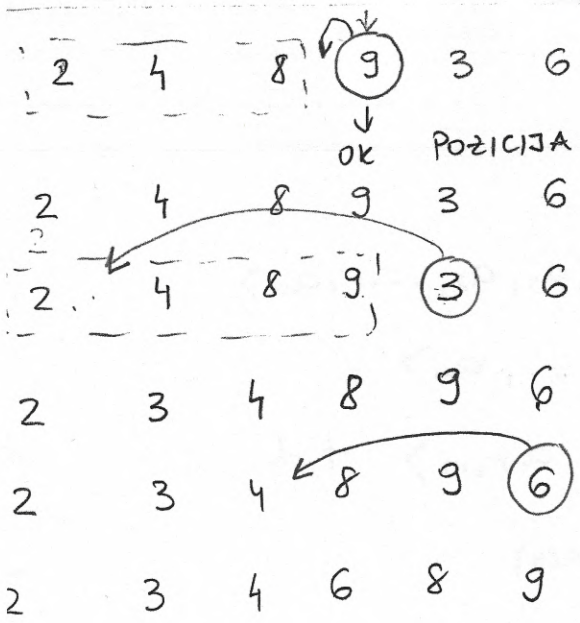
Ulaz: 8 2 4 9 3 6

8 | 2 | 4 9 3 6 ← ovdje pretpostavljamo da je podniz koji se sastoji od prvog elementa već sortiran

2 8 | 4 | 9 3 6

2 | 8 | 4 | 9 3 6

2 4 | 8 | 9 3 6



Zadatak: Napisati pseudokod INSERTION-SORT ALGORITMA

Upute: koristiti "strukturu podataka" polja A (Uvod u računarstvo)

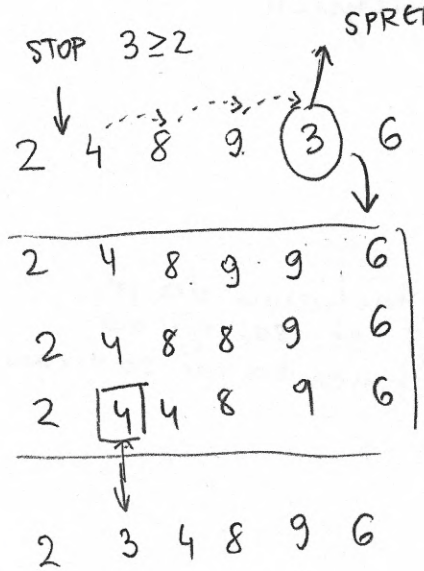
```
for j ← 2 to length(A)
    ▷ umetnuti element A[j] u sortirano polje A[1.... j-1]
end for
```

ovo je samo komentar: idemo to implementirat

Pogledajmo na jedan od koraka u algoritmu:

2 4 8 9 (3) 6

* pronaći poziciju
* napraviti pomak elemenata u već sortiranom polju → kojih elemenata? Onih koji su veći od 3



PSEUDOKOD:

```
do key ← A[j]
   i ← j-1
   while i>0 and A[i] > key
   | do A[i+1] ← A[i]
   |    i ← i-1
   end while
   A[i+1] ← key
```

kompletan pseudokod:

```
INSERTION-SORT(A)
for j = 2 to length(A)        ← duljina polja A, uglavnom se označava s n, tj. length(A) = n
    do key ← A[j]
    ▷ umetnuti element A[j] u sortirano polje A[1,...,j-1]
    i  i ← j-1
       while i > 0 and A[i] > key
           do A[i+1] ← A[i]
              i ← i-1
       end while
       A[i+1] ← key
end for
```

MERGE-SORT ALGORITAM

-ideja: - podjeli pa vladaj
- rekurzivan pristup
- podijeli: problem na manje potprobleme
- vladaj: riješi potprobleme rekurzivno
  ↓ BOTTOM
  tj: rješavaj sve dok ne dođeš do potproblema koji je dovoljno malen i tijek rješenje je trivijalno
- kombiniraj: rješenja potproblema u rješenje originalnog problema

