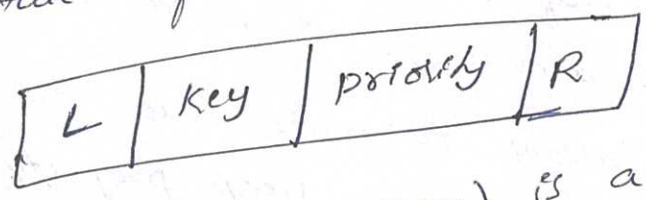
Treaps

- A treap is a data structure which combines binary tree and binary heap.
- A Treap is a binary tree that maintains simultaneously the property of binary search tree and heap.
- Treap is a balanced binary search tree like AVL and Red-Black trees.
- Treaps use Randomization and Binary Heap property to maintain balance with high probability.
- The time complexity of search, Insert and delete operation is O(Logn).
- The structure of a node in Treap is



- Formally a treap (tree + Heap) is a binary tree whose nodes maintain two values
  1) Key - follows standard BST ordering (left is smaller and right is greater)
  2) Priority - Randomly assigned value that follows Max-Heap or Min-Heap property.

Basic operations on Treap are

1. Searching
2. Insertion
3. Deletion:

- Treaps uses rotations to maintain Max-Heap property during Insertion and deletion.
- To search for a given key value, apply standard search process of BST with key value by ignoring the priorities.

Insertion in a Treap (Max Heap).

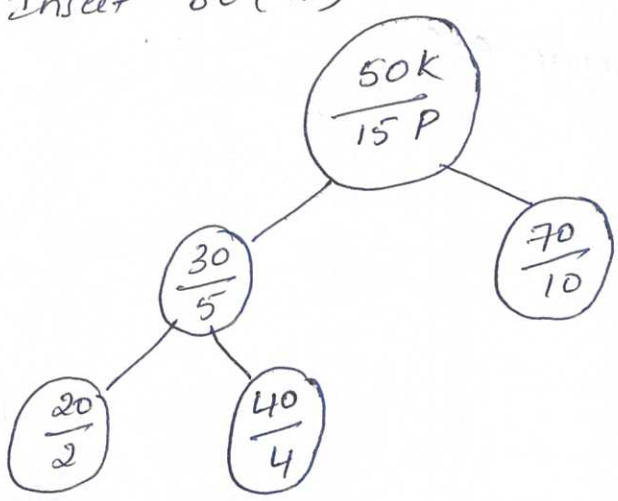
- To insert a new key x with priority y into the treap:

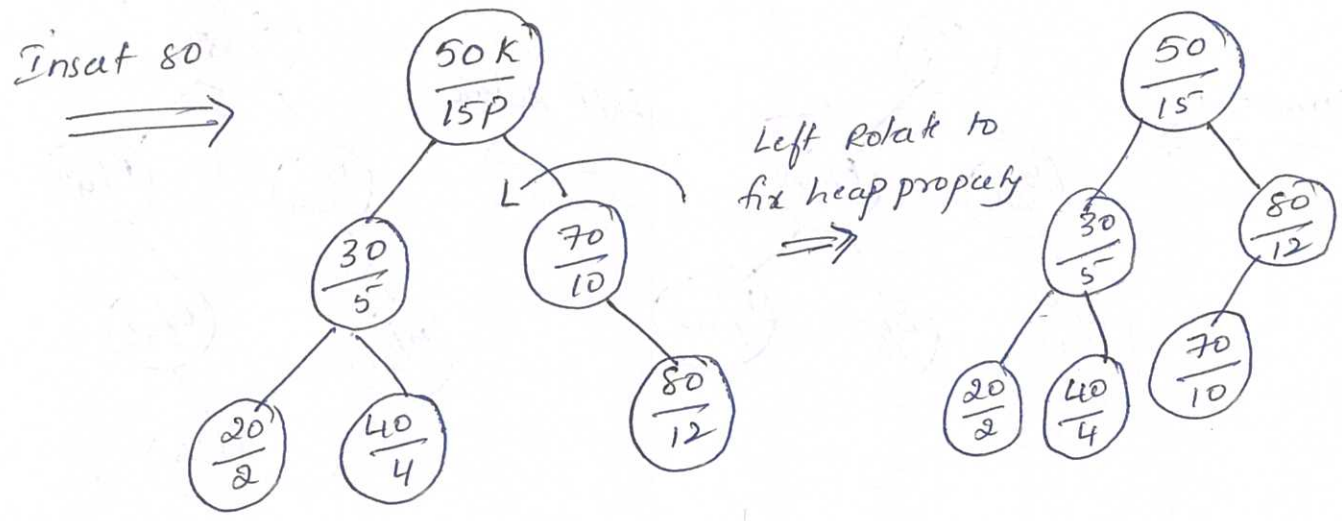
step1: Create a new node with key equals to x and priority value equals to a random value.

