

BST Traversal

Inorder traversal:

Tree searching

Tree-search (x, k) <sup>x is a node
 k is key</sup>

if $k = x.key$ or $x == \text{Null}$
return x

if $k < x.key$

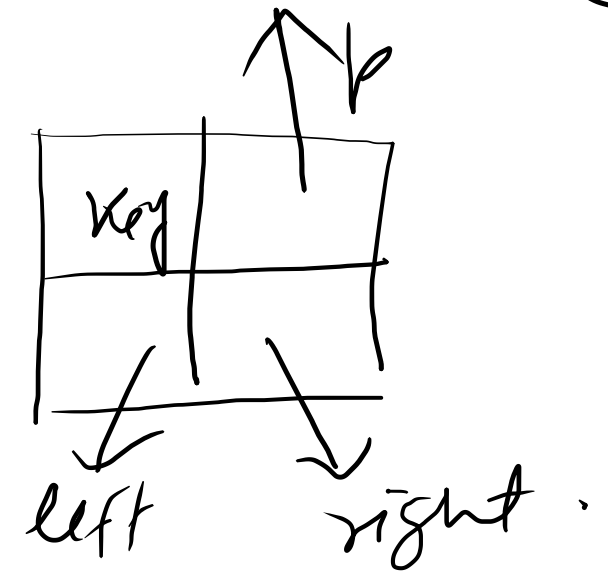
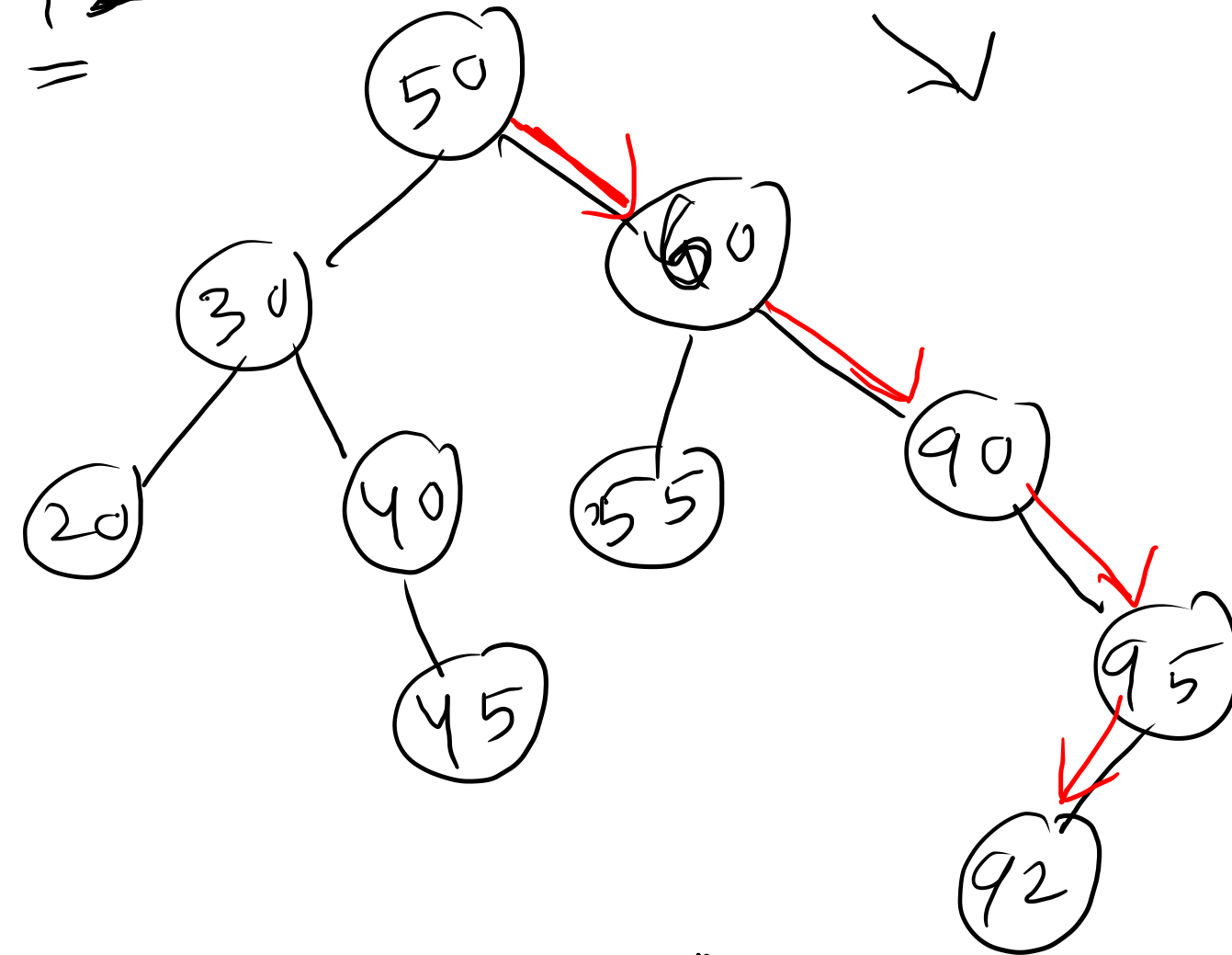
tree-search ($x.left, k$)

