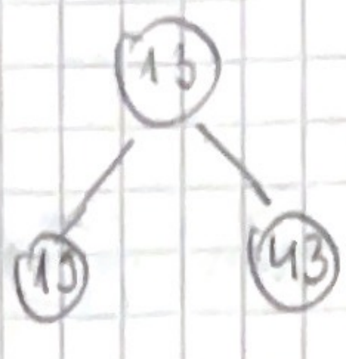
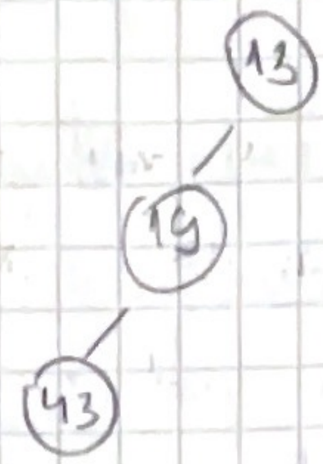
SPA2 - ZADAĆA 3

1. keys : {13, 19, 43, 63, 66, 87, 92}

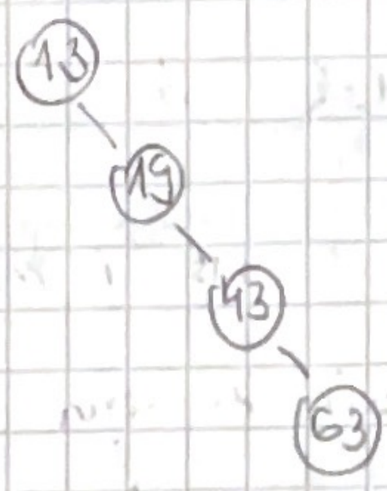
h : 2



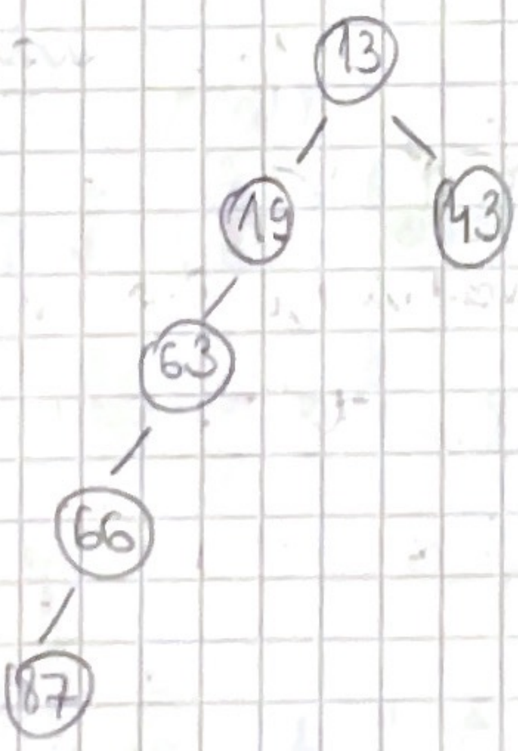
h : 3



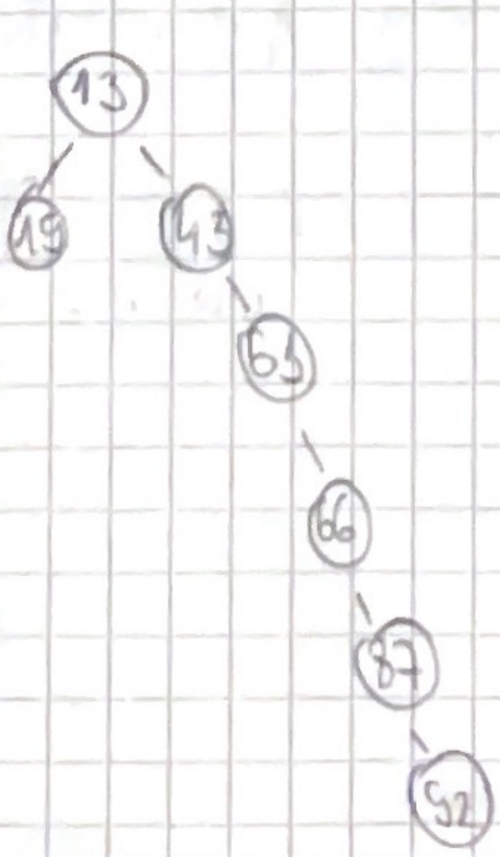
h : 4



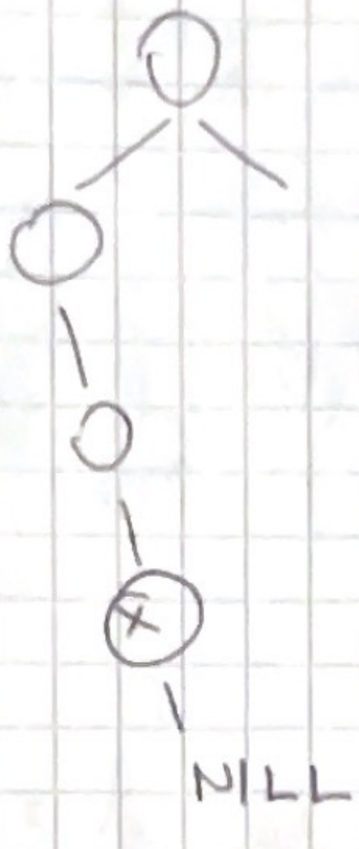
h : 5



h : 6



2.



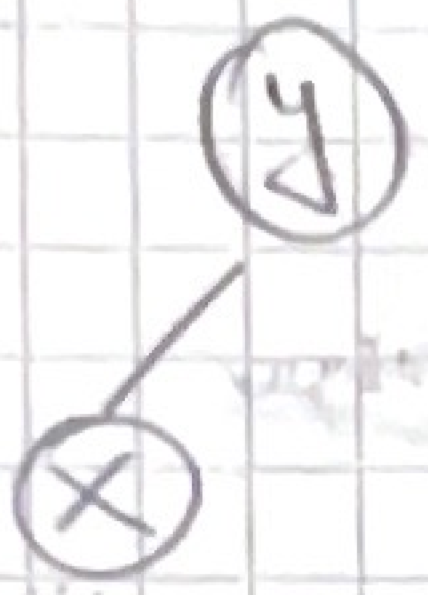
y mora biti predak od x, dbo y nije predak od x
Tada ne bi vrijedila svojstva BST-a. Neka y nije
predak od x, z najbliži predak od x i y. Slijedi
$x < z < y$, tj. y ne bi bio sljedbenik od x

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

Sada pretpostavimo da y nije najmanji predak od x
od kojeg je lijevo dijete također predak od x
Neka čvor z je najmanji predak od x, a čije je
lijevo dijete također predak od x. Tada bi čvor z
morao biti u lijevom podstablu od y pa je $k < y$
