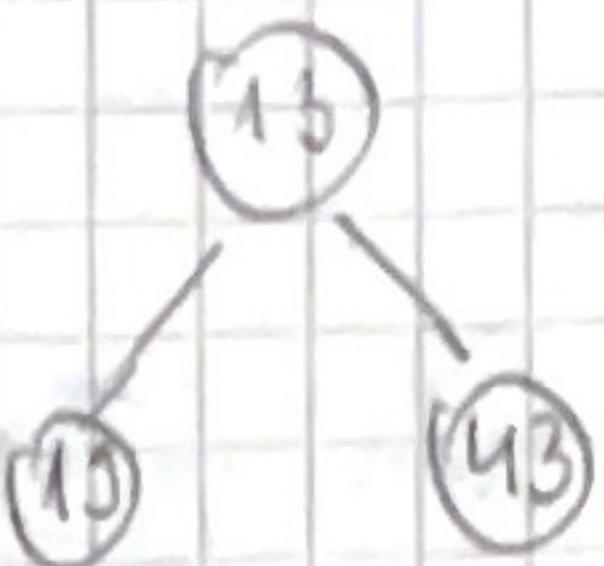


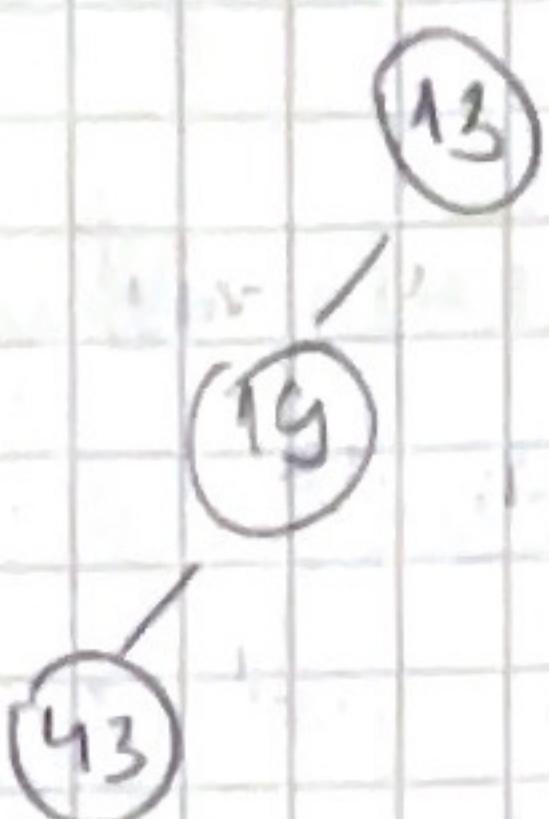
SPA2 - ZADACA 3

① keys : {13, 19, 43, 63, 66, 87, 92}

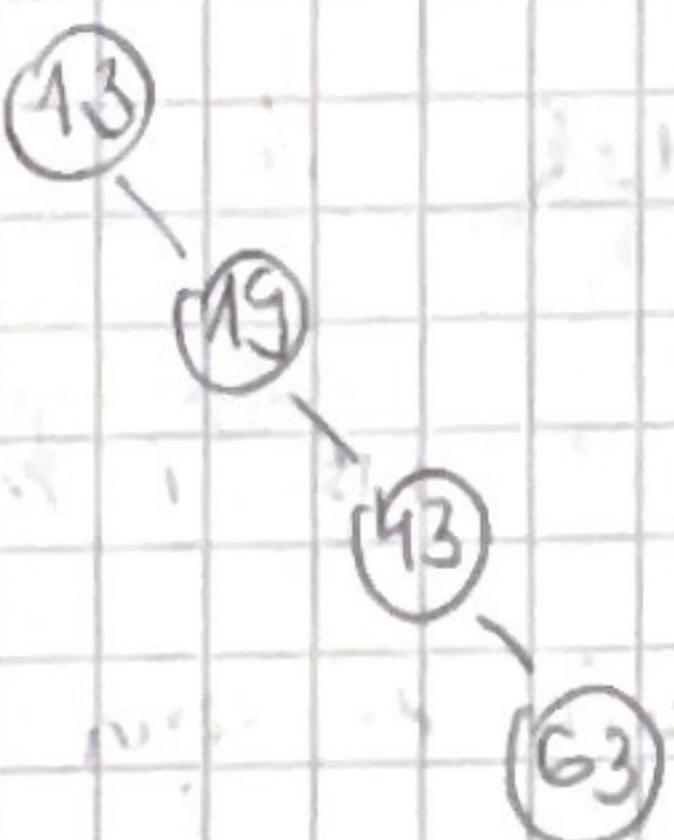
h: 2



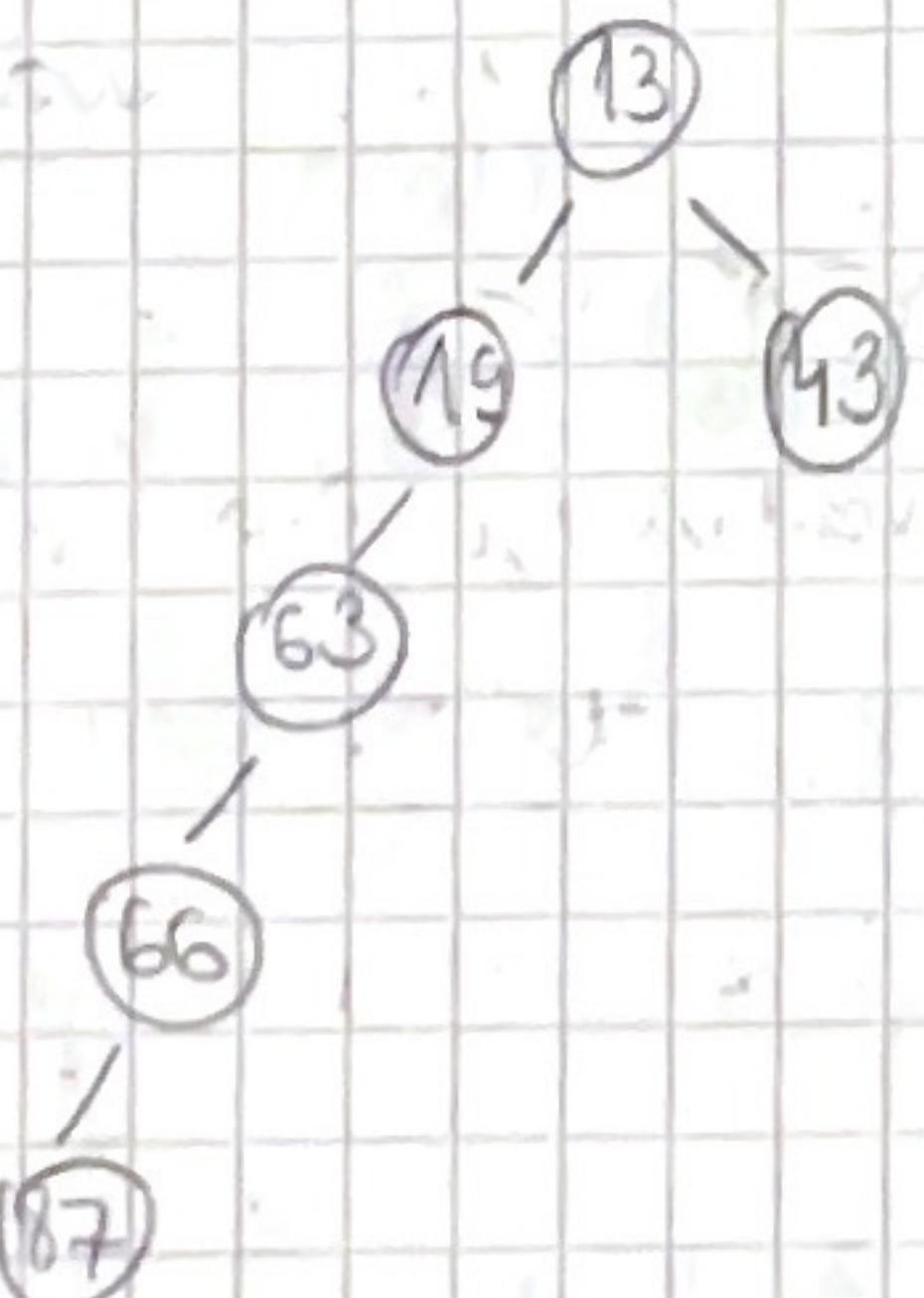
h: 3



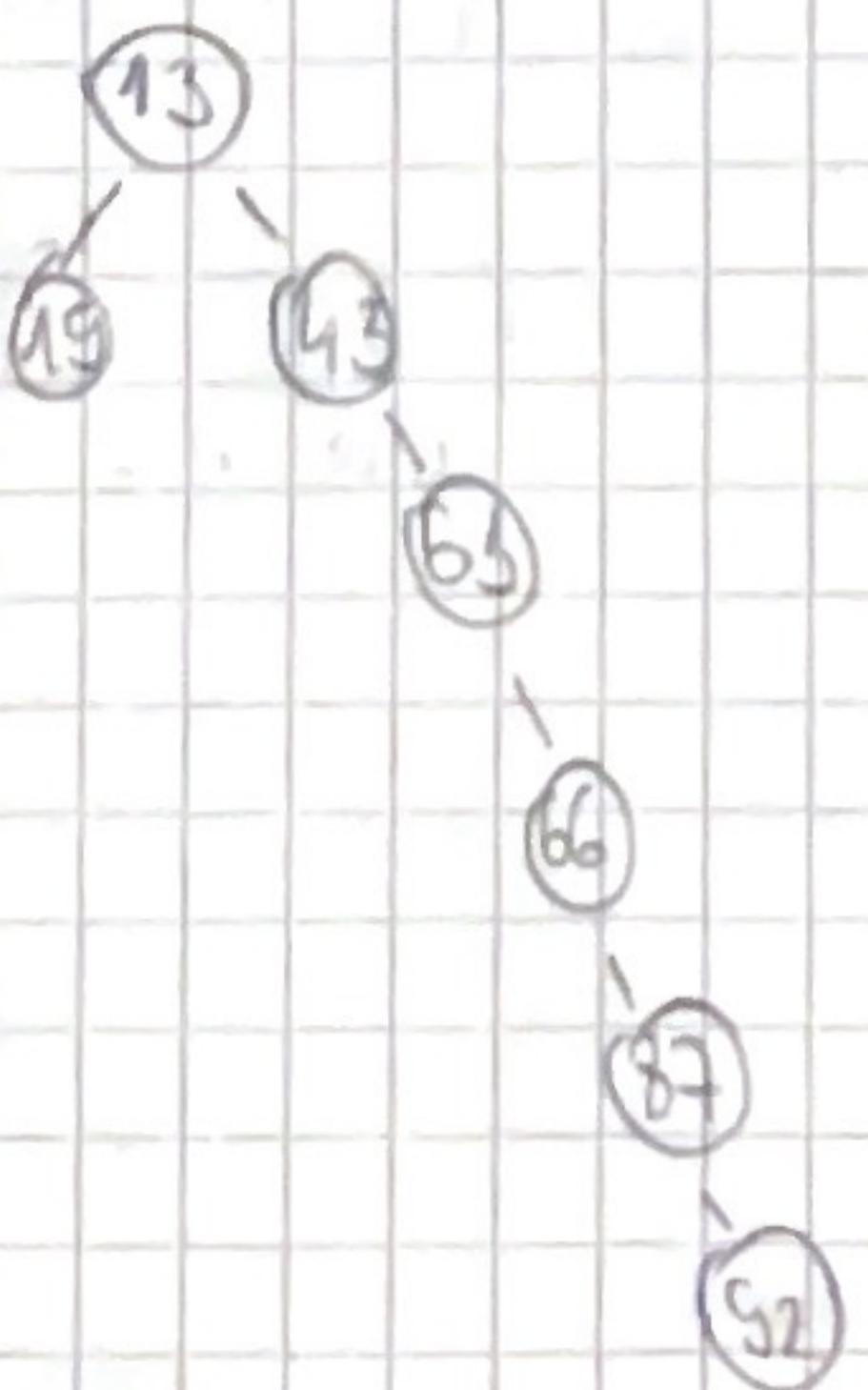
h: 4

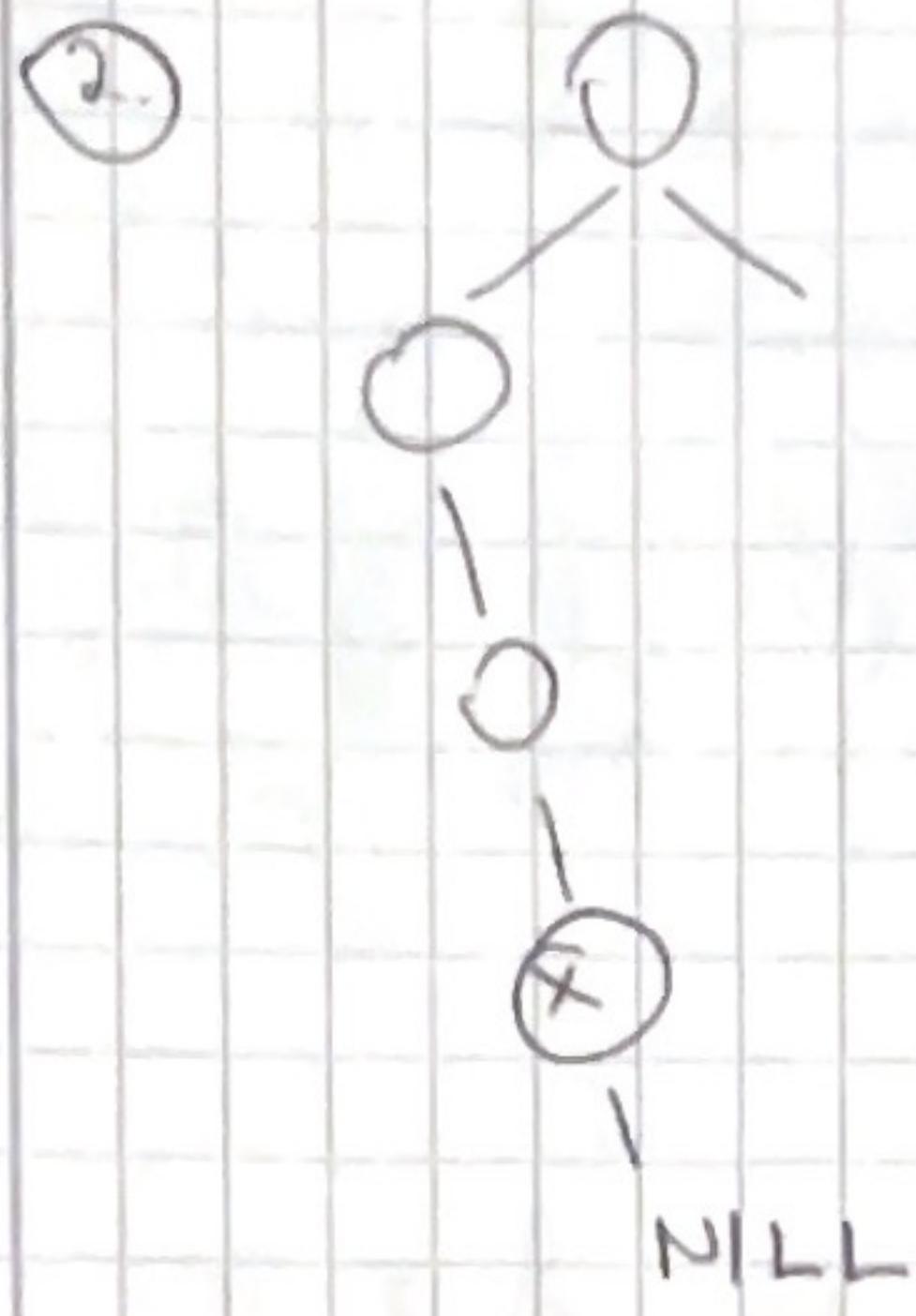


h: 5



h: 6





y mora biti predak od x, a z y nije predak od x
 Tada ne bi vrijedila svojstva BST-a. Neke y nije
 predak od x, z najblizi predak od x i y. Slijedi