else

tree-search ($x.right, k$)

Running time: $O(\text{height})$

$$K = 92$$



Iterative

iter-tree-search(x, k)

while $x \neq \text{Null}$ or $x.\text{key} \neq k$

 if $k < x.\text{key}$

$x = x.\text{left}$

 else

$x = x.\text{right}$

return x

Running time: $O(\text{height})$

Not-exact search

$K = 48$

`not-exact-search(x, K)`

`y = null`

`while x != null`

`y = x`

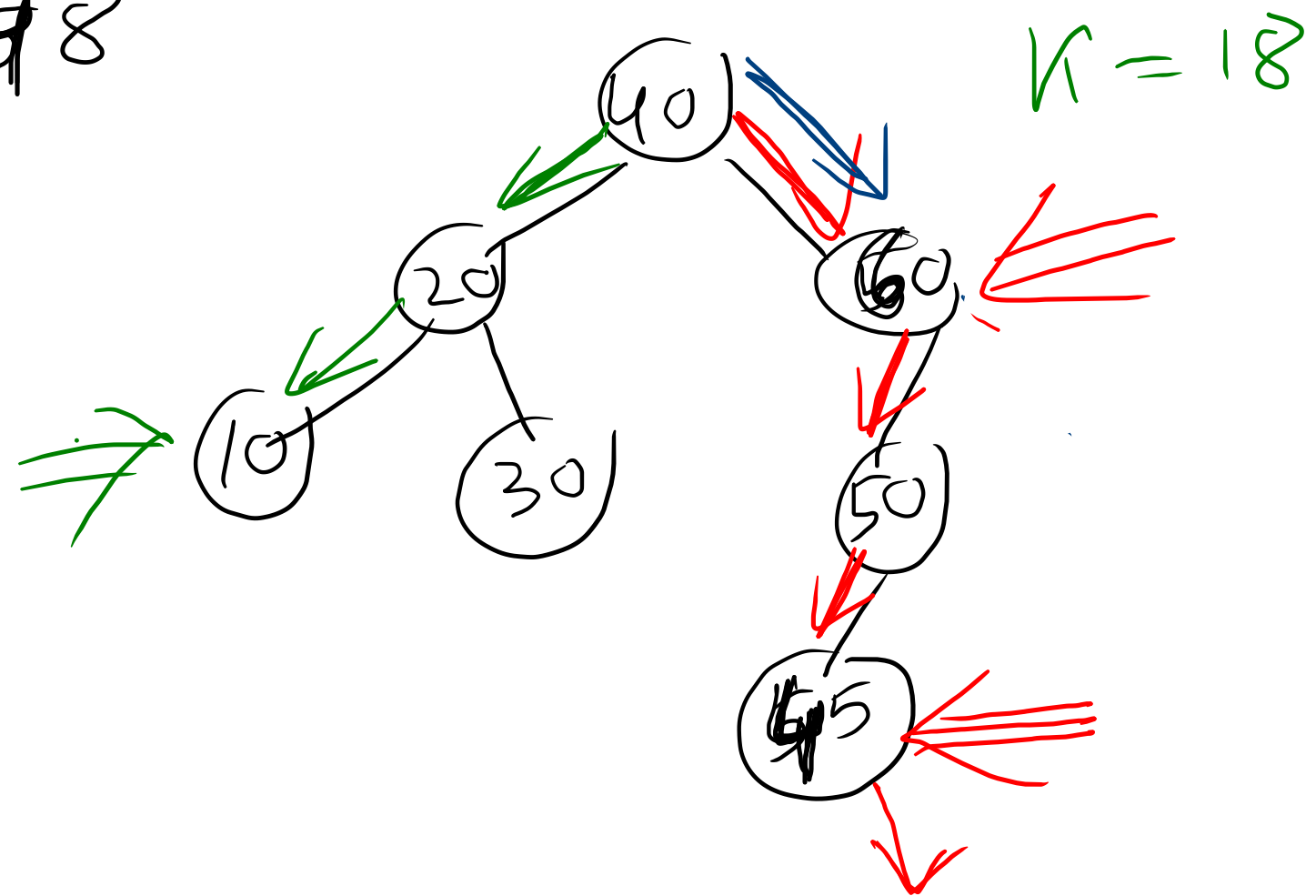
`if $x < K$`

`x = x.left`

`else`

`x = x.right`

`return y`

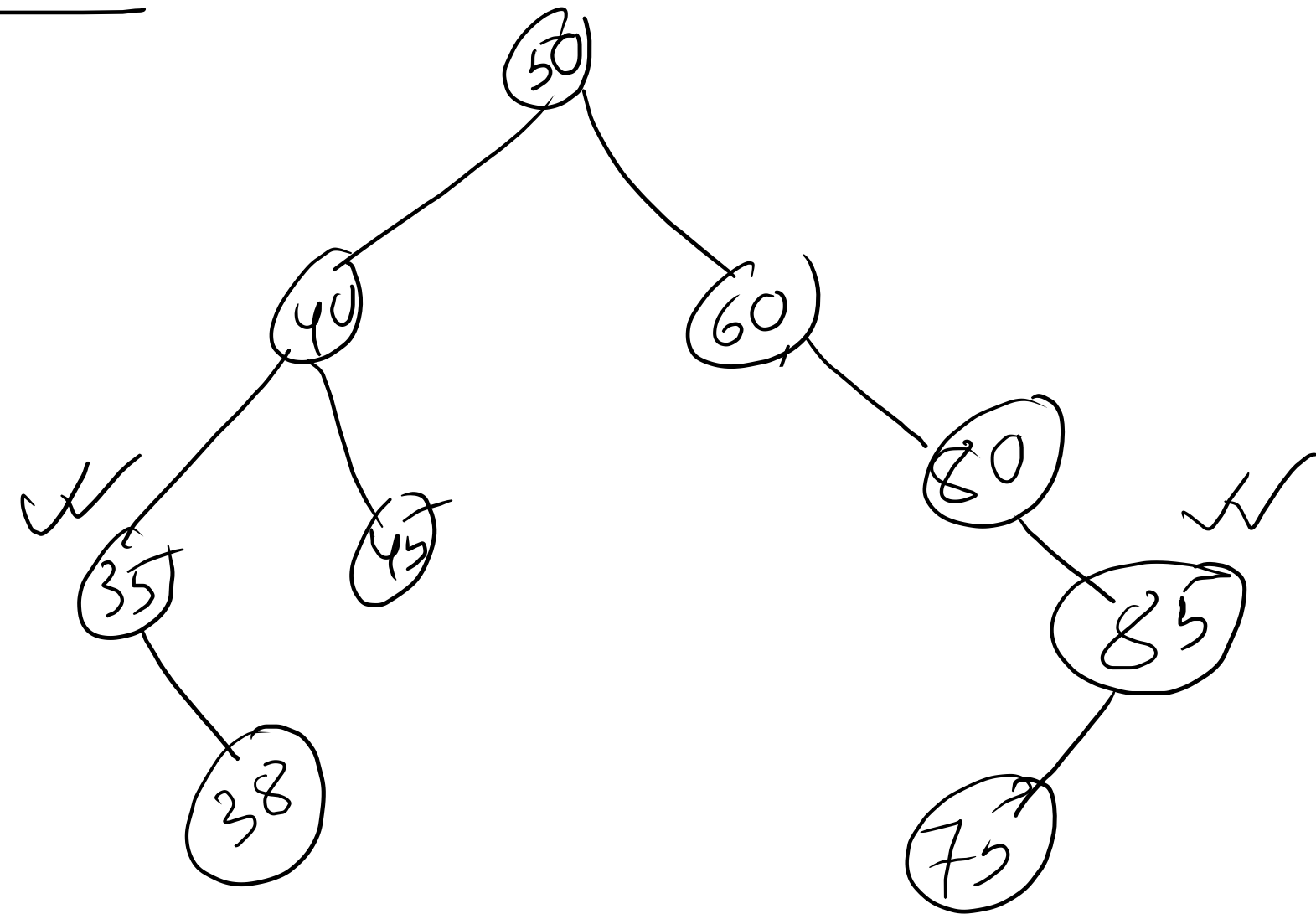


Running time: $O(\text{height})$

Tree minimum and maximum

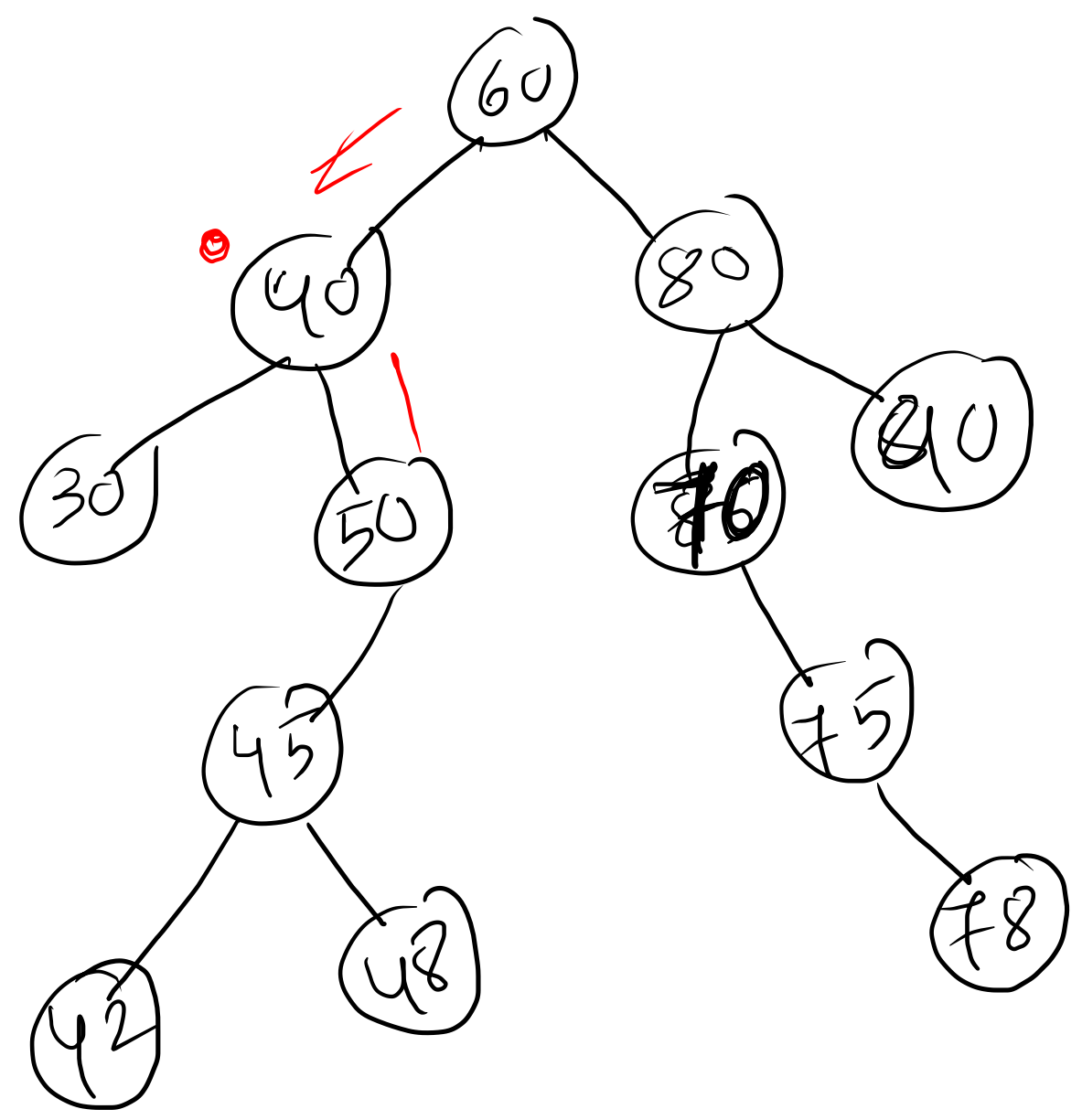
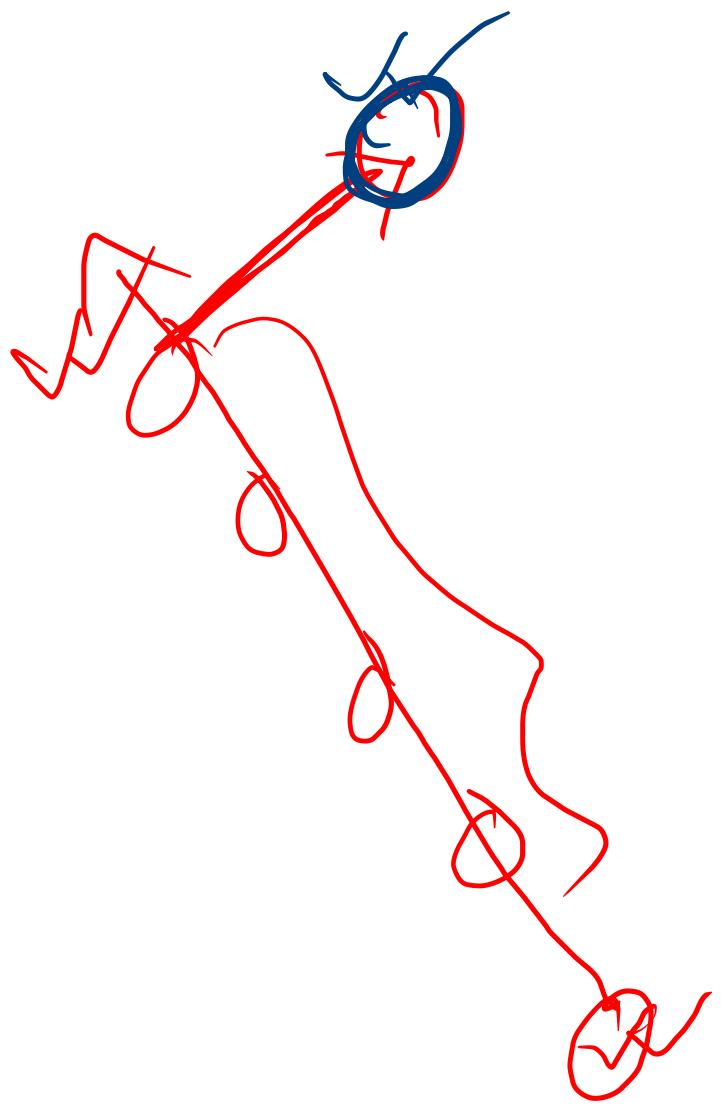
Tree minimum(x)
while $x \cdot \text{left} \neq \text{Null}$
└─ $x = x \cdot \text{left}$
return x

Tree maximum(x)
while $x \cdot \text{right} \neq \text{Null}$
└─ $x = x \cdot \text{right}$
return x



time: $O(\text{height})$

Successor



45 — 48
 40 — 42
 50 — 60
 78 — 80
 90 — Null

Tree-successor(x)

if $x.\text{right} \neq \text{Null}$

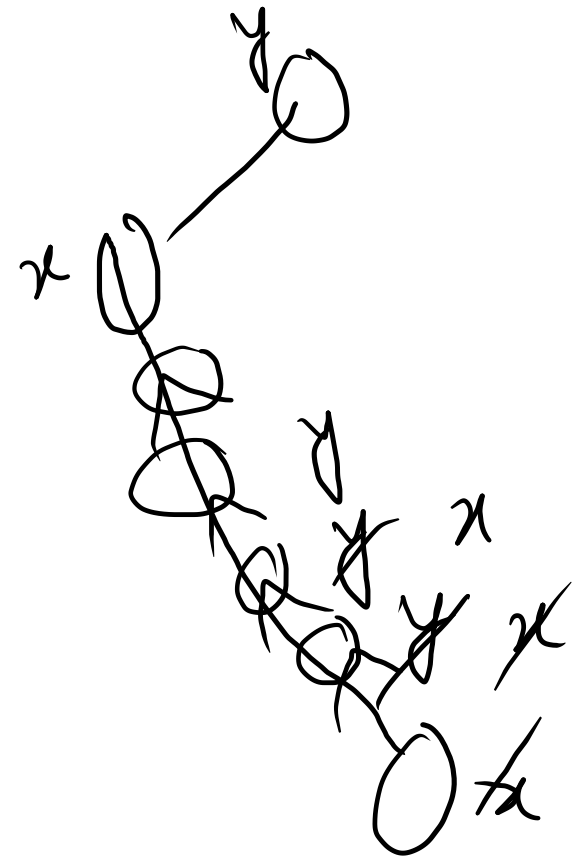
└ return tree-minimum($x.\text{right}$)

$y = x.p$

while $y \neq \text{Null}$ and $x \neq y.\text{left}$

└ $x = y$
 $y = x.p$

return y



Running time: $O(\text{height})$

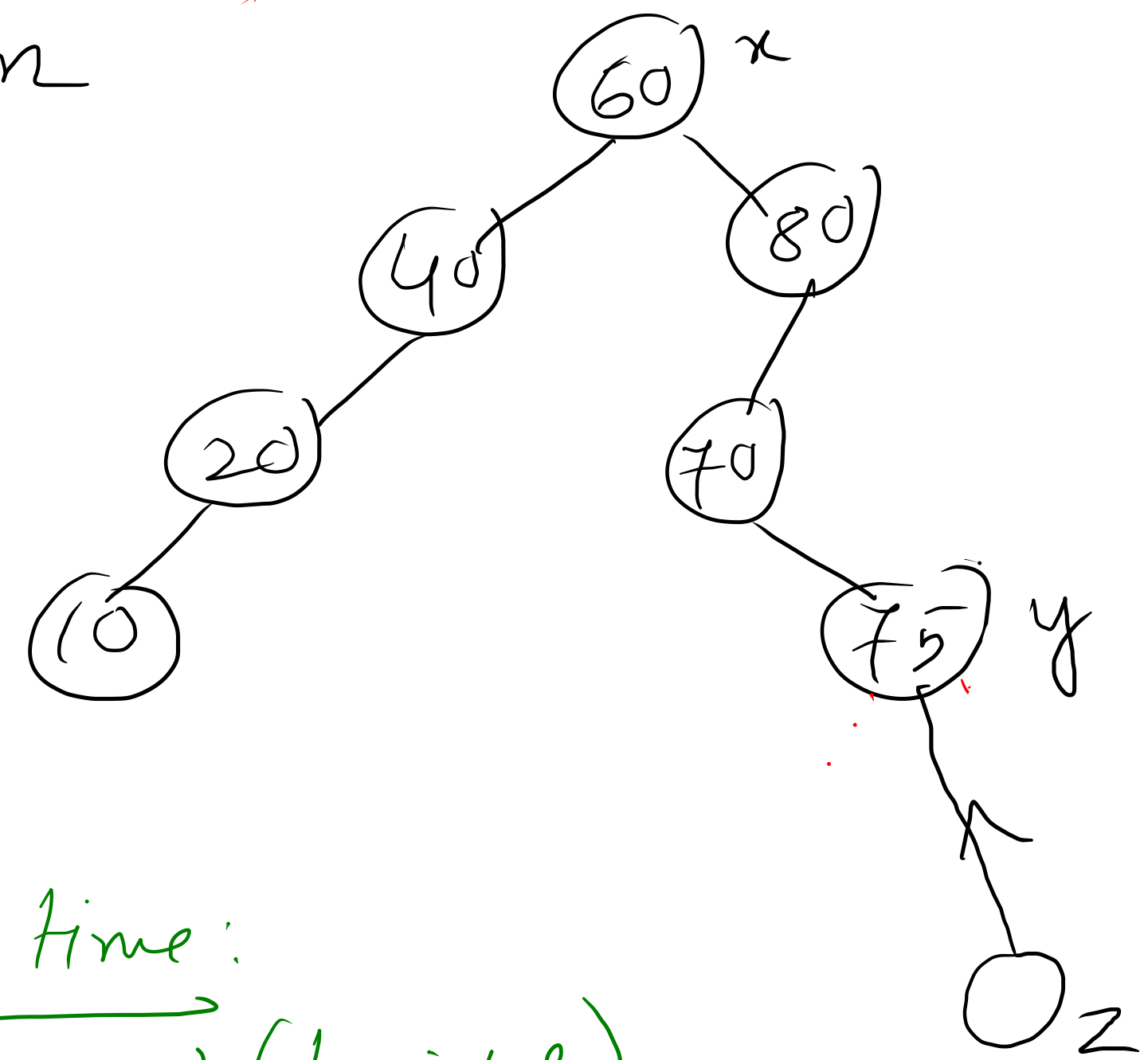
Tree-predecessor

How.

Dynamic Operations

BST tree insertion and tree deletion

Insert - 78
 Insert - 74



Tree insertion
 Tree-insertion(T, z) → tree T
→ node.

```

x = T.root
y = Null
while x != Null
  y = x
  if z.key < x.key
    x = x.left
  else
    x = x.right
  
```

→ if y == Null
 T.root = z
 z.p = y
 if z.key < y.key
 y.left = z
 else
 y.right = z

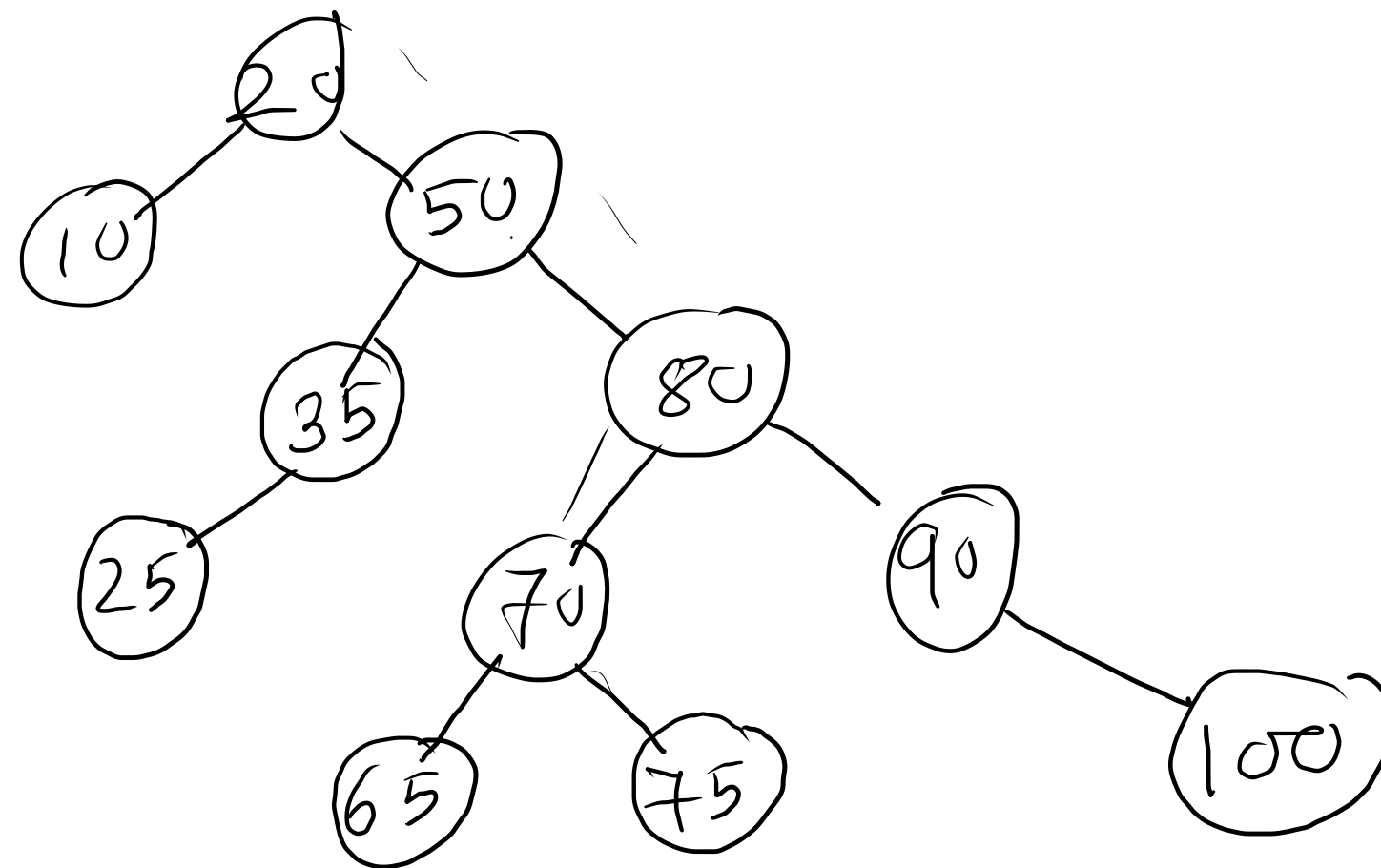
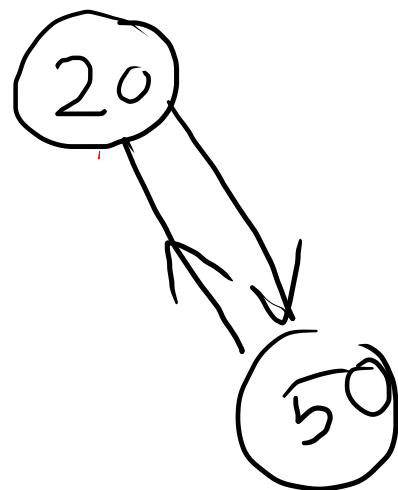
Running time:

$O(\text{height})$
 $\log n \leq \text{height} \leq n$

insert

20, 50, 80, 70, 90, 10, 65, 75
35, 25, 100

root



5 20 10 35 6 2 9 6 4

BST sorting running time

$O(n \cdot h_{\max})$