a tu dolazimo do kontradikcije sa pretpostavkom
odnosno da je y sljedbenik od x

Sada je y najmanji predak od x, a čije je lijevo
dijete također predak od x tada je x čvor s max.
sljedbenikom u lijevom podstablu od y
Iz prethodnoga možemo zaključiti, da je y sljedbenik
od x, a čije je dijete predak od x, najmanji
predak od x.

3. x list → nema djece
y je roditelj od x



- x je list tj. nema djece, ne postoji x.right koji bi tada bio veći od x a manji od y

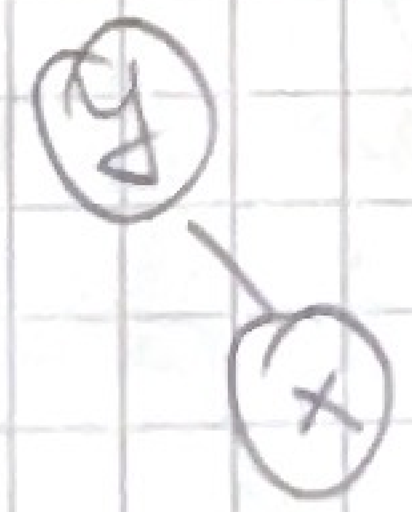
s obzirom da je y roditelj od x slijedi y > x

y.right > y

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

y je najmanji predak od x u T

- u alg. successor while petlja se neće vrtiti nijednom, nato će se vratiti y



ako je x desno dijete od y, while petlja u alg. predecessor također se neće vrtiti nijednom, isto će se vratiti y