 $x < z < y$, tj. y ne bi bio slijedbenik od x

y.left mora biti predak od x prema pravilu BST-a

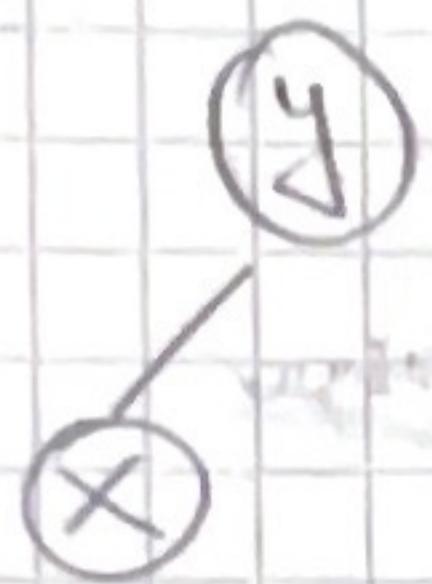
Tko pretpostavimo da y nije najnizi predak od x
 od kojeg je lijevo dijete točkotvor predak od x
 Stoga čvor z je najnizi predak od x, a čije je
 lijevo dijete točkotvor predak od x. Tada bi čvor z
 morao biti u lijevom podstabku od y pa je $z < y$
 a tuc dođemo do kontradikcije sa pretpostavkom
 odnosno da je y slijedbenik od x.

Tko je y najnizi predak od x, a čije je lijevo
 dijete točkotvor predak od x tada je x čvor s max.
 bljećem u lijevom podstabku od y.

Iz prethodnog možemo reći, da je y slijedbenik
 od x, a čije je dijete predak od x, najnizi
 predak od x.

3. x list \rightarrow nemu dijce.

y je roditelj od x .



- x je list ty. nemu dijce, ne postoji

x .right koji bi tada bio nai od

x a manji od y

je obratno da je y roditelj od x

stjedi $y > x$

$y.right > y$

za svakog predka od y vrijedi da je
 y manji od njega

y je najmanji predak od x u T

- u alg. successor while petlji ne može vratiti
nijednom, nato će se vratiti y



sto je x desno dijete od y , while
petlji u alg. predecessor također ne može
vratiti nijednom, isto će se vratiti y

④ Neha je x čvor na kojem smo pozvali tree - successor, a neha je y n -ti naslednik od x .
 Štočter neha je z najveći predak x i y (raodenički).
 Vrastopni poziv može potrošiti više od 2 puta
 kroz pogodnu granu. Uvodi se da će biti blizu nje
 izmedu x i y pregledat se ne manje jednom, a
 to će se dogoditi kada budemo prolazili od x
 do z ili od y do z . Duljina prelaska tog puta
 ograničena je sa h dokle vrijeme izvršenja
 možemo reapisati kao $3k + 2h = O(k+h)$

5. Iz 4. ravnopravno da su u n vrastopnih pozivu
 tree - successor je $O(h+k)$, i ovo je imamo $n-1$
 poziva tj. $O(n-1+h)$, a ubravom mu to da je
 $n \geq h$ dobijemo $O(n)$. Točkač možemo srediti
 da je VSA $\Rightarrow O(n)$ jer alg. prolazi manje
 2 puta kroz svaku granu i svaki.

Tree - minimum prolazi kroz ljevu granu kada
 li dođe do minimuma

BST imao $n-1$ mihova pa je to gornje meste
 $O(n)$

Snoder prolazi kroz cijeli BST, tj. n pa
 je donja mesta $S(n)$

VSA : $O(n)$

6. Insert trädet put od rootne do roditelju īvora
kojeg će ubaciti. Search će proći isti put,
a onda će inserati ubaćeni īvor kojeg je
insert ubacio. Dakle search će inserati $n+1$
īvora.

7. INORDER-TREE-WALK(x)

1. if $x \neq \text{NIL}$
2. INORDER-TREE-WALK($x.\text{left}$)
3. print $x.\text{key}$
4. INORDER-TREE-WALK($x.\text{right}$)

VSA za n īvora: $\Theta(n)$

TREE-INSERT(T, z)

1. $y = \text{NIL}$
 2. $x = T.\text{root}$
 3. while $x \neq \text{NIL}$
 4. $y = x$
 5. if $z.\text{key} < x.\text{key}$
 6. $x = x.\text{left}$
 7. else $x = x.\text{right}$
- } h prima

8. $z.P = y$

9. if $y = \text{NIL}$

10. $T.\text{root} = z$

11. else $z.\text{key} < y.\text{key}$

12. $y.\text{left} = z$

13. else $y.\text{right} = z$

Ako vremenu n inseruje
 \rightarrow VSA: $\Theta(n \cdot h)$

VSA: $\Theta(h)$

WORST CASE:

prematrano situacija u kojoj je niz n brojeva sortiran, tako da je u formi INSERT-a napraviti BST sive n, h=n

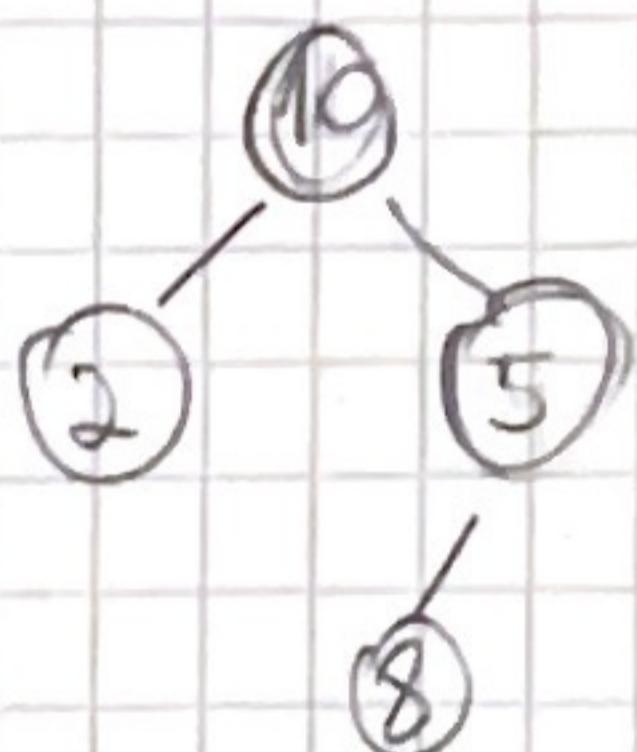
$$VSA: \Theta(n^2+n) = \Theta(n^2)$$

BEST CASE:

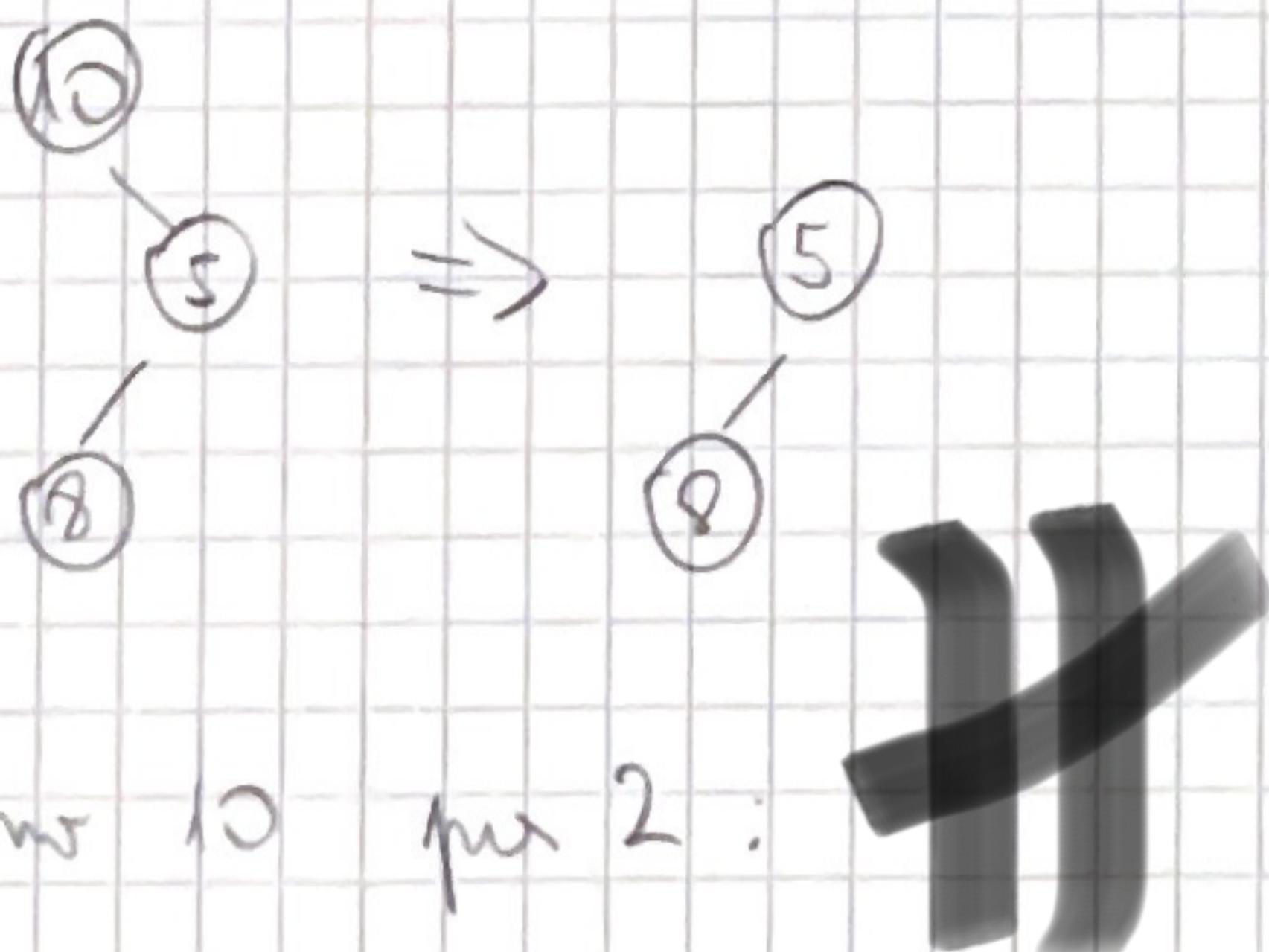
istak n poziva stvorit će balansirano stablo
 $\Rightarrow h \leq \log n$

$$VSA: \Theta(n \log n + n) = \Theta(n \log n)$$

- Operacija brisanja el. iz BST-a nije komutativna



da obrišemo 2 pre 10:



da obrišemo 10 pre 2:

