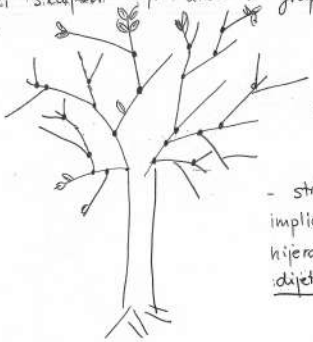
STABLA

- dinamički skupovi koji ulaze u grupu nelinearnih struktura podataka

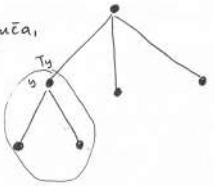


- stablo je povezan aciklički graf

- struktura podataka koja implicitno podrazumijeva hijerarhijski odnos roditelj-dijete

- svojstva stabla: T

  - svaki čvor x sadrži ključ key - x.key
  - postoji specijalan element zvan korijen stabla - T.root
  - svaki nekorijenski čvor predstavlja novi korijenski čvor za podstablo T
  - svaki čvor x, osim ključa, sadrži pokazivače na roditelja i na djecu



  - korijen je element koji nema roditelja
  - list je element koji nema djece

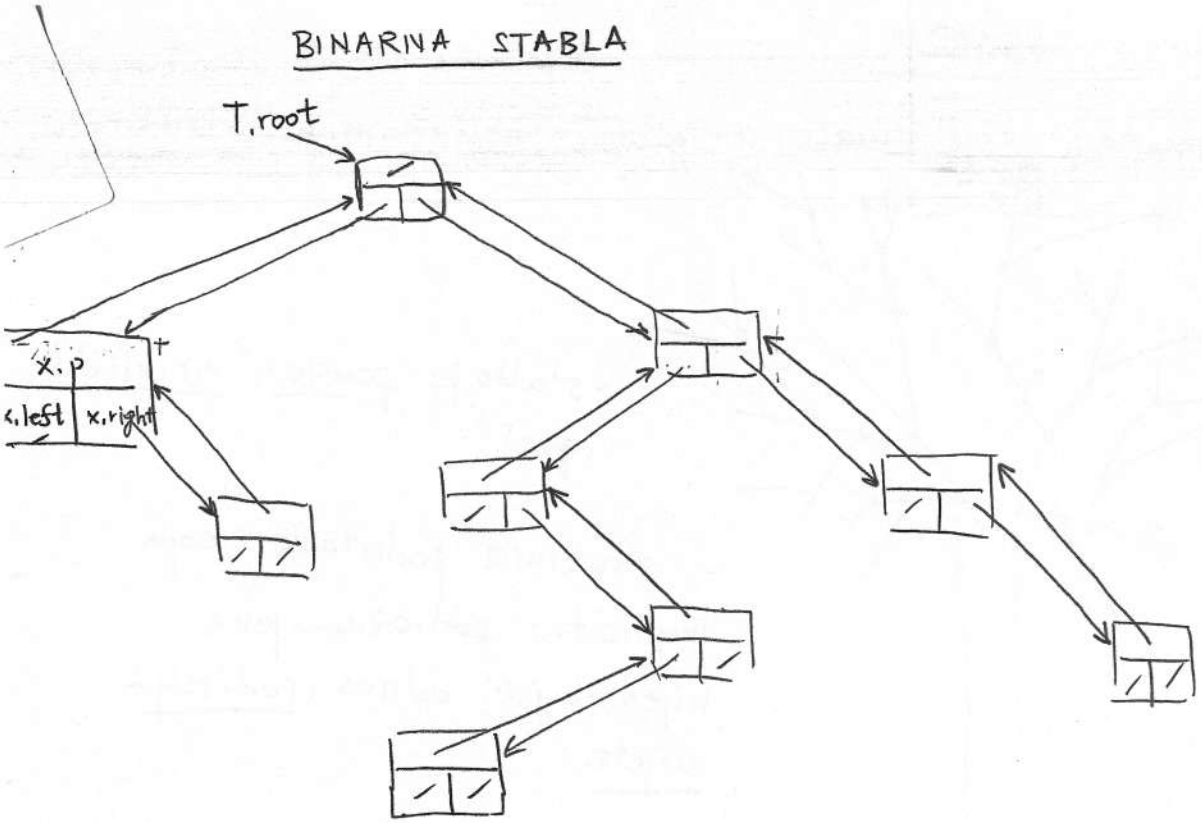
stupanj čvora = broj podstabala pri čvoru u danom stablu

stupanj stabla = maksimalan stupanj čvora u stablu

stablo je strukturirano po nivoima

dubina/visina stabla je maksimalan broj nivoa u stablu

BINARNA STABLA



stabla je 2

no stablo T je povezana struktura podataka koja elemente i pokazivač na korijen T.root

element x stabla T sadrži:

- x.p - pokazivač na roditelja
- x.left - pokazivaĭ na lijevo dijete od x
- x.right - pokazivaĭ na desno dijete od x

x.p = NIL onda je x korijen stabla

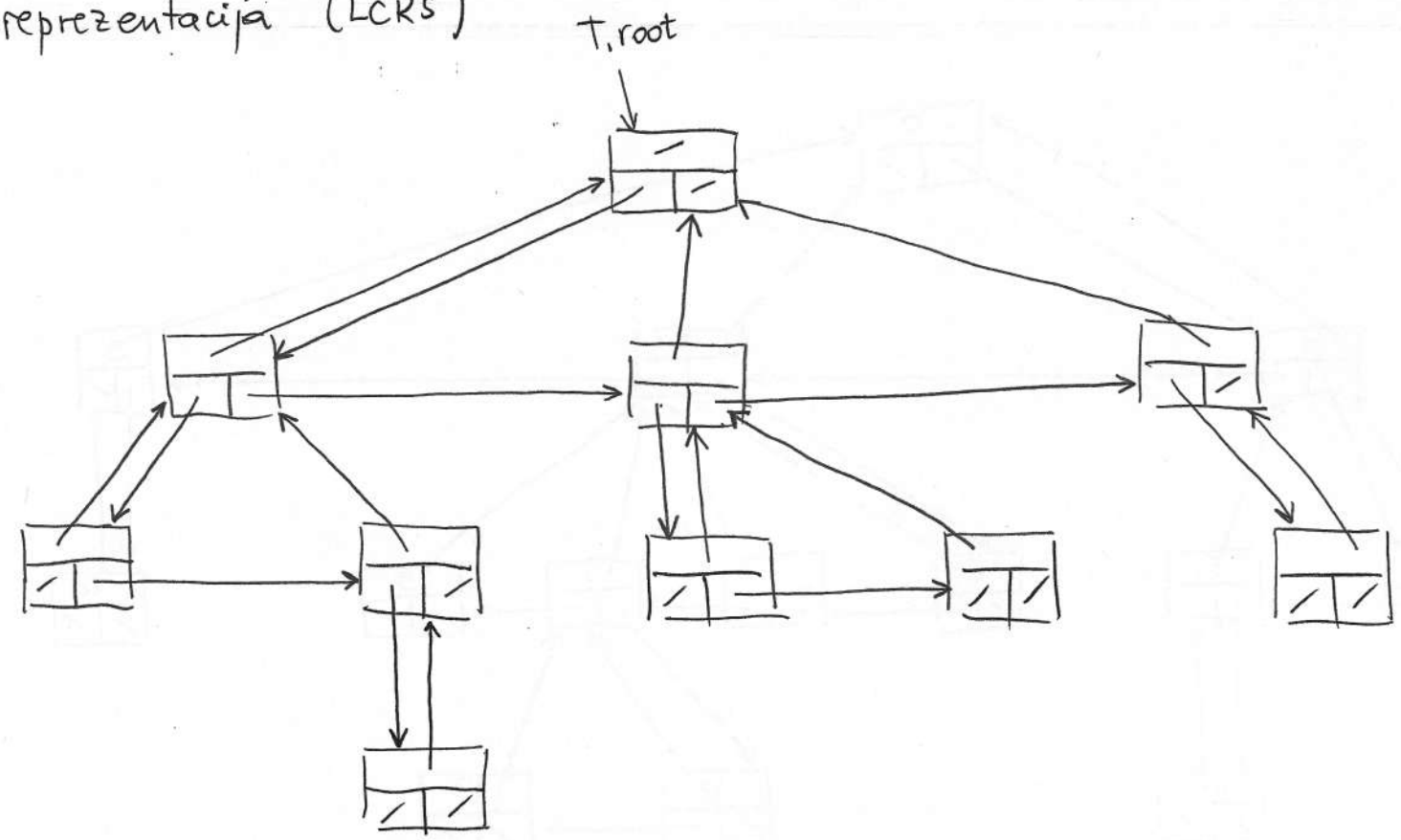
nema lijevo (desno) dijete onda je x.left = NIL

t = NIL)