step 2; Perform standard BST insert. (Create a new node at the leaf position where the binary search determines a node for x should exist)

step3: Use Rotations to make sure that inserted node's priority follows max heap property. As long as x is not the root of the tree and has a larger priority number than its parent z, perform a tree rotation that reverses the parent-child relationship between x and z.

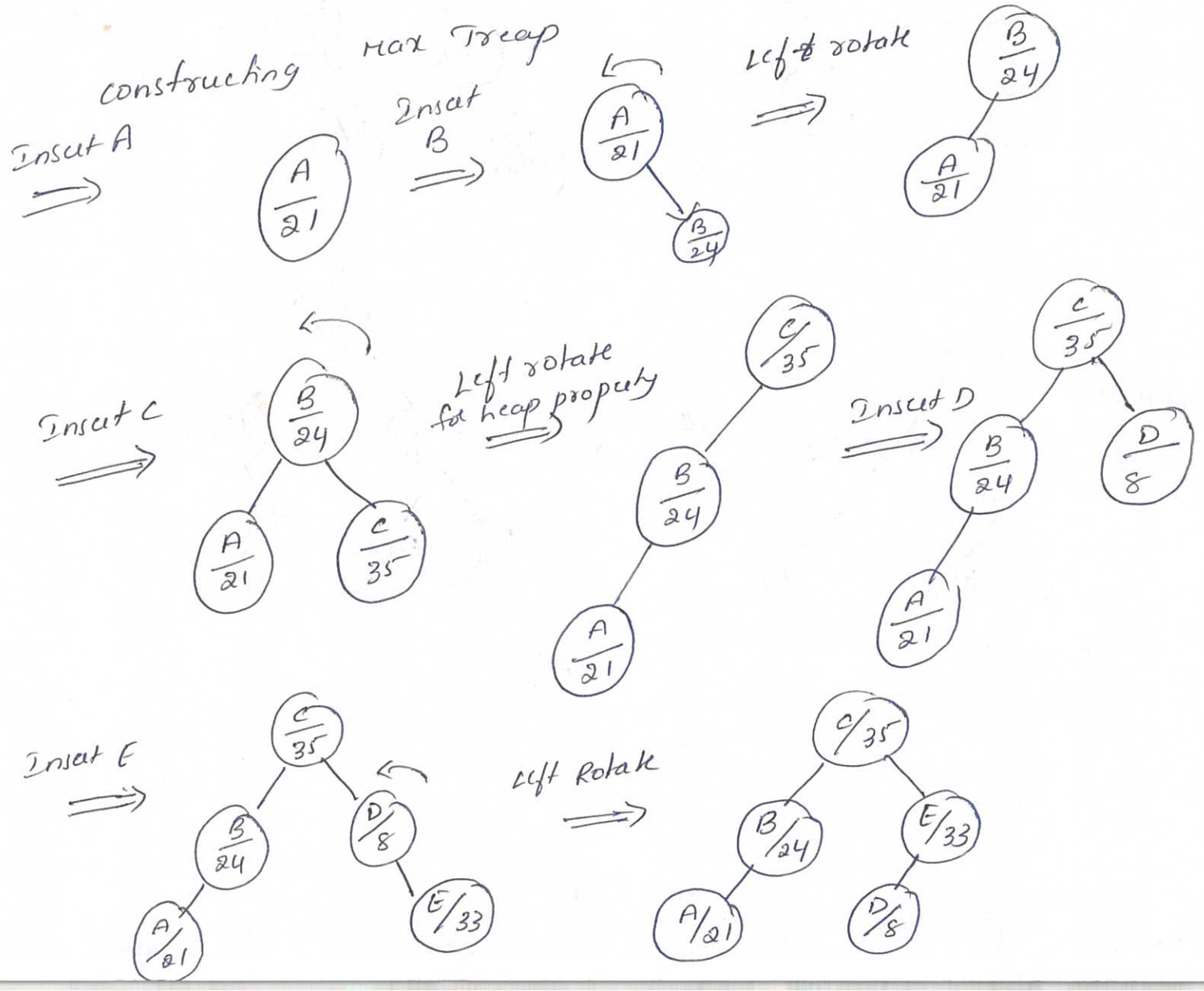
Ex Insert 80(12) into the following Treap.

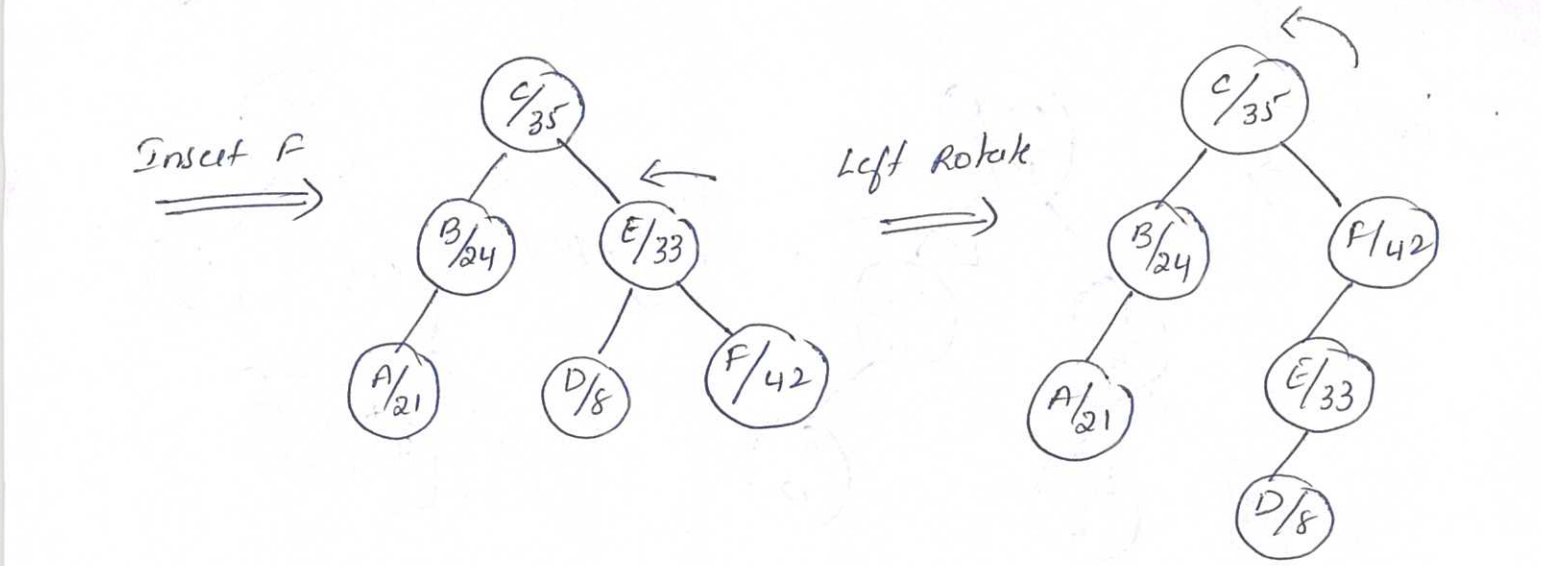


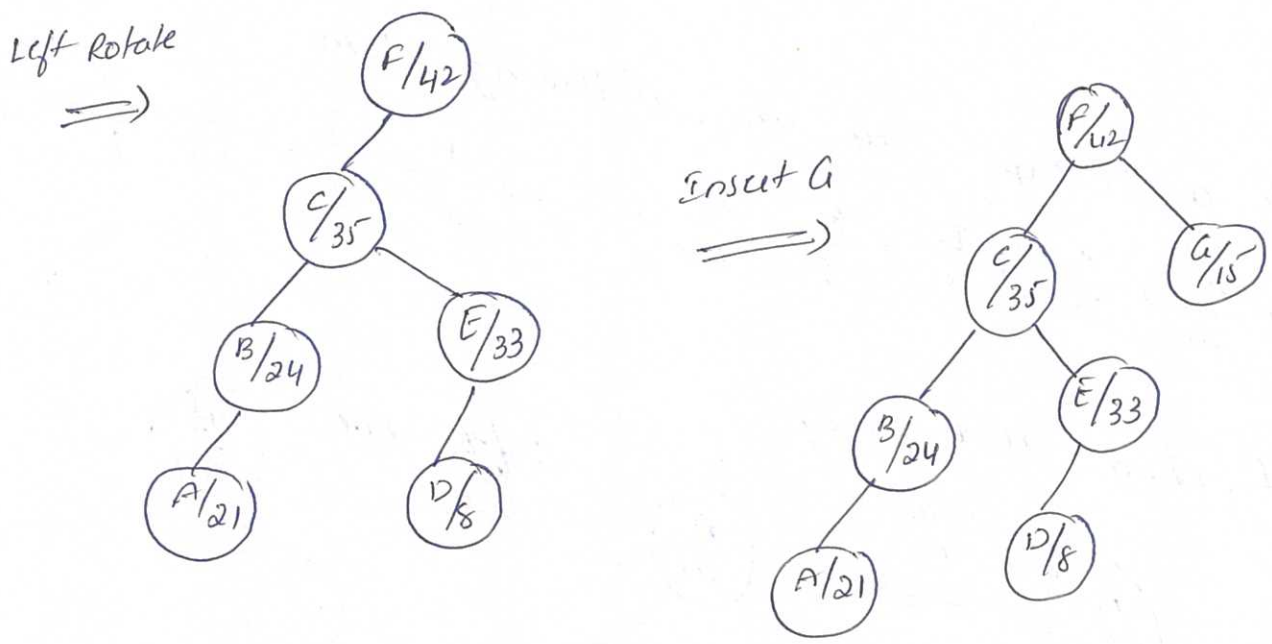


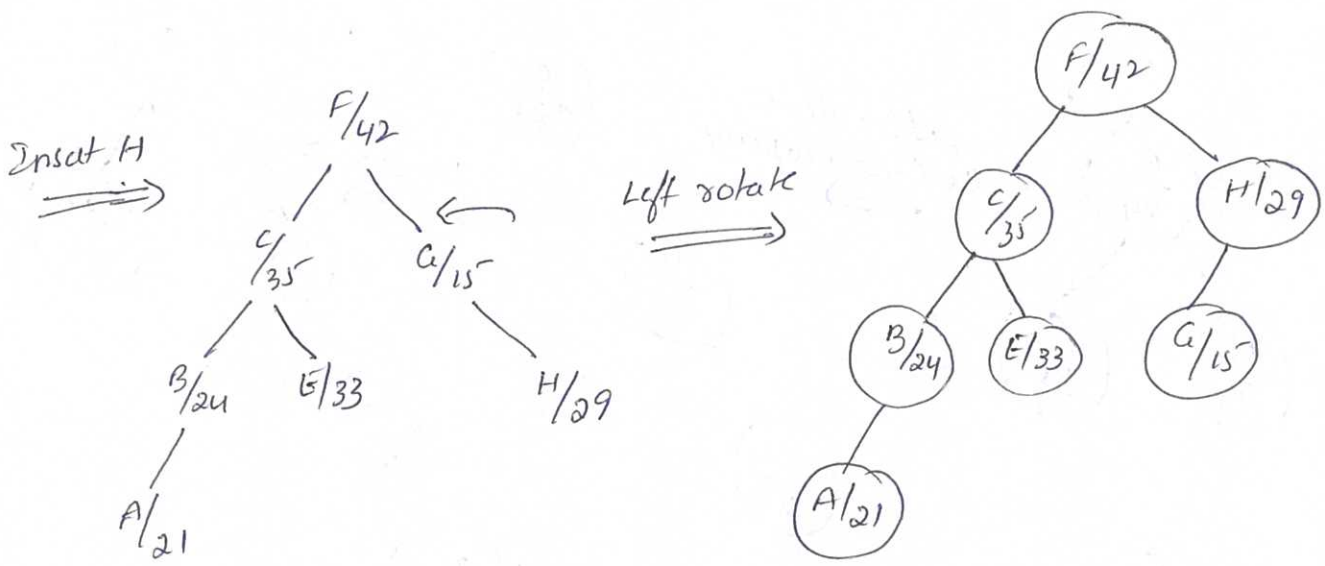
Ex Creat a Treap with the following data

