

## ANALIZA SLOŽENOSTI ALGORITAMA (15 bodova)

```

int k, s(0);
for(int i=1; i<=n; i++) {
    k=1;
    while(k<n) {
        for(j=1; j<=n-k; j++) s+=i-k+1;
        k*=3;
    }
}

```

$\log_3 n$

### RJEŠENJE:

Potpovativmo da je  $T(n)$  vrijeme izvršavanja navedenog algoritma.

Vnjska for petija će se izvršiti sigurno  $(n-1)$  puta, operacija dodjele imna konstantno vrijeme kao i inicijalizacija varijabli na početku.

BEST CASE: U slučaju kada je  $n=1$ , tada se neće izvršiti for petija ni while petija, vrijeme će biti konstantno  $T(n) = c$ .

### AVERAGE CASE:

$$\begin{aligned}
 T(n) &= c \cdot (n-1) + (n + (n-1) + (n-3) + (n-9) + \dots + ) \\
 &= cn - c + n \cdot \log_3 n \quad \text{dužina } \log_3 n \text{ jer se } k \text{ troduplicira} \\
 &= cn - c + n \cdot \log_3 n \approx \underline{\underline{O(n \log n)}} \quad O(n)
 \end{aligned}$$

## ANALIZA SLOŽENOSTI ALGORITAMA (15 bodova)

```

int k, s(0);
for(int i=1; i<=n; i++) {
    k=1;
    while(k<n) {
        for(j=1; j<=n-k; j++) s+=i-k+1;
        k*=3;
    }
}

```

$\log_3 n$

### RJEŠENJE:

Potpovativmo da je  $T(n)$  vrijeme izvršavanja navedenog algoritma.

Vnjska for petija će se izvršiti sigurno  $(n-1)$  puta, operacija dodjele imna konstantno vrijeme kao i inicijalizacija varijabli na početku.

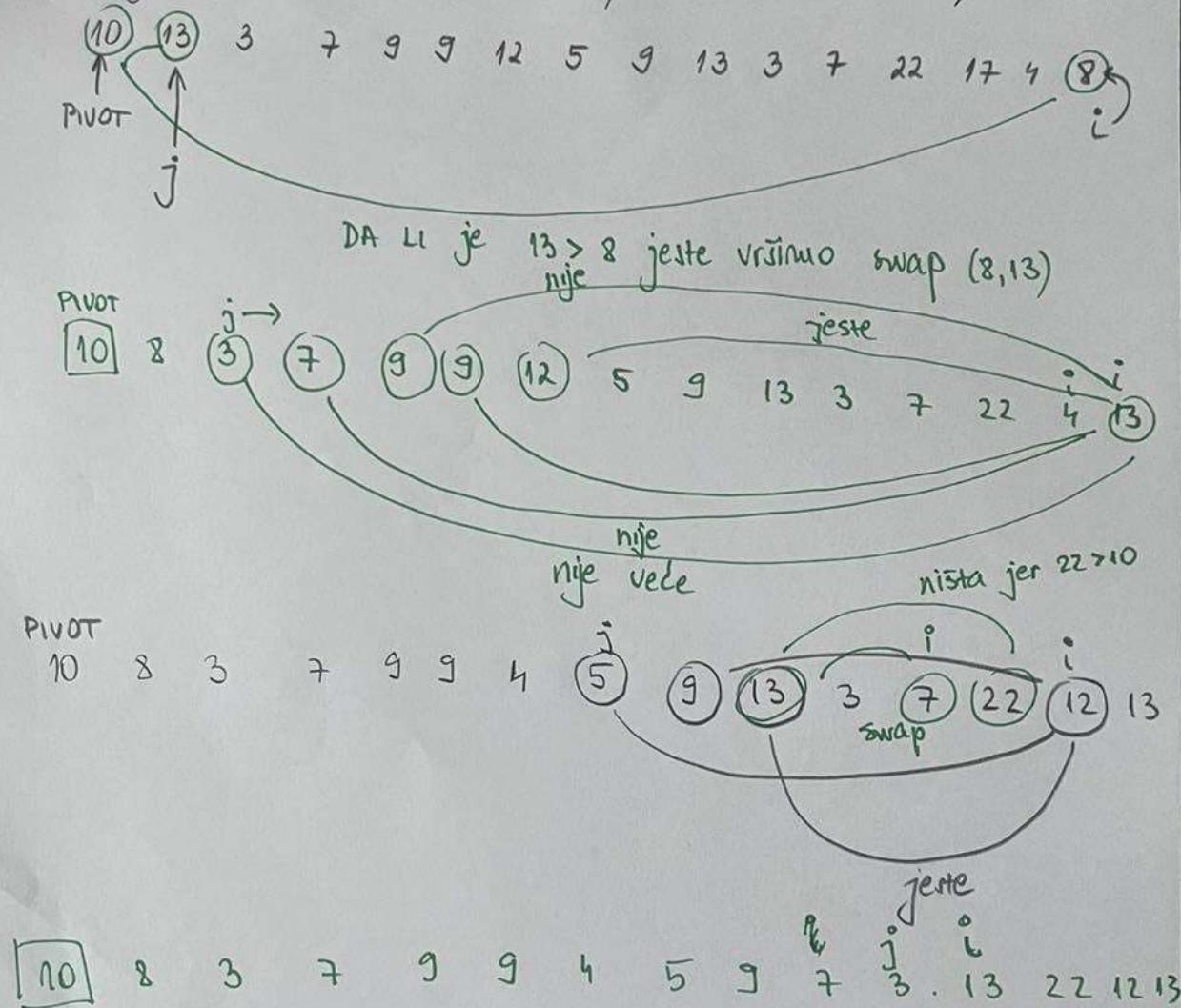
BEST CASE: U slučaju kada je  $n=1$ , tada se neće izvršiti for petija ni while petija, vrijeme će biti konstantno  $T(n) = c$ .

### AVERAGE CASE:

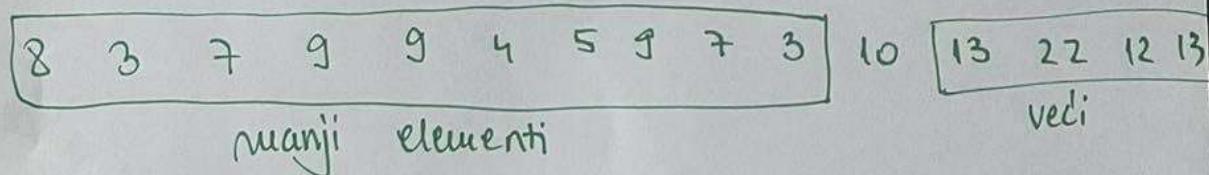
$$\begin{aligned}
 T(n) &= c \cdot (n-1) + (n + (n-1) + (n-3) + (n-9) + \dots + ) \\
 &= cn - c + n \cdot \log_3 n \quad \text{dužina } \log_3 n \text{ jer se } k \text{ troduplicira} \\
 &= cn - c + n \cdot \log_3 n \approx \underline{\underline{O(n \log n)}} \quad O(n)
 \end{aligned}$$

c) Dati postupak particije datog niza koristeći Quick sort partitioning algoritam uzimajući prvi element kao pivot.  
 $10, 13, 3, 7, 9, 9, 12, 5, 9, 13, 3, 7, 22, 17, 4, 8$ .

Ovaj algoritam treba odabrati prvi element kao pivot.



Li stajemo pivota prebacujemo na  $i-1$  poziciju



## HASH TABLE

### DJSTRUKO HASHIRANJE

Uti su ključevi: 3, 2, 9, 6, 11, 13, 7, 12. Veličina hash tabele je

$$N=10, h_1(k) = 2k+3$$

$$h_2(k) = 3k+1$$

Unutri k u prvo prazno mjesto od  $(u+v*i) \% N$ ,  
gde je  $i = 0, N-1$ .

Vrijednost izračunata  
u  $h_1(x)$  i  $h_2(k)$  u  $h_2(k)$

0	
1	9
2	
3	11
4	12
5	6
6	
7	2
8	
9	3

KLJUČ	LOKACIJA U	V	PROBA
3	9	—	1
2	7	—	
9	1	—	1
6	5	—	1
11	5	1	2, 3
13	9	0	NE MOŽENO INSEKTOMAT
7	7	2	NE MOŽENO
12	7	7	

predstavlja  
broj pokušaja  
da unetemo  
ovaj ključ  
u hash

tabelu  
(ako je oduzah  
slobodno  
upisujemo 1)

ako možemo ključ  
upisati u hash tabelu  
onda ne računamo v  
ako je zauzeto  
njegovo mjesto onda  
računamo v.

u ovom slučaju koristimo  
dvostruko hashiranje, jer je  
pozicija 5 zauzeta. Način na  
koji to radimo je:

$$(u+v*i) \bmod N$$

$$h_1(11) = (2 \cdot 11 + 3) \bmod N = 10$$

$$h_2(11) = (3 \cdot 11 + 1) \bmod N = 4$$

$$(u+v*i) \bmod 10 = (5 + 4 \cdot 0) \bmod 10 = 5 \rightarrow \text{zauzeto}$$

idemo da je i=1

$$(5 + 4 \cdot 1) \bmod 10 = 9 \rightarrow \text{zauzeto}, \text{ idemo da je i=2.}$$

ya.  
 kundarnom hash funkcijom  
 nju podataka iz hash table  
 vloboditi odgoj  
 joj nerto.

1	72	
2		
3		
4	26	
5	11	
6	44	
7		
8	22	
9	27	
10	92	
11	30	
12		
13	17	
14	89	
15	31	
K		

KLJUČ	$h_1(x)$	$h_2(x)$	VR/DEONOR	VLOTRUKO
8	8	—	22	1
73	5	—	11	1
10	10	—	92	1
56	5	4	27	2
44	10	4	89	2
32	15	—	31	1
26	9	4	17	2
57	6	—	44	1
66	15	6	26	2
39	5	3	30	2
15	15	3	72	3

↳ kod ključa 56 moramo izvrjiti dvostruko  
 hashiranje kojiteći sekundarnu funkciju  
 $h_2(x) = 6 - (x \bmod 6)$ .

$$h_2(56) = 6 - (56 \bmod 6) = 4$$

$$\hookrightarrow j=0 \Rightarrow (5+4*0) \bmod 17 = 5 \rightarrow zauzeto$$

$$\hookrightarrow j=1 \Rightarrow (5+4*1) \bmod 17 = 9 \rightarrow slobodno$$

↳ Dvostruko hashiranje za ključ 44. Imamo:  
 $h_2(44) = 6 - (44 \bmod 6) = 4$

$$\hookrightarrow j=0 \Rightarrow (10+4*0) \bmod 17 = 10 \rightarrow zauzeto$$

$$\hookrightarrow j=1 \Rightarrow (10+4*1) \bmod 17 = 14 \rightarrow slobodno$$

↳ Dvostruko hashiranje za ključ 26.  
 $h_2(26) = 6 - (26 \bmod 6) = 4$

$$\hookrightarrow j=0 \Rightarrow (9+4*0) \bmod 17 = 9 \rightarrow zauzeto$$

$$\hookrightarrow j=1 \Rightarrow (9+4*1) \bmod 17 = 13 \rightarrow slobodno$$

$$\begin{array}{c} \rightarrow [14] \rightarrow 1 \\ \rightarrow [22] \rightarrow [32] \end{array}$$

$14 \bmod 10 = 4$   
 $32 \bmod 10 = 2$   
 $71 \bmod 10 = 1$   
 $46 \bmod 10 = 6$

$\rightarrow [46]$

SAVANJE  
DANOST POK

→ u slučaju da nam je zauzet indeks na koji trebamo izvršiti umetanje hasha dopunjavamo tonu

$$j=1 \Rightarrow [1+1*1] \bmod 13 = 2 \rightarrow \text{slободно}$$

Kod kyuća 31 takođe imamo problem, jer je indeks 5 popunjeno u hash tabeli. Koristimo vada dvostruko hashiranje.

$$h_2(31) = 1+3 = 4$$

$$[h_1(x) + h_2(x) \cdot j] \bmod N$$

$$j=0 \Rightarrow [5+4*0] \bmod 13 = 5 \rightarrow \text{zauzeto}$$

$$j=1 \Rightarrow [5+4*1] \bmod 13 = 9 \rightarrow \text{slободно}$$

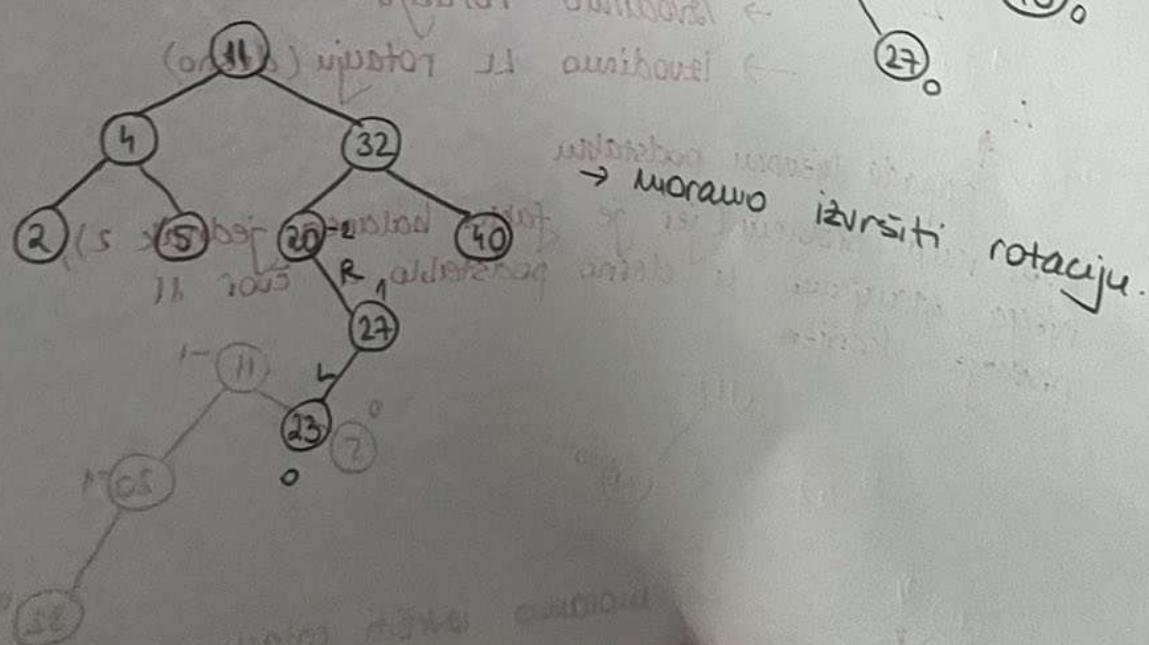
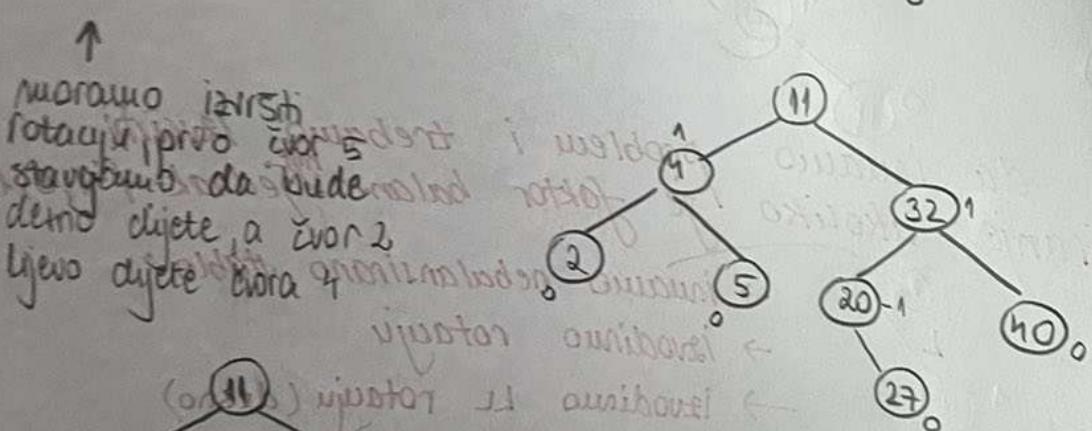
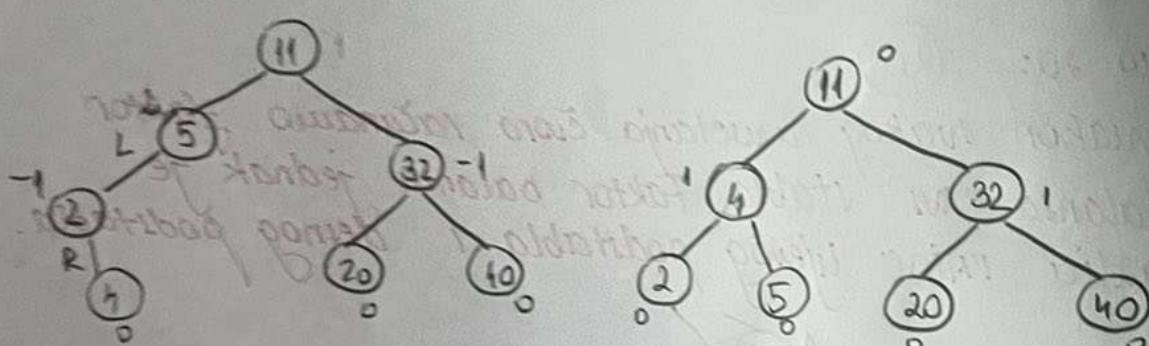
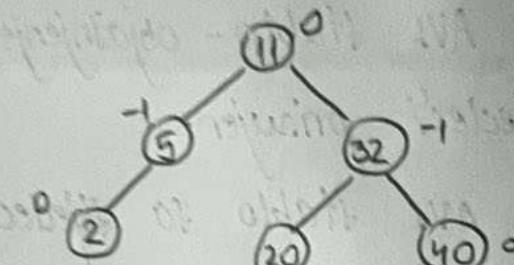
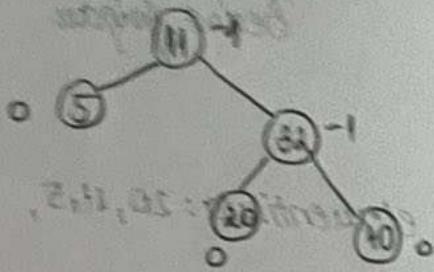
↪ Kod kyuća 100 imamo problem jer je pozicija 9 zauzeta, koristimo dvostruko hashiranje.

$$h_2(100) = 1+0 = 1$$

$$\hookrightarrow j=0 \Rightarrow (9+1*0) \bmod 13 = 9 \rightarrow \text{zauzeto}$$

$$\hookrightarrow j=1 \Rightarrow (9+1*1) \bmod 13 = 10 \rightarrow \text{zauzeto}$$

$$\hookrightarrow j=2 \Rightarrow (9+1*2) \bmod 13 = 11 \rightarrow \text{slободно}$$



Moramo izvršiti rotaciju prvo čvor 5 stavljanjem da bude rođak desne dijete, a čvor 2 lijevo dijete čvora 5.

Unutar rotacije je moralo izvršiti rotaciju.

Veka je da ove brojeve da brojevi nisu u da ligleda stablo nako  
kako ligleda stablo nako  
→ visinska rotacija

(5+4\*2) mod 10 = 3 → slobodno tamo ubacimo

Ako klijuc  $K=13$ , i dalje imamo problem pa koristimo dvostruko hashiranje.

$$(2 \cdot 13 + 3) \text{ mod } 10 = 9$$

$$(3 \cdot 13 + 1) \text{ mod } 10 = 0$$

$$(u + v * i) \text{ mod } 10 = 9 \rightarrow uvijek je 9 za bilo koje } i = \overline{0, N-1}$$

Zaključujemo da klijuc 13 NE MOŽENO INSERTOVATI U HAJT TABELU.

Za klijuc  $K=7$  imamo zauzetot mogemo koristiti dvostruko hashiranje.

$$(7+2*0) \text{ mod } 10 = 7 \rightarrow zauzeto.$$

$$i=1 \quad (7+2*1) \text{ mod } 10 = 9 \rightarrow zauzeto$$

$$i=2 \quad (7+2*2) \text{ mod } 10 = 1 \rightarrow zauzeto$$

$$i=3 \quad (7+2*3) \text{ mod } 10 = 3 \rightarrow zauzeto$$

$$i=4 \quad (7+2*4) \text{ mod } 10 = 5 \rightarrow zauzeto$$

$$i=5 \quad (7+2*5) \text{ mod } 10 = 7 \rightarrow zauzeto$$

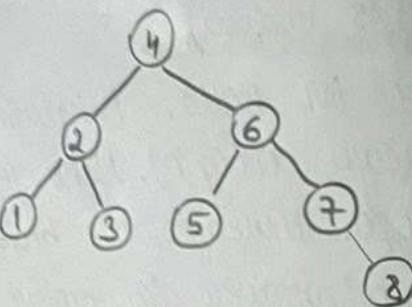
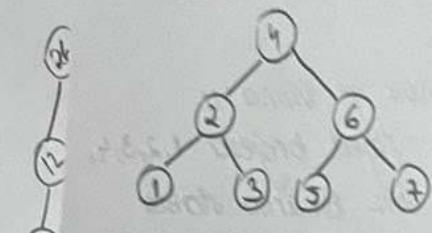
$$i=6 \quad (7+2*6) \text{ mod } 10 = 9 \rightarrow zauzeto$$

$$i=7 \quad (7+2*7) \text{ mod } 10 = 1 \rightarrow zauzeto$$

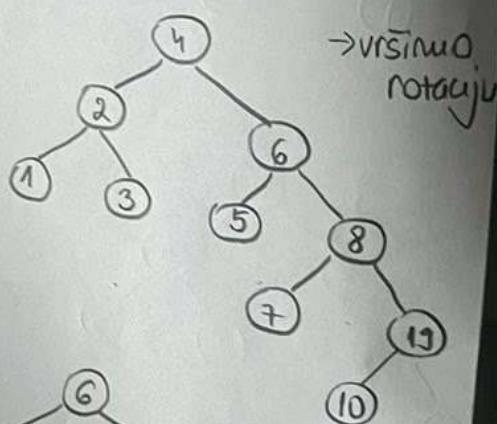
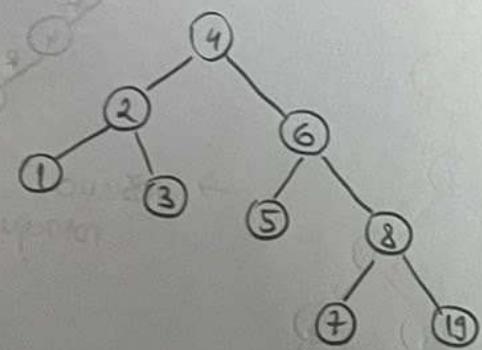
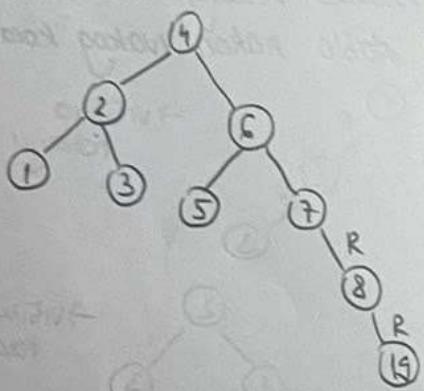
$$i=8 \quad (7+2*8) \text{ mod } 10 = 3 \rightarrow zauzeto$$

$$i=9 \quad (7+2*9) \text{ mod } 10 = 5 \rightarrow zauzeto$$

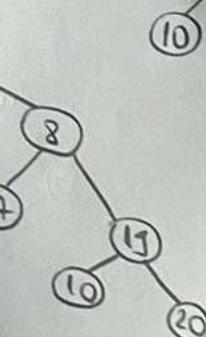
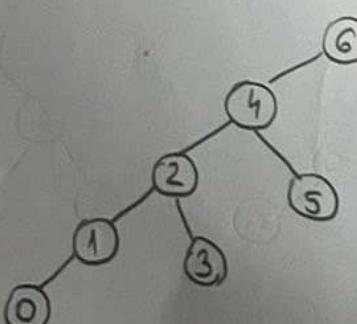
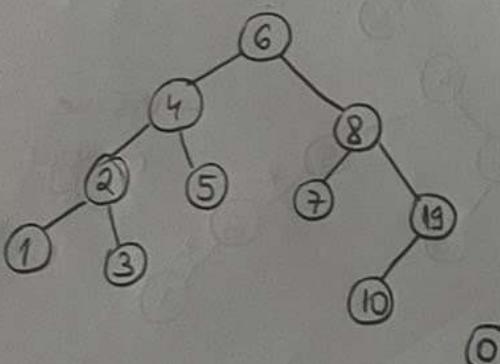
Ne možemo insertovati 7 jer nema odgovarajuću poziciju za njega



→ vršinu rotaciju



→ vršinu rotaciju



→ vršinu rotaciju

kako

načite

$$(x) + h_2(x) \cdot j] \bmod N$$

$$j=0 \Rightarrow (5+1*0) \bmod 17 = 5 \rightarrow \text{zauzeto}$$

$$j=1 \Rightarrow (5+4*1) \bmod 17 = 9 \rightarrow \text{slobodno}$$

↳ Imamo problem i kod ključa 4 jer je pozicija 10 zauzeta. Koristimo dvostruko hashiranje. Imamo:

$$h_2(44) = 4 - (44 \bmod 4) = 4$$

$$\hookrightarrow j=0 \Rightarrow (10+4*0) \bmod 17 = 10 \rightarrow \text{zauzeto}$$

$$\hookrightarrow j=1 \Rightarrow (10+4*1) \bmod 17 = 14 \rightarrow \text{slobodno}$$

↳ Kod ključa 26 koristimo dvostruko hashiranje.

$$h_2(26) = 4 - (26 \bmod 4) = 2$$

$$\hookrightarrow j=0 \Rightarrow (9+2*0) \bmod 17 = 9 \rightarrow \text{zauzeto}$$

$$\hookrightarrow j=1 \Rightarrow (9+2*1) \bmod 17 = 11 \rightarrow \text{slobodno}$$

↳ Pozicija 6 je niskorijena, pa za ključ 57 koristimo dvostruko hashiranje.

$$h_2(57) = 4 - (57 \bmod 4) = 3$$

$$\hookrightarrow j=0 \Rightarrow (6+3*0) \bmod 17 = 6 \rightarrow \text{zauzeto}$$

$$\hookrightarrow j=1 \Rightarrow (6+3*1) \bmod 17 = 9 \rightarrow \text{zauzeto}$$

$$\hookrightarrow j=2 \Rightarrow (6+3*2) \bmod 17 = 12 \rightarrow \text{slobodno}$$

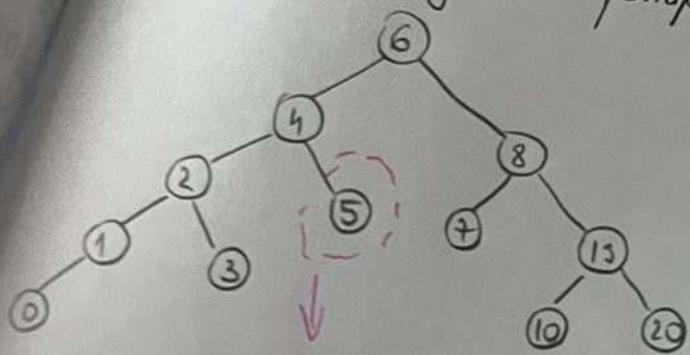
↳ Za ključ 66 koristimo dvostruko hashiranje.

$$h_2(66) = 4 - (66 \bmod 4) = 2$$

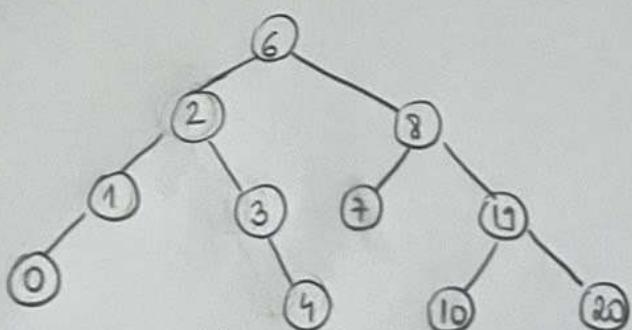
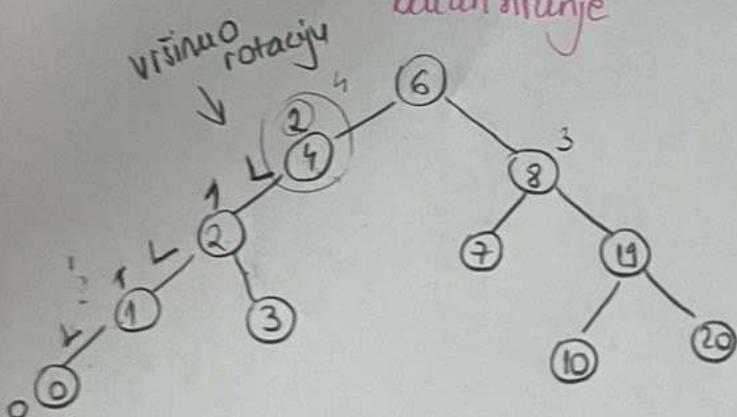
$$\hookrightarrow j=0 \Rightarrow (15+2*0) \bmod 17 = 15 \rightarrow \text{zauzeto}$$

$$\hookrightarrow j=1 \Rightarrow (15+2*1) \bmod 17 = 0 \rightarrow \text{slobodno}$$

Pitate kako bi izgledao postupak brisanja elementa 5.



ukinjamo element 5,  
nakon toga računamo  
faktor balansiranosti  
ako je različit od {1,0,-1} vršinu  
balansiranje



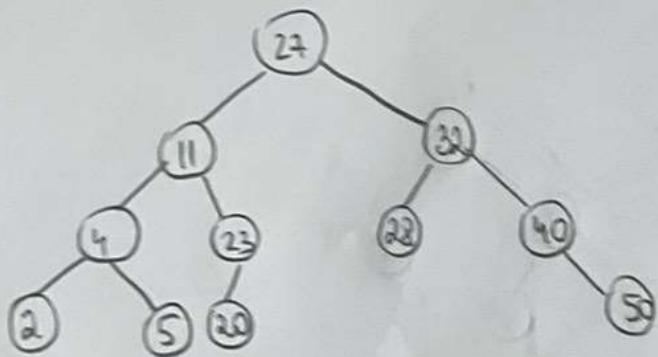
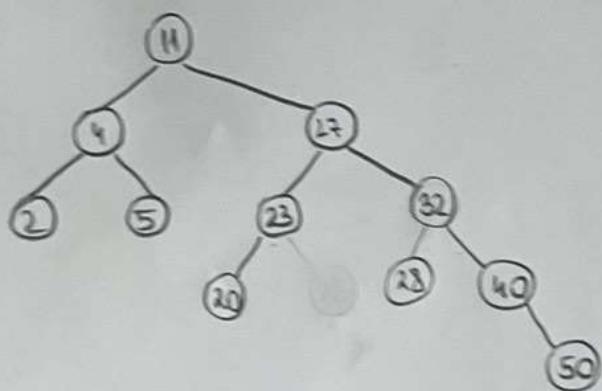
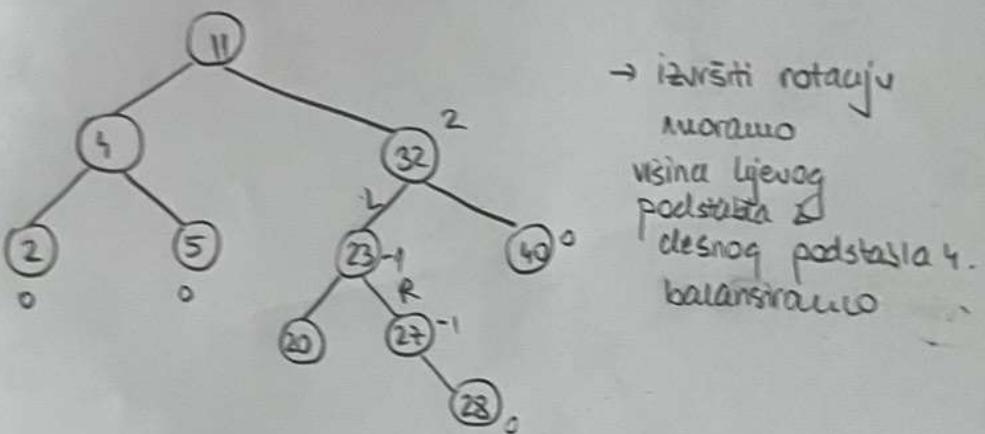
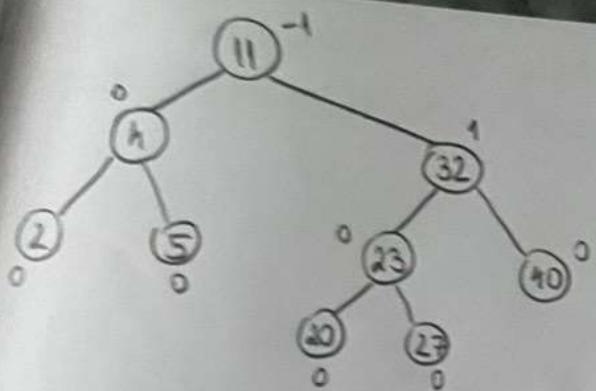
→ nisam 100%  
sigurna !!

N u  
) resultat  
) kao  
) žanret.  
table

mo primijeniti dvostruko hajiranje, jer  $k=12$  ne  
možemo interzovati.

$$i=0 \quad (7+7*0) \bmod 10 = 7 \rightarrow \text{gauzeto}$$

$$i=1 \quad (7+7*1) \bmod 10 = 4 \rightarrow \text{flobodno}$$

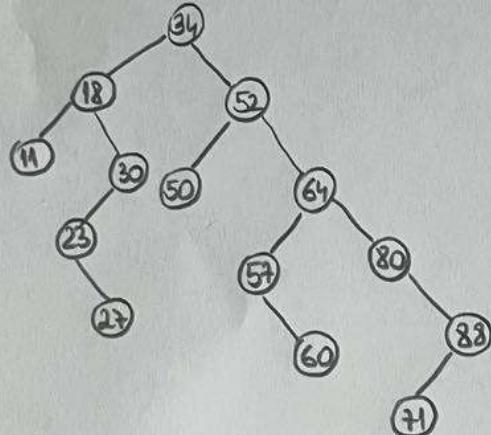


## BINARNO STABLO PRETRAGE (15 bodova)

Neka je dat sljed brojeva  $34, 52, 50, 18, 64, 57, 30, 60, 11, 23, 80, 27, 88, 71$ .

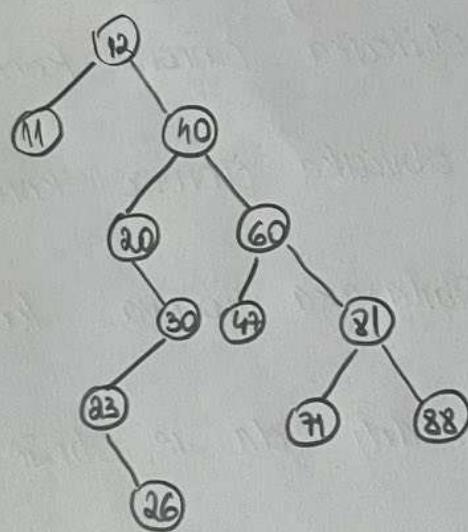
- Umetnite ove brojeve u binarno stablo pretrage uz prepostavku da brojevi mailaze redom kako su navedeni (potrebno je nacrtati kako tako dobijeno binarno stablo izgleda).
- Prikažite redoslijed obilaska čvorova koritenjem preorder obilaska.
- Prikažite redoslijed obilaska čvorova koritenjem inorder obilazak.
- Prikažite redoslijed obilaska čvorova koritenjem postorder obilazka.
- Opnite što treba uraditi da ve obrće čvor koji sadrži vrijednost 34.

a)

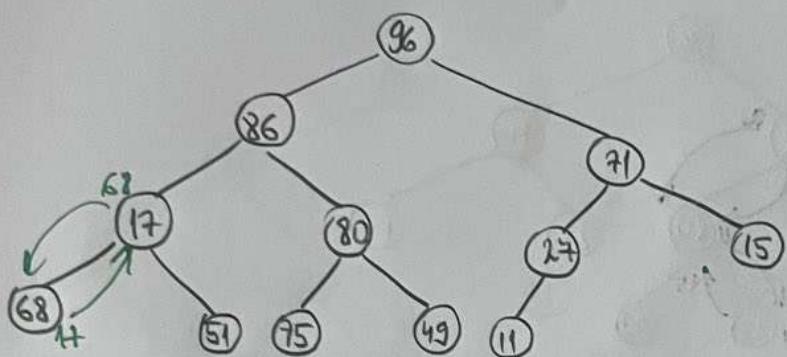
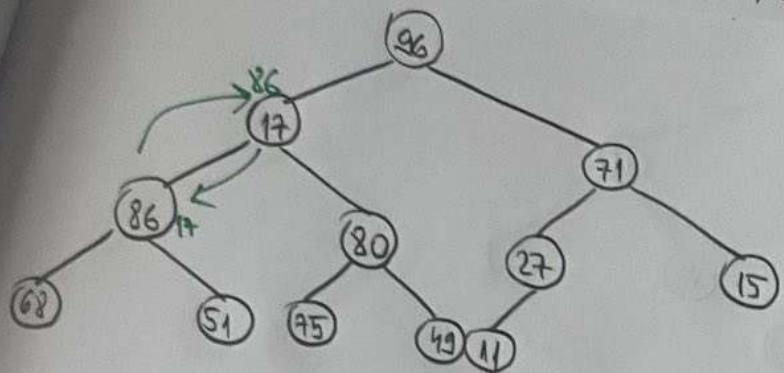


- PREORDER:  $34, 18, 11, 30, 23, 27, 52, 50, 64, 57, 60, 80, 88, 71$
- INORDER:  $11, 18, 27, 23, 30, 34, 50, 52, 60, 57, 64, 80, 88, 71$
- POSTORDER:  $11, 27, 23, 30, 18, 50, 60, 57, 71, 88, 80, 64, 52, 34$
- Potrebno je da u lijevom podstablu pronademo najveći element koji je list ili u desnom podstablu najmanji element koji je list. Dakle, pokazivač preuzijera je alocirana za element 34. Stablo će izgledati:

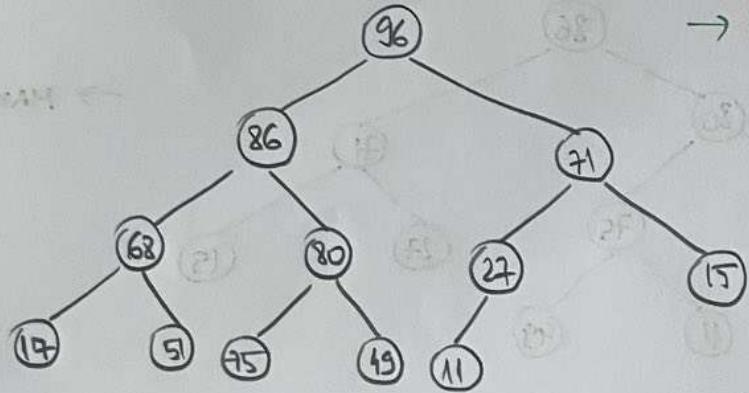
e) Budući da čvor 4 ima samo desno dijete, a ne ma lijevog djeteta bit će dovoljno da pokazivač preusmjerimo da pokazuje na njegovo desno dijete, tj. čvor sa vrijednošću 12 i da oslobođeno memorije koja je alocirana za čvor 4. Stablo nakon brisanja čvora 4 će izgledati:



redite skupinu brojeva 44, 91, 94, 20, 27, 42, 74, 58, 41, 53, 17,



→ MAX-HEAP.



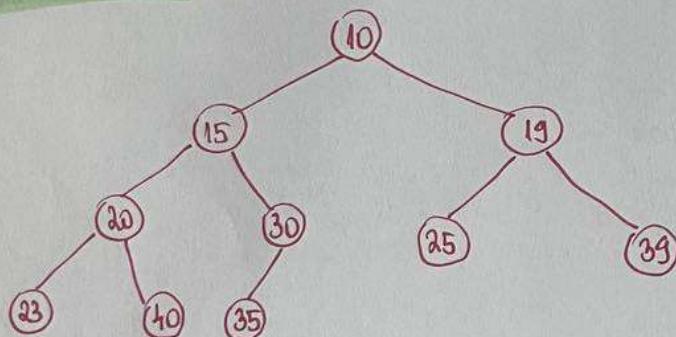
Uklanjanje elementa sa najvećeg prioriteta, tj. konjena - elementa 96, a na njegovu poziciju stavljanje elementa sa najvećim indeksom u heapu, tj. element 11. Nakon što izvršimo pozicioniranje elementa 11 bit će potrebno da uradimo određene zamjene kako bismo ponovo dobili max-heap.

Sprijau Berna

### HEAP

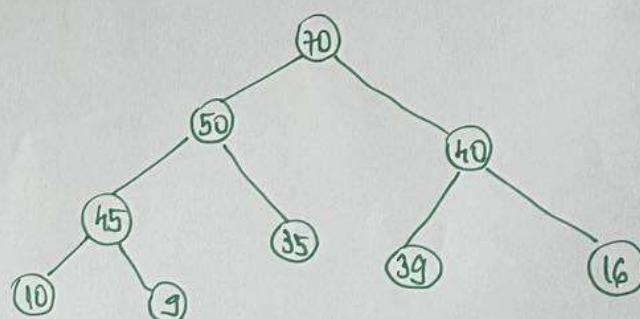
- Imamo dve vrste heapa: min-heap i max-heap.
- Kod min-heapa za svaki čvor i vrijednost čvora i mora biti veća ili jednaka od vrijednosti njegovog roditelja.
- Kod max-heapa za svaki čvor i vrijednost čvora i mora biti manja ili jednaka od vrijednosti njegovog roditelja.

### MIN-HEAP



10, 15, 19, 20, 30, 25, 39, 23, 40, 35 - nizovna reprezentacija

### MAX-HEAP



70, 50, 40, 45, 35, 39, 16, 10, 9 - nizovna reprezentacija

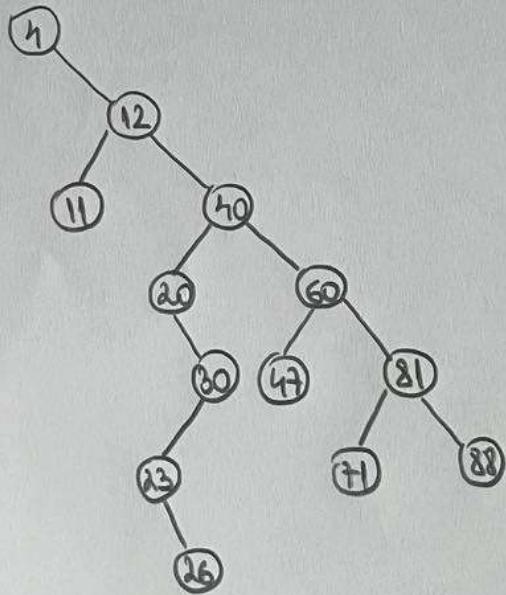
reprezentacija

### BINARNO JSTABLO PRETRAGE (15 bodova)

Neka je dat slijed brojeva 4, 12, 40, 20, 60, 47, 30, 60, 11, 23, 81, 26, 88 i 71.

- Umetnite ove brojeve u binarno stablo pretrage uz pretpostavku da brojevi nailaze redom kako su navedeni (potrebno je nacrtati kako tako dobijeno binarno stablo izgleda).
- Prikažite redoslijed obilaska čvorova koristenjem preorder obilazaka.
- Prikažite redoslijed obilaska čvorova koristenjem inorder obilazaka.
- Prikažite redoslijed obilaska čvorova koristenjem postorder obilazaka.
- Opisite što treba uraditi da se obrće čvor koji sadrži vrijednost 4.

a)



b) Preorder: 4, 12, 11, 40, 20, 30, 23, 26, 60, 47, 81, 71, 88

c) Inorder: 4, 11, 12, 23, 26, 30, 20, 40, 47, 60, 71, 81, 88

d) Postorder: 11, 26, 23, 30, 20, 47, 71, 88, 81, 60, 40, 12, 4

#### 4. DIZAJN ALGORITAMA (20 bodova)

Dat je niz od  $n$  prirodnih brojeva od 1 do  $n$  (koji ne moraju biti sortirani) pri čemu se jedan element ponavlja dva puta dok jedan element nedostaje. Opisati algoritam koji u vremenu  $O(n)$  i sa  $O(1)$  dodatnog prostora pronađe odgovarajući element. Dozvoljeno je koristiti samo varijable celobrojnog tipa (No overflow allowed).

ALGORITAM: Prolazimo kroz niz u linearnom vremenu. Plikom prolaska kroz niz koritit ćemo apolutnu vrijednost svakog elementa kao indeks i vrijednost koja se nalazi na tomu indeksu pretvorit ćemo u pozitivnu. Da bi pronašli element koji nedostaje, proći ćemo opet kroz niz i potražiti pozitivnu vrijednost.

### 3. DIZAJN ALGORITMA (20 bodova)

Neka su data dva niza  $x$  i  $y$  svaki veličine  $n$  i svaki od datch nizova je sortiran. Opisati algoritam koji u vremenu  $O(\lg n)$  pronađi medijanu unije elemenata skupova  $x$  i  $y$ .

Prvo ćemo dobiti medijane dva sortirana niza, a zatim ih usporediti.

ALGORITAM: Neka su niz1 i niz2 ulazni nizovi.

- 1) Izračunamo prvo medijanu  $m_1$  ulaznog niza „niz1”, a zatim  $m_2$  medijanu ulaznog niza „niz2”.
- 2) Ako su  $m_1$  i  $m_2$  jednaki onda je to to, naći smo medijanu.
- 3) Ako je  $m_1 > m_2$  medijana je prisutna u jednom od dva niza:
  - a)  $niz1[0 \dots \lfloor n/2 \rfloor]$  - od prvog elementa niza do  $m_1$ ,
  - b)  $niz2[\lfloor n/2 \rfloor \dots n-1]$  - od  $m_2$  do zadnjeg elementa niza  $niz2$
- 4) Ako je  $m_2 > m_1$  medijana je prisutna u jednom od dva niza:
  - a)  $niz1[\lfloor n/2 \rfloor \dots n-1]$  - od  $m_1$  do posljednjeg elementa niza1
  - b)  $niz2[0 \dots \lfloor n/2 \rfloor]$  - od prvog elementa niza niz2 do  $m_2$
- 5) Gornji postupak se ponavlja dok veličina oba podniza ne postane 2.
- 6) Ako je veličina oba niza 2 onda ćemo koristiti formulu:  
$$\text{medijana} = (\max(niz1[0], niz2[0]) + \min(niz1[1], niz2[1]))/2;$$

f) Objasnite što su stek i red, i kakve su im primjene.

STEK je struktura podataka kod koje se elementi mogu umjetati ili uklanjati samo na jednomu kraju strukture.

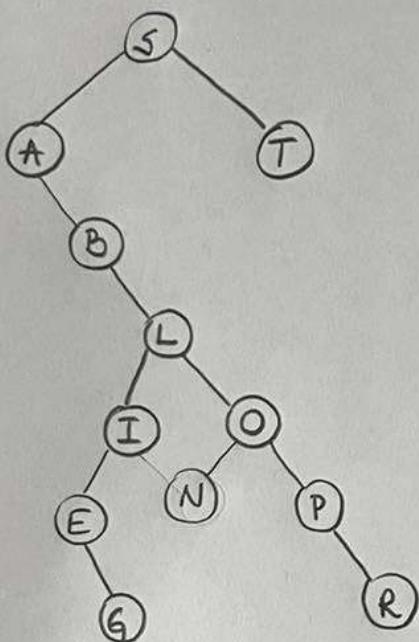
Taj kraj se naziva vrh steka (top). Drugi kraj strukture podataka se naziva dno (bottom). Za stek možemo reći da imaju LIFO organizaciju. To znači da se elementi va steka uklanjuju obrnutim redoslijedom od umetanja.

PRIMJENA: implementacija kalkulatora, Balansiranje simbola, evaluator, parser...

RED je struktura podataka koja je slična steku. Za razliku od steka, red imao dva pristupna kraja. Na jednomu kraju strukture koja se naziva zatelje (tail) vrti se umjetanje novih elemenata, dok se izbacivanje elemenata iz reda vrti na drugomu kraju strukture koja se naziva čelo (head). Pritom na dva kraja omogućava da se elementi iz reda uklanjuju istim redoslijedom u kojem su i umjetnuti. To znači da se iz reda uklanja onaj element koji je u red umjetnut najranije (FIFO). Red obavlja funkciju buffera.

## BINARNO STABLO

- a) Raasporedite slova unutar stringa "STABLO BINARNE PRETRAGE" u binarno stablo pretrage uzimajući slova iz stringa redom sjeva na desno, uzimajući abecedni poredak slova kao kriterij za razvrstavanje.



- b) Prikazite redoslijed prolaska kroz čvorove ovog stabla za svaku od tri karakteristične strategije obilaska stabla: PREORDER, INORDER, POSTORDER.

PREORDER: S, A, B, L, I, E, G, O, N, P, R, T

INORDER: A, B, L, I, E, G, N, O, P, R, S, T

POSTORDER: G, E, I, N, R, P, O, L, B, A, T, S.

b) Klasična definicija produkta dva kompleksna broja  
 $(a+bi)(c+di) = (ac-bd)+(ad+bc)i$  i zahvaljujući obično množenju  
Pokazite kako se isti produkt može izračunati uz  
pomoć svega 3 množenja (Uputa: koristite istu ideju pri  
realizaciji divide and conquer strategije za množenje dva  
velika broja ili dva polinoma).

$$\text{REALNI DIO: } ac - bd$$

$$\text{IMAGINARNI DIO: } ad + bc.$$

Ako stavimo da je  $s_1 = ac$ ,  $s_2 = bd$  dobijamo realni dio  
kao razliku ova dva proizvoda, tj.

$$\text{realni dio} = s_1 - s_2 = ac - bd.$$

Ako stavimo da je  $s_3 = (a+b)(c+d)$  dobijamo i imaginarni  
dio:

$$\text{Imaginarni dio} = s_3 - s_1 - s_2 = (a+b)(c+d) - ac - bd.$$

Ovaj proces se naziva Karatsuba algoritam za  
množenje.

b) Objavite što je osnova stabilnosti algoritama za sortiranje. Koji su od poznatih brojkih algoritama za sortiranje stabilni a koji nisu? Da li su algoritmi Selection sort i Insertion sort stabilni? Obrazložite odgovor.

Za algoritam za sortiranje se kaže da je stabilan ako se dva objekta sa jednakim ključevima pojavljuju u istom redoslijedu u sortiranom nizu kao što ve pojavljuju u ulaznom nizu koji ve sortira.

Stabilnost je uglavnom važna kada imamo parove vrijednosti ključeva sa mogućim duplicitiranim ključevima (kao što su imena ključeva i njegovi detalji kao vrijednosti). Želimo da sortiramo ove objekte po ključevima.

**STABILNI:** Bubble sort, Merge sort, Insertion sort, Count sort, ...

**NESTABILNI:** Quick sort, Heap sort, Selection sort, ...

d) Objasnite zašto pri brijanju podataka iz hash tabele nije dovoljno samo sloboditi odgovarajući slot nego je potrebno počuhati joj nešto, šta?

Da bi iz hash tabele obrnuli podatak nije dovoljno samo da slobodimo odgovarajući slot već je potrebno postaviti unique marker koji se zove ~~tonubitne~~ unijesto null, te će naručiti da je na tom mjestu podatak obrisan i da bi se slot trebao preskočiti tokom pretrage.

e) Da li kod algoritma Merge sort trajanje sortiranja zavisi od toga kako su elementi raspoređeni u nizu ili ne?

Kod merge sorta trajanje sortiranja ne zavisi od toga kako su elementi raspoređeni. Vrijeme će i u najboljem i u najgorem slučaju biti  $n \log n$ . Spajanje elemenata niza u Merge sortu će manu uvijek uzeti vrijeme  $n$ , a polovica je  $n \log n$ .

#### 4. REKURZIVNE RELACIJE (15 bodova)

a) Ako je  $g(n)$  arhimedotski pozitivna funkcija definisati skupove  $O(g(n))$ ,  $\Omega(g(n))$ ,  $\Theta(g(n))$ .

b) Nacrtati rekursivno stablo za rekurziju  $T(n) = T(n/3) + T(2n/3) + \gamma n$ , te naci arhimedotski najbolji gornju i donju granicu rješenje ove rekursivne relacije. To dokazati indukcijom.

Definicija 1: Neka je  $g(n)$  realna funkcija definisana na nekom podskupu skupa prirodnih brojeva. Sa  $O(g(n))$  se označava skup realnih funkcija  $f(n)$  definisanih na nekom podskupu skupa realnih brojeva takvi da postoji pozitivna konstanta  $c$  i prirodan broj  $n_0$  takav da je:

$$O \leq f(n) \leq c \cdot g(n) \text{ za svako } n \geq n_0.$$

DEF 2: Neka je  $g(n)$  realna funkcija definisana na nekom podskupu skupa prirodnih brojeva. Sa  $\Omega(g(n))$  se označava skup realnih funkcija  $w(g(n)) = f(n)$  takvih da postoji pozitivna konstanta  $c$  i prirodan broj  $n_0$  takav da je:

$$O \leq c \cdot g(n) < f(n), \quad \forall n \geq n_0$$

DEF 3: Neka je  $g(n)$  realna funkcija definisana na nekom podskupu skupa prirodnih brojeva. Sa  $\Theta(g(n))$  se označava skup realnih funkcija  $f(n)$  definisanih na nekom podskupu skupa realnih brojeva takvih da postoji konstanta  $c > 0$  i prirodan broj  $n_0$  takvi da vrijedi:

$$O \leq c \cdot g(n) \leq f(n), \quad \forall n \geq n_0$$

## 2. DIVIDE AND CONQUER STRATEGIJA (25 bodova)

- a) Objasnite ukratko na čemu se zasniva divide and conquer strategija za rješavanje problema i navedite primjere barem 3 algoritma zasnovana na ovoj strategiji.
- Divide and conquer je strategija rješavanja problema koja se temelji na podjeli problema na manje, lakše rješive potprobleme rješavanju svakog od njih zasebno, a zatim kombiniraju njihovih rješenja kako bi se dobio konacan rezultat. Ova strategija se cesto koristi za rješavanje problema s velikim skupinama podataka ili složenih problema.

Primjeri algoritama koji se oslanjaju na strategiju „divide and conquer“:

1. MERGE SORT: Niz se dijeli na pola, zatim se svaka polovina sortira rekursivno. Nakon toga, sortirane polovine se spajaju u konačno sortirani niz.

2. QUICK SORT: Niz se podjeli na dva podniza oko odabranog pivot elementa. Podnizovi se zatim sortiraju rekursivno koristeći isti algoritam. Nakon toga, sortirani podnizovi se spajaju na odgovarajući način.

3. BINARY SEARCH: Lista je podijeljena na pola, a zatim se pretraživanje nastavlja na polovinama koje mogu sadržati traženi element. Ovaj proces se rekursivno nastavlja sve dok traženi element ne nađete ili ve lista potpuno ne iscrpi.

c) Razmislite u kojem slučaju užeda od samo jednog množenja otvarenog pod  $\theta$ , može biti veoma značajna.

Primer jedne ovakve situacije je u upotrebi brze Fourierove transformacije ili diskretne Fourierove transformacije.

Ovi algoritmi se često koriste u obradi signala i obradi slike. Tokom izvođenja ovih transformacija složeni brojevi se često množe. Jedna tehnika koja se koristi razdvajaće konjunktnih brojeva u realne i imaginarnne dijelove. Ovaj postupak omogućava efikasniju primjenu algoritma jer se konjunktni brojevi množi zasebno, što može rezultirati uštedom u vremenu izvođenja.

## 6. ALGORITMI ZA SORTIRANJE (15 bodova)

a) Objavite osnovne prednosti i manje Algoritma Selection sort i Insertion sort.

### SELECTION SORT:

- prednost selection sort algoritma je da dobro radi na malim listama
- buduci da je in-place u najetu nije potrebna dodatna privremena memorija osim one sto je potrebna za cuvanje originalne liste
- nedostatak selection sort algoritma je slaba efikavnost kada radi sa velikim listama. Selection sort zahtjeva  $n^2$  koraka za sortiranje n elemenata. Osim toga, na njegovu izvedbu utice pocetni raspored elemenata u listi. zbog toga je selection sort pogodan za listu od nekoliko elemenata koji imaju naranicatu redoslijed.

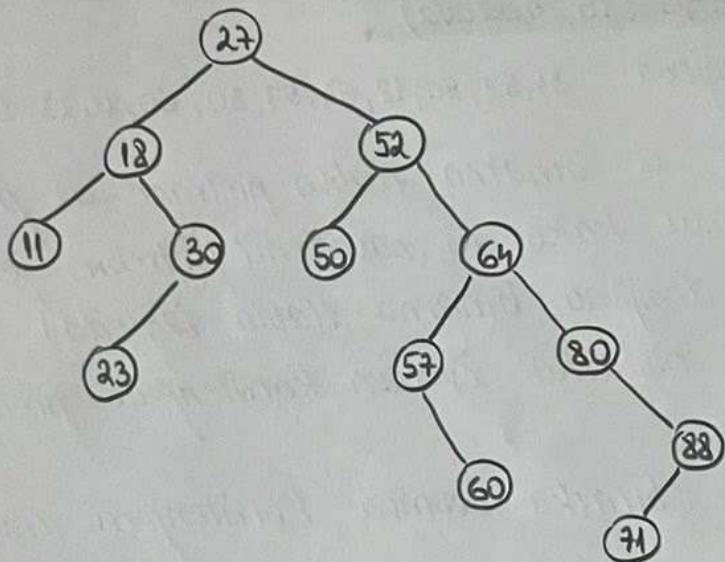
### INSERTION SORT:

- prednost je njegova jedinstvenost
- dobre performanse u radu sa malim listama
- insertion sort je in-place algoritam, tako da je zahtjev za prostorom minimalan
- manja je sto za n elemenata je vrijeme izvrsavanja  $n^2$ .

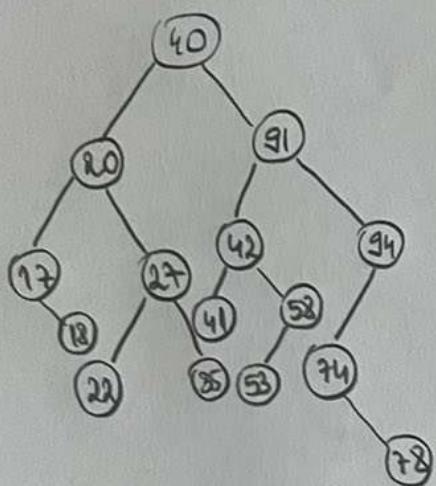
d) Definisati šta je heap i navesti moguće primjene ove strukture podataka (osim za realizaciju heap sort postupkom).

Heap je struktura podataka koja nam omogućava da brzo i lako dođemo do maksimalnog elementa/minimalnog elementa. Heap se predstavlja kao stablo na sljedeći način: za svaki čvor, njegova djeca su uvek manja od tog čvora ukoliko govorimo o max-heapu, odnosno veća ukoliko govorimo o min-heapu.

Heap nam omogućava dodavanje elemenata u  $O(n \log n)$  vremenu. Ako želimo obrisati najveći(najmanji) element, to možemo dobiti u konstantnom vremenu, tj.  $O(1)$ .



9) Rasporedite skupinu brojeva 40, 91, 94, 20, 27, 42, 74, 58, 41, 53, 17, 22, 78, 35 i 18 u binarno stablo pretrage. Nakon toga, prikažite redoslijed prolaska kroz čvorove ovog stabla za svaku od 3 karakteristične strategije obilaska stabla: preorder, inorder. i postorder.

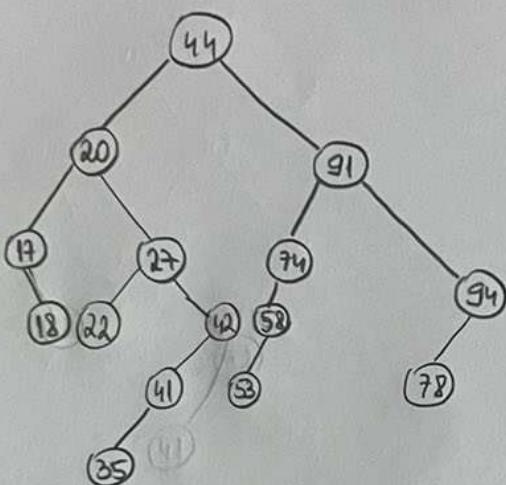


PREORDER: 40, 20, 17, 18, 22, 35, 91, 42, 41, 58, 53, 94, 74, 78

INORDER: 18, 17, 20, 22, 27, 35, 40, 41, 42, 53, 58, 91, 78, 74, 94

POSTORDER: 18, 17, 22, 35, 27, 20, 41, 53, 58, 42, 78, 74, 94, 91, 40

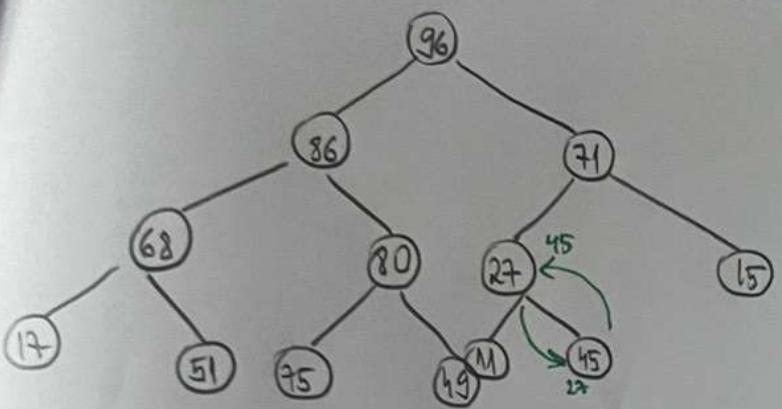
g) Rasporedite skupinu brojeva  $44, 91, 94, 20, 27, 42, 74, 58, 41, 53, 17, 22, 78, 35; 18$  u binarno stablo pretrage. Nakon toga, prikažite redoslijed prolaska kroz čvorove ovog stabla za svaku od tri karakteristične strategije obilaska stabla: preorder, inorder i postorder.



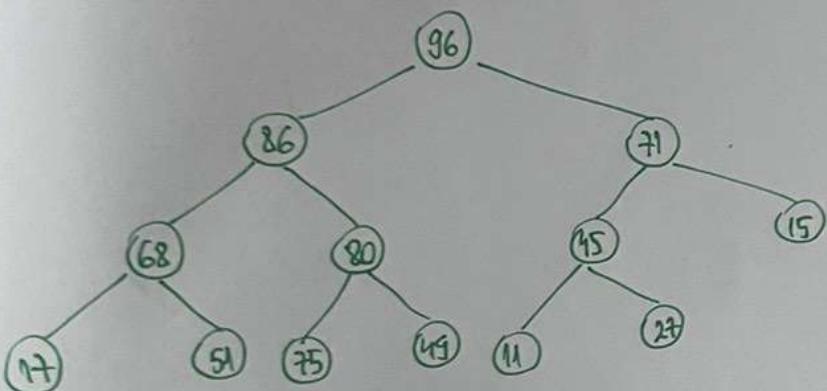
PREORDER:  $44, 20, 17, 18, 27, 22, 42, 41, 35, 91, 74, 58, 53, 94, 78$

INORDER:  $18, 17, 20, 22, 27, 35, 41, 42, 44, 53, 58, 74, 91, 78, 94$

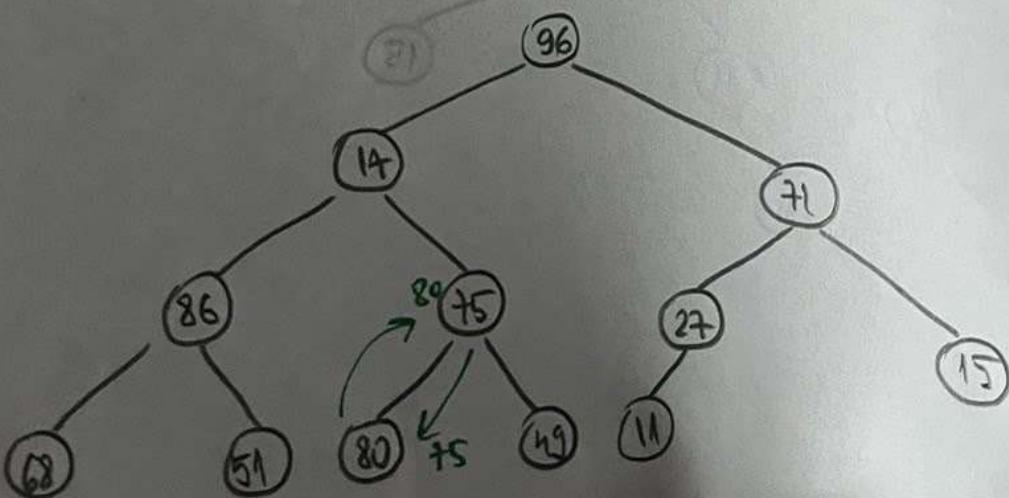
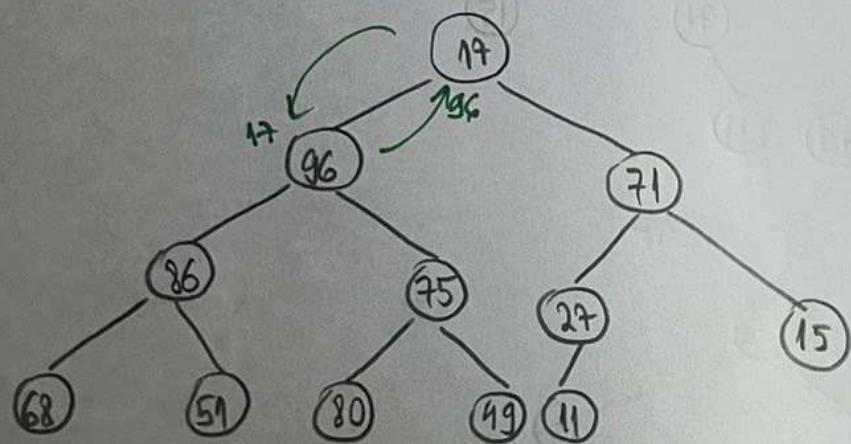
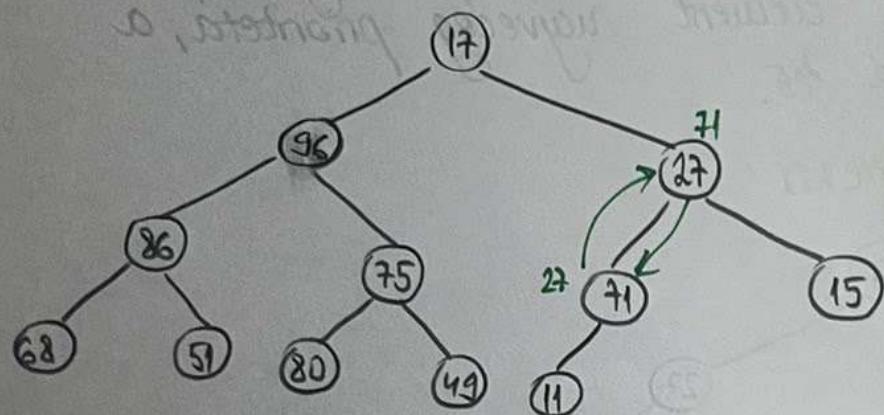
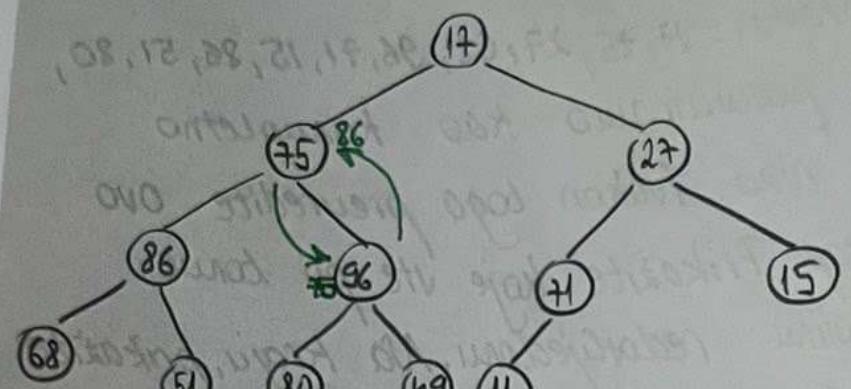
POSTORDER:  $18, 17, 22, 35, 41, 42, 27, 20, 53, 58, 74, 78, 94, 91, 44$

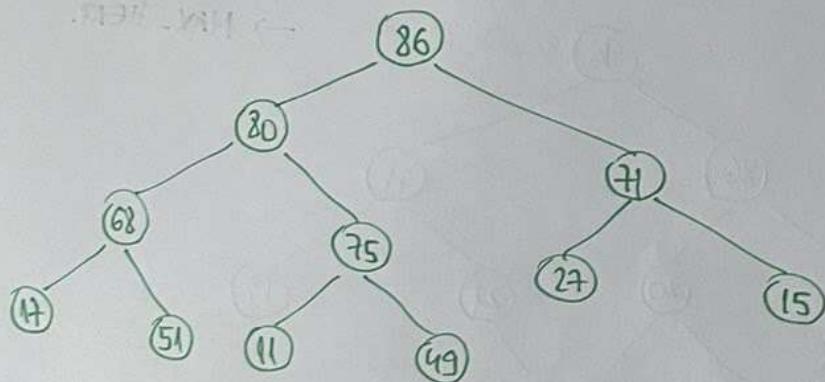
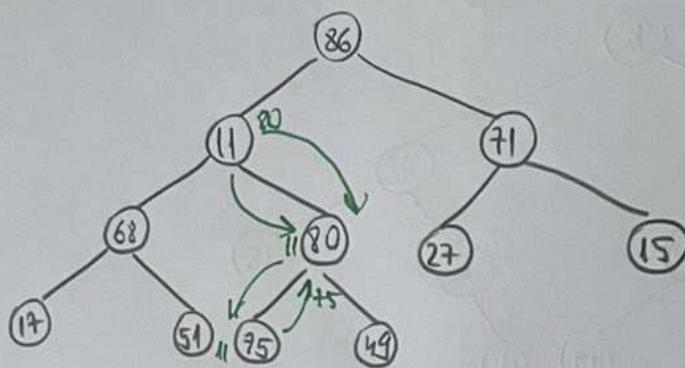
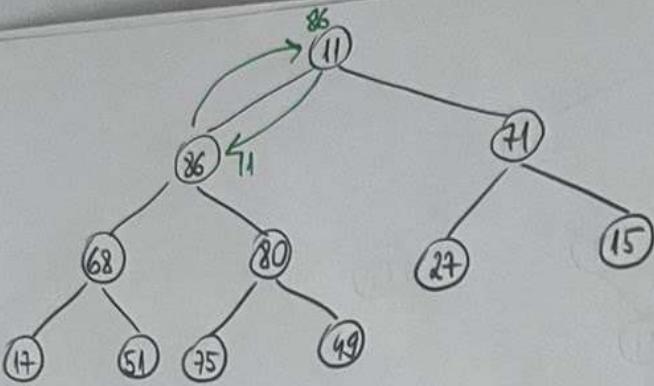


→ MAX- HEAP.



PREDRI  
INORDE  
POSTORS





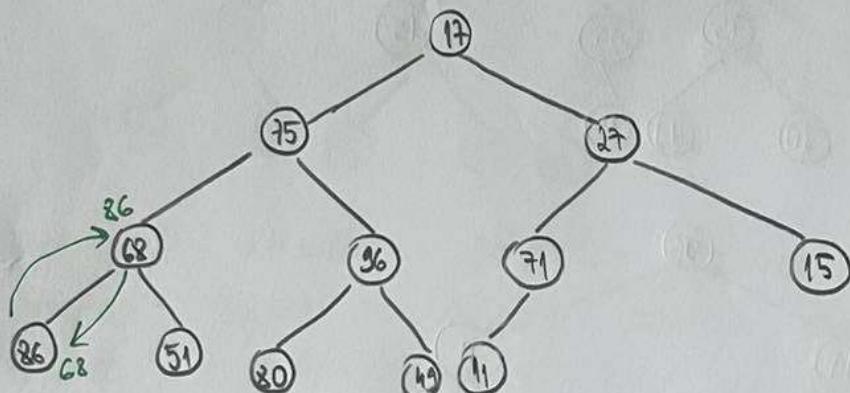
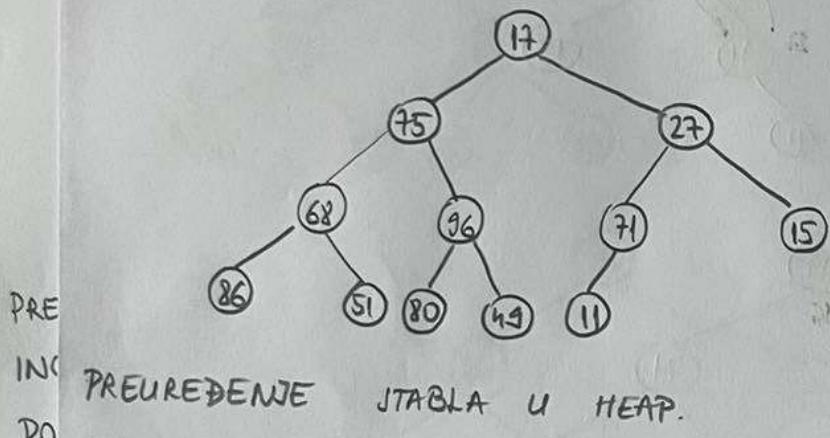
→ MAX-HEAP.

↳ Unutar uvećanja elementa 45

- dodajemo ga na kraj niza
- inicijalizirati majveci indeksi
- dodajemo ga kao deino dijete elementu 71.
- nakon toga vršimo zamjene da dobijemo max-heap jer će poređak biti narušen.

g) Ra  
22, 71 h) Prikazite kako bi niz brojeva: 17, 75, 27, 68, 96, 71, 15, 86, 51, 80, 49 i 11 izgledao ako ga posmatramo kao kompletno binarno stablino u formi niza. Nakon toga preuredite ovo stablo tako da postane heap. Prikazite koje ste pri tome sve izvrsili i kojim redoslijedom. Na kraju, prikazite koje znamene treba izvršiti ukoliko iz kreiranog heap-a želimo prvo ukloniti element najvećeg prioriteta, a zatim ubaciti element 45.

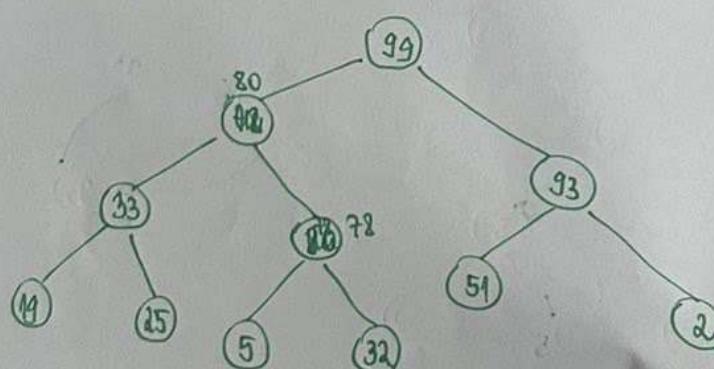
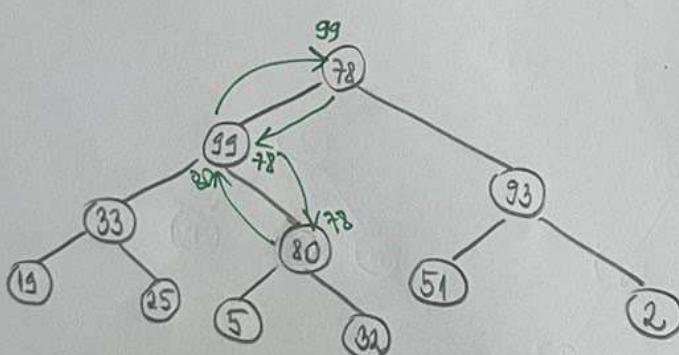
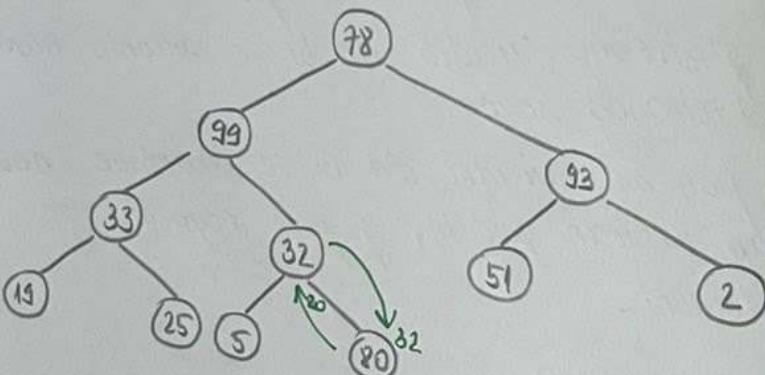
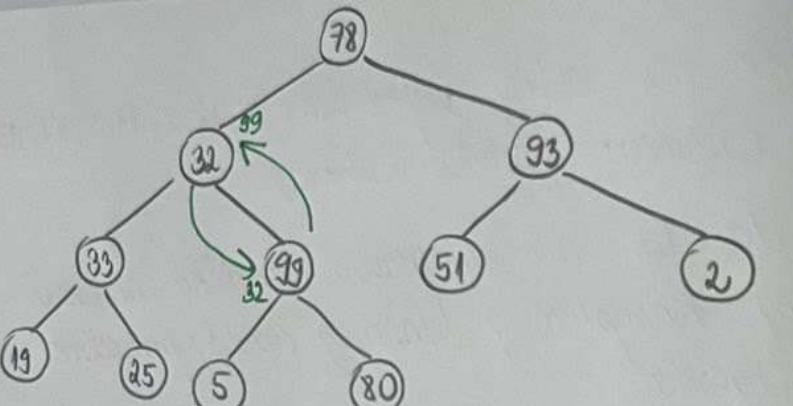
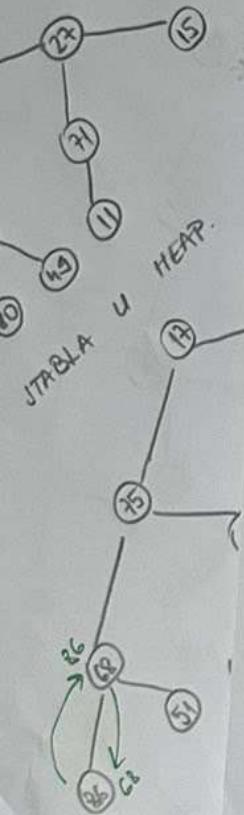
#### KOMPLETNO BINARNO STABLO



pr  
Na kraj u pr  
kreirao heap-a  
eleg prioriteta, a

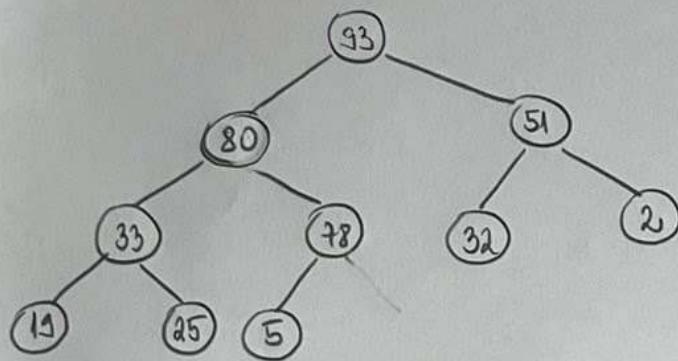
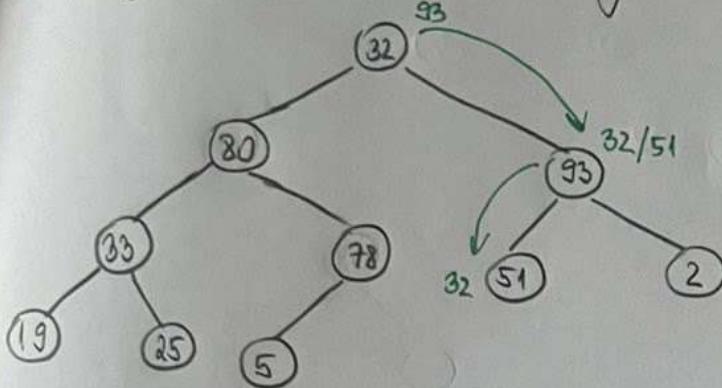
Uklanjanje

13.  
na

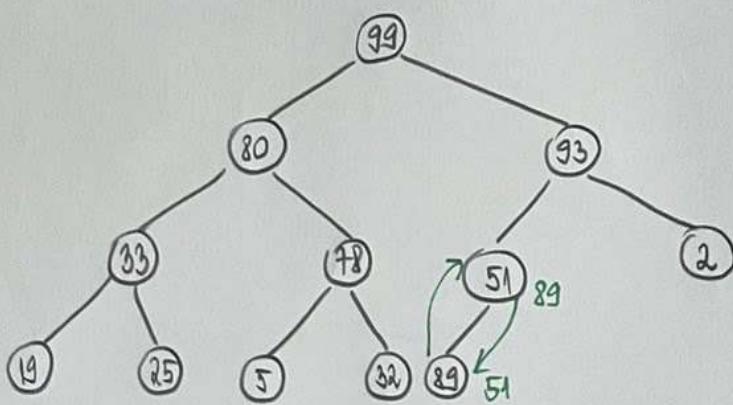


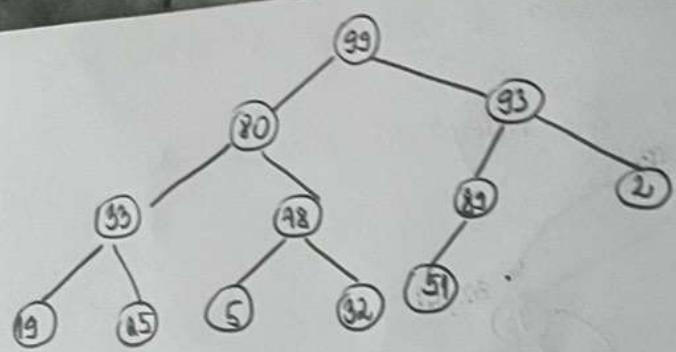
→ MAX-HEAP.

Uklanjanje najvećeg elementa iz heap-a, tj. elementa  
na mjesto elementa 99 stavljamo element 32.



d) element 89 dodajemo na kraj miza, tj. kao lijevo  
djelje elementa 32



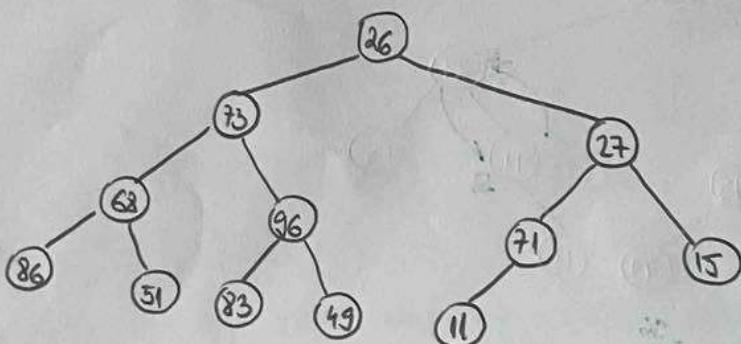


### HEAP (15 bodova)

Prepostavimo da neki niz sadrži vrijednosti: 26, 73, 27, 68, 96, 71, 15, 86, 51, 83, 49 i 11 (date u rastućem poretku indeksa).

- Prepostavljajući da taj niz predstavlja staticku nizovnu reprezentaciju nekog kompletног binarnog stabla, prikažite kako to stablo izgleda.
- Reorganizirajte to stablo da postane heap.
- Prikažite što je potrebno uraditi da bi se uklonio najveći element iz tako kreiranog heapa.
- Prikažite što je potrebno uraditi da bi se umetnuo novi element 78 u heap kreiran pod b) tј. prije nego što je uklonjen najveći element.

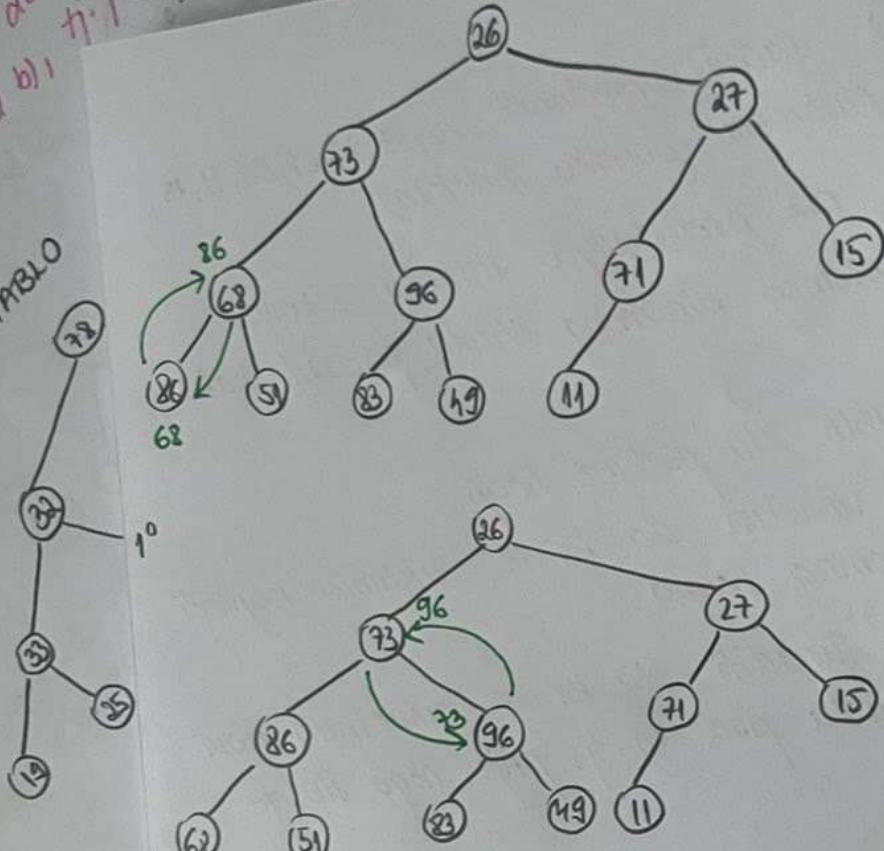
#### a) KOMPLETNO BINARNO STABLO



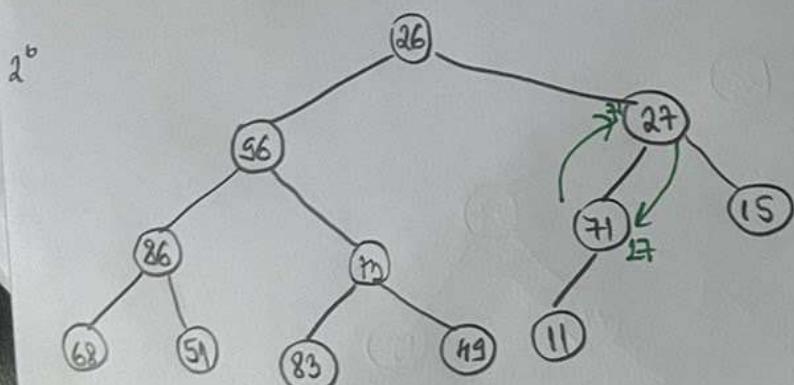
- Reorganizaciju moramo predstaviti koristeći određene zauyeue čvorova. Reorganizirajmo dato kompletно binarno stablo tako da postane max-heap.
- Počinjemo od listova i vrinjuo zauijenc.

na portane neg  
o uradit do bi se  
nog hepa.  
strebno uradit pod b), tj. preje nigo in  
ap kreiran element.

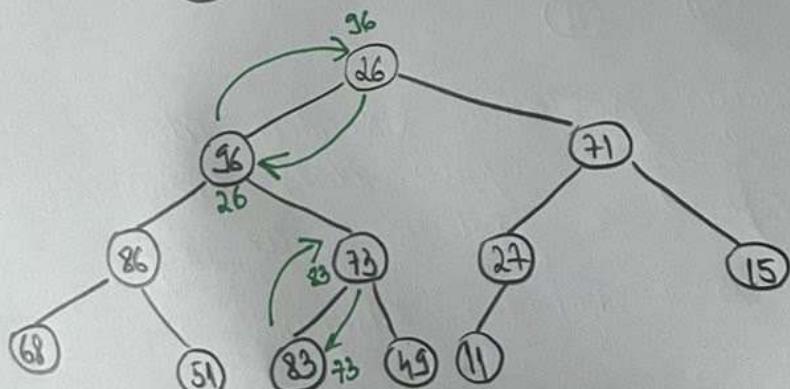
BINARNO IMBALO

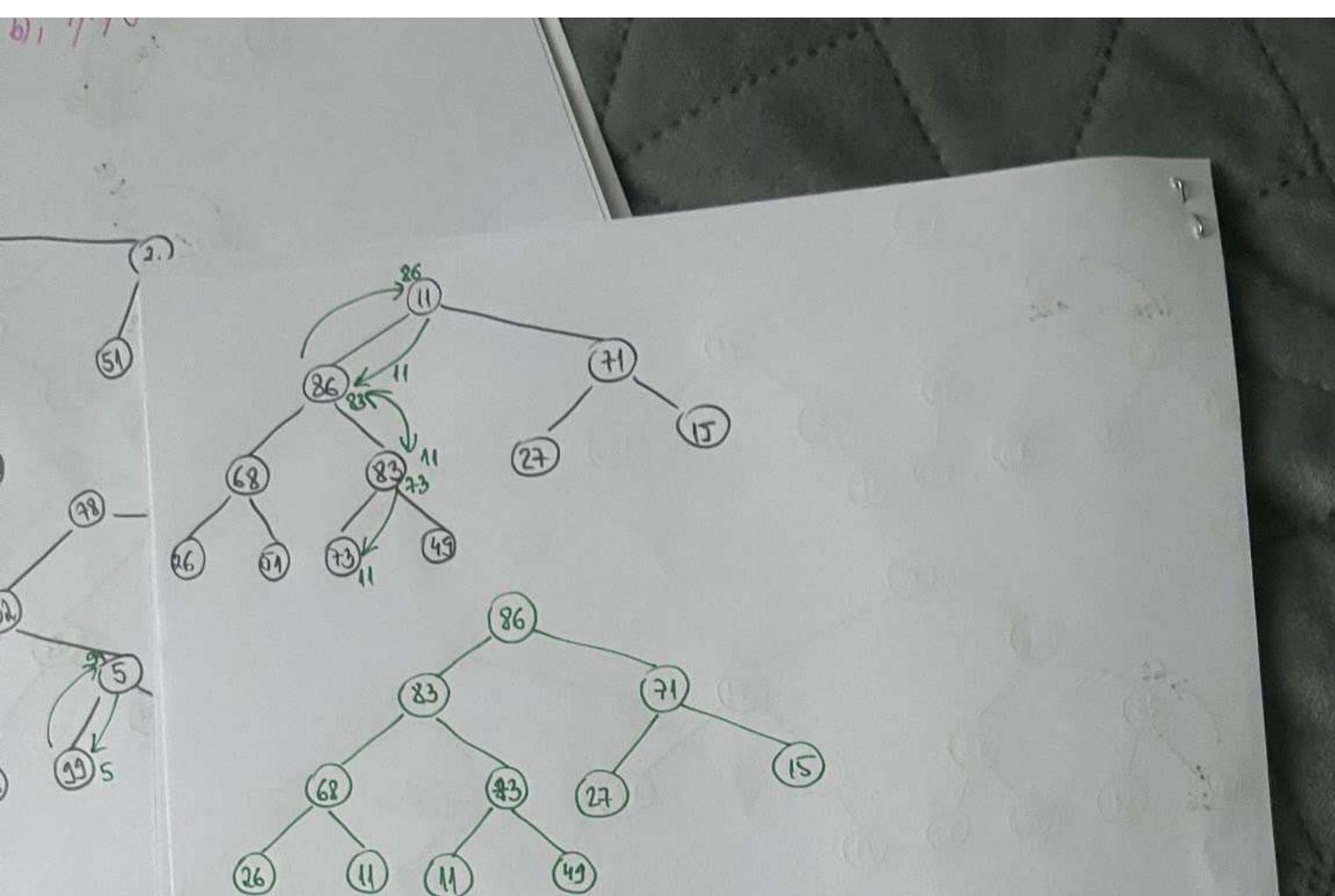


b)

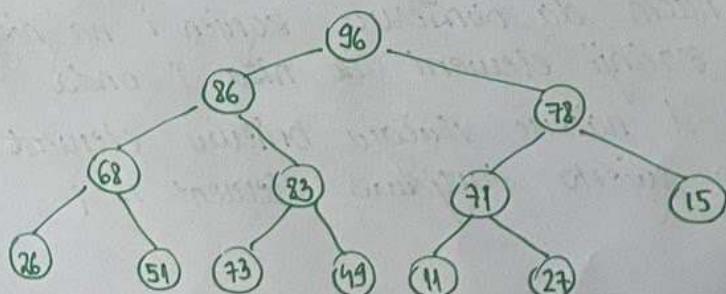
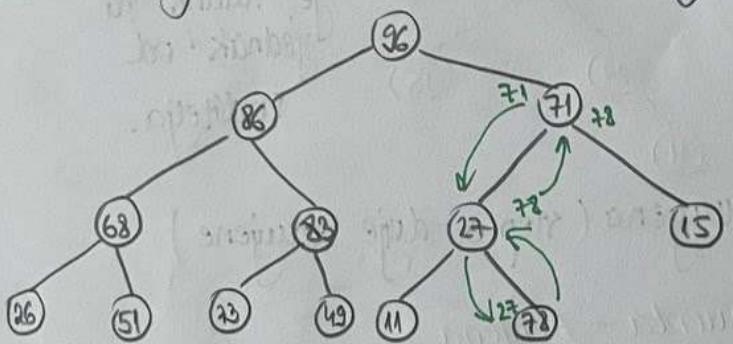


c)





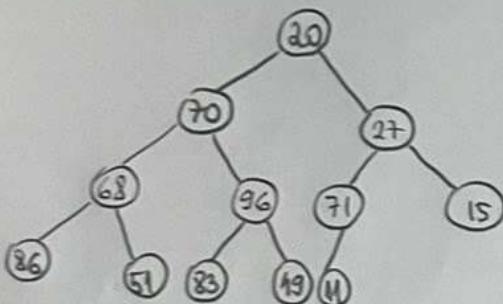
d) Unuetanje elementa 78 - dodajemo ga na kraj niza tako da on učinu prvi slobodan indeks, a potom vrćimo zaujene kako bi bili zadovoljeni uslovi max-heapa.



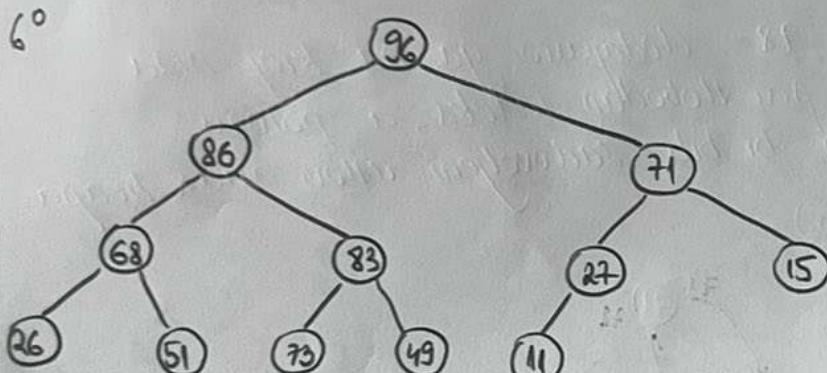
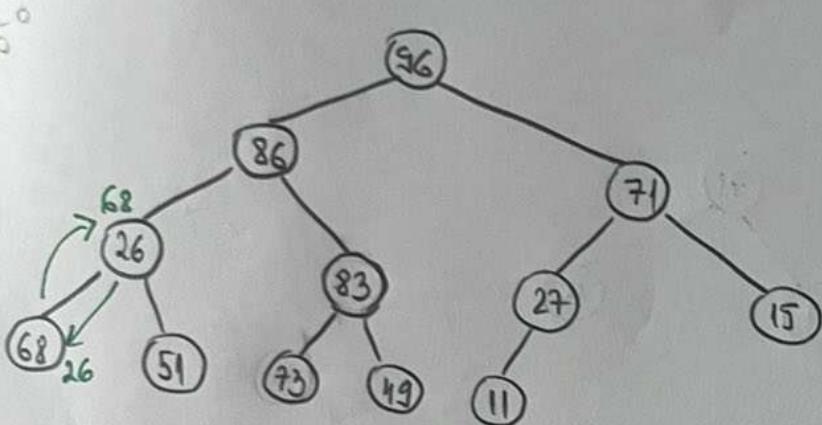
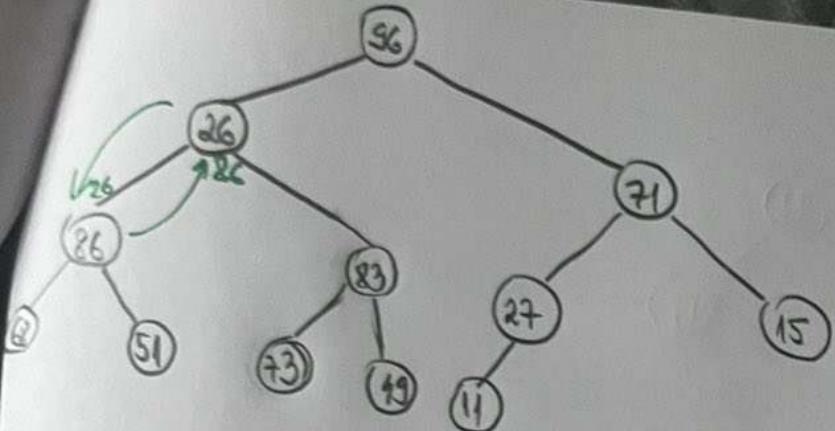
V uo ljevo V

b) Prikazite kako bi niz brojeva: 20, 70, 27, 68, 96, 71, 15, 86, 51, 23, 49 i 11 izgledao ako ga ponavljamo kao kompletno binarno stablo u formi niza. Nakon toga preuredite ovo stablo tako da postane heap. Prikazite koje ste pri tome sve zauvjene trebali izvršiti ukoliko iz kreiranog heapa želimo prvo ukloniti najveći element, a zatim ubaciti element 78.

#### KOMPLETNO BINARNO STABLO



Pogledati prethodne primjere radi se na isti način.



→ MAX-HEAP

savki i-ti čvor  
je manji ili  
jednak od  
roditelja.

Izbjrgeno ukupno 7 zauvjena (step 3 - dvije zauvjene)

c) Brisanje najvećeg elementa - korijena.

To radimo na način da obrnemo korijen i na njegovo mjesto stavimo zadnji element iz niza i onda vrniemo zauvjene. U našem slučaju brižemo element 96, a na njegovo mjesto stavljamo element 11 i vrniemo zauvjene.

jer se na isti način se radi.

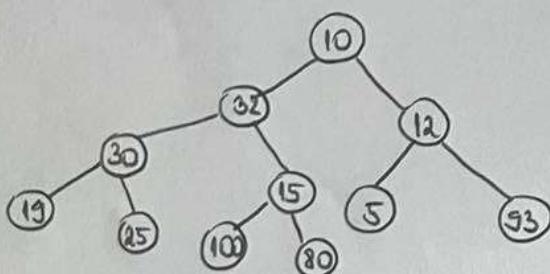
v - u ujeno v'

### HEAP (15 bodova)

Prepostavimo da neki niz sadrži vrijednosti: 10, 32, 12, 30, 15, 5, 93, 19, 25, 100 i 80 (date u rastućem poretku indekса).

- Prepostavljajući da taj niz predstavlja statičku nizomu reprezentaciju nekog kompletног binarnog stabla, prikažite kako to stablo izgleda.
- Aorganizirajte to stablo da postane heap.
- Prikažite što je potrebno uraditi da bi se uklonio najveći element iz tako kreiranog heapa.
- Prikažite što je potrebno uraditi da bi se unesuo novi element 89 u heap pod b), tj. prije nego što je uklonjen najveći element.

#### a) KOMPLETНО BINARNO STABLO



Pogledati primjere koji su urađeni na isti način se radi.

(ljevo rotacija)  
uio ljevo

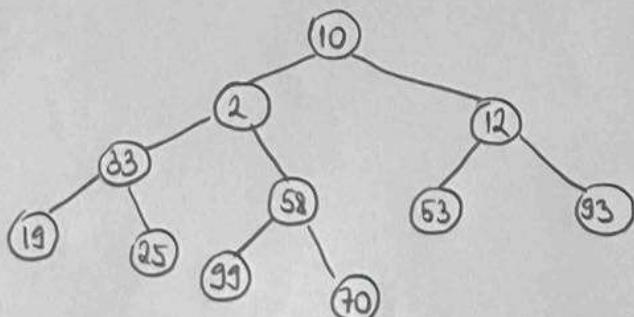
### HEAP (15 bodova)

Prepostavimo da neki niz sadrži vrijednosti 10, 2, 12, 33, 58, 53, 93, 19, 25, 99 i 70 (date u rastućem poretku indeksa).

- Prepostavljajući da taj niz predstavlja staticku nizovnu reprezentaciju nekog kompletног binarnog stabla, prikažite kako to stablo izgleda.
- Reorganizirajte to stablo tako da postane heap (prioritet je najveći element).
- Prikažite što je potrebno uraditi da bise uklonio najveći element iz tako kreiranog heapa.
- Prikažite što je potrebno uraditi da bi se unutru novi element u heap kreiran pod b), tj. prije nego što je uklonjen najveći element.
- Koja je vremenska složenost pravljenja heapa od niza sa n elemenata? Odgovor obradite.
- Objasniši kako radi heaps sort algoritam. Dokazite da mu je vreme izračunava u najgorem slučaju  $O(n \lg n)$ .

#### a) KOMPLETNO BINARNO STABLO

- popunjavanju nivoje redom elementima kako su navedeni u nizu.



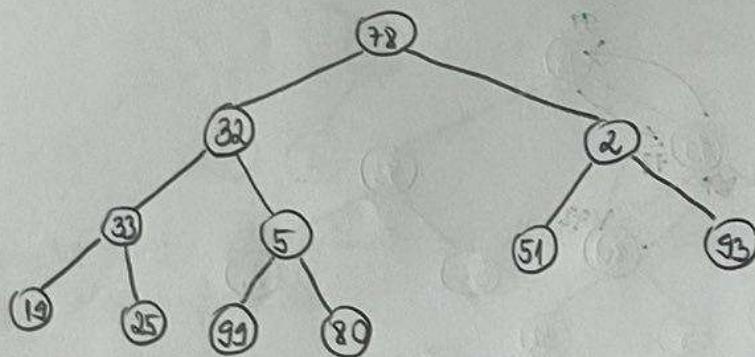
Pogledati prethodne prihvijere na isti način se radi.

### HEAP (15 bodova)

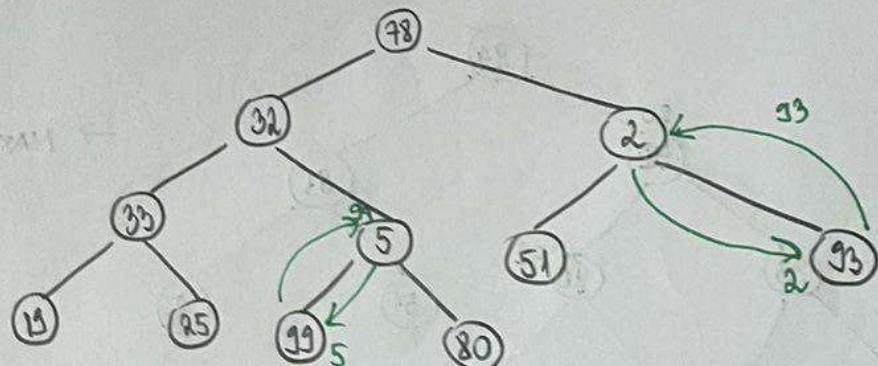
Pripremavimo da neki niz sadrži vrijednosti: 78, 32, 2, 33, 5, 51, 93, 19, 25, 99 i 80 (date u rasporedu poreklu indeksa).

- Prepostavljajući da taj niz predstavlja staticku nizovnu reprezentaciju nekog kompletog binarnog stabla, prikažite kako to stablo izgleda.
- Reorganizirajte to stablo da postane heap.
- Prikažite što je potrebno uraditi da bi se uklonio najveći element iz tako kreiranog heapa.
- Prikažite što je potrebno uraditi da bi se umetnuo novi element 89 u heap kreiran pod b), tj. prije nego što je uklonjen najveći element.

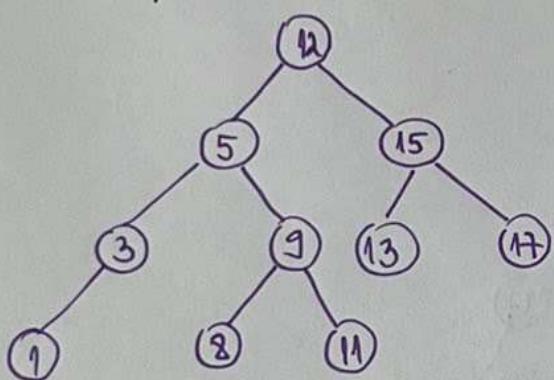
#### a) KOMPLETNO BINARNO STABLO



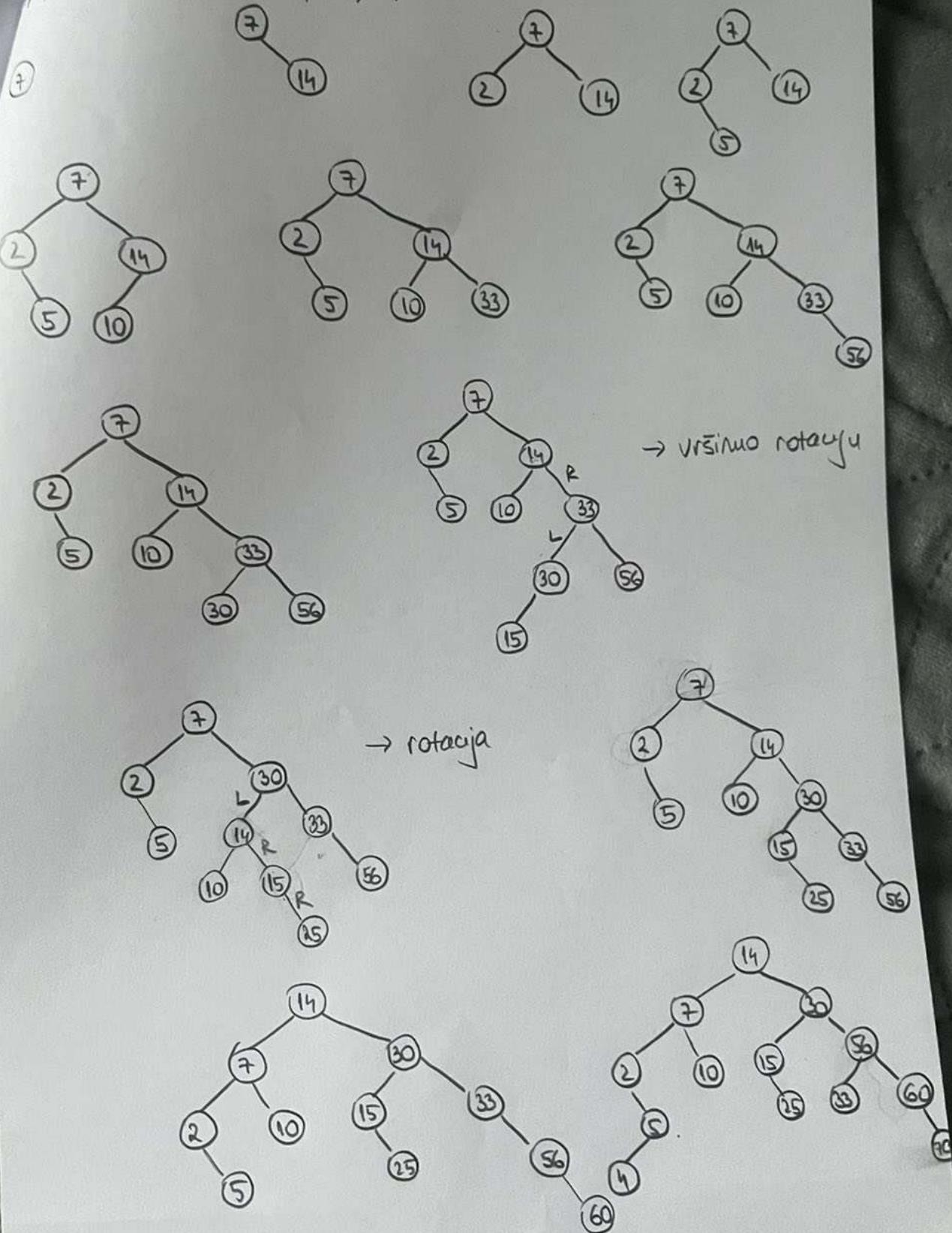
b)



-npr. briješmo čvor koji imao samo jedno dijete (desno ili lijevo), samo pozicioniramo njegovo dijete na njegovo mjesto.



izlistati AVL stablo od niza sljedećih elemenata:  
 2, 5, 10, 35, 56, 30, 15, 25, 68, 70, 4.



## AVL stablo - objašnjenje.

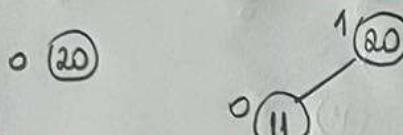
Benina Špijana

Pogledajmo slijedeći primjer:

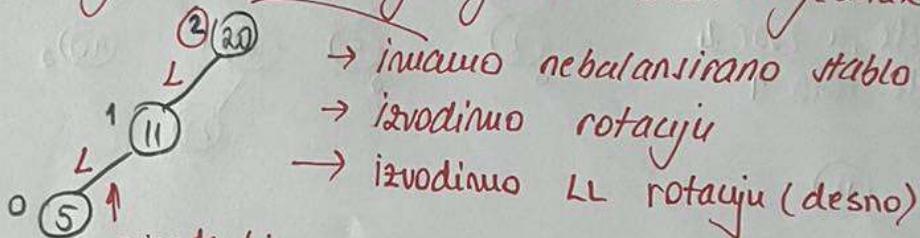
- konstruirati AVL stablo sa sljedećim elementima: 20, 11, 5, 32, 40, 2, 4, 27, 23, 28, 50.

→ unutar 20:  $\begin{array}{c} 20 \\ | \\ 11 \end{array}$

VAŽNO: nakon svakog umetanja čvora računamo faktor balansa AVL stabla. Faktor balansa jednak je razlici visina lijevog podstabla i desnog podstabla.



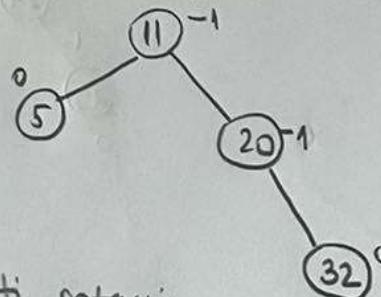
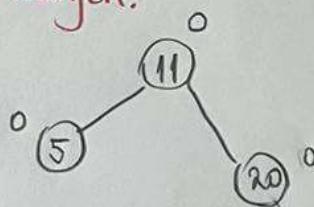
Kažemo da imamo problem i trebamo izvršiti balansiranje ukoliko je faktor balansa  $\neq -1, 0, 1$ .



- imamo nebalansirano stablo
- izvodimo rotaciju
- izvodimo LL rotaciju (desno)

pripada lijevom podstablu

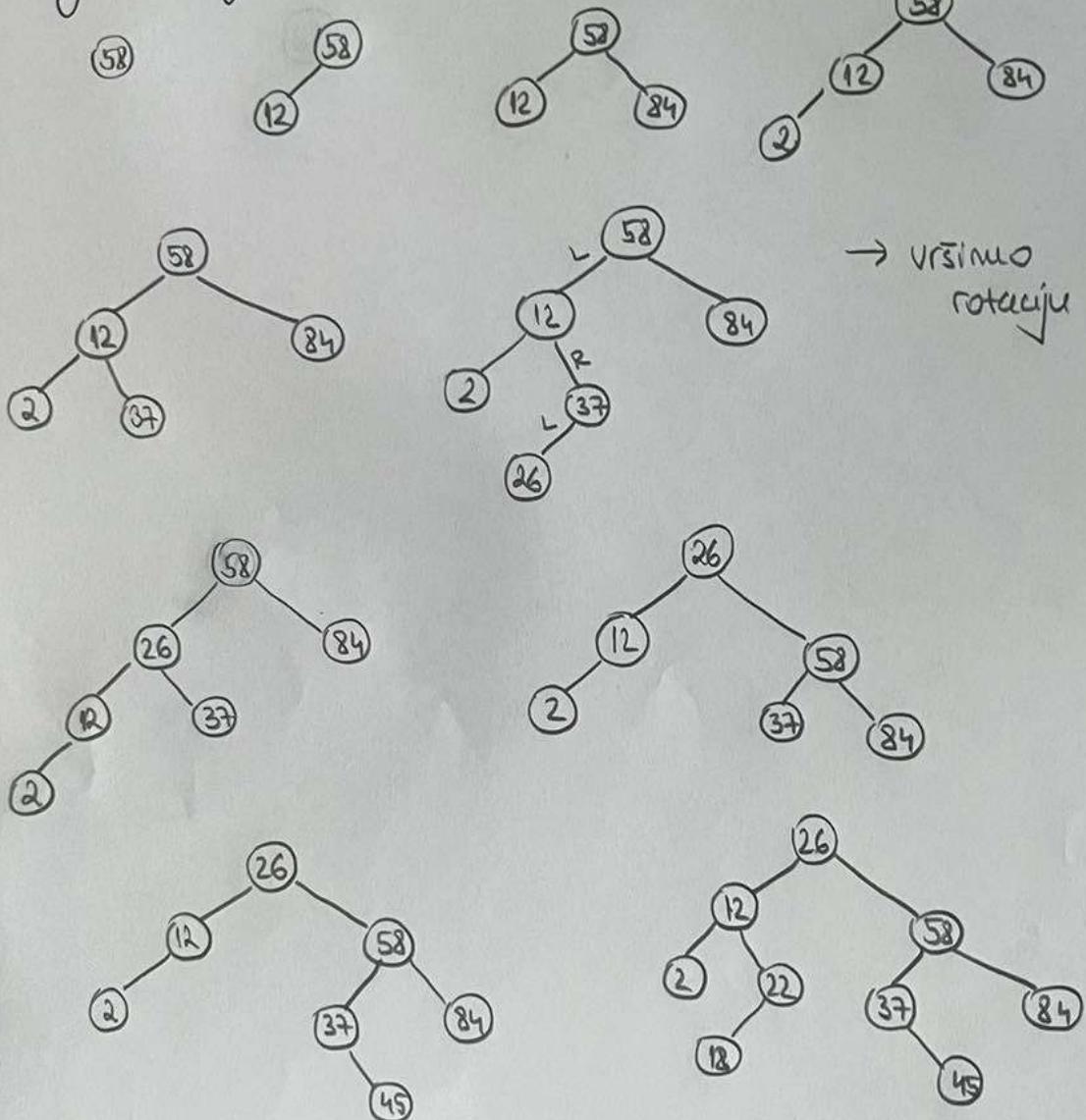
20 → čvor imao problem (jer je faktor balansa jednak -2), njega stavljamo u desno podstablo, a čvor 11 postaje koriđen.



- moramo izvršiti rotaciju
- RR rotaciju izvodimo
- čvor koji je problematičan (lijevo rotacija)
- 20 je problem njega stavljamo lijevo

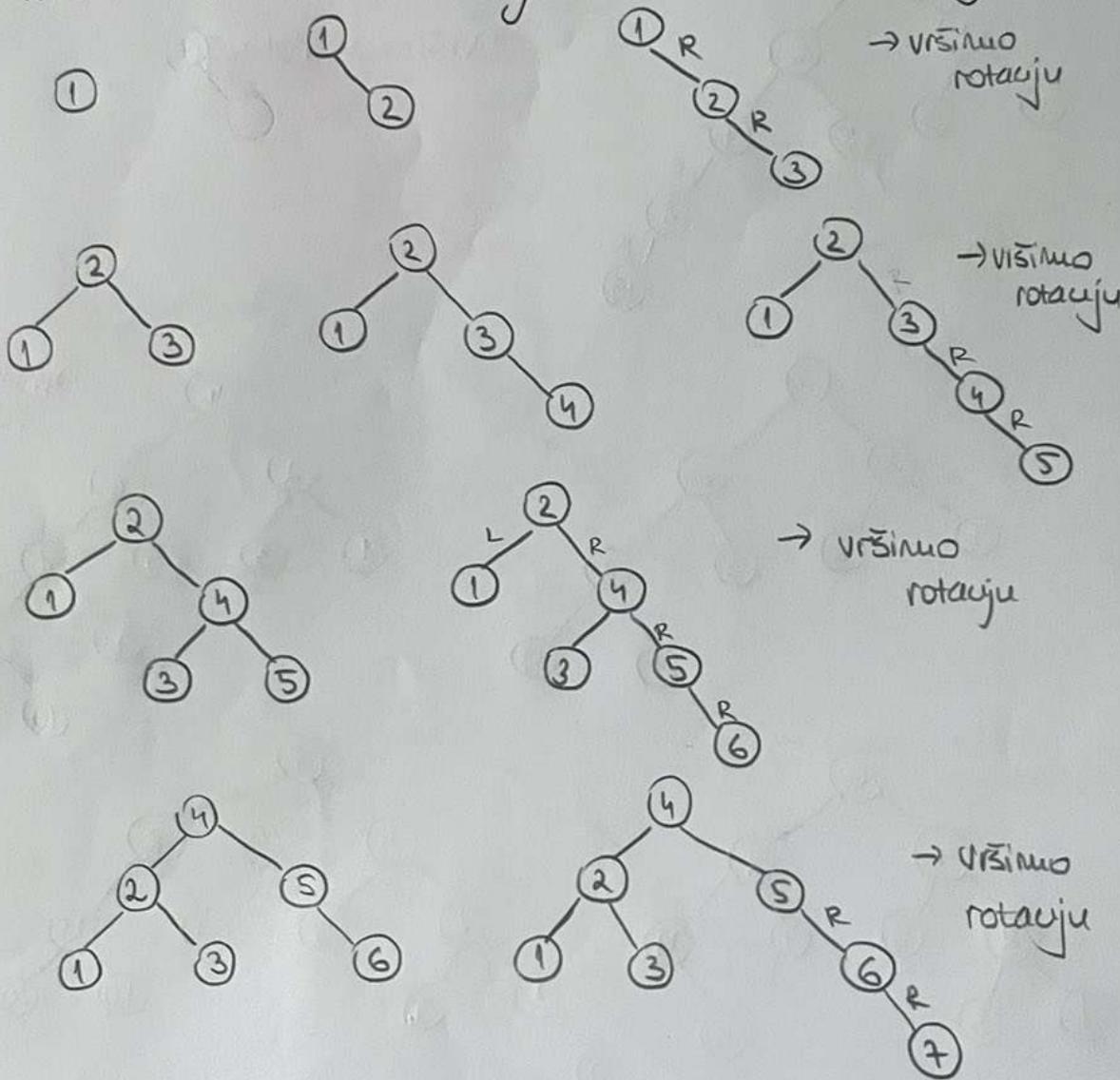
### AVL stablo (10 bodova)

Neka je dat sljed brojeva 58, 12, 2, 37, 26, 45, 32 i 18. Unutnjite ove brojeve u AVL stablo, uz prepostavku da brojevi nailaze redom kako su navedeni. Prikazite kako izgleda stablo nakon svakog unutjanja.



### 6. AVL stablo (15 bodova)

Dati definiciju AVL stabla. Pokazati da mu je vršina u najgorem slučaju  $O(\lg n)$ . Neka je dat redoslijed brojeva 1, 2, 3, 4, 6, 7, 8, 19, 10, 20, 0. Umjesto ove brojeve u AVL binarno stablo pretrage, uz pretpostavku da brojevi nailaze redoslijedomu kako su navedeni. Prikazite kako izgleda stablo nakon svakog koraka.



## HASH TABELE

### DJSTRUKO HASHIRANJE

Dati su ključevi: 3, 2, 9, 6, 11, 13, 7, 12. Veličina hash tabele je

$$N=10. \quad h_1(k) = 2k+3$$

$$h_2(k) = 3k+1.$$

Uneseni k na prvo prazno mjesto od  $(u+v*i) \% N$ ,  
gde je  $i = 0, N-1$ .

vrijednost izračunata  
izračunata u  $h_1(x)$  i.  $h_2(k)$

0	
1	9
2	
3	11
4	12
5	6
6	
7	2
8	
9	3

KLJUČ	LOKACIJA L	V	PROBA
3	9	—	1
2	7	—	
9	1	—	1
6	5	—	1
11	5	4	23
13	9	0	NE MOŽENO
7	7	2	NE MOŽENO
12	7	7	

predstavlja broj pokusaja da unesemo ovaj ključ u hash

tabelu (ako je odmah slobodno upisujemo 1)

ako možemo upisati u hash tabelu onda ne računamo V

ako je zauzeto njegovo mjesto onda računamo V.

u ovoj slučaju koristimo dvostruko hashiranje, jer je pozicija 5 zauzeta. Način na koji to radimo je:

$$(u+v*i) \bmod N = 10$$

$$h_1(u) = (2 \cdot 11 + 3) \bmod 10 = 5$$

$$h_2(u) = (3 \cdot 11 + 1) \bmod 10 = 4$$

$$(u+v*i) \bmod 10 = (5+4*0) \bmod 10 = 5 \rightarrow \text{zauzeto}$$

idemo dalje  $i=1$

$$(5+4*1) \bmod 10 = 9 \rightarrow \text{zauzeto}, \text{ idemo dalje } i=2.$$

## HAŠT TABELE

Benja pisan

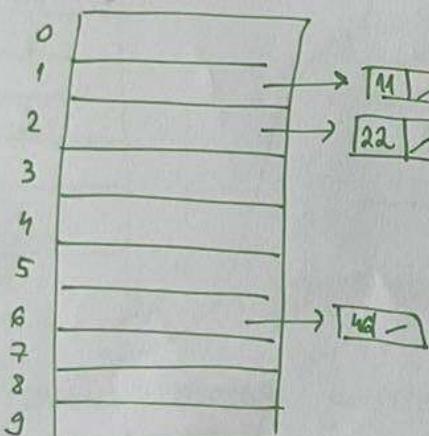
- 1) Odvojeno nizanje
- 2) Linearno isprobavanje
- 3) Dvostruko hashiranje.

### ODVOJENO NIZANJE:

Potrebno je da izračunamo lokacije u našoj hash tabeli koje zauzimaju date vrijednosti.

To računamo kao  $H(\text{kљuc}) = \text{kљuc MOD } N$  (gde  $N$  predstavlja kapacitet naše hash tabele).

Uzmimo npr. neka je data slijedeća hash tabela sa  $N=10$ .  
22, 11, 32, 71, 46



$$\begin{aligned} 22 \text{ MOD } 10 &= 2 \\ 11 \text{ MOD } 10 &= 1 \\ 32 \text{ MOD } 10 &= 2 \\ 71 \text{ MOD } 10 &= 1 \\ 46 \text{ MOD } 10 &= 6 \end{aligned}$$

### LINEARNO ISPRAVANJE

	VRIJEDNOST	POK
0		
1	11	
2	22	
3	32	
4	71	
5		
6	46	
7		
8		
9		

→ u slučaju da nam je zauzet indeks na koji trebamo izvršiti umetanje hash popunjavamo tom vrijednosću na prvo slobodno mjesto

## HAJH TABELE

Benina prijam

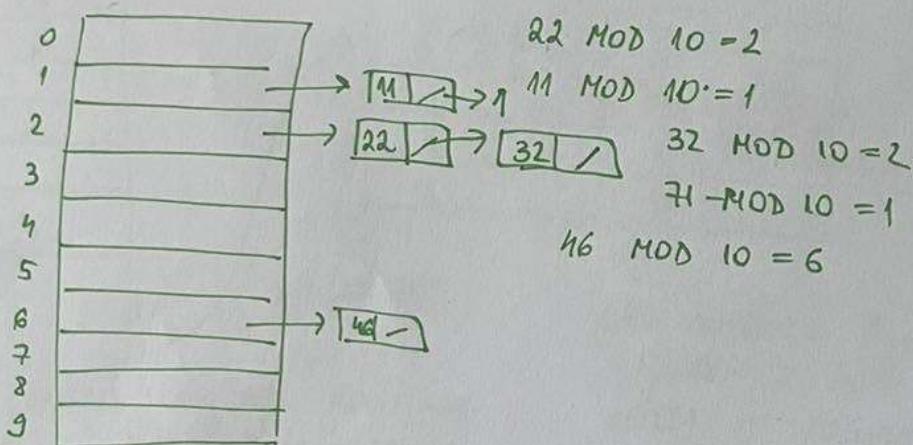
- 1) Odvojeno nizanje
- 2) Linearno isprobavanje
- 3) Dvostruko hashiranje.

### ODVOJENO NIZANJE:

Potrebno je da izračunamo lokacije u našoj hash tabeli koje zauzimaju date vrijednosti.

To računamo kao  $H(\text{klijen}) = \text{klijen} \bmod N$  (gde je  $N$  predstavlja kapacitet naše hash tabele).

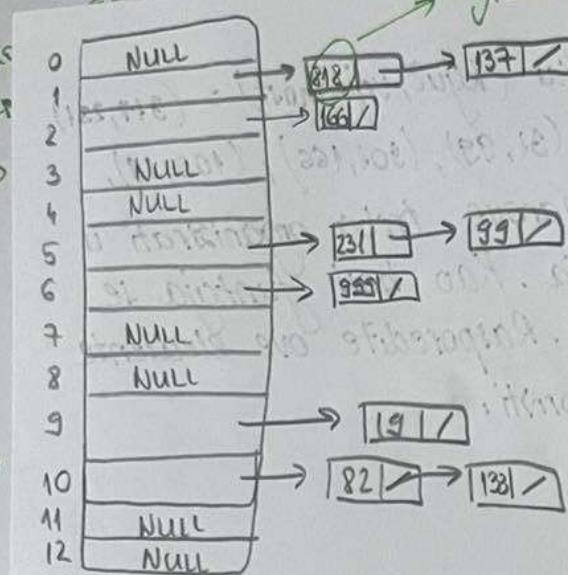
Uzmuimo npr. neka je data sljedeća hash tabela sa  $N=10$ .  
 22, 11, 32, 71, 46



### LINEARNO ISPRAVANJE

	VRJEDNOST	POK
0	11	
1	22	
2	32	
3	71	
4		
5		
6	46	
7		
8		
9		

→ u slučaju da nam je zauzet indeks na koji trebamo izvršiti unetanje hash popunjavanju tom vrijednost u prvo slobodno mjesto



VRIJEDNOST	POK
NULL	
218	
137	
166	
NULL	
231	
99	
999	
NULL	
19	
82	
133	
NULL	

DOUBLE HASHING:

$$\text{hash1}(\text{klijec}) + j \cdot \text{hash2}(\text{klijec})$$

Indeks koji rimožimo u slučaju  
kada nam je zauzeta  
lokacija na koju pokusavamo  
ubaciti element

gleđamo koja je vrijednost za klijec  
i ovdje je upravljanje

→ LINEARNO ISPROBAVANJE

→ kada nađemo na  
problem da  
nam je  
zauzeto mjesto  
idemo na prvo  
slijedeće koje je  
slobodno, npr. 5  
zauzeto idemo na 6  
ako je 6 zauzeto  
idemo dalje dok  
ne dođemo do  
prvog slobodnog  
mjetsta.

STRUKO MATIHETANJE  
provjeravanje  
možemo  
u koliko je  
funkcija  
pre

DVOSTRUKO HASHIRANJE  $h_2(x) = 1 + (x \bmod 4)$ .

- provjeravamo da li vrijednost pod određenim ključem možemo umetnuti u hash tabelu
- ukoliko je ta pozicija zauzeta onda koristimo sekundarnu funkciju  $h_2(x)$  i to na sljedeći način:

$$[h_1(x) + j \cdot h_2(x)] \bmod N \text{ gdje } j = \overline{0, N-1}, \text{ a } N$$

predstavlja kapacitet, odnosno broj polja hash tabele.

0	
1	818
2	166
3	
4	
5	231
6	999
7	
8	
9	99
10	82
11	19
12	133

KLJUČ	$h_1(x)$	$h_2(x)$	VRJEDNOST	ISPOBAVANJE
317	5	—	231	1
192	10	—	82	1
49	10	2	133	2
313	1	—	818	1
144	1	1	137	2
31	5	1	99	2
301	2	—	166	1
700	9	1	19	3
19	6	—	999	1

$$h_1(x) = x \bmod N$$

\* Kod ključa 49 vidimo da je indeks 10 u hash tabeli zauzet pa koristimo dvostruko hashiranje.

$$h_2(x) = 1 + (x \bmod 4) = 1 + 1 = 2$$

$$[h_1(x) + j \cdot h_2(x)] \bmod N$$

$$j=0 \Rightarrow (10 + 2 \cdot 0) \bmod \underset{=10}{N} = 10 \rightarrow \text{zauzeto}$$

$$j=1 \Rightarrow (10 + 2 \cdot 1) \bmod \underset{=10}{N} = 12 \rightarrow \text{slobodno}$$

\* Kod ključa 144 vidimo da je indeks 1 u hash tabeli zauzet pa primjenjujemo dvostruko hashiranje.

$$h_2(144) = 1 + (144 \bmod 4) = 1$$

$$j=0 \Rightarrow [1 + 1 \cdot 0] \bmod 13 = 1 \rightarrow \text{zauzeto}$$

### H/SH TABELA (12 bodova)

Dati su sljedeći parovi u obliku (ključ, vrijednost): (317, 231), (192, 82), (49, 133), (313, 818), (144, 137), (31, 99), (301, 166), (100, 19), (19, 999). Pretpostavimo da ove parove treba organizirati u hash tabelu kapaciteta  $N=13$  polja. Kao hash funkcija se koristi funkcija  $h(x) = x \bmod N$ . Rasporedite ove elemente u hash tabelu ukoliko se koristi:

- a) Odvojeno nizanje
- b) Linearno interpoliranje
- c) Dvostruko hashiranje sa sekundarnom hash funkcijom  $h_2(x) = 1 + (x \bmod 4)$ .
- d) Objasnite zašto pri brisanju podataka iz hash tabele nije dovoljno samo sloboditi odgovarajući slot, nego je potrebno poduzeti još nešto.

#### a) ODVOJENO NIZANJE

- posmatramo u parovima ključeve, njih posmatramo po modulu  $N$  (kapacitet hash tabele) na osnovu toga dobijamo indekse na koje trebamo izvršiti unetanje
- u slučaju da je neki indeks prethodno popunjen idemo na prvi slobodan
- u hash tabelu upisujemo vrijednost iz para ne ključ.

$$317 \bmod 13 = 5$$

$$301 \bmod 13 = 2$$

$$192 \bmod 13 = 10$$

$$100 \bmod 13 = 9$$

$$49 \bmod 13 = 10$$

$$19 \bmod 13 = 6$$

$$313 \bmod 13 = 1$$

$$144 \bmod 13 = 1$$

$$31 \bmod 13 = 5$$

ien

39.

↳ Dvostrukko hashiranje za ključ 39.

$$h_2(39) = 4 - (39 \bmod 4) = 1$$

↳  $j=0 \Rightarrow (5+1*0) \bmod 17 = 5 \rightarrow \text{stavzeto}$

↳  $j=1 \Rightarrow (5+1*1) \bmod 17 = 6 \rightarrow \text{stavzeto}$

↳  $j=2 \Rightarrow (5+1*2) \bmod 17 = 7 \rightarrow \text{stavzeto}$

↳ Dvostrukko hashiranje za ključ 15.

$$h_2(15) = 4 - (15 \bmod 4) = 1$$

↳  $j=0 \Rightarrow (15+1*0) \bmod 17 = 15 \rightarrow \text{stavzeto}$

↳  $j=1 \Rightarrow (15+1*1) \bmod 17 = 16 \rightarrow \text{stavzeto}$

ANALIZA JLOŽENOJTE ALGORITAMA (20 bodova)

```

int k, s(0);
for (int i=1; i<=n; i++) {
    k = n-1;
    while (k > 0) {
        for (j=1; j<=2*i; j*=2) s += i-k; log2n
        k /= 3;
    }
}

```

RJEJENJE:

BEST CASE:  $n=1$ ,  $T(1)=c$ .  
 AVERAGE CASE:

$$T(n) = c \cdot (n-1) \cdot \log_3 n \cdot \log n \rightarrow \text{dupla se } n \text{ for petriji}$$

jer se k smanjuje

$$T(n) \propto O(n \log^2 n)$$

monu  
podataka iz hali t  
odgovarajuću slotu.  
boditi nevojno.

ovinu  
sacitet  
indeks  
da  
pri  
tabel

juč.  
mod  
mo  
n  
313  
14.

### b) LINEARNO ISPROBAVANJE

0	72
1	
2	
3	
4	
5	11
6	22
7	27
8	44
9	17
10	92
11	89
12	30
13	
14	
15	31
16	26

- pozicija za ključ 56 (index 5) je zauzet i tražimo iza tog indeksa prvi slobodan na koji možemo upisati vrijednost 27  
→ indeks 6 popunjen pa vrijednost 27 upisujemo na indeks 7.

### c) DVOSTRUKO HAJHIRANJE SA sekundarnom funkcijom $h_2(x) = h - (x \bmod 4)$

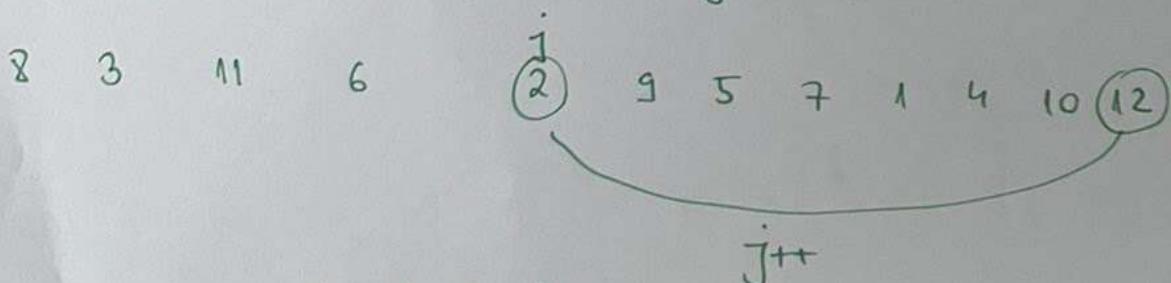
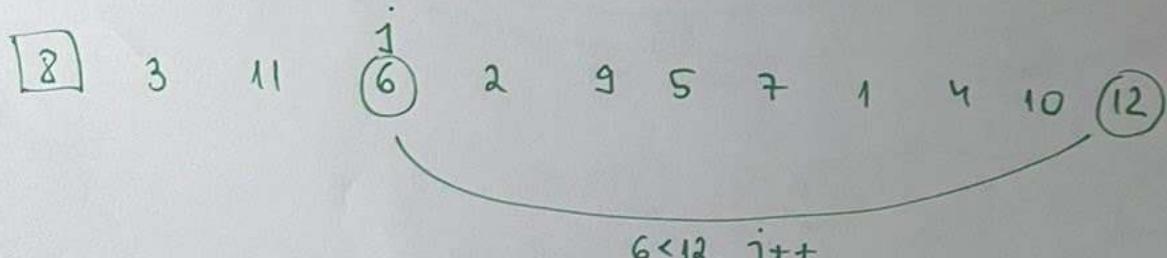
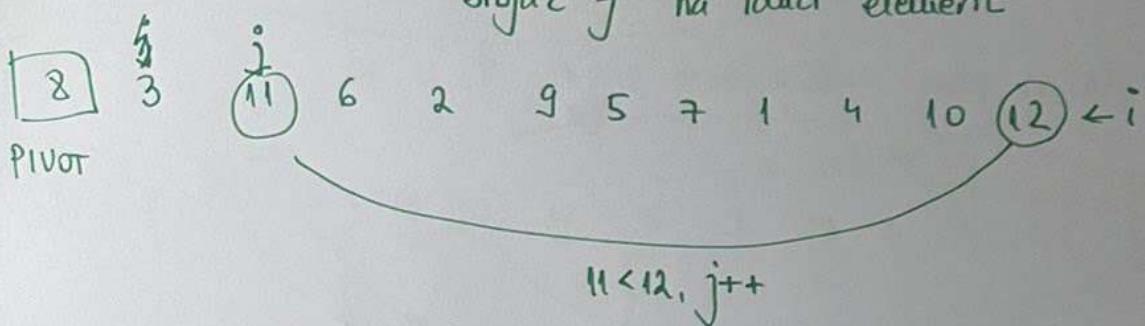
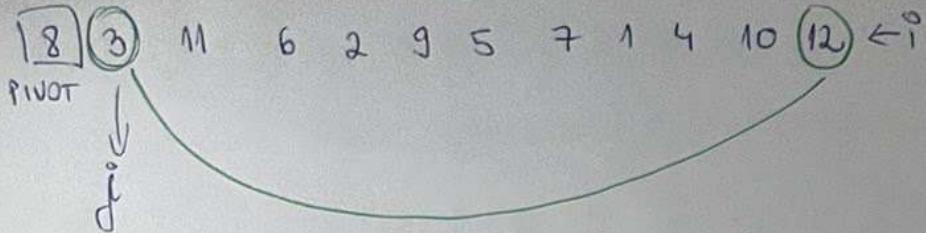
0	26
1	
2	
3	
4	
5	11
6	22
7	30
8	
9	27
10	92
11	17
12	44
13	
14	89
15	31
16	72

KLJUČ	$h_1(x)$	$h_2(x)$	VRIJEDNOST	BRAV PROBAVANJA
6	6	—	22	1
73	5	—	11	1
10	10	—	92	1
56	5	4	27	2
44	10	4	89	2
32	15	—	31	1
26	9	2	17	2
57	6	3	44	3
66	15	2	26	2
39	5	1	30	3
15	15	1	72	2

↳ kod ključa 56 vidimo da moramo iskoristiti dvostruko hajhiranje, jer je index 5 već popunjeno.

$$h_2(56) = h - (56 \bmod h) = 4$$

## QUICKSORT PARTITION ALGORITAM



Nema ovde nikakvih zamjena.

### b) Recursive Max

$T(n) \rightarrow$  vrijeme izvršavanja datog algoritma

$T(1) = c \rightarrow$  vraćamo  $A[1]$  jer je konstantno vrijeme  
za  $n=1$

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

$$T(n-1) = 2c + T(n-2) \dots = \dots =$$

$$= k \cdot c + T(n-k)$$

$$\sum_{i=0}^{n-1} 1 \cdot c = c \cdot (n-1) = \underline{\underline{cn - c}}$$

$$T(n) \in O(n)$$

### 3) Divide And Conquer

$T(1) = c \rightarrow$  za  $n=1$  treba naći konstantno vrijeme da vratišmo prvi element niza

$$T(n) = c + T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}\right)$$

$$T(n) = c + 2T\left(\frac{n}{2}\right)$$

$$T\left(\frac{n}{2}\right) = 2c + 2T\left(\frac{n}{2^2}\right)$$

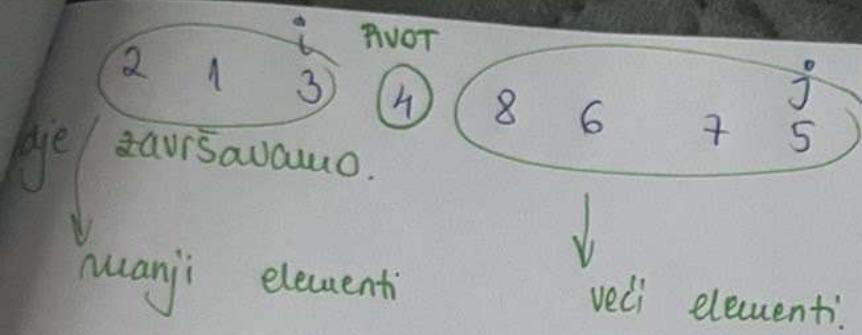
;

$$k = \log_2 n$$

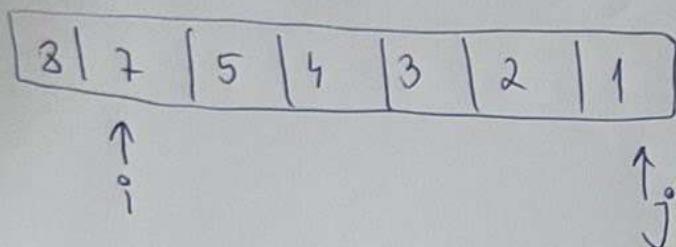
$$\log_2 n - 1$$

$$\sum_{i=0}^{\log_2 n - 1} 1 \cdot c = c \cdot \log_2 n$$

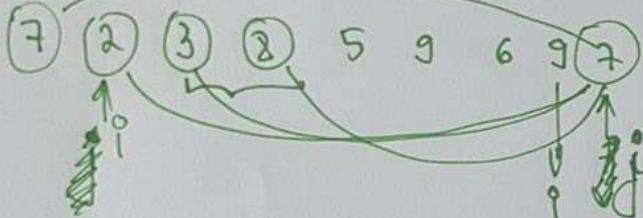
$$T(n) \in O(\lg n)$$



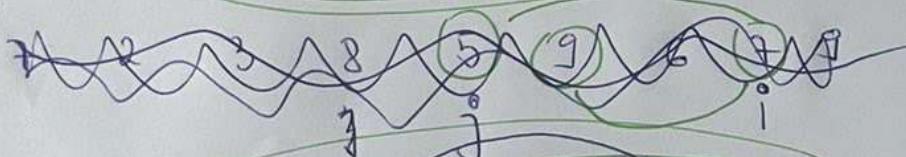
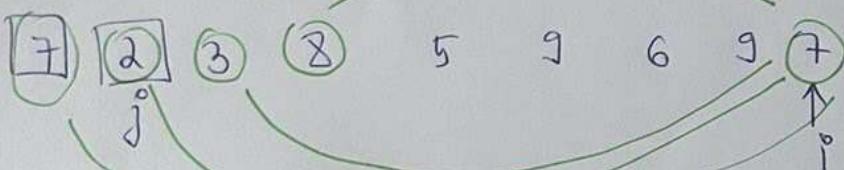
PRVI ELEMENT KAO PIVOT.



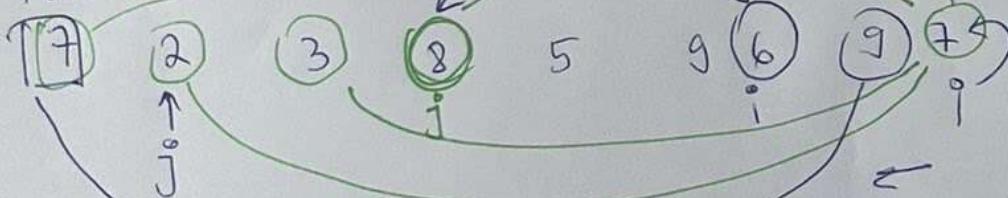
1 7 5 4 3 2 8



PIVOT

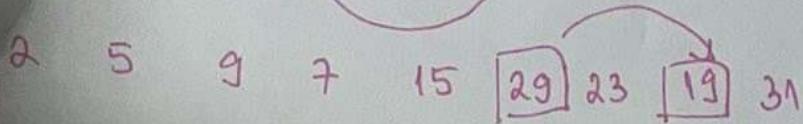
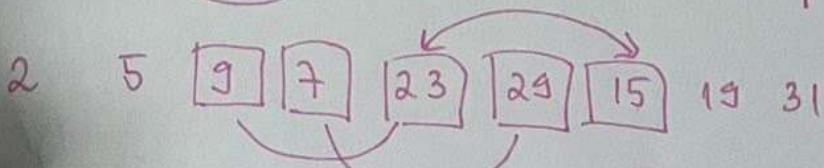
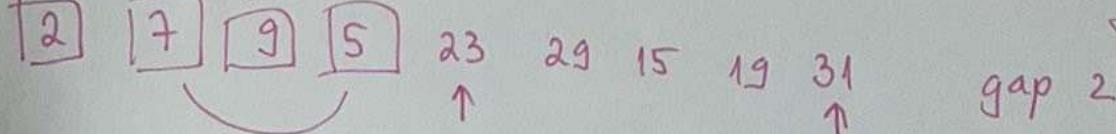
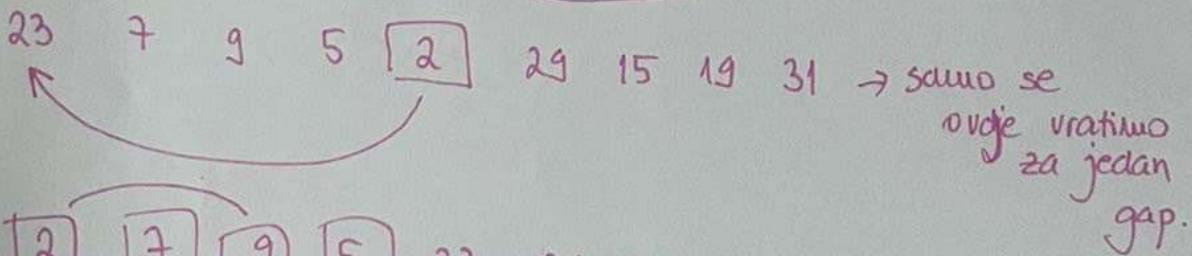
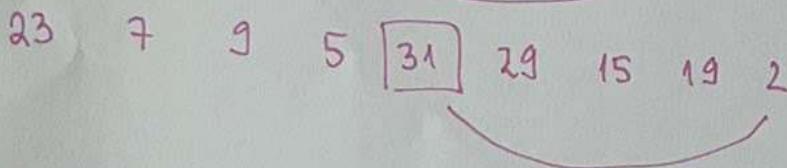
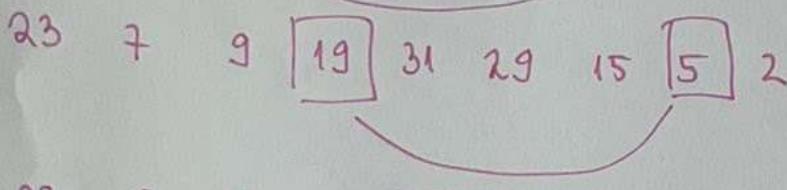
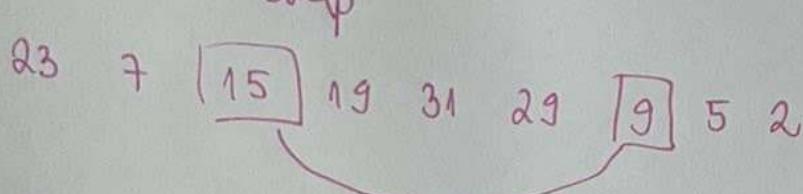
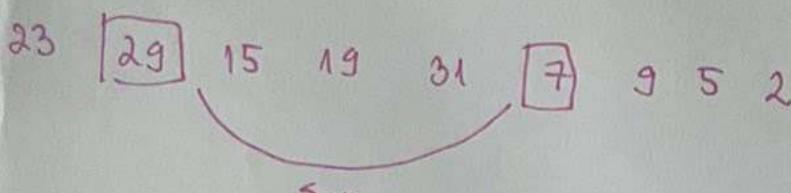
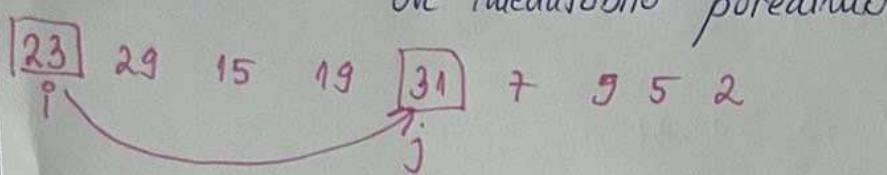
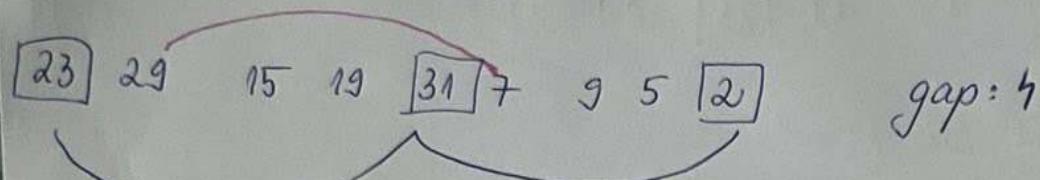


PIVOT

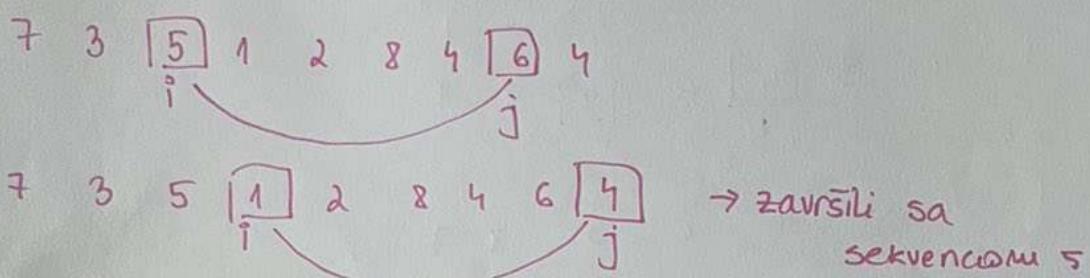
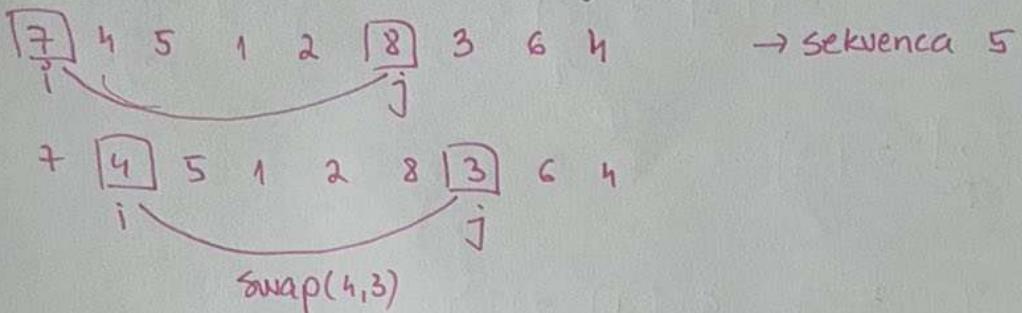


veci ponjera seda je

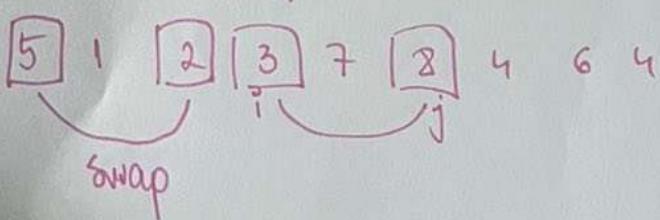
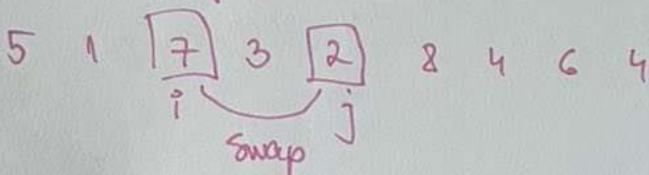
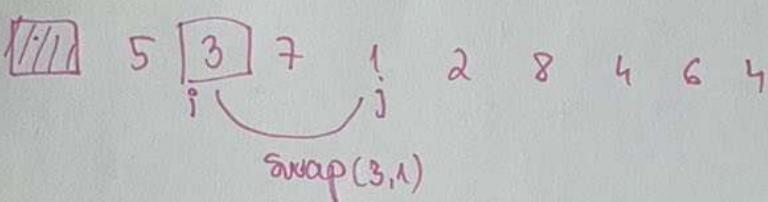
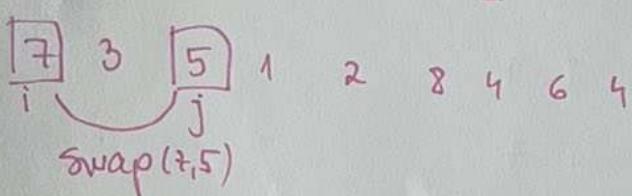
### SHELL SORT



e) Dat je niz elemenata  $7, 4, 5, 1, 2, 8, 5, 3, 6$  i 4. Sortirajte ovaj niz shell sort postupkom koristeći opadajuću sekvencu  $5, 2, 1$ . Prikazite sve korake obavljene u sortiranju.



↳ prelazimo na sekvencu 2



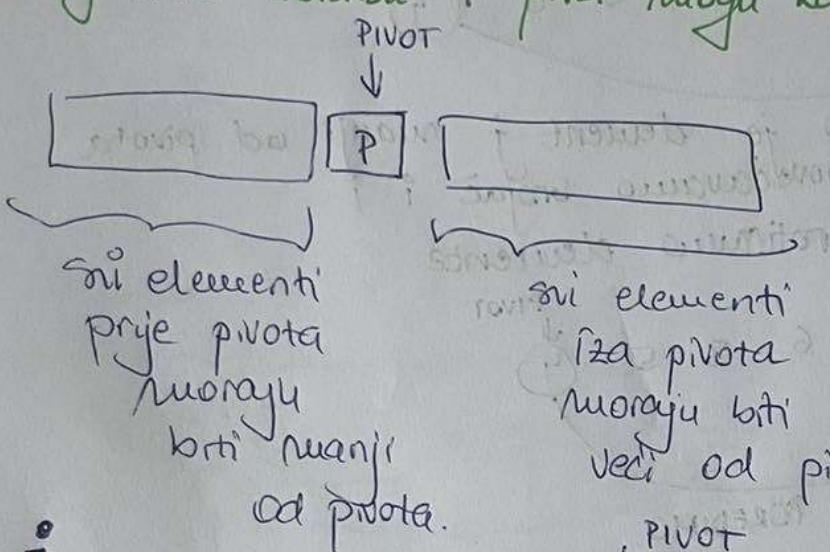
## QUICK SORT PARTITION ALGORITHM

7	1	3	5	2	6	4
0	1	2	3	4	5	6

→ PIVOT

pi - indeks na koji možemo postaviti pivota

Potencijalni indeks na koji možemo staviti pivota recimo da je 0. To znači da se element koji se nalazi na potencijalnom indeksu i pivot mogu zamijeniti.

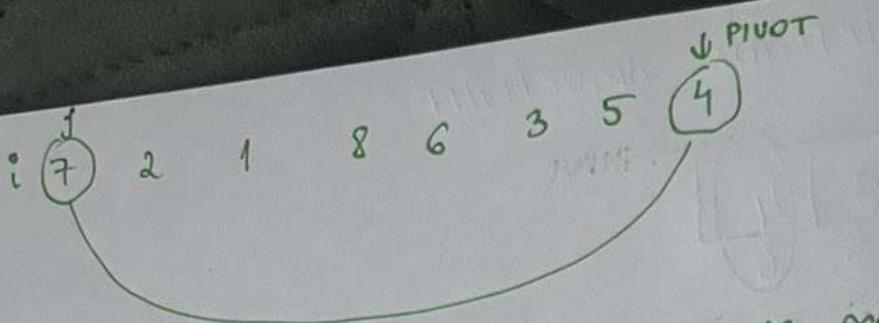


i	j	7	2	1	8	6	3	5	4
---	---	---	---	---	---	---	---	---	---

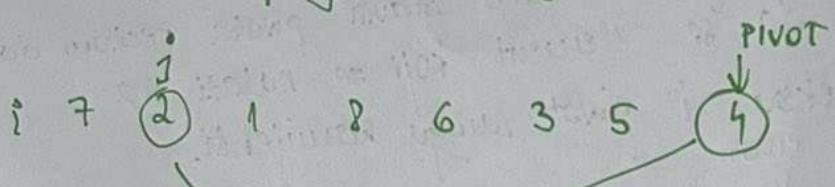
moraće imati dva brojača, jedan je i

Poredimo uvjek pivot i onaj element koji je na poziciji j.

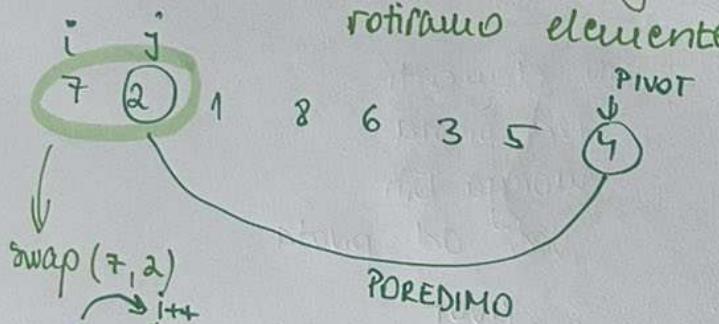
ako je element j na tjoj poziciji veći  
od pivota ne radimo nista i povećavamo  
brojac j.



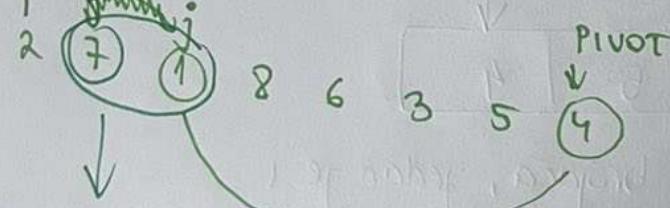
da li je  $a[j] > \text{pivot}$ , ne radi ništa  
ponajezavimo brojač na slijedeći element



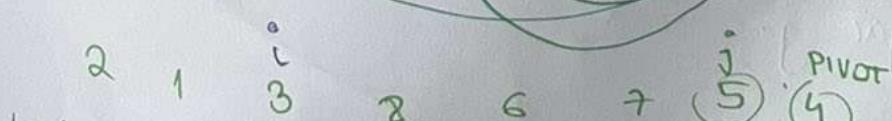
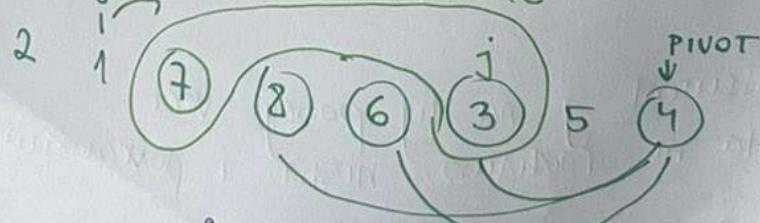
ako je element j manji od pivot-a  
povećavamo brojač i i  
rotiramo elemente



POREDIMO



POREDIMO



kad dodemo do kraja pivot ubacujemo na  
i+1 poziciju u našem slučaju t2 a 3

### ANALIZA JUŽENOSTI ALGORITAMA (15 bodova)

Za svaki od sljedećih kodova detaljno izračunati vrijeme izvršavanje i dati najbolje asinhaptotičke projene tog vremena u najgorem slučaju koristeći asinhaptotičke notacije u zavisnosti od  $n$ :

(a) Funny ( $A[1..n]$ )

1. for  $i \leftarrow 1$  to  $n$
2.   for  $j \leftarrow i$  to  $n$  do  $A[j] = A[i] + A[j]$

(b) Recursive Max ( $A[1..n]$ )

1. if  $n=1$  then return  $A[1]$  else
2.  $a \leftarrow \text{RecursiveMax}(A[2..n])$
3. if  $a > A[1]$  then return  $a$  else return  $A[1]$ .

c) DivideAndConquerMax ( $A[1..n]$ )

1. if  $n=1$  then return  $A[1]$  else
2.  $a \leftarrow \text{DivideAndConquerMax}(A[1..(n/2)])$
3.  $b \leftarrow \text{DivideAndConquerMax}(A[(n/2)+1..n])$
4. if  $a > b$  then return  $a$  else return  $b$ .

Sve asinhaptotičke notacije izvesti bez koritenja master teorema.

(a) Funny

$$\begin{aligned} T(n) &= C_1 \cdot (n-1) \cdot (n-1) = \\ &= C_1 \cdot (n^2 - n - n + 1) = C_1 \cdot (n^2 - 2n + 1) = C_1 \cdot n^2 - 2n \cdot C_1 + C_1 = \\ &= \underline{\underline{O(n^2)}} \end{aligned}$$

$$T(n) = C_1 \cdot (n-1) + T(n-1)$$

$$T(n-1) = C_2 \cdot (n-2) + T(n-2)$$

:

$$T(n-k) = C_2 \cdot (n-k) + T(n-k)$$

$$\sum_{i=0}^{n-1} C \cdot (n-i) = Cn \cdot (n-1) - C \cdot \frac{(n-1)(n-2)}{2}$$

$$\approx O(n^2)$$

7

12

4 9 3 10 6 5 1 8 2 11

PIVOT

12 > 11, vršimo zamjenu, no  
kako je 11 veće od pivota  
pomjeramo i (tj. i--) sve  
do prvog elementa koji je  
manji od elementa na  
poziciji j i manji ili  
jednak od pivota

PIVOT

12

4 9 3 10 6 5 1 8 2 11

swap(12,2)

7

2

j

4

9

3

10

6

5

1

8

12

i

11

$i < 12, j++$

2

4

9

3

10

6

5

1

8

7

12

11

mjesto ispred j  
zadnje ubacujemo  
pivot

### ANALIZA JEDŽENOSTI ALGORITAMA (15 bodova)

Analizirati vrijeme izvršavanja sljedećih algoritama za niz A veličine n:

```
FUNCTION STAN-SORT(A, low, high)
1. if (low = high) then
2.   return
3. end if
4. if (low+1=high) then
5.   if (A[high]< A[low]) then
6.     swap A[low] AND A[high]
7.   end if
8.   return
9. end if
10. t1 = low + high-low+1
11. t2 = low + 2 · high-low+1
12. STAN-SORT(A, low, t2)
13. STAN-SORT(A, t2, high)
14. STAN-SORT(A, low, t1)
```

$$\left. \begin{array}{l} \\ \\ \end{array} \right\} 3 \cdot T\left(\frac{2n}{3}\right)$$

Neka je  $T(n)$  vremenska složenost STAN-SORT algoritma.

$$T(n) = 3 \cdot T\left(\frac{2n}{3}\right) + c$$

$$T(n) \approx \Theta(n^{2/3})$$

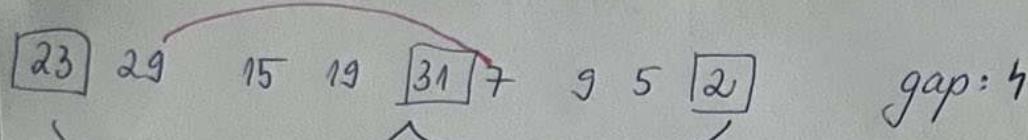
2 5 9 7 15 19 23 29 31

↑  
nije sortiran niz, ~~ati A~~

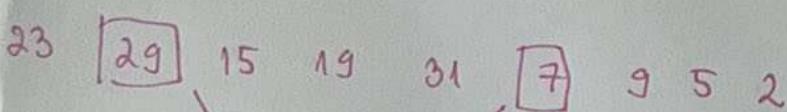
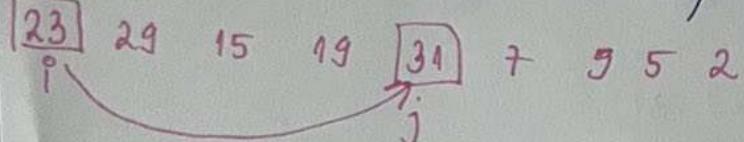
gap = 1 → poređimo redom.

2 5 7 9 15 19 23 29 31

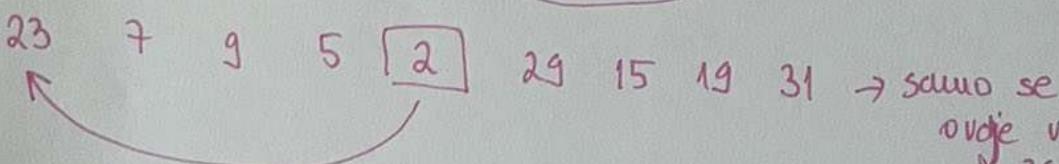
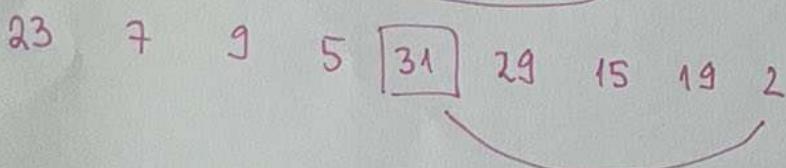
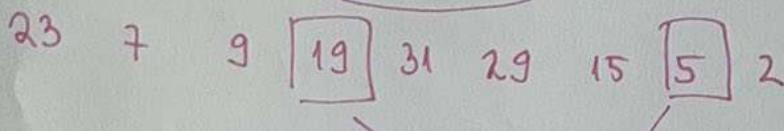
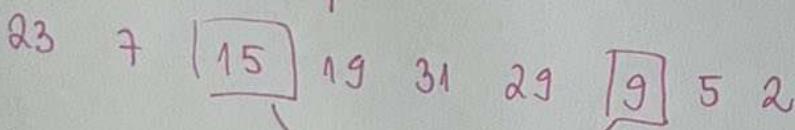
## SHELL SORT



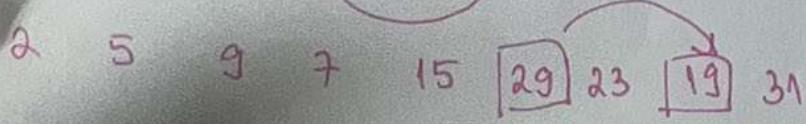
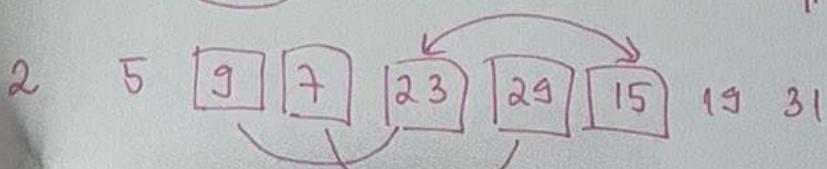
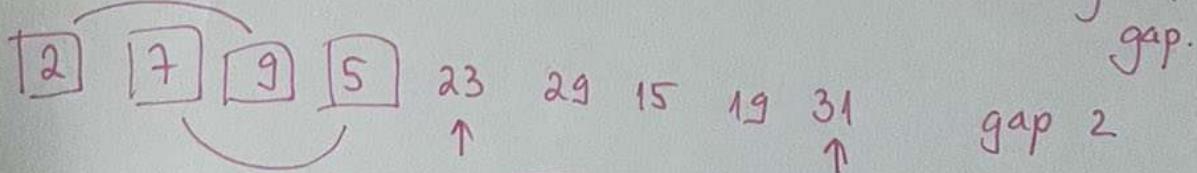
ove naredimo poređinje



swap



gap.



$$b) \lg n! \leq n \log n$$

$$\underbrace{\log 1 + \log 2 + \dots + \log n}_{x} \leq n \log n$$

$$x = \log 1 + \log 2 + \dots + \log n$$

$$\frac{n}{2} \log \frac{n}{2} \leq x \leq n \log n$$

$$n \log n \leq n \log n$$

$$c) \sum_{i=1}^n i \lg i \in O(n^2 \log n)$$

$$\underbrace{\lg 1 + 2 \lg 2 + 3 \lg 3 + \dots + n \lg n}_{x} \leq n^2 \log n$$

$$x = \lg 1 + 2 \lg 2 + \dots + n \lg n$$

$$\frac{n \lg n}{2} \lg \frac{n \lg n}{2} \leq x \leq n \lg n \cdot \lg(n \lg n) =$$

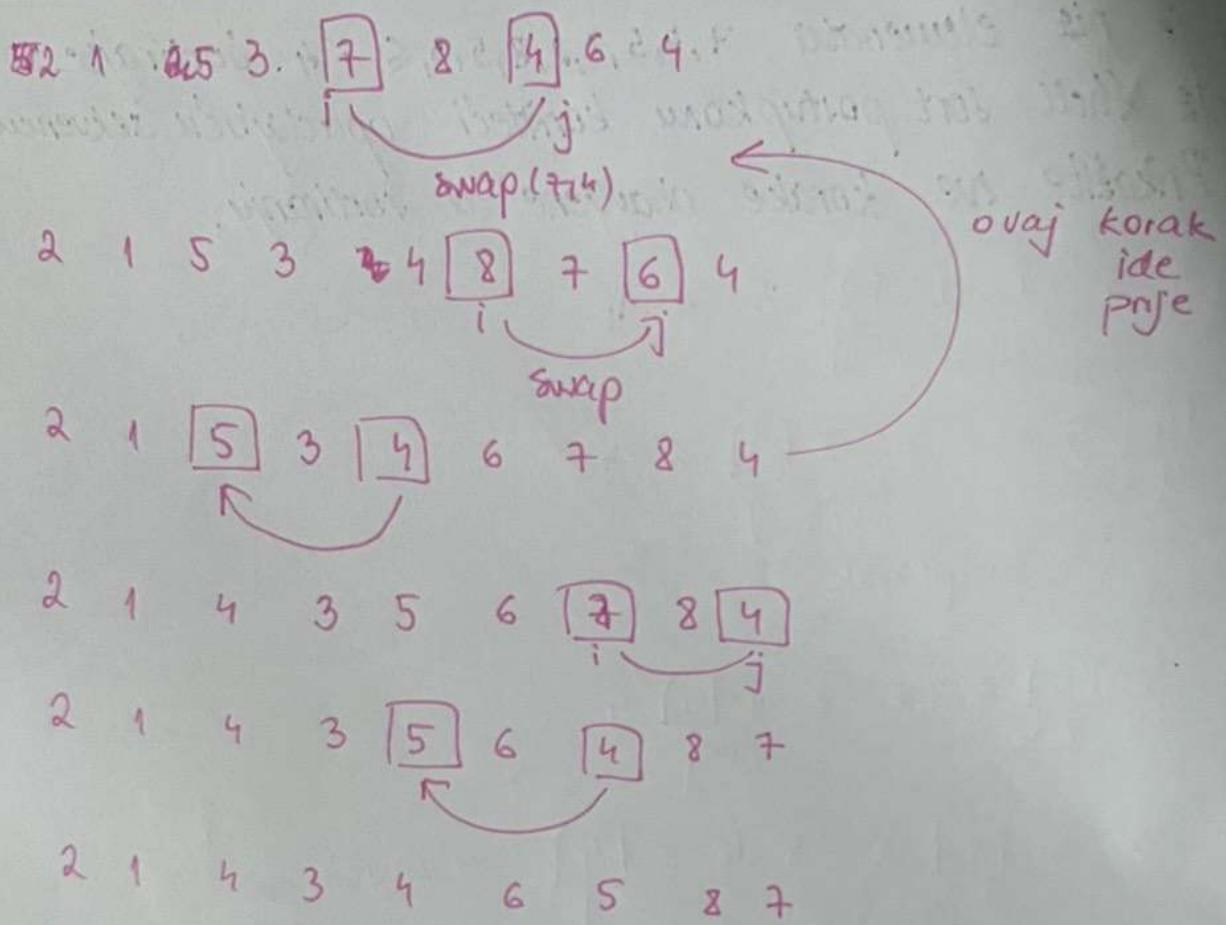
$$= n \lg n (\lg n + \lg(\lg n)) =$$

$$= n \lg^2 n + n \lg n \cdot \lg \lg n \leq n^2 \log n$$

vrijedi

d) Za algoritme sortiranja: selection sort, bubble sort, shell sort, quick sort u dvadesetoj tabeli unijeti informacije da li su sortiraju na mjestu; da li su stabilni; koje je najbolje vrijeme izvršavanja algoritma; koje je prosječno vrijeme izvršavanja algoritma; koje je najgorje vrijeme izvršavanja algoritma, pri čemu ~~vrijeme~~ vrijeme izvršavanja algoritma izražati koritenjem O notacije.

	IN PLACE	STABILAN	NAJGORE VRIJEME	PROSJEČNO VR.	NAJBOLJE VR.
SELECTION	Da	Ne	$O(n^2)$	$O(n^2)$	$O(n^2)$
INSERTION	Da	Da	$O(n^2)$	$O(n^2)$	$O(n)$
BUBBLE	Da	Da	$O(n^2)$	$O(n^2)$	$O(n^2)$
SHELL	Da	Ne	$O(n^2)$	$O(n \log n)$	$O(n \log n)$
MERGE	Ne	Da	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$
QUICK	Da	Ne	$O(n^2)$	$O(n \log n)$	$O(n \log n)$



↳ sekvenca 1

1 → 2 → 3 → 4 → 5 → 6 → 7 →  
 1 2 3 4 5 6 7  
 1 2 3 4 5 6 7  
 1 2 3 4 5 6 7  
 1 2 3 4 5 6 7

→ kraj, sortiran niz.

a) Formulirati i dokazati Master teoremu.

Teorem: Neka je  $T(n)$  dato rekurzivnu relaciju

$$T(n) = aT(n/b) + f(n),$$

gdje je  $a \geq 1, b > 1$  i  $f(n)$  je asimptotski pozitivna funkcija.

- 1) Ako je  $f(n) = O(n^{\log_b a - \varepsilon})$  za  $\varepsilon > 0$ , tada je  $T(n) = \Theta(n^{\log_b a})$ .
- 2) Ako je  $f(n) = \Omega(n^{\log_b a} \lg^k n)$ ,  $k \geq 0$  tada je  $T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$ .
- 3) Ako je  $f(n) = \Omega(n^{\log_b a + \varepsilon})$  za  $\varepsilon > 0$  i ako je  $af(n/b) \leq cf(n)$  za neku konstantu  $c < 1$ , dorozno veliko  $n$ , tada je  $T(n) = \Theta(f(n))$ .

Dokaz:

$$\begin{aligned} 1) g(n) &= \sum_{j=0}^{\log_b n - 1} a^j f(n/b^j) = O\left(\sum_{j=0}^{\log_b n - 1} q^j \left(\frac{n}{b^j}\right)^{\log_b a - \varepsilon}\right) \\ &\sum_{j=0}^{\log_b n - 1} q^j \left(\frac{n}{b^j}\right)^{\log_b a - \varepsilon} = n^{\log_b a - 1} \left(\frac{ab^\varepsilon}{b^{\log_b a}}\right)^j = \\ &= n^{\log_b a - \varepsilon} \sum_{j=0}^{\log_b n - 1} (b^\varepsilon)^j = \\ &= n^{\log_b a - \varepsilon} \left(\frac{b^{\varepsilon \log_b n} - 1}{b^\varepsilon - 1}\right) = \\ &= n^{\log_b a - \varepsilon} \left(\frac{n^\varepsilon - 1}{b^\varepsilon - 1}\right) \end{aligned}$$

$$\begin{aligned} 2) g(n) &= \sum_{j=0}^{\log_b n - 1} a^j f(n/b^j) = \Theta\left(\sum_{j=0}^{\log_b n - 1} a^j \left(\frac{n}{b^j}\right)^{\log_b a}\right) \\ &\sum_{j=0}^{\log_b n - 1} a^j \left(\frac{n}{b^j}\right)^{\log_b a} = n^{\log_b a} \sum_{j=0}^{\log_b n - 1} \left(\frac{a}{b^{\log_b a}}\right)^j = \\ &= n^{\log_b a} \sum_{j=0}^{\log_b n - 1} 1 = n^{\log_b a} \log_b n. \end{aligned}$$

$$3) f(n) = \Omega(n^{\log_b a + \epsilon})$$

$$g(n) = \sum_{j=0}^{\log_b n - 1} a^j f(n/b^j)$$

$$\leq \sum_{j=0}^{\log_b n - 1} c^j f(n)$$

$$\leq f(n) \sum_{j=0}^{\log_b n - 1} c^j$$

$$= f(n) \left( \frac{1}{1-c} \right)$$

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