T.root = NIL onda je stablo T prazno

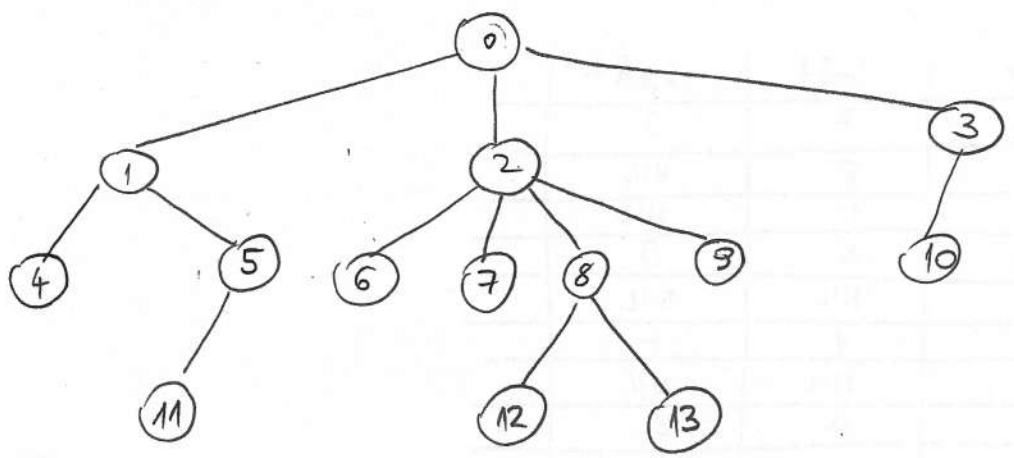
STABLA NEOGRANIČENOG STUPNJA

- LEFT-CHILD-RIGHT-SIBLING (Lijevo-dijete-desni-brat) reprezentacija (LCRS)

