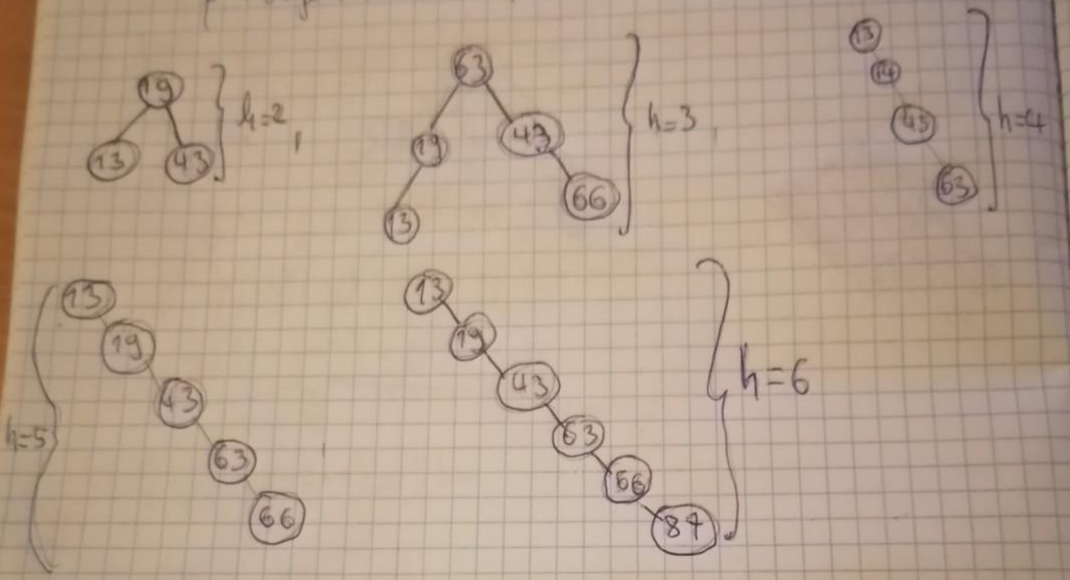
Zadaća 2

Zad. 1 Za skup ključeva {13, 19, 43, 63, 66, 84, 92} nacrtajte binarna stabla pretraživanja visine 2, 3, 4, 5, 6.



Zad. 2. Neka je T BST u kojem su svi ključevi različiti. Dokažite da ako čvor x nema desno dijete i y je sljedbenik od x, onda je y najniži predak od x čije je dijete isto predak od x.

Dokaz:

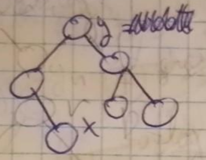
Iz algoritma successor imamo:

x nema desno dijete ⇒

```
define x_s;
y = x.p;
while (y ≠ nil and y.right == x):
    x_s = y;
    y = y.p;
```

y je sljedbenik od x

x nema desno dijete ⇒ algoritam ulazi u drugu granu



BSO, NEKA JE OVO PODSTABLO nekog stabla T.

Iz ovog crteža, jasno se vidi da je jedini put do x preko y, koji mu je predak i sljedbenik. Struktura je važeća za bilo koje podstablo jer x mora biti dio lijeve grane od svog sljedbenika u ovom slučaju ⇒ tvrdnja je zadovoljena.

Zad. 3

x list ⇒ nema lijevo i desno dijete

x može biti lijevo ili desno dijete od y.

1° x lijevo dijete od y:

x list ⇒ x je sigurno prethodnik od y ⇔ y je sljedbenik od x ✓

⇔ tvrdnja 1

2° x je desno dijete od y

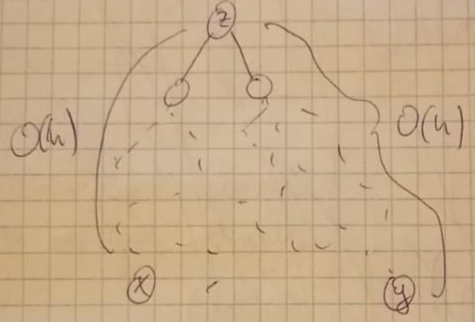
x list ⇒ x je sigurno sljedbenik od y ⇔ y je prethodnik od x ⇔ tvrdnja 2

Zad. 4

Neka je x čvor na kojem smo pozvali tree-successor, y k-ti sljedbenik od x. Neka je z najmanji zajednički predak od x i y.

TREE-SUCCESSOR SE PONAŠA KAO OBILAZAK STABLA, što znači da nikad jedan ~~[illegible]~~ node nećemo posjetiti više od 3 puta (prilikom dolaska, povratak rekurzije iz lijevog i iz desnog djeteta).

Kako duljina jednog puta može biti najviše h, vrijeme možemo ocijeniti kao: $O(\max(3k, 2h)) = O(3k+2h) = O(k+h)$



Zad. 5

S pretpostavkom znamo i vrijeme izvršavanja algoritma TREE-MINIMUM je $O(h)$.

Konstruirajmo funkciju zadanu u zadatku:

```
ISPIS (T):
  Node* n = T.TREE-MINIMUM()
  for i in range 1 to n:
      n = TREE-SUCCESSOR(n);
```

U zad 4. smo dokazali: vrijeme izvršavanja k uzast. poziva TREE-SUCC JE $O(k+h)$ za proizv. k.

Specijalno za $k = n-1$, imamo: $O(n-1+h) = O(n+$

Kako je $n \geq h$ (broj čvorova $\geq$ visini)

$\Rightarrow$ ISPIS IMA VSA $O(n)$.

Zadatak 6

Insert: (koristim isti s predavanja)

```
tree_insert(T, z) {
  y = nil;
  x = T.root;
  while(x != nil) {
   y = x;
   if(z.key < x.key) {
    x = x.left
   }
  else x = x.right
  }
  z.p = y;
  if(y == nil) {
   T.root = z;
  }
  elif(z.key < y.key) {
   y.left = z;
  }
  else y.right = z;
}
```

Search: (također s predavanja)

```
tree_insert(x, k) {
   if(x.key == k || x == nil) return x
   if(k < x.key) return tree_search(x.left, k)
   else return tree_search(x.right, k)
}
```

Analiza

Iako je jedan algoritam rekurzivan, a jedan iterativan, vidljivo je da se oba algoritma izvršavaju u O(h) vremenu, gdje je h visina stabla.
Nadalje, iz uvjeta u petlji prvog i rekurziji drugog vidimo da će oba algoritma proći isti put da dođu do nodea koji im treba,
do na rubni uvjet rekurzije.
Kako kod searcha taj element po pretpostavci, unutar stabla, to znači da će search proći istim nodeovima, a kao jedan više računamo novokreirani node.
Prethodna rečenica dokazuje tvrdnju.

Zadatak 7

<u>Insert koristim iz prošlog zadatka</u>

Inorder print

```
inorder_print(x){
  if(x != nil) {
    inorder_print(x.left);
    print(x.key)
    inorder_print(x.right);
  }
}
```

Analiza

Analiza

Inorder print:

Inorder print ima vremensku slozenost O(n). To je nekako i očito jer moramo proći svaki node u stablu, znači imamo n rekurzivnih poziva od kojih svaki ima složenost O(1).

Insert

Jedan insert ima vremensku složenost O(h) gdje je h visina stabla. Kako inserata radimo n, imat ćemo vremensku složenost O(n*h).
U najboljem slučaju, visina stabla je lg n, gdje je n broj elemenata. To bi značilo savršenu ili gotovo savršenu balansiranost našeg stabla,
tj. da imamo približno jednako elemenata sa svake strane. Najgori slučaj je lanac, gdje je trošak inserta jednak kao i kod povezane liste [O(n)]

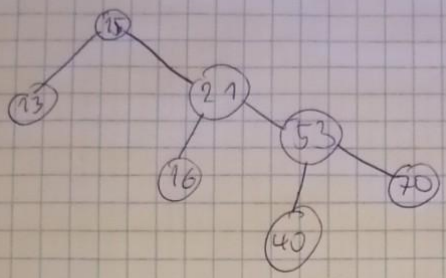
Vremenska složenost procedure

Za relativno mali broj insert operacija, insert operacija će početi dominirati u VSA.
Best case: Svaki insert košta O(lg n) => VSA algoritma je O (n * lgn)
Worst case: Svaki insert košta O(n) => VSA algoritma je O (n^2)

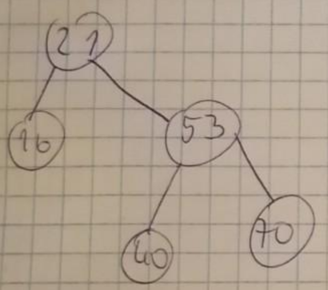
Zad. 8

BRISANJE NIJE KOMUTATIVNO:

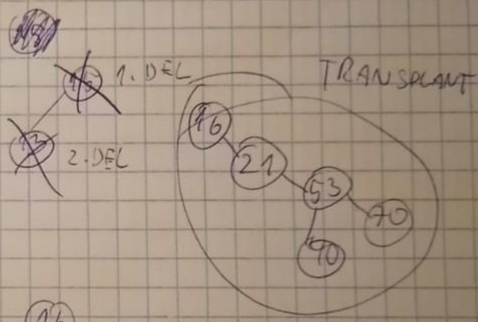
KONTRA PRIMJER:



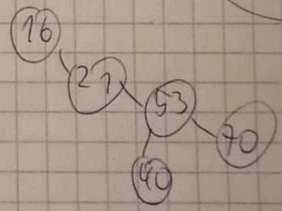
BRISANJEM 13 PA 15 IMAMO:



OBRATNO IMAMO:



=>



$\frac{9}{10}$ n+1

a = 1