UP

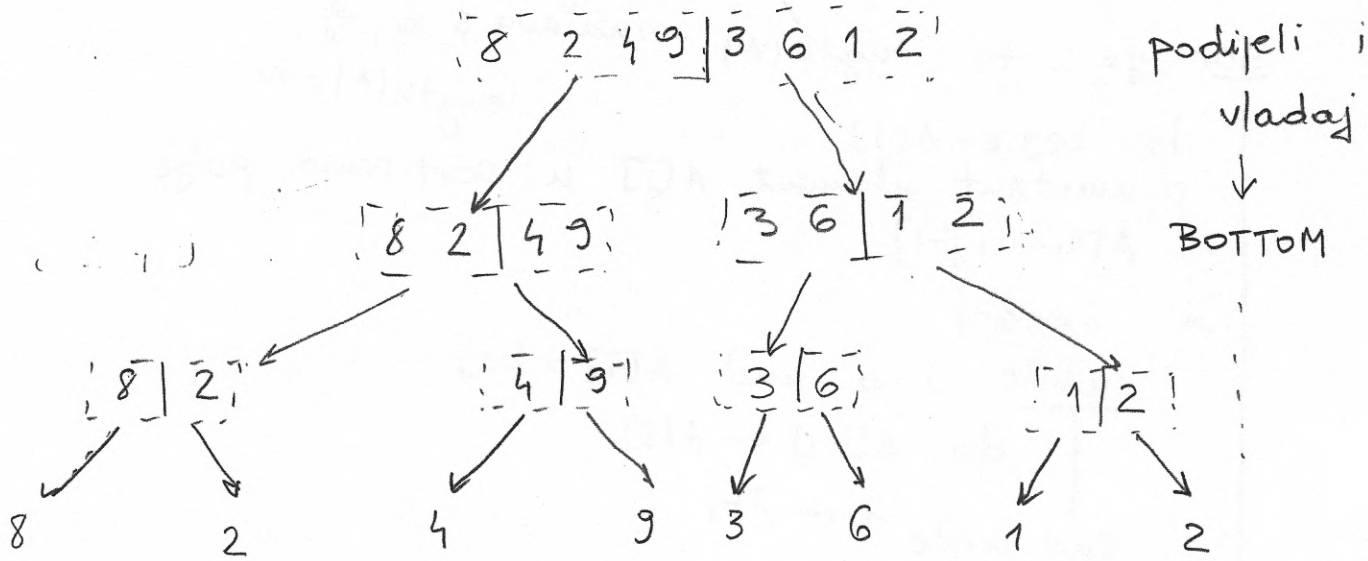
ključni korak MERGE

Primjer:

8 2 4 9 3 6 1 2

← proširili smo polje iz primjera s početka predavanja za 2 elementa

Što znači savladati problemom?