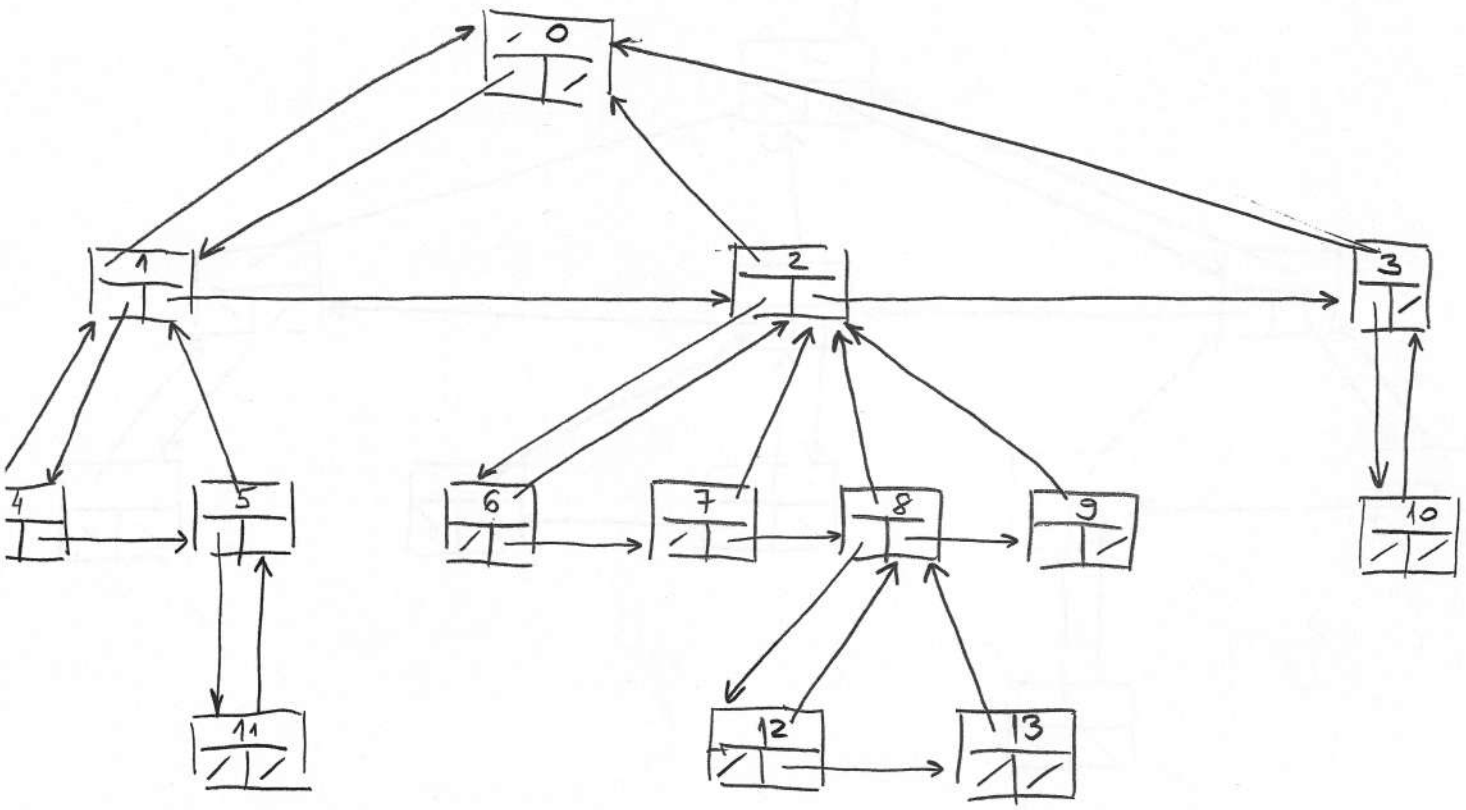


- stupanj stabla nije ograničen
- stablo T sadrži pokazivač na korijen T.root
  - x.p - pokazivač na roditelja
  - x.left-child - pokazivač na lijevo dijete od x
  - x.right-sibling - pokazivač na desnog brata od x

Primjer: Dano je stablo T na sljedećoj slici:



Utvrdite x.p, x.left-child, x.right-sibling, dubinu/visinu stabla, listove gdje je x.key = 6 i prikažite ga u LCRS reprezentaciji.



