ALGORITMI I STRUKTURE PODATAKA

STUDIJSKI PROGRAMI:

SOFTVERSKO INŽENJERSTVO, RAČUNARSKA TEHNIKA, INFORMATIKA I MATEMATIKA

NASTAVNIK: DOC. DR ULFETA MAROVAC, UMAROVAC@NP.AC.RS

OBAVEŠTENJE

- Posle svake nastavne jedinice obavezno je popuniti test provere znanja
- Odgovore na test slati na e-mail umarovac@np.ac.rs sa naznakom

  Algoritmi I strukture podataka u word dokumentu pod nazivom

  Ime_prezime_test_ASP_br_testa.doc
- Test pitanja se nalaze na poslednjem slajdu
- Vaše prisustvo i aktivno ucečće će se evidentirati na osnovu urađenog testa
- Dodatni material se nalazi na http://moodle.np.ac.rs/ na
- Departmanu za tehničke nauke –Softversko inženjerstvo-**Algoritmi i strukture podataka**

Za sve dodatne informacije pišite na e-mail umarovac@np.ac.rs

SORTIRANJE

- Sortiranje se može definisati kao proces preuređivanja skupa podataka po nekom utvrdjenom poretku.
- Jedna od osnovnih svrha sortiranja je omogućavanje efikasnijeg pretraživanja.
- Sortiranje je korisno ako treba proveriti jednakost dva skupa podataka.

➢ Cilj:
- ✓ efikasnije pretraživanje
- ✓ provera jednakosti
- ✓ sistematizovani prikaz

ALGORITMI ZA SORTIRANJE

- Zbog velike važnosti, razvijen je čitav spektar metoda različite efikasnosti. Posebna pažnja se posvećuje vremenskoj složenosti koja se kreće u relativno širokom opsegu, od O(n) do O(*n*2).
- Izbor algoritma sortiranja uzima u obzir i prostornu složenost jer neki algoritmi zahtevaju veći ili manji dodatni prostor da bi se sortiranje obavilo.

- Poredak (↗,↘)određen vrednostima polja ključa

SORTIRANJE

- U skupu ključeva koji se sortira nije nužno da sve vrednosti budu različite. Za metod sortiranja se kaže da je stabilan ako početni poredak zapisa sa istom vrednošću ključa ostane očuvan u sortirnom nizu.
- Sortiranje:
  - ✓ $K_1, ..., K_n \Rightarrow K_{p1} \leq ... \leq K_{pn}$
  - ✓ $p1, ..., pn$ – permutacija od $1..n$

➢ Stabilnost – $i < j$ i $K_{pi} = K_{pj} \Rightarrow pi < pj$

PODELA SORTIRANJA

- Po mestu sortiranja gde se nalaze podaci koji se uređuju postoje dve grupe metoda:

❖ **Unutrašnje sortiranje** – se koristi za podatke koji mogu da stanu u operativnu memoriju, obicno u formi niza ili tabele

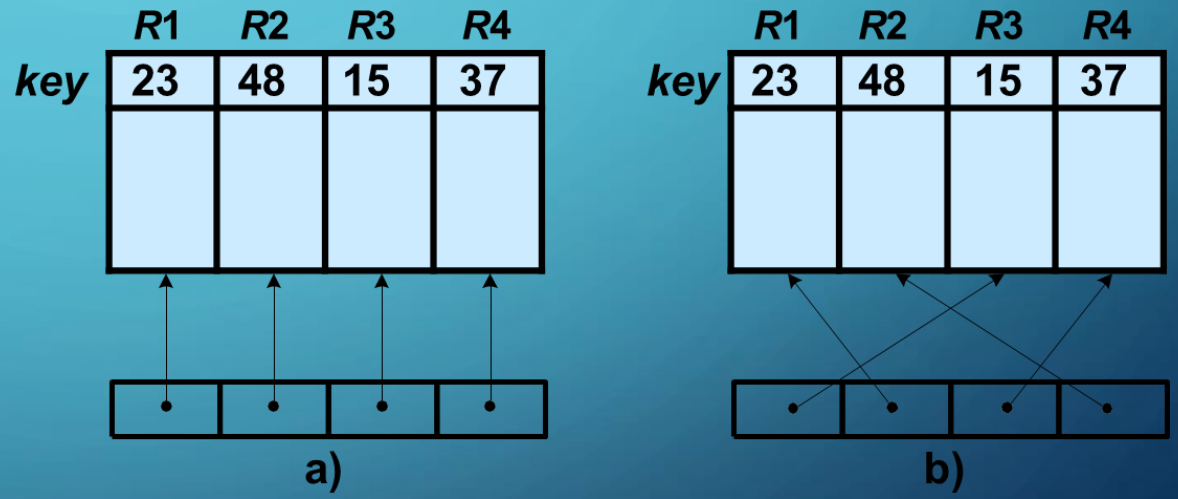
❖ **Spoljašnje sortiranje** – uređuje velike skupove podataka koji se nalaze na spoljašnjim memorijama organizovane u datoteke

UNUTRAŠNJE SORTIRANJE

- Sortiranje kada su svi podaci u operativnoj memoriji
- Sortiranje nizova i lista
- Sortiranje in situ
- Indikatori performanse
  - broj koraka
  - broj poređenja ključeva
  - broj premeštanja ključeva

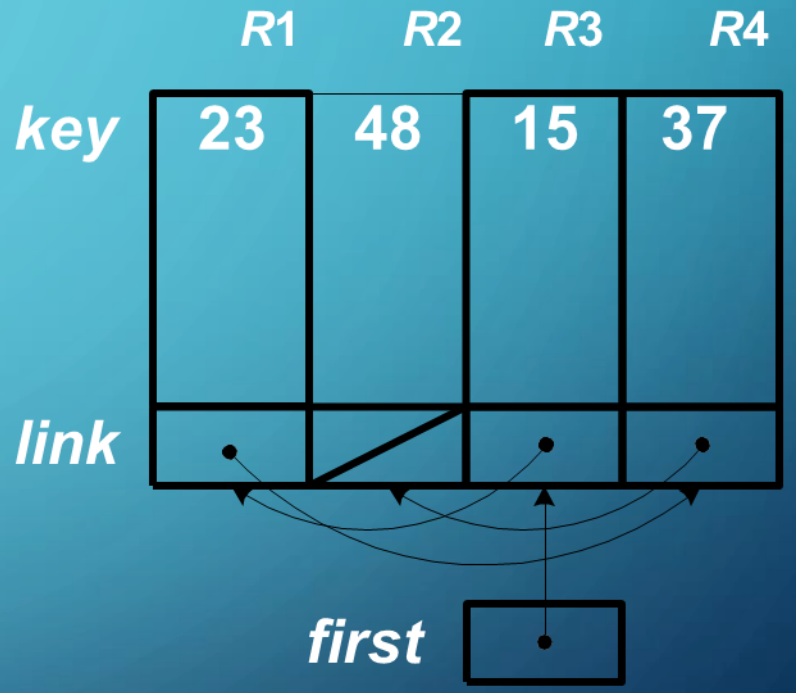
UNUTRAŠNJE SORTIRANJE

- Sortiranje po adresi
- Izbegava premeštanje zapisa



UNUTRAŠNJE SORTIRANJE

- Ulančavanje zapisa po poretku
- Izbegava premeštanje zapisa



SORTIRANJE POREĐENJEM

- Isključivo zasnovano na poređenju ključeva
- Podela:
  - ✓ metodi umetanja
  - ✓ metodi selekcije
  - ✓ metodi zamene
  - ✓ metodi spajanja
- Direktni metodi
  - ✓ Jednostavni
  - ✓ Lošije perfomanse

- **Pokazuje se da su najbolje performanse kod ovih metoda u srednjem i najgorem slučaju ograničene na $O(n \log n)$.**

METODI UMETANJA

- Ova grupa metoda se zasniva na principu postepenog uređivanja niza, tako što se u svakom trenutku održava uređeni i neuređeni deo.
- Princip – po jedan element iz neuređenog dela umeće se u uređeni deo
- Predstavnici ove grupe metoda su: **direktno umetanje i umetanje sa smanjenjem inkrementa.**

DIREKTNO UMETANJE

```
INSERTION-SORT(a)
for i =2 to n do
        K = a[i]
        j = i - 1
       while (j >0) and (a [j]) >K) do
              a [j +1] =a[j]
              j =j -1
       end_while
  a [j +1] =K
end_for
```

- U početku se uređeni deo sastoji samo od prvog elementa niza, a u neuređeni deo spadaju svi ostali elementi (n-1).
- U svakoj iteraciji u okviru unutrašnje petlje uzima se tekući element kao prvi element iz neuređenog dela, pa se upoređuje redom sa elementima uređenog dela, prvo sa poslednjim, pa sa pretposlednjim, itd., sve dok se ne naiđe na prvi element u uređenom delu koji nije veći od tekućeg elementa.
- Pritom se elementi uređenog dela koji su veći od tekućeg elementa pomeraju za po jedno mesto naviše i tako prave mesto za njegovo umetanje.
- Sortiranje se završava kad nema više elemenata u neuređenom delu.

DIREKTNO UMETANJE

DIREKTNO UMETANJE

- Najbolji slučaj – uređen niz, $C_{min} = n - 1 \Rightarrow O(n)$
- Najgori slučaj – obrnuto uređen niz,

$$C_{max} = M_{max} = \Sigma(i - 1) \Rightarrow O(n^2)$$

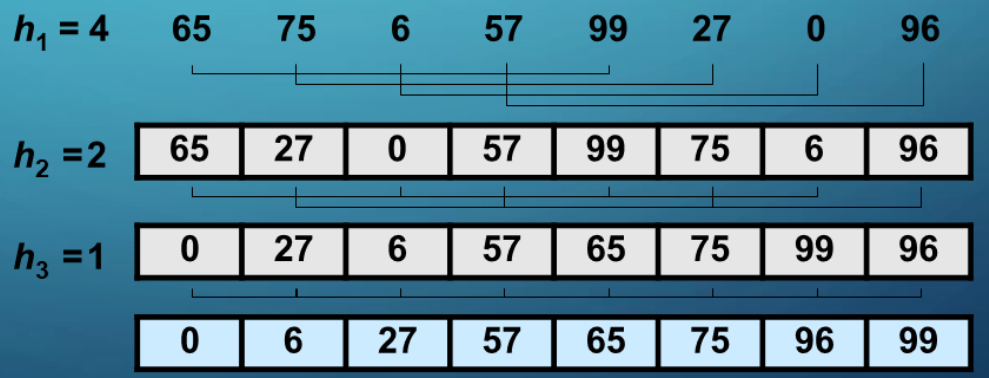
- Prosečan slučaj bolji od najgoreg samo za faktor 1/2
- Vrlo dobar za male nizove i skoro uređene nizove
- **Metod je stabilan** jer kad ključ koji se umeće dođe do jednakog ključa u uređenom delu, on se stavlja neposredno iza njega.

POBOLJŠANJA

- Problemi
  - ✓ broj poređenja
  - ✓ broj premeštanja
- Binarno pretraživanje uređenog dela
  - ✓ smanjuje broj poređenja, ali ne i premeštanja
  - ✓ isti red složenosti
  - ✓ za uređen niz čak i lošije
- Jednostruko ulančana lista umesto niza
  - ✓ vektor indeksa simulira pokazivače
  - ✓ smanjuje broj premeštanja, ali ne i poređenja
  - ✓ dodatni prostor

SHELLSORT

- Umetanje sa smanjenjem inkrementa h
- Grupe elemenata na ekvidistantnom razmaku h
- Grupe se sortiraju metodom direktnog umetanja
- Inkrementi se smanjuju sve do 1



SHELLSORT

```
SHELL-SORT(a, h)
for i =1 to t do
        inc =h[i]
        for j =inc +1 to n do
                y =a[j]
                k =j - inc
                while (k ≥ 1) and (y <a[k]) do
                        a[k +inc] =a[k]
                        k =k - inc
                end_while
                a[k +inc] =y
        end_for
end_for
```

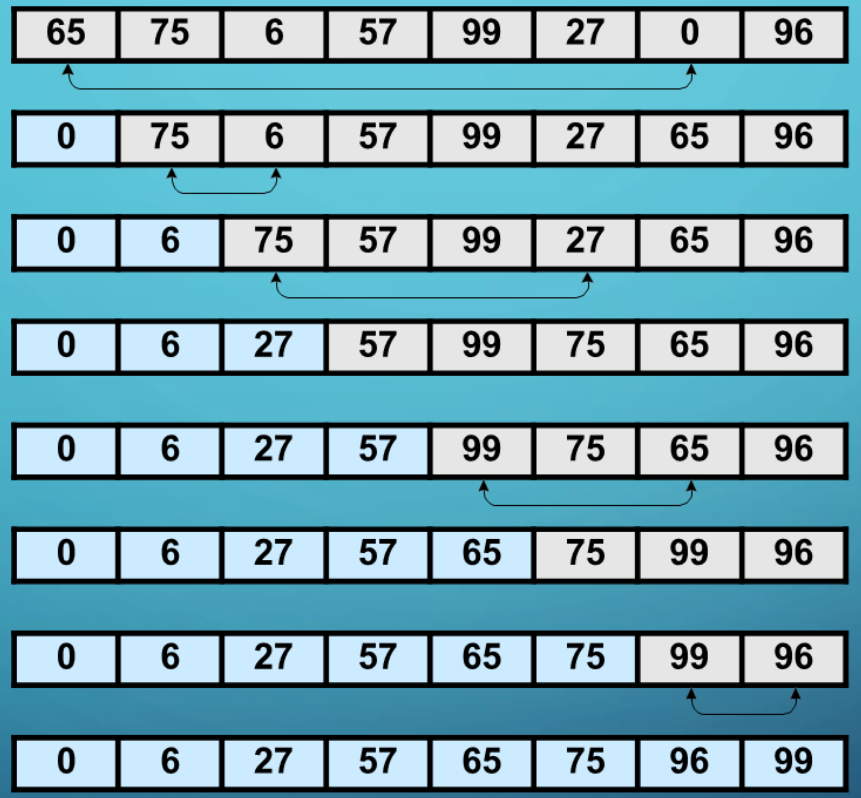
SHELLSORT

- Direktno sortiranje
  - ✓ male nesortirane grupe
  - ✓ veće dosta sortirane grupe
- ✓ Sekvenca inkrementa $h_1, h_2, \ldots, h_t$
  - ✓ $h_{i+1} < h_i, \quad 1 \leq i < t$
  - ✓ $h_t = 1$
- ✓ Knuth – $h_{i-1} = 3h_i + 1$, $h_t = 1$, $t = \log_3 n - 1$
- ✓ Bolji uzajamno prosti inkrementi
- ✓ Složenost $O(n(\log n)^2)$ (empirijski $O(n^{1.3})$)

METODI SELEKCIJE

- Princip – selektuje najmanji element iz neuređenog i stavlja ga na kraj uređenog dela
- Ponekad procesiranje neuređenog dela
  - u strukturu koja olakšava selekciju (prioritetni red)
- Direktna selekcija
- Nema dodatnog procesiranja neuređenog dela
- Sličnosti i razlike sa direktnim umetanjem
- Poređenja u neuređenom delu
  - (ne može da počne dok nema sve podatke)

DIREKTNA SELEKCIJA



DIREKTNA SELEKCIJA

```
SELECTION-SORT(a)
for i =1 to n -1 do
        min =a[i]
        pos =i
        for j =i +1 to n do
                if (a[j] <min) then
                        min =a[j]
                        pos =j
                end_if
        end_for
        a[pos] =a[i]
        a[i] =min
end_for
```

DIREKTNA SELEKCIJA

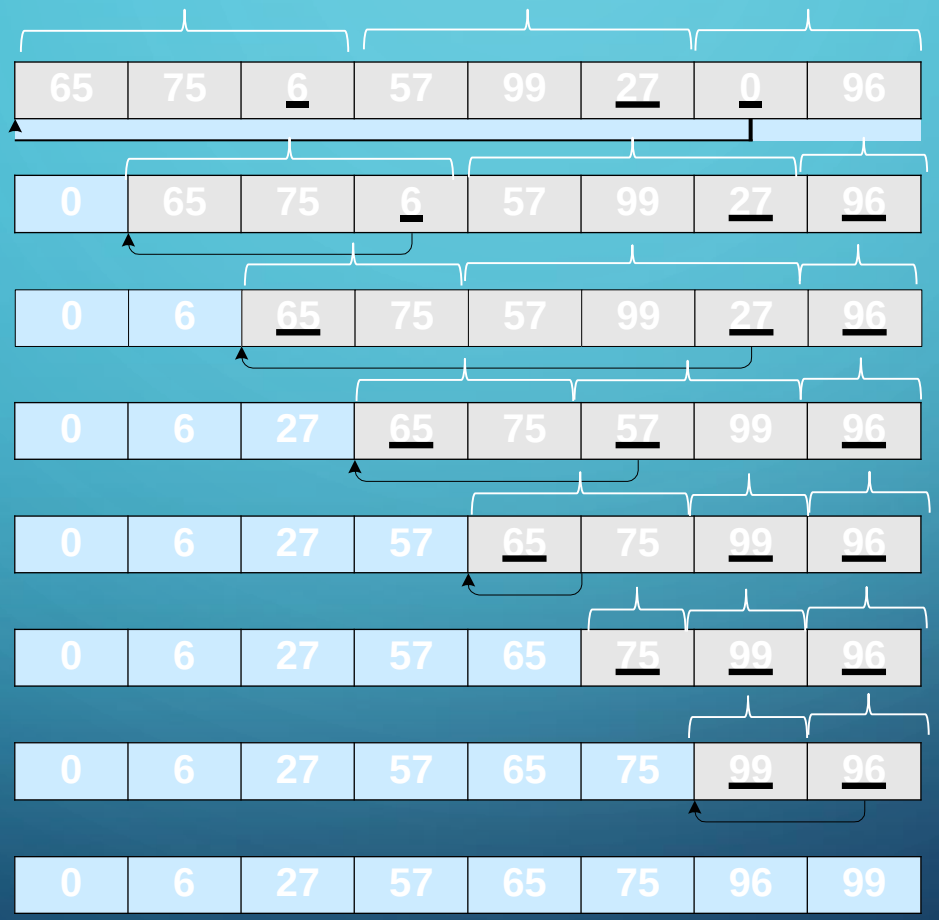
- Jedna zamena i $n - i$ poređenja po koraku
- $C = \Sigma(n - i) \Rightarrow O(n^2)$
- Nema razlike između najboljeg i najgoreg slučaja
- Problem – broj poređenja

Kvadratna selekcija

- Selekcija po grupama – lokalni i globalni minimumi

- Složenost $O(n\sqrt{n})$

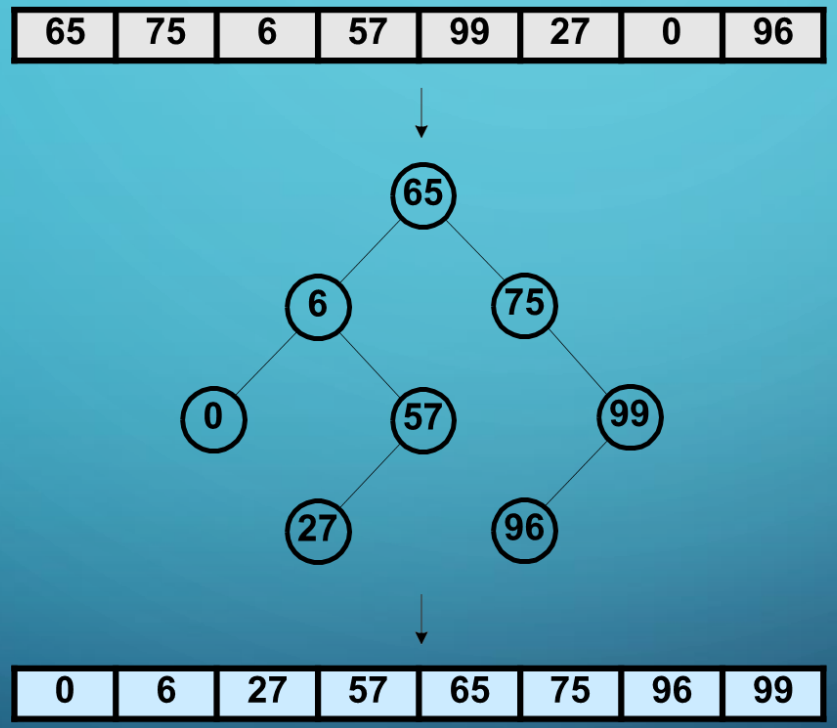
KVADRATNA SELEKCIJA



SORTIRANJE POMOĆU B S T

- Elementi neuređenog niza se umeću u stablo binarnog pretraživanja
- Inorder obilazak
- Isti ključevi:
  - ✓ u desno podstablo
  - ✓ ulančana lista
- Složenost O(n log n), ali nije garantovana
- Pogodan za dinamičke skupove podataka

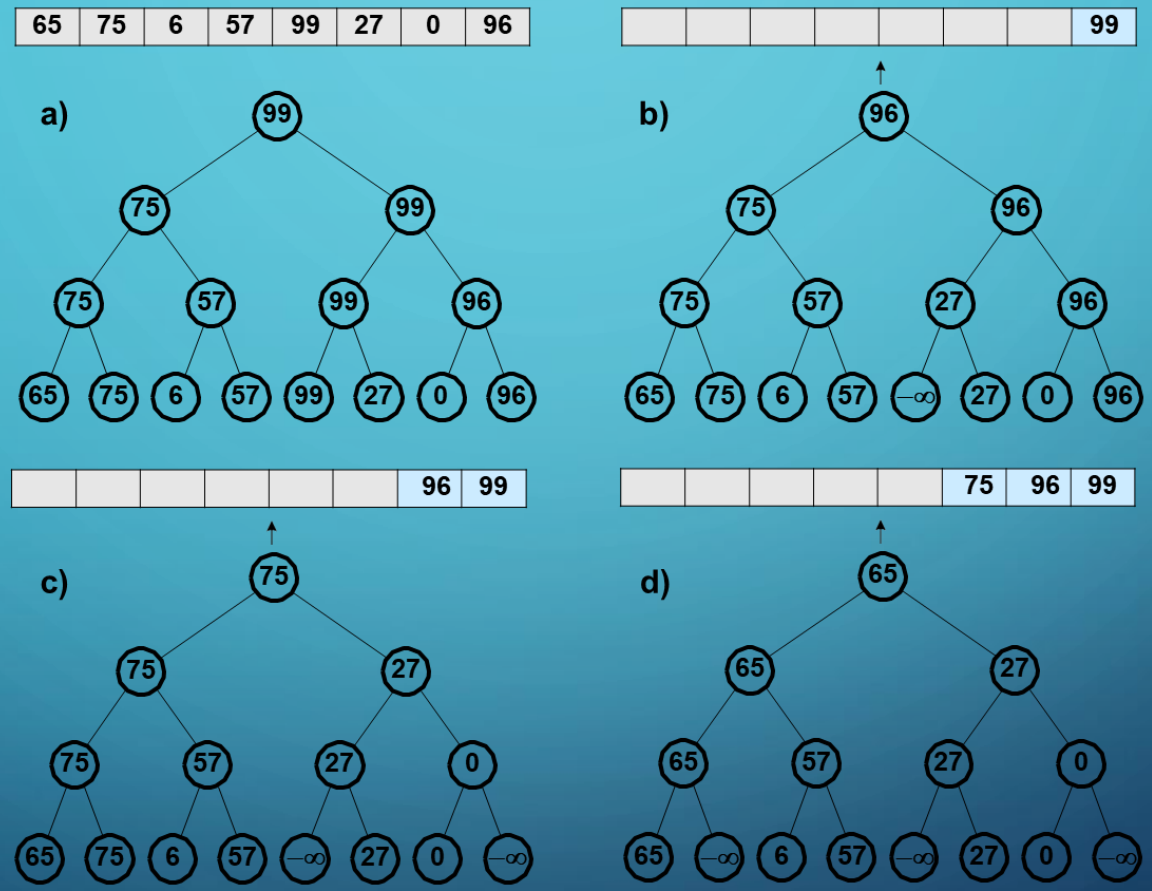
SORTIRANJE POMOĆU B S T



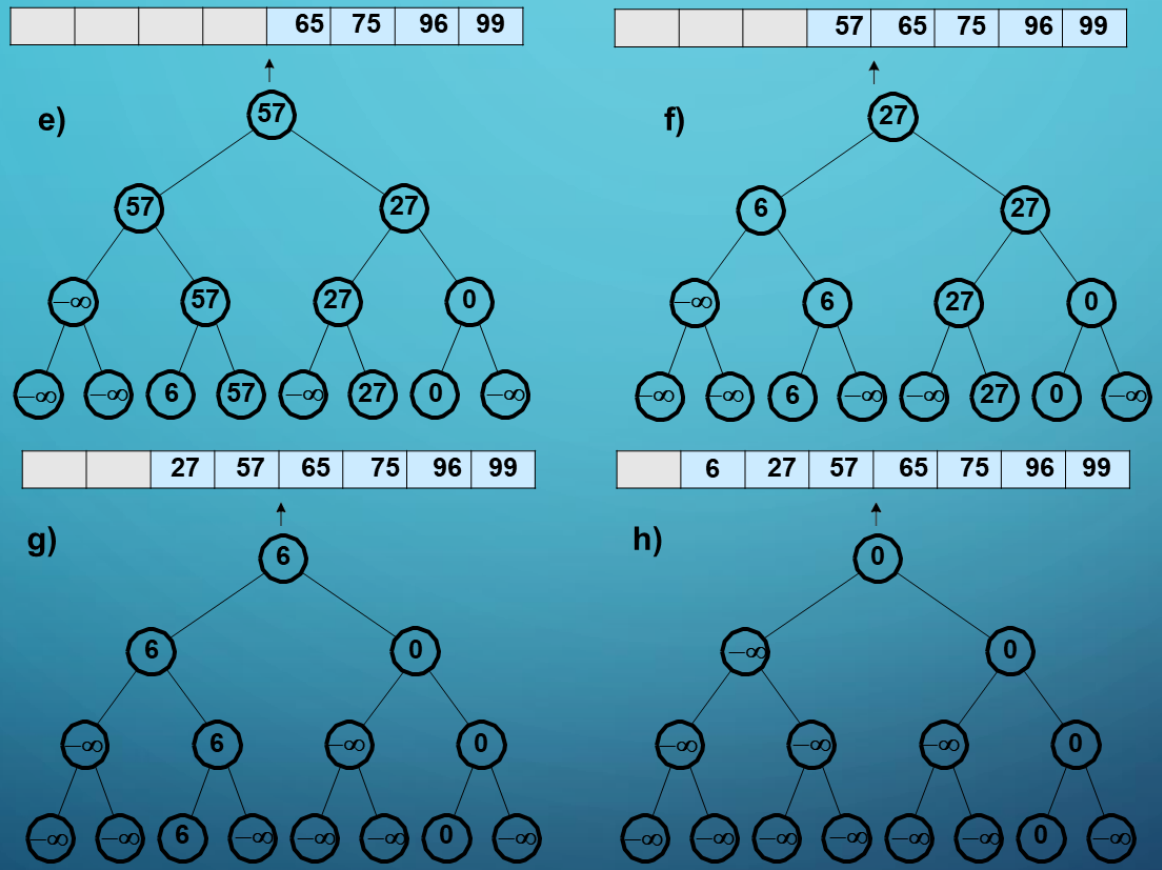
STABLO SELEKCIJE

- Problem garantovane performanse
- Balansirano stablo
- Poređenja po kup–sistemu
- Selekcija iz korena
- Ažuriranje po jednoj putanji od lista do korena
- Performanse
  - ✓ generisanje stabla – O(n)
  - ✓ selekcija – O(n log n)

STABLO SELEKCIJE



STABLO SELEKCIJE



HEAPSORT

- Nedostaci stabla selekcije
  - dodatni prostor
  - nepotrebna poređenja
- Struktura selekcije - *heap*
  - kompletno ili skoro kompletno binarno stablo
  - otac veći ili jednak sa oba sina
- Sekvencijalna implementacija $a\,[1{:}n]$
  - $a\,[i] \geq a\,[2i]$ i $a\,[i] \geq a\,[2i+1]$, $1 \leq i < 2i < 2i+1 \leq n$
- Efikasna implementacija prioritetnog reda
- Sortiranje na mestu

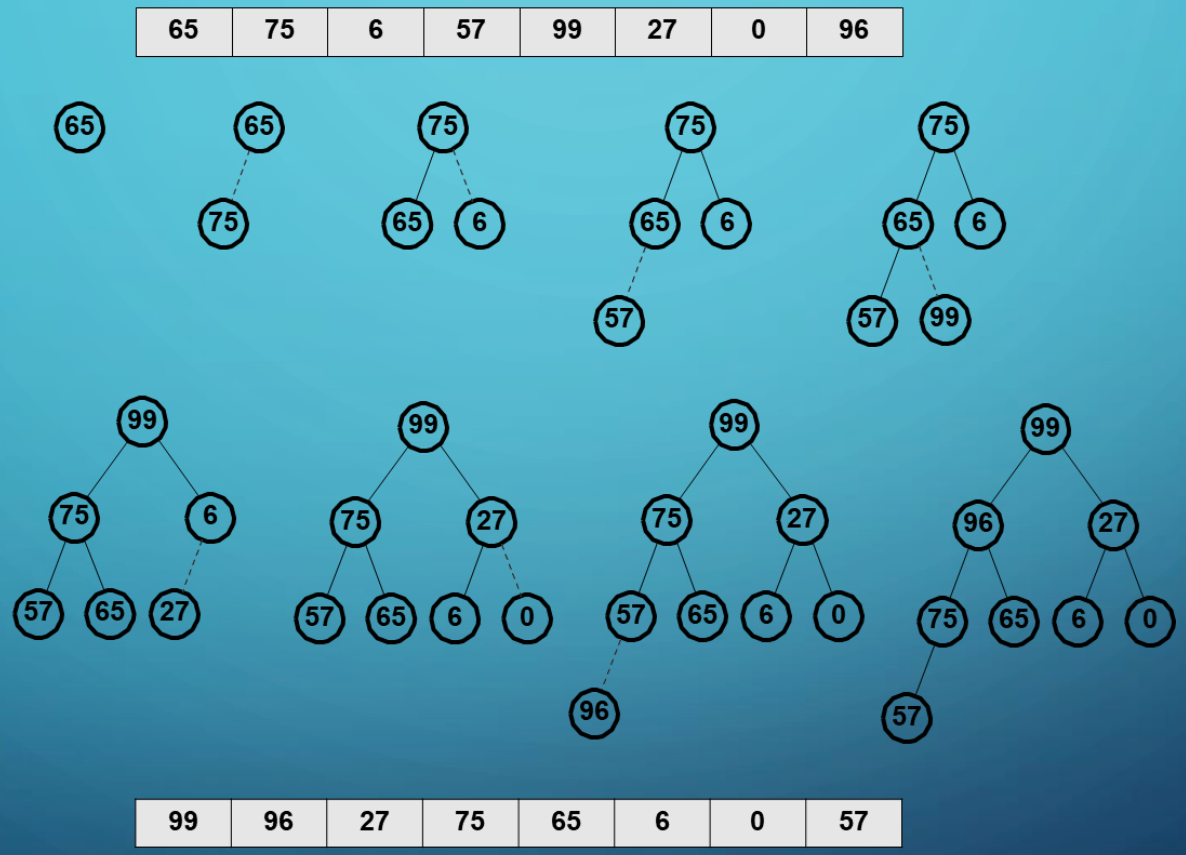
HEAPSORT

- Dve faze algoritma:
  - ✓ generisanje *heap*-a
  - ✓ procesiranje *heap*-a

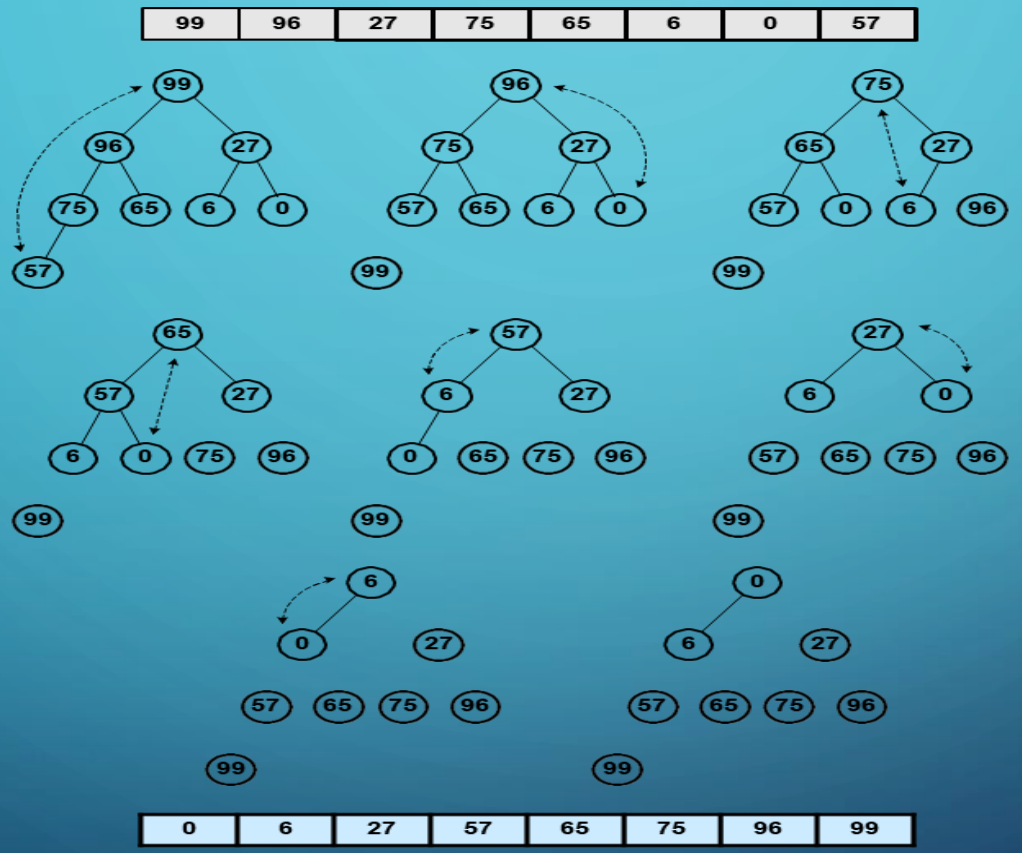
```
HEAPSORT(a)
for i = 2 to n
    do  nhe =
    a[i]  s = i
    f = s/2
    while ((s > 1) and (a[f] < nhe)) do
        a[s]    =
        a[f]  s = f
        f    =
    s/2
    end_whil
    e   a[s]  =
    nhe
end_for
```

```
for i =  n downto 2 do
        last =a[i]
        a[i] = a[1]
        f = 1
        if ((i –1) ≥ 3 and (a[3] > a[2])) then
                s = 3
        else
                s = 2
        end_if
        while (s ≤ i – 1) and (a[s] > last) do
                a[f] = a[s]
                f = s
                s = 2f
                if ((s + 1) ≤ i – 1) and (a[s + 1] > a[s]) then
                        s = s + 1
                end_if
        end_while
        a[f] = last
end_for
```

HEAPSORT



HEAPSORT


99 96 27 75 65 6 0 57
99 96 27 75 65 6 0 57
96 75 27 57 65 6 0
99
75 65 27 57 0 6 96
99
65 57 27 6 0 75 96
99
57 6 27 0 65 75 96
99
27 6 0 57 65 75 96
99
6 0 27 57 65 75 96
99
0 6 27 57 65 75 96
99
0 6 27 57 65 75 96 99

HEAPSORT

- Alternativna realizacija
  - ✓ ADJUST pretvara u *heap* stablo sa korenom *i*
    - kada su mu oba podstabla već heap-ovi
  - ✓ poziva se i pri generisanju i pri procesiranju
- ✓ Performanse
  - ✓ generisanje – O(n log n) (sa ADJUST čak O(n))
  - ✓ procesiranje – O(n log n)
- ✓ Prosečan broj zamena 0.5n log n
- ✓ Garantovan najgori slučaj O(n log n)
- ✓ Za izdvajanje k najvećih ključeva (k << n) – ~O(n)

```
ADJUST(a, i, n)
K =a[i]
j =2i
while(j ≤ n) do
    if((j < n) and (a[j]< a[j+1])) then
        j =j +1
    end_if
    if(K ≥ a[j]) then a[j/2]
        =K  return
    else
        a[j/2]=a[j]
        j =2j
    end_if
end_while
a[j/2]=K
```

```
HEAPSORT-1(a)
for i =n/2 downto 1 do
    ADJUST(a, i, n)
end_for
for i =n -1 downto 1 do
    a[i+1]↔ a[1]
    ADJUST(a, 1, i)
end_for
```

TEST PITANJA

1. Čemu služi sortiranje? Kako se postiže stabilnost algoritma?
2. Kako se može podeliti sortiranje?
3. Šta predstavlja sortiranje po adresi?
4. Koji je princip sortiranja metodom umetanja?
5. Dajte primer niza i sortirajte ga metodom direktnog umetanja
6. Prethodni niz sortirajte shellsort algoritmom
7. Koji je princip metoda selekcije?
8. Sortirajte prethodni niz direktnom selekcijom
9. Sortirajte niz pomocu stable selekcije.
10. Izvršite sortiranje heapsort-om.