(4) Neka je x čvor na kojem smo pozvali
tree-successor, a neka je y n-ti nasljednik od x.
Također neka je z najniži predak x i y (zajednički).
Uzastopni pozivi neće prolaziti više od 2 puta
kroz pojedinu granu. Svaki vrh čiji ključ nije
između x i y pregledat će se najviše jednom, a
to će se dogoditi kada budemo prolazili od x
do z ili od y do z. Duljina prelaska tog puta
ograničena je s h dakle vrijeme izvođenja
možemo zapisati kao $3k + 2h = O(k+h)$ //

(5.) Iz 4. znamo da za n uzastopnih poziva
tree-successor je $O(h+k)$, ovdje imamo n-1
poziva tj. $O(n-1+h)$, s obzirom na to da je
$n \geq h$ slijedi $O(n)$. Također možemo vidjeti
da je VSA => $O(n)$ jer alg. prolazi najviše
2 puta kroz svaku granu i vrh.
Tree-minimum prolazi kroz lijevu granu kako
bi došao do minimuma.

BST ima n-1 vrhova pa je to gornja međa
$O(n)$
Inorder prolazi kroz cijeli BST, tj. n pa
je donja međa $\Omega(n)$

VSA : $O(n)$ //

(6) Insert traži put od roota do roditelja čvora kojeg će ubaciti. Search će proći isti put, a onda će ispisati ubačeni čvor kojeg je insert ubacio. Dalje search će ispisati n+1 čvorova.

(7) INORDER-TREE-WALK(x)

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

VSA za n čvorova: $\Theta(n)$

TREE-INSERT(T, z)

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

} h puta

8. z.p = y
9. if y == NIL
10. T.root = z
11. elif z.key < y.key
12. y.left = z
13. else y.right = z

VSA: O(h)

Ako imamo n insertova
=> VSA: O(n.h)

WORST CASE:

promatramo situaciju u kojoj je niz n brojeva sortiran, tada će n pozivu INSERTa napravit BST visine n, $h = n$

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

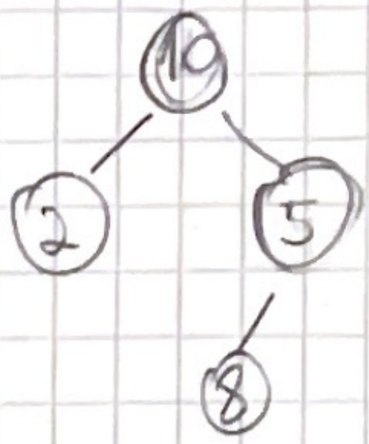
BEST CASE:

ishl n poziva stvorit će balansirano stablo

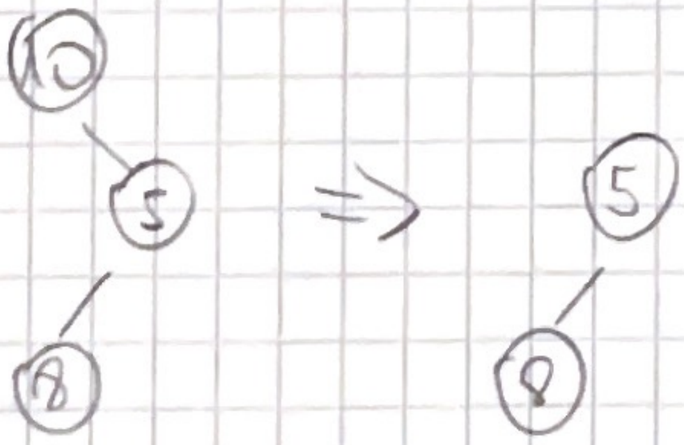
$\Rightarrow h \leq \log n$

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

② Operacija brisanja el. iz BST-a nije komutativna



ako obrišemo 2 pa onda 10:



ako obrišemo 10 pa 2: