CSE 101 3-7-25

Rotations:

note:

$$Keys(\alpha) \le Key[y] \le Keys(\beta) \le Key[x] \le Keys(\gamma)$$

So Rotations preserve BST properties.

- Summarize RightRotate(T, x):

p $\begin{cases} x.left = y.right \\ y.right.Parent = x \end{cases}$

Parent $\begin{cases} y.Parent = x.Parent \\ x.Parent.(left\ or\ right) = y \end{cases}$

x, y $\begin{cases} y.right = x \\ x.Parent = y \end{cases}$

13.3 Insertion

2 cases

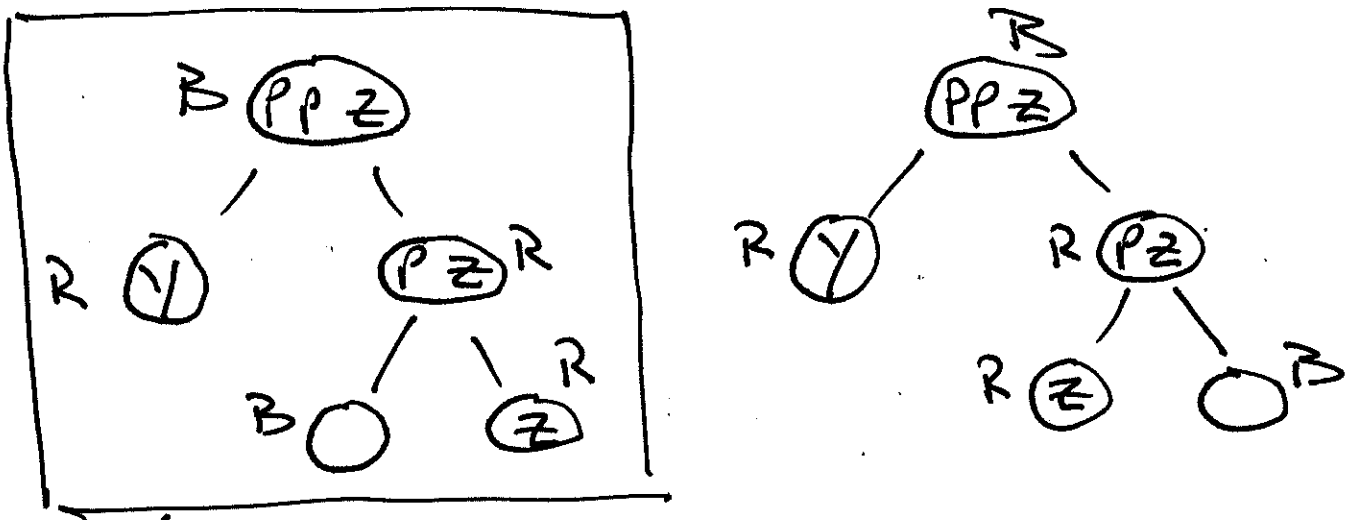
- z.parent is left child of Parent: cases 1, 2, 3

  let y be z.parent right sibling, i.e. z's uncle.

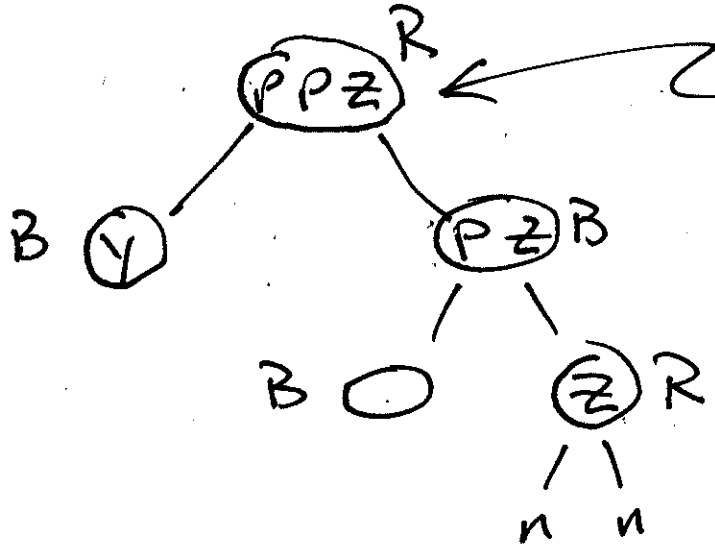
- z.parent is right child of its Parent: cases 4, 5, 6

  let y be z.parent left sibling i.e. z uncle.

Case 4 : y is Red



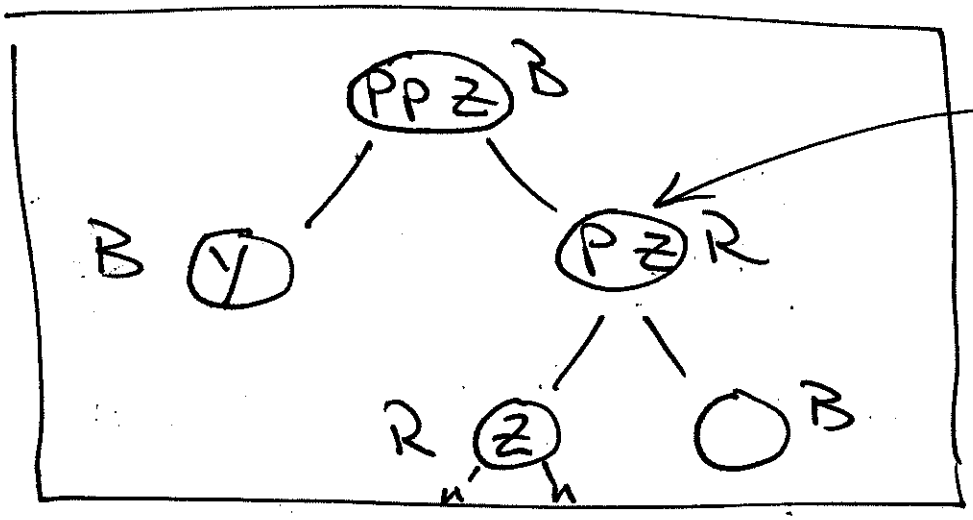
fix colors of y, Pz, PPz. let local variable z 'climb up' to PPz.



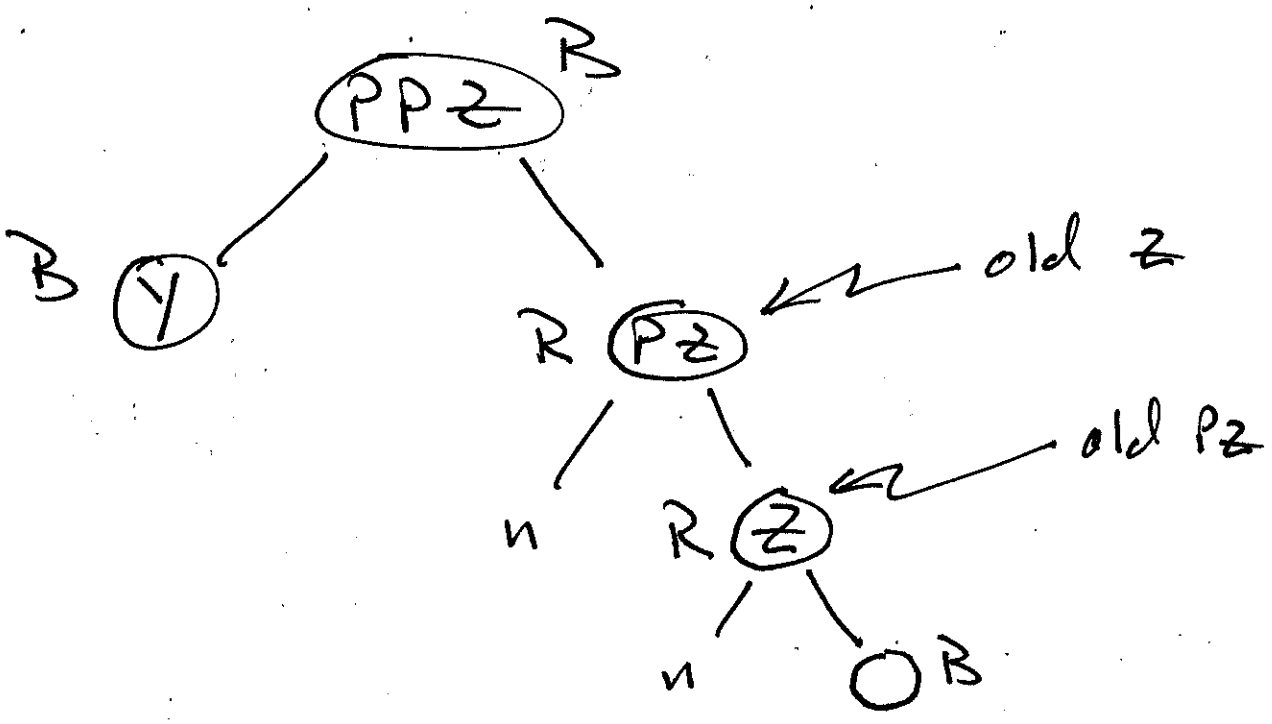
new z

may execute another iteration of while loop.

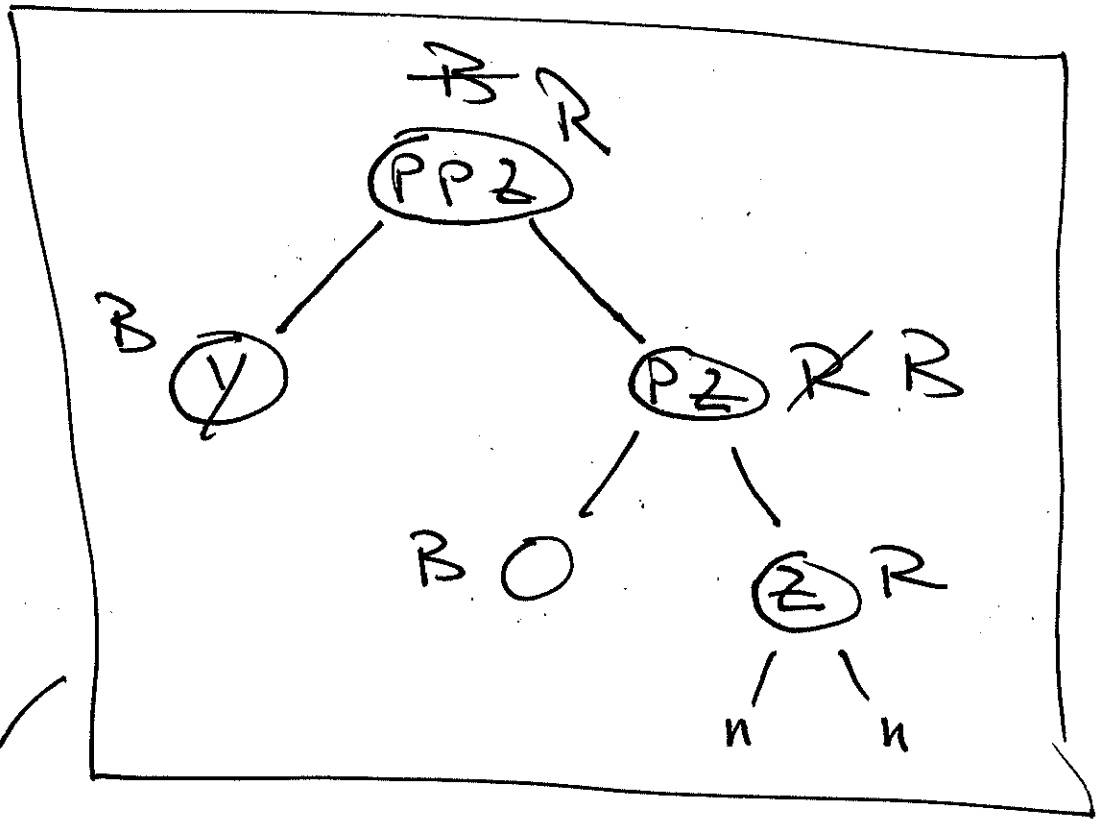
Case 5: y is Black, z == z.Parent.left



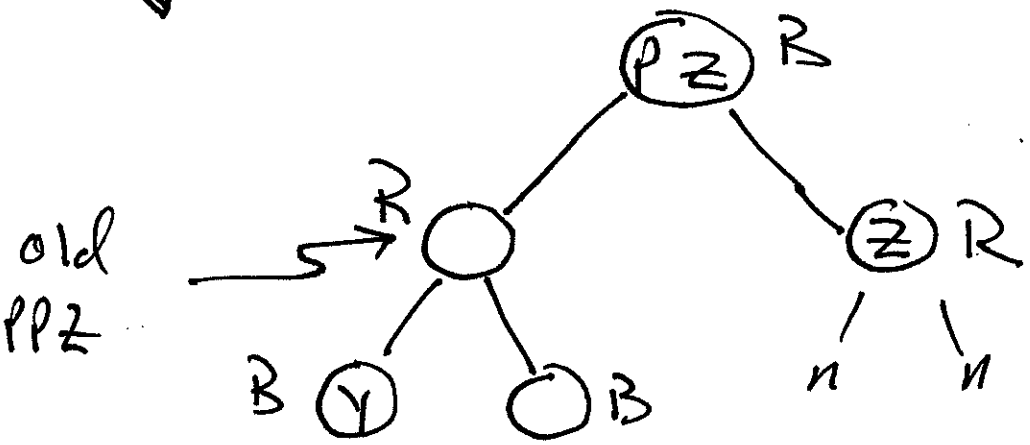
convert to case 6 (z == z.Parent.right)



Case 6: y is Black, z == z.parent.right



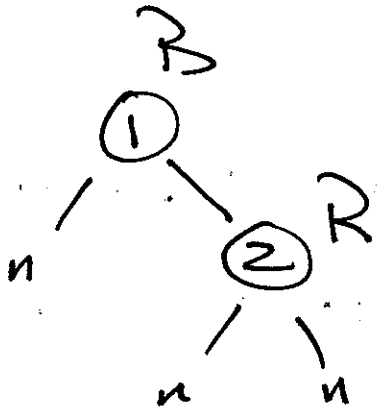
color Pz Black, PPz Red, then left Rotate about PPz.



Ex Insert 1, 2, 3, 4, 5 into an initially empty RBT.

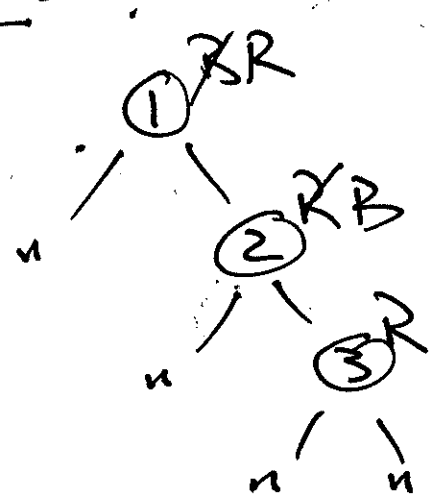
Insert 1:

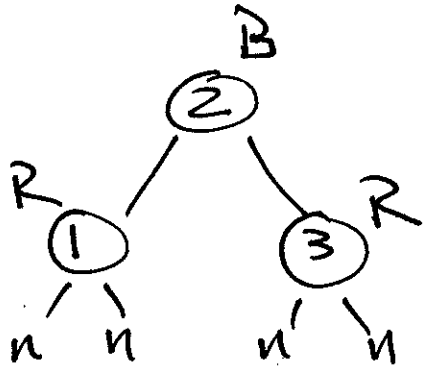
Insert 2:



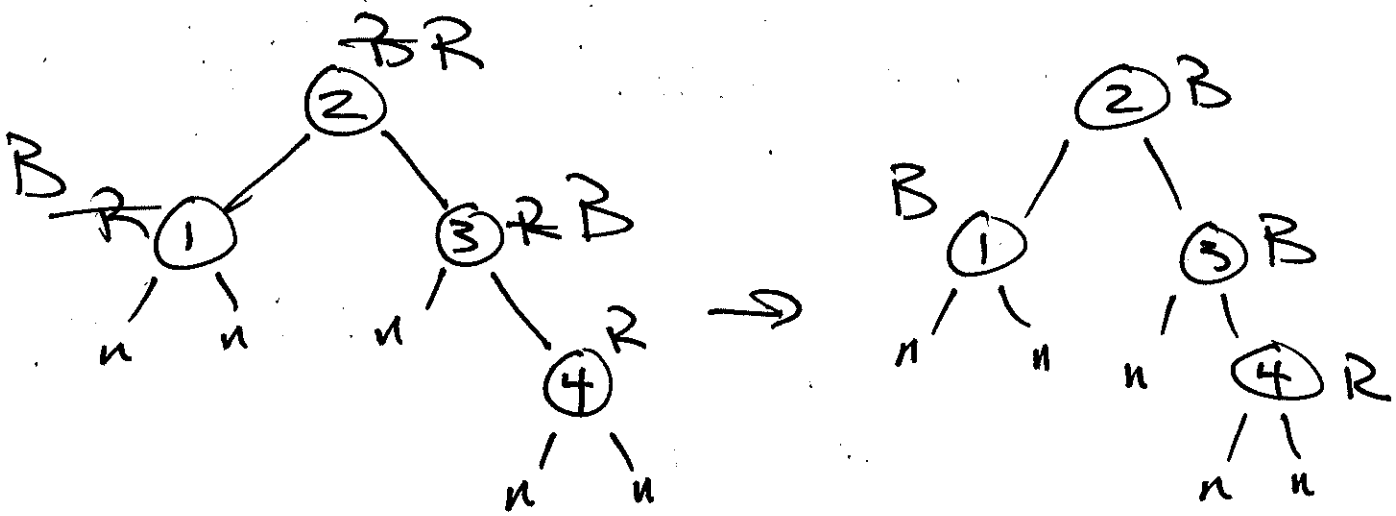
Insert 3:

Case 6

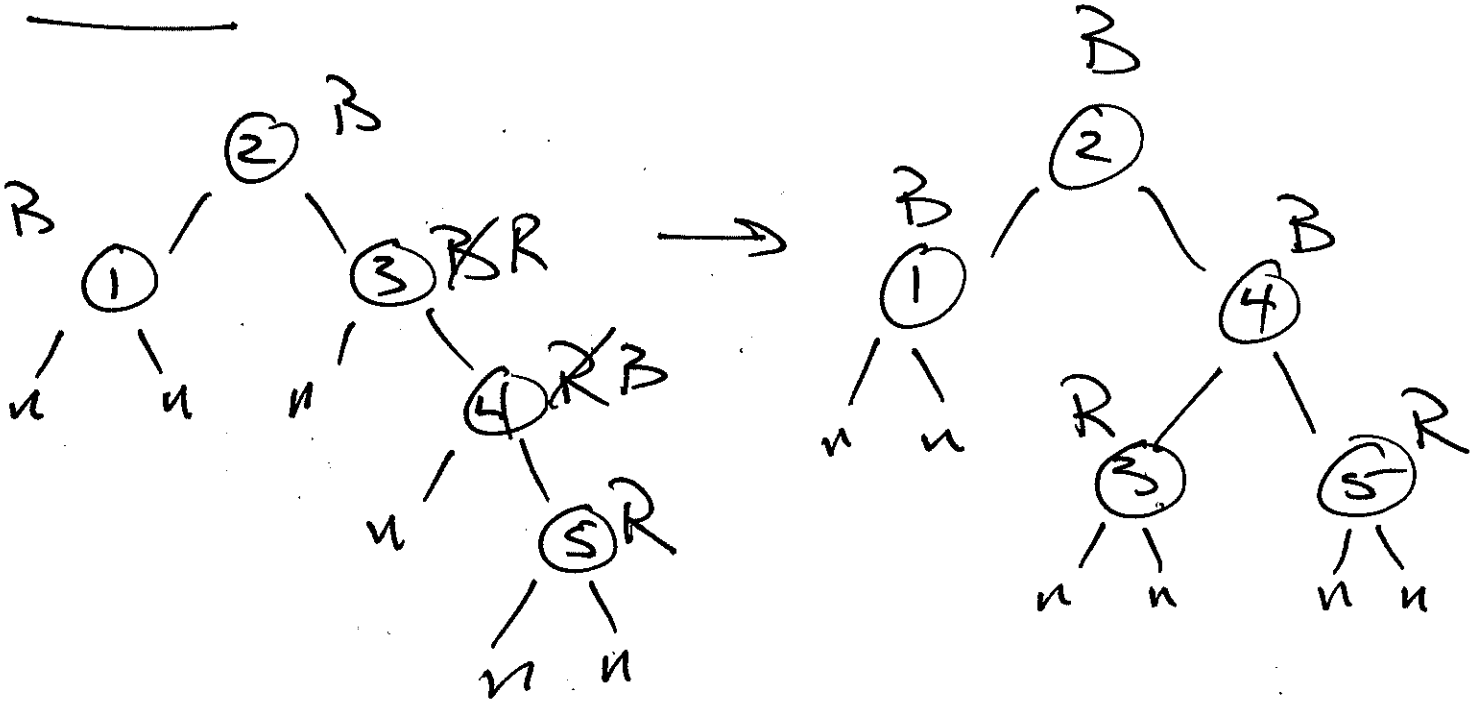




Insert 4 (case 4)



Insert 5



Case 6

CSE 101 3-10-25

- SETs : close Sun. 11:59 PM
- PA8 : ext 2 days.

Heaps & Priority Queues

A Priority Queue is an ADT that maintains a finite set S of records.

each record in S has an attribute called its key

$$x = (\cdot, \cdot, \cdot, \cdot, key) \in S$$

record — satellite data

$$S = \{\cdot, \cdot, x, \cdot, \cdot\}$$

S is a state in the Priority queue. $Key[x] = x.key$ defines the Priority of x in S.

Two types : max P.Q.
min P.Q.

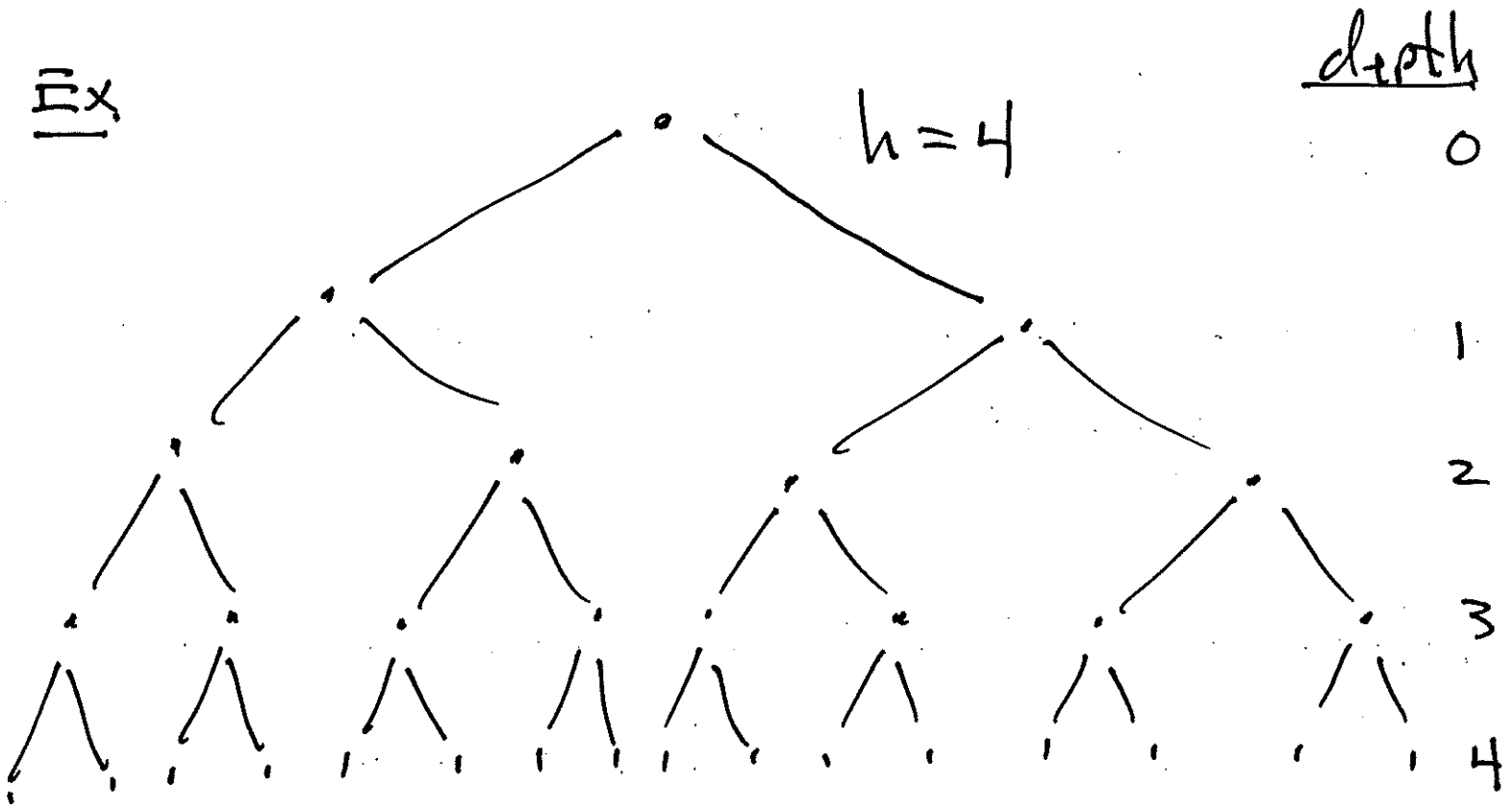
A max P.Q. has following operations

- Insert (S, x)
- Max (S): pure access fcn.
- ExtractMax (S): returns & deletes max key.
- IncreaseKey (S, x, k): if k > Key[x], changes Key[x] to k otherwise does nothing.

Heaps : chap 6

Defn A complete Binary Tree (CBT) is a B.T. where all leaves have same depth, all internal nodes have 2 children

Ex



observe

$$(\#\text{nodes at depth } d) = 2^d$$

if #nodes in tree is n and height of tree is h, then

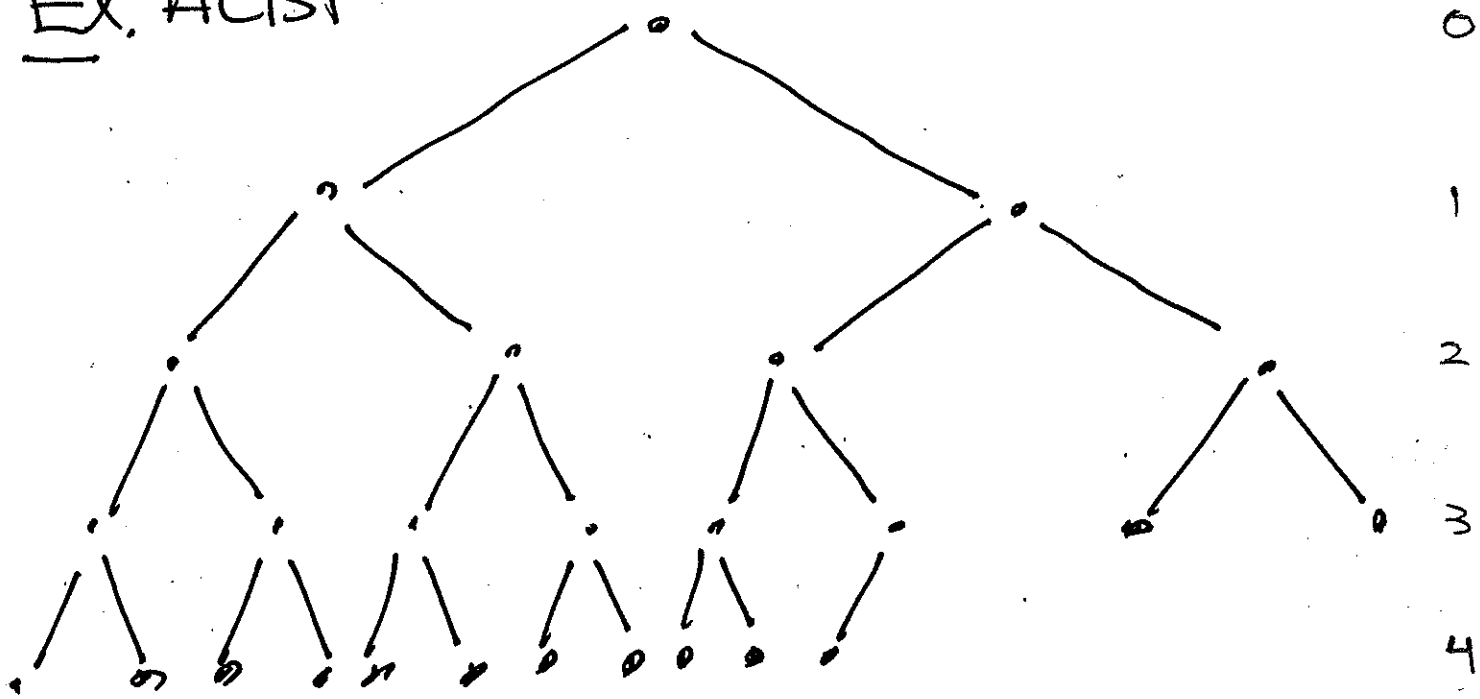
$$n = \sum_{d=0}^{h} 2^d = \frac{2^{h+1}-1}{2-1} = 2^{h+1}-1$$

An Almost Complete Binary Tree

(ACBT) is a B.T. that is filled at all levels, except possibly the deepest, which is filled from left to right.

depth

Ex. ACBT



The number of nodes n in an ACBT of height h satisfies

$$2^h - 1 < n \leq 2^{h+1} - 1$$

$$\therefore \quad 2^h \leq n < 2^{h+1}$$

$$\therefore \quad h \leq \lg(n) < h+1$$

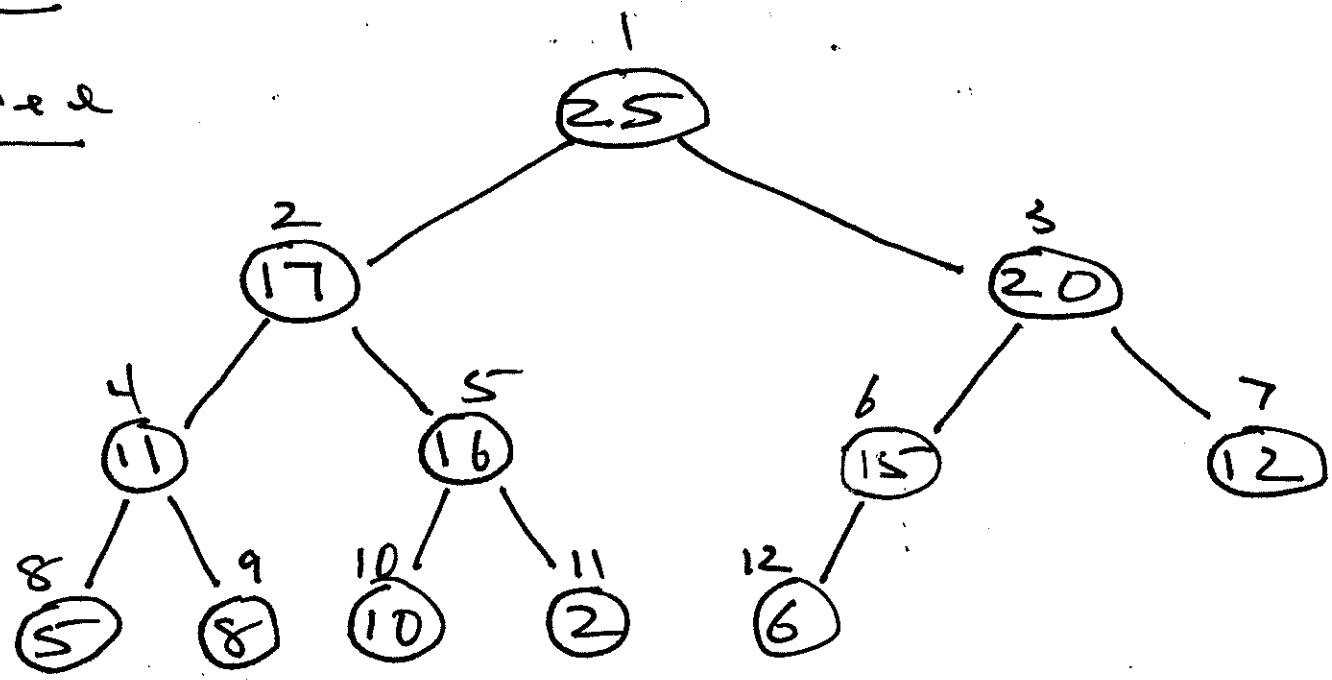
$\therefore$ $\boxed{h = \lfloor \lg n \rfloor}$

Binary Heaps (max, min)

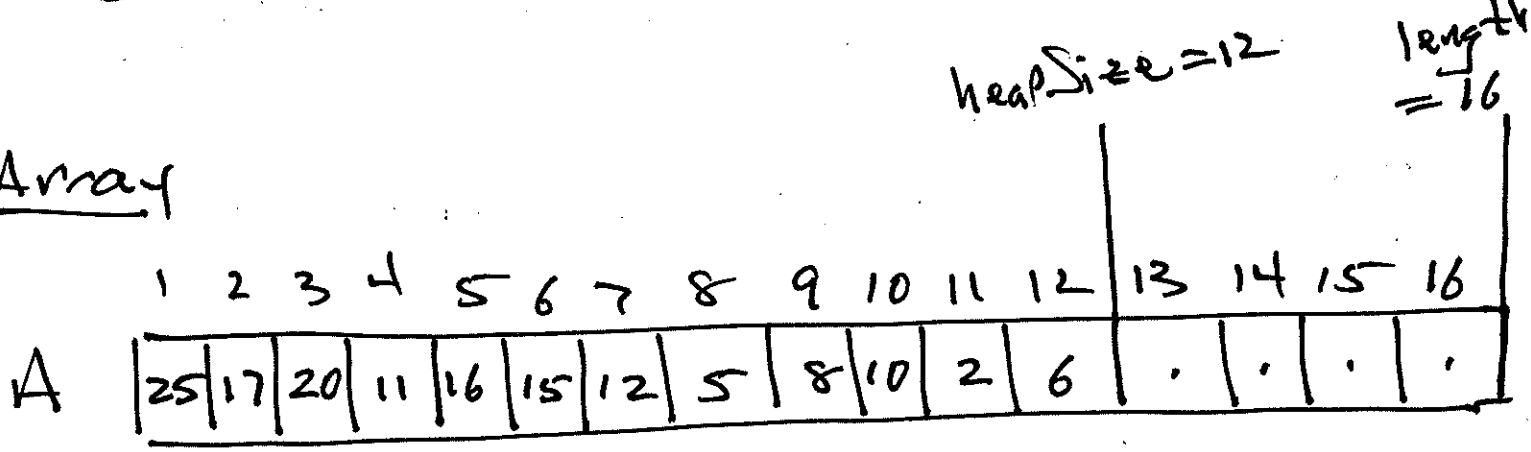
A heap is an array object that represents an ACRT.

Ex.

Tree



Array



the array has 2 attributes

- length[A]
- heapSize[A]

- helper functions

$$\text{Parent}(i) = \lfloor \tfrac{i}{2} \rfloor \quad (\text{for } i \geq 2)$$

$$\text{left}(i) = 2i$$

$$\text{right}(i) = 2i + 1$$

- heap properties

max $A[\text{Parent}(i)] \geq A[i]$ ✓

min $A[\text{Parent}(i)] \leq A[i]$