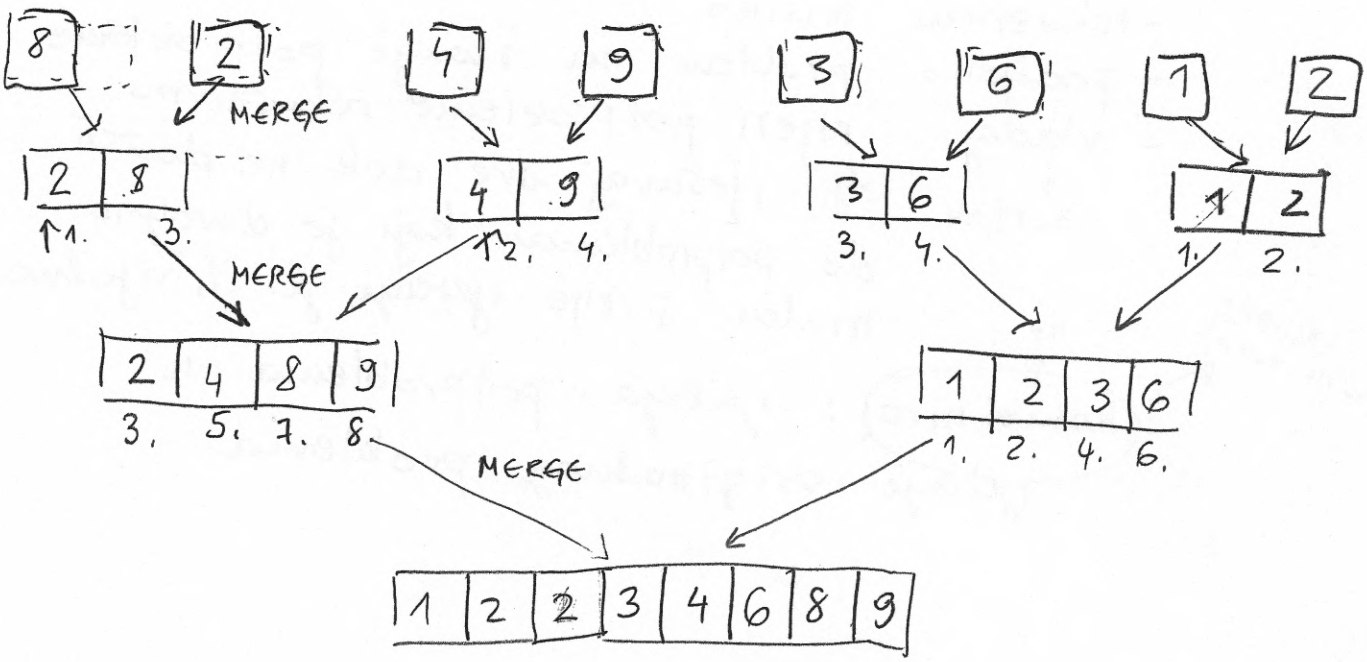


podijeli i vladaj ↓ BOTTOM

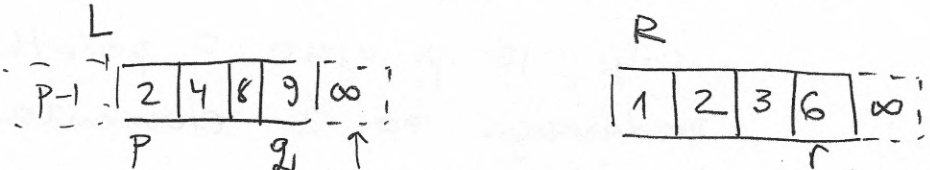
polja duljine 1 su već sortirana?

Kako spojiti rješenja potproblema u rješenje originalnog problema?

ključna operacija MERGE



Zadatak: Napisati pseudokod ~~of~~ metode MERGE



* niti jedan element ne može biti veći od ∞
* označava kraj podniza

MERGE (A, p, q, r) $p \le q < r$

$n_1 \leftarrow q - p + 1$ ◁ duljina niza L

$n_2 \leftarrow r - q$

▷ n_1+1 je duljina niza L

▷ n_2+1 je duljina niza R

▷ L i R su pomoćna polja

I for $i \leftarrow 1$ to n_1
    do $L[i] \leftarrow A[p + i - 1]$

II for $j \leftarrow 1$ to n_2
    do $R[j] \leftarrow A[q + j]$

$L[n_1+1] \leftarrow \infty$ ◁ dodatni element ← kad dođemo do ∞ znamo da je L prazan

$R[n_2+1] \leftarrow \infty$

$i \leftarrow 1$

$j \leftarrow 1$

III for $k \leftarrow p$ to r
    do if $L[i] \le R[j]$
        then $A[k] \leftarrow L[i]$
            $i \leftarrow i+1$
        else $A[k] \leftarrow R[j]$
            $j \leftarrow j+1$
    end if

end for

Zadatak: Napisati pseudokod metode MERGE-SORT

- MERGE-SORT (p, r)

if $p < r$ ← zašto ovaj uvjet? Kad je $p \geq r$ onda smo na „dnu rekurzije"

then $q \leftarrow \lfloor \frac{p+r}{2} \rfloor$

MERGE-SORT (A, p, q)

MERGE-SORT (A, q+1, r)

MERGE (A, p, q, r)

end if

ANALIZA ALGORITAMA

* korektnost
* vremenska složenost algoritma
* (prostorna složenost algoritma)

Korektnost INSERTION-SORT algoritma ↓ LOOP INVARIANT

Tema (Invarijanta INSERTION-SORT algoritma)

Na početku svake iteracije for petlje potpolje $A[1, \ldots, j-1]$ sastoji se od prvih $j-1$ elemenata ulaznog polja A ali u sortiranom poretku.

Dokaz: matematičkom indukcijom

$j \leftarrow 2$ potpolje ulaznog polja A $A[1]$ je sortirano i to je prvi element

$j \leftarrow k$ pretpostavka: vrijedi tvrdnja

$j \leftarrow k+1$ koraci indukcije: nakon umetanja $A[k+1]$ prvog elementa u sortirano polje $A[1, \ldots, k]$ polje $A[1 \ldots k+1]$ (p)ostaje sortirano

za $j = n$ polje $A[1, \ldots, n]$ postaje sortirano !

Algoritam je korektan □

Vremenska složenost INSERTION-SORT ALGORITMA

* direktna metoda:
  * vrijeme izvršavanja izraziti kao funkciju veličine ulaza
  * u našem slučaju veličinu ulaza „mjerimo" brojem elemenata u polju A

Teorem:

Vrijeme izvršavanja INSERTION-SORT algoritma na polju A duljine n može se opisati funkcijom

$$T(n) = \begin{cases} c_1 n + c_2, & \text{u najboljem slučaju} \\ c_3 n^2 + c_4 n + c_5, & \text{u najgorem slučaju} \\ c_6 n^2 + c_7 n + c_8, & \text{„u prosjeku"} \end{cases}$$

za neke konstante $c_i \in \mathbb{R}$, $i = 1, \ldots, 8$

dokaz:

```
INSERTION - SORT                                   cost      times
1: for j ← 2 to length(A)                          c1        n
2:     do key ← A[j]                               c2        n-1
3:        i ← j-1                                  c3        n-1
4:        while i>0 and A[i] > key                 c4        Σ_{j=2}^{n} t_j
5:           do A[i+1] ← A[i]                      c5        Σ_{j=2}^{n} (t_j - 1)
6:              i ← i-1                            c6        Σ_{j=2}^{n} (t_j - 1)
7:        end while
8:        A[i+1] ← key                             c7        n-1
```

- t_j ← broj provjera uvjeta u **while** petlji u j-toj iteraciji **for** petlje, $j = 2, \ldots, n$

Ukupno vrijeme:

$$T(n) = c_1 n + c_2(n-1) + c_3(n-1) + c_4 \sum_{j=2}^{n} t_j + c_5 \sum_{j=2}^{n} (t_j - 1) + c_6 \sum_{j=2}^{n} (t_j - 1) + c_7(n-1)$$

Vrijeme ovisi o t_j, $j = 2, \ldots, n$

* najbolji slučaj $t_j = 1$, $j = 2, \ldots, n$ ⇝ polje sortirano
* najgori slučaj $t_j = j$, $j = 2, \ldots, n$ ⇝ polje sortirano u obrnutom redoslijedu

* „u prosjeku" (average case) $t_j = \frac{j}{2}$ $j = 2, \dots, n$

Kad? ako su brojevi slučajno generirani

pola manje od A[j] } $t_j = \frac{j}{2}$

-||- veće -||-

Sad znamo izračunati $T(n)$:

najbolji slučaj:

$$T(n) = f_1 n + (f_2 + f_3)(n-1) + f_4(n-1) + f_7(n-1)$$

$$= \underbrace{(f_1 + f_2 + f_3 + f_4 + f_7)}_{c_1} n + \underbrace{(-c_2 - c_3 - c_4 - f_7)}_{c_2}$$

$$= c_1 n + c_2$$

tali slučajevi analogno ▽ □

---

simptotsko ponašanje:

$$T(n) = \begin{cases} \Theta(n) & , \text{ najbolji} & \text{worst-case} \\ \Theta(n^2) & , \text{ prosječan} & \text{average-case} \\ \Theta(n^2) & , \text{ najgori} & \text{best-case} \end{cases}$$

Korektnost MERGE-SORT ALGORITMA

ma (Invarijanta MERGE algoritma)

Na početku svake iteracije III for petlje algoritma MERGE otpolje A[p.... k-1] sadrži k-p najmanjih elemenata iz polja [1... n_1+1] i polja R[1.... n_2+1] u sortiranom poretku.