key = 6

left-child = NIL

right-sibling = 7

p = 2

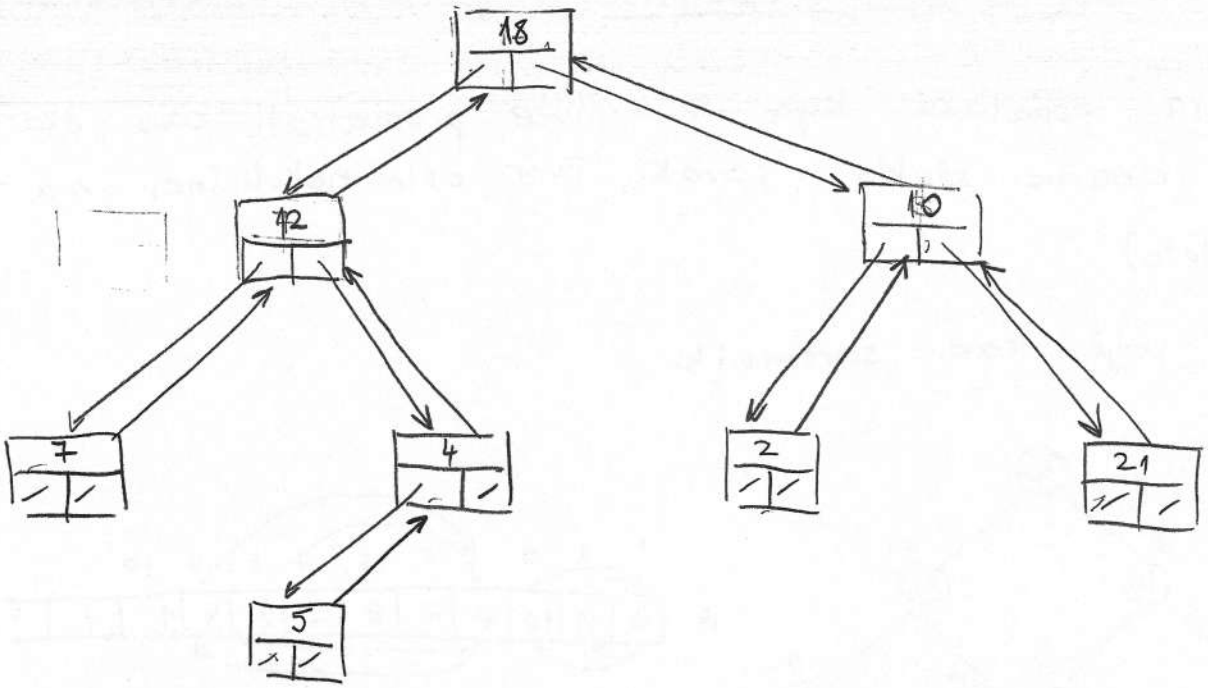
listovi: 4, 11, 6, 7, 12, 13, 9, 10

dubina/visina: 3

Primjer:

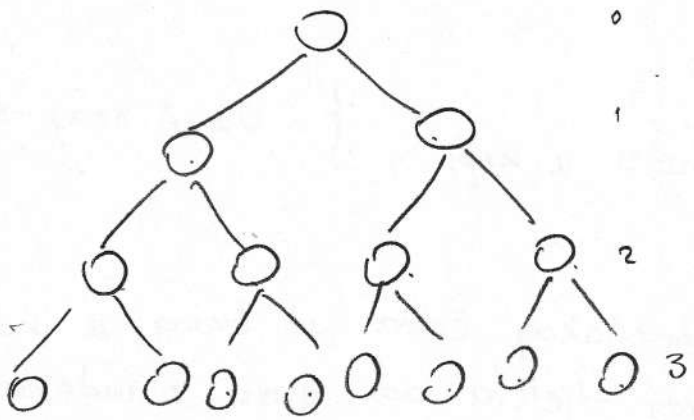
Nacrtajte binarno stablo s korijenom indeksa 6, zadano sljedećom tablicom

index	key	left	right
1	12	7	3
2	15	8	NIL
3	4	10	NIL
4	10	5	9
5	2	NIL	NIL
6	6	1	4
7	7	NIL	NIL
8	14	6	2
9	21	NIL	NIL
10	5	NIL	NIL



STABLA BINARNOG PRETRAŽIVANJA

- Vrijeme izvršavanja svih osnovnih operacija proporcionalna je visini stabla (u slučaju potpunih binarnih stabala $\Theta(\lg n)$)



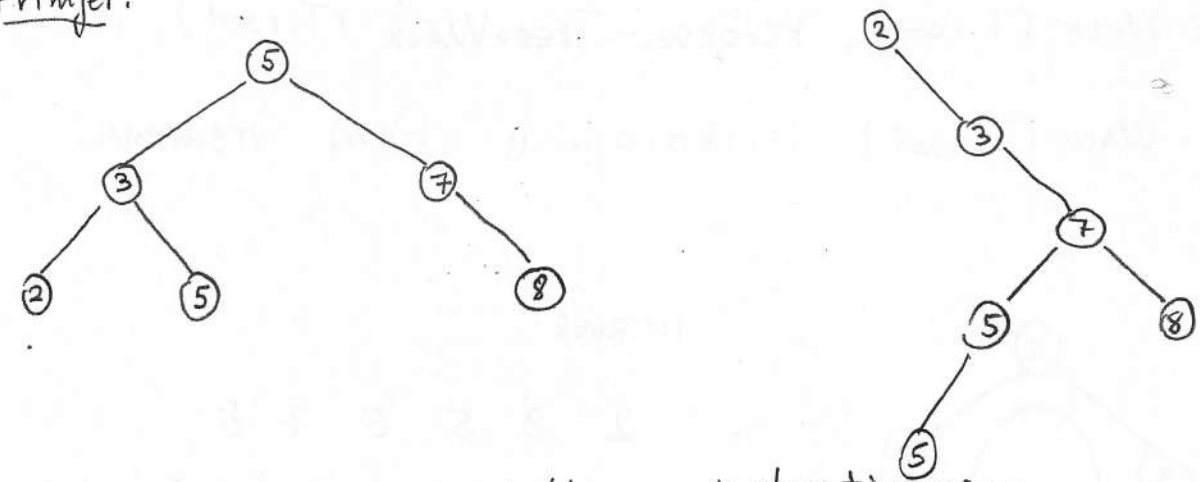
$\Theta(\lfloor \lg n \rfloor)$

BINARNA STABLA PRETRAŽIVANJA

- binarna stabla koja imaju sljedeće svojstvo:

Neka je x čvor u binarnom stablu T. Ako je y čvor sadržan u lijevom podstablu od x, onda je $y.key \leq x.key$. Ako je y čvor sadržan u desnom podstablu od x, onda je $x.key \leq y.key$.

Primjer:



- obilasci binarnog stabla pretraživanja
  - * INORDER - lijevo podstablo + korijen + desno podstablo
  - * PREORDER - korijen + lijevo podstablo + desno podstablo
  - * POSTORDER - lijevo podstablo + desno podstablo + korijen

```
INORDER-TREE-WALK (x)
if   x ≠ NIL
   then   INORDER-TREE-WALK (x.left)
          print x.key
          INORDER-TREE-WALK (x.right)
```

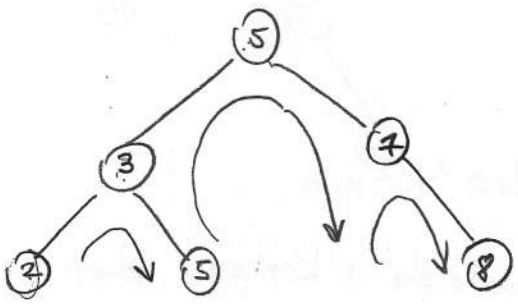
```
PREORDER-TREE-WALK (x)
if   x ≠ NIL
   then   print x.key
          PREORDER-TREE-WALK (x.left)
          PREORDER-TREE-WALK (x.right)
```

```
POST ORDER-TREE-WALK (x)
  if   x≠NIL
    then  POSTORDER-TREE-WALK (x.left)
          POSTORDER-TREE-WALK (x.right)
          print  x.key
```

Teorem. (Samo INORDER-TREE-WALK)

Neka je T stablo od n čvorova. Tada se obilasci INORDER-TREE-WALK (T.root), PREORDER-TREE-WALK (T.root) i POSTORDER-TREE-WALK (T.root) izvršavaju u $\Theta(n)$ vremenu.

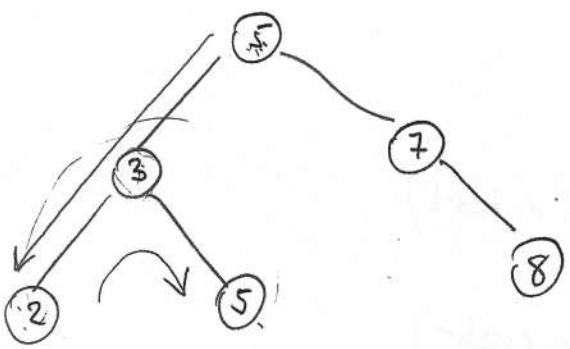
Primjer:



INORDER:

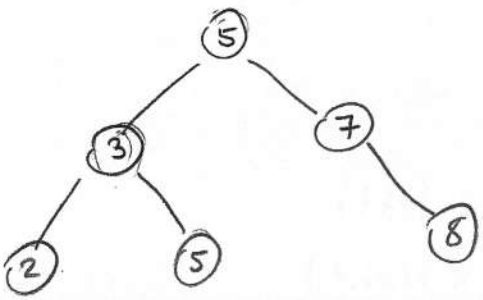
2 3 5 5 7 8

sortirani čvorovi po ključu



PREORDER:

5 3 2 5 7 8

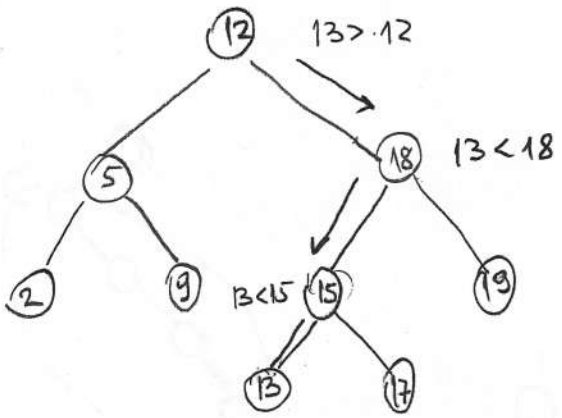


POSTORDER:

2 5 3 8 7 5

UBACIVANJE ELEMENATA U BINARNO STABLO PRETRAŽIVANJA

Primjer:



Ubaciti čvor s ključem 13

1. pronaći odgovarajuću poziciju
2. ubaciti element

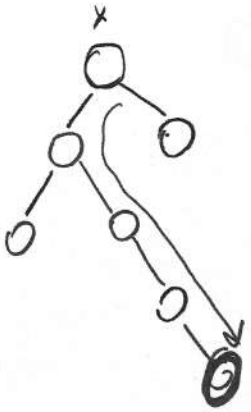
TREE-INSERT (T, z)

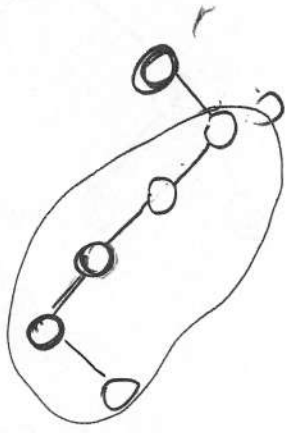
```
y ← NIL
x ← T.root
while x ≠ NIL
    y ← x
    if z.key < x.key
        then x ← x.left
        else x ← x.right
end while
y ← z.p
if y = NIL (ako je T bilo prazno stablo)
    then T.root = z
    else if z.key < y.key
            then y.left ← z
            else y.right ← z
end if
```

TREE-PREDECESSOR (x)

```
if x.left ≠ NIL
    return TREE-MAXIMUM(x.left)        } O(n)
end if

y ← x.p
while y ≠ NIL and x = y.left
    x ← y
    y ← y.p
end while
return y
```





Minimalni i maksimalni element

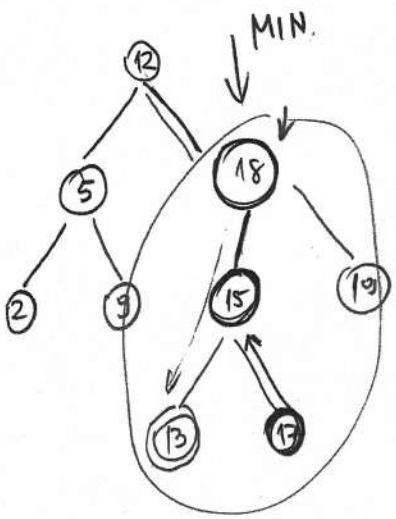
Ulaz: korijen podstabla x

Izlaz: pokazivač na čvor s (najmanjim / najvećim) ključem u stablu s korijenom x

```
TREE-MINIMUM (x)
    while x.left ≠ NIL
        x ← x.left
    end while
    return x
```

```
TREE-MAXIMUM (x)
    while x.right ≠ NIL
        x ← x.right
    end while
    return x
```

Sljedbenik čvora



Sljed. 12 je 13

Sljed od 17 je 18

Moguća su dva slučaja:

1.) čvor x nema desno podstablo

2.) ~~—~~ ima ~~—~~ ~~—~~ ✓

Propozicija

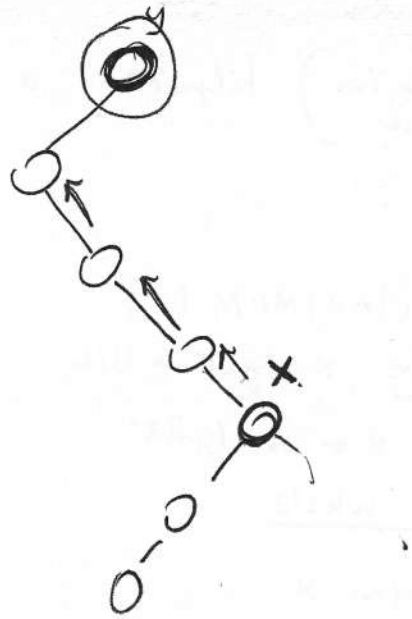
Neka je T BSP s čvorovima čiji su ključevi međusobno različiti. Ako je desno podstablo s korijenom x u T prazno i x ima sljedbenika y, onda je y prvi roditelj (predak) od x čije je lijevo dijete također predak od x.

Dokaz:

Ako je y sljedbenik od x onda je x prethodnik od y. Lijevo podstablo od y je neprazno.

x_1 je maksimalan element u lijevom podstablu od y. Ukoliko pogledamo TREE-MAXIMUM (y) uvjerit ćemo

je. u. istinitost tvrdnje.



TREE-SUCCESSOR (x)

if x.right ≠ NIL
| return TREE-MINIMUM (x.right)
end if

} O(h)

y ← x.p
while y ≠ NIL and x = y.right
| x ← y
| y ← y.p
end while
return y

} O(h)

Analiza vremena izvršavanja:

TREE-SEARCH : O(h)

TREE-MINIMUM: O(h)

TREE-MAXIMUM: O(h)

TREE-SUCCESSOR: O(h)

TREE-PREDECESSOR: O(h)

Teorem: Neka je T BSP visine h. Tada se operacije SEARCH, MINIMUM, MAXIMUM, SUCCESSOR i PREDECESSOR izvršavaju O(h) vremenu.

Teorem

Ako je x korijen binarnog stabla koje ima n čvorova, INORDER-TREE-WALK(x) izvršava se u $\Theta(n)$ vremenu.

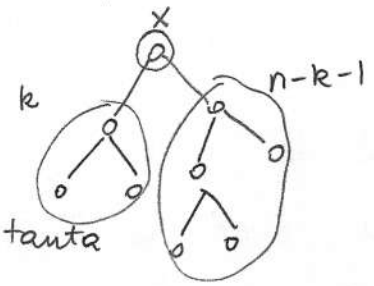
dokaz: (metodom supstitucije)

$T(n)$ ← vrijeme izvršavanja

$n=0 \Rightarrow T(0) = c$, c konstanta ($x \neq NIL$)

$n>0 \Rightarrow$

$$T(n) = T(k) + T(n-k-1) + d$$



d konstanta

Metoda supstitucije (mat. indukcija) $T(n) = \Theta(n) = (c+d)n + c$

$n=0$ $\quad T(0) = (c+d)\cdot 0 + c = c$ ✓

Pretpostavka: Tvrdnja vrijedi za n i tj:

$$T(n) = (c+d)\cdot n + c \quad //$$

$0 < k < n$,

korak:

$$\begin{aligned} T(n+1) &= T(k) + T(n+1-k-1) + d \\ &= T(k) + T(n-k) + d \\ &= (c+d)k + c + (c+d)(n-k) + c + d \\ &= (c+d)(k+1) + (c+d)(n-k) + c \\ &= (c+d)(n-k+k+1) + c \\ &= (c+d)(n+1) + c \end{aligned}$$

$\Rightarrow \quad T(n) = \Theta(n)$ $\square$

pregled operacija:

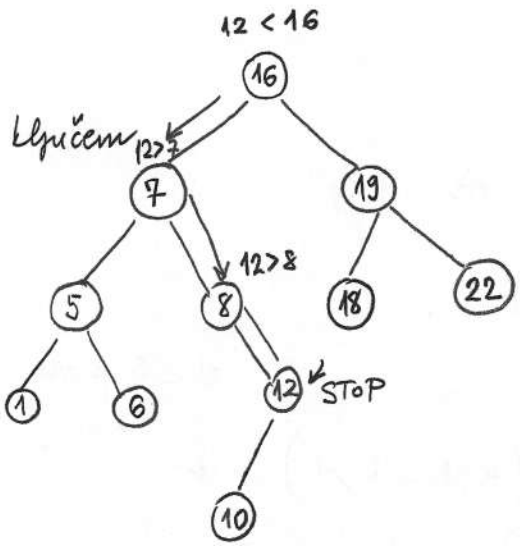
- obilasci BSP-a: - INORDER
  - POSTORDER
  - PREORDER
- pretraživanje
- minimum i maksimum
- sljedbenik i prethodnik
- umetanje i brisanje

Pretraživanje:

ulaz: korijen x ključ k

izlaz: pokazivač na čvor u T kojemu je ključ k, ili NIL ukoliko takav ne postoji

primjer:



Uputa: koristi svojstvo BSP

```
TREE-SEARCH (x, k)
if x = NIL or k = x.key
|    return x
end if
if k < x.key
|    TREE-SEARCH (x.left, k)
else
|    TREE-SEARCH (x.right, k)
end if
```

zadatak:

Napiši iterativnu verziju.

```
ITERATIVE-TREE-SEARCH (x, k)
while x ≠ NIL and k ≠ x.key
|    if k < x.key
|    |    x ← x.left
|    else
|    |    x ← x.right
|    end if
end while
return x
```