zdalje, L[i] i R[j] su najmanji elementi u poljima L i koji još nisu kopirani u polje A.

kaz: Cormen, Leiserson, Rivest, Stein, 2nd ed., 30. str.

ema: (Korektnost MERGE-SORT algoritma)

MERGE-SORT algoritam vraća sortirano polje A.

z: Iz leme slijedi da MERGE(A, 1, n, $\lfloor \frac{n+1}{2} \rfloor$) vraća sortirano polje ▽ □

Vremenska složenost MERGE-SORT algoritma

MERGE-SORT (A, p, r) $\quad T(n)$

if $p < r$

then $q \leftarrow \lfloor \frac{p+r}{2} \rfloor \rightarrow \Theta(1)$

MERGE-SORT (A, p, q) } $\rightarrow T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil)$

MERGE-SORT $(A, q+1, r)$ }

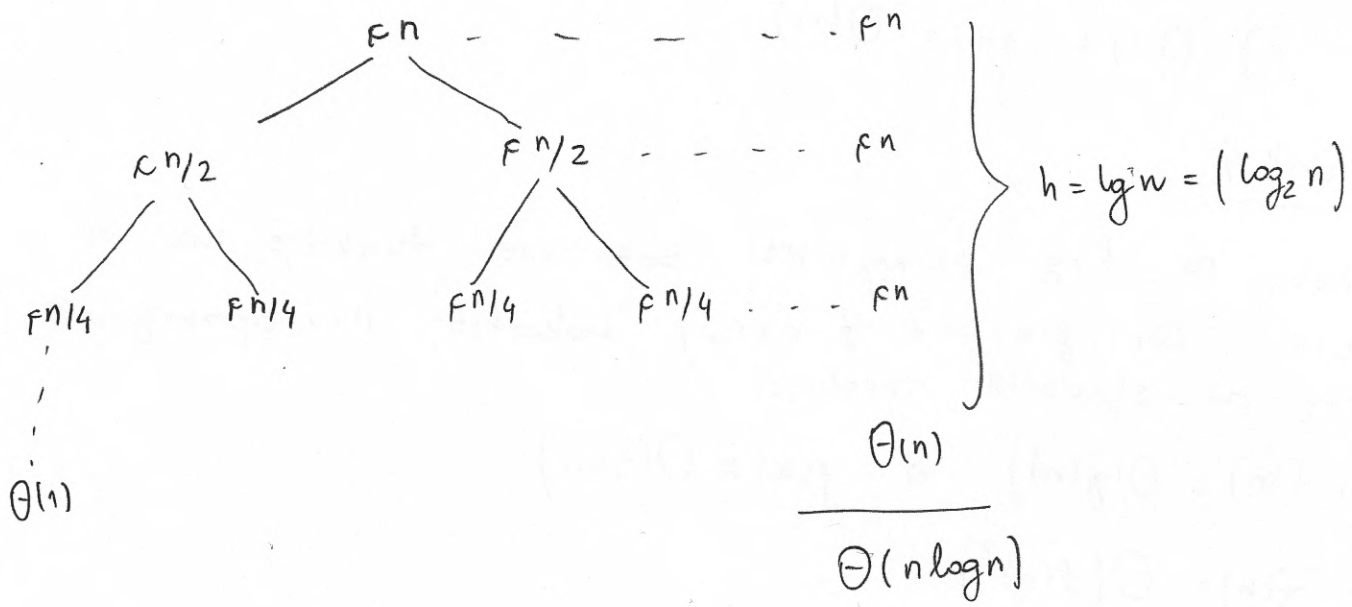
MERGE $(A, p, q, r) \rightarrow \Theta(n)$

* rekurzivna relacija za $T(n)$

$$T(n) = \begin{cases} \Theta(1), & n = 1 \\ 2T(n/2) + \Theta(n), & \text{inače} \end{cases}$$

* odrediti zatvorenu formulu za $T(n)$, tj: iskazati $T(n)$ kao funkciju od n

* metoda stabla rekurzije



$\Theta(n \log n)$

Teorem MERG-SORT algoritam vraća sortirano polje u $\Theta(n \lg n)$ vremenu

Skica: prethodno razmatranje

□