$\frac{A}{21}, \frac{B}{24}, \frac{C}{35}, \frac{D}{8}, \frac{E}{33}, \frac{F}{42}, \frac{G}{15}, \frac{H}{29}, \frac{I}{35}, \frac{J}{13}, \frac{K}{9}, \frac{L}{16}$

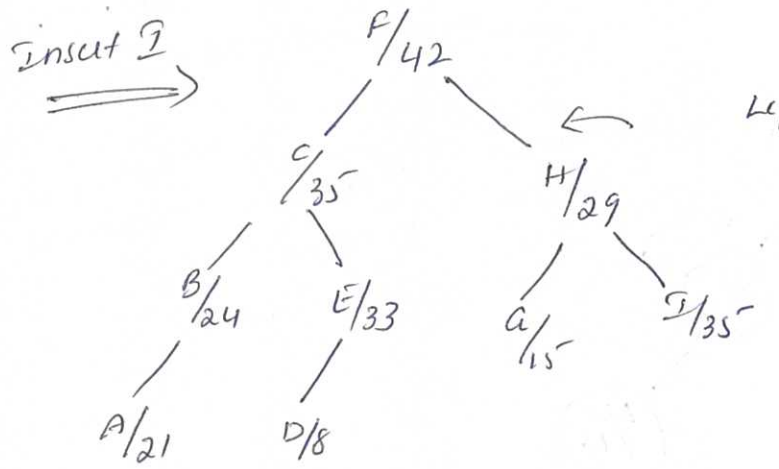



Insert F
C/35
B/24
E/33
A/21
D/8
F/42
Left Rotate
C/35
B/24
F/42
A/21
E/33
D/8

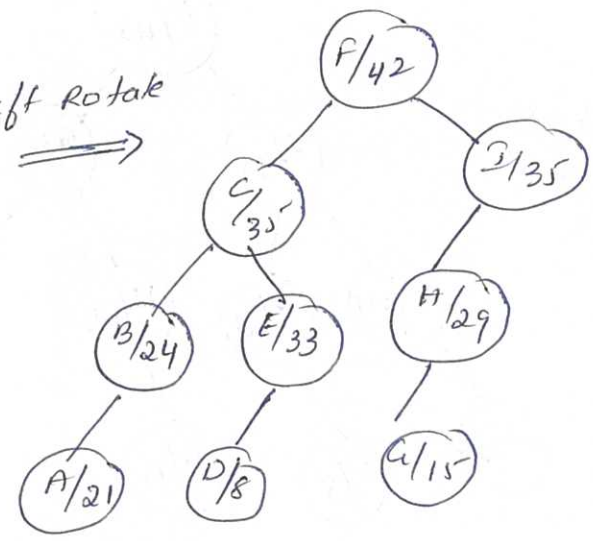

Left Rotate
F/42
C/35
B/24
E/33
A/21
D/8
Insert G
F/42
C/35
G/15
B/24
E/33
A/21
D/8


Insert H
F/42
C/35
G/15
B/24
E/33
H/29
A/21
Left rotate
F/42
C/35
H/29
B/24
E/33
G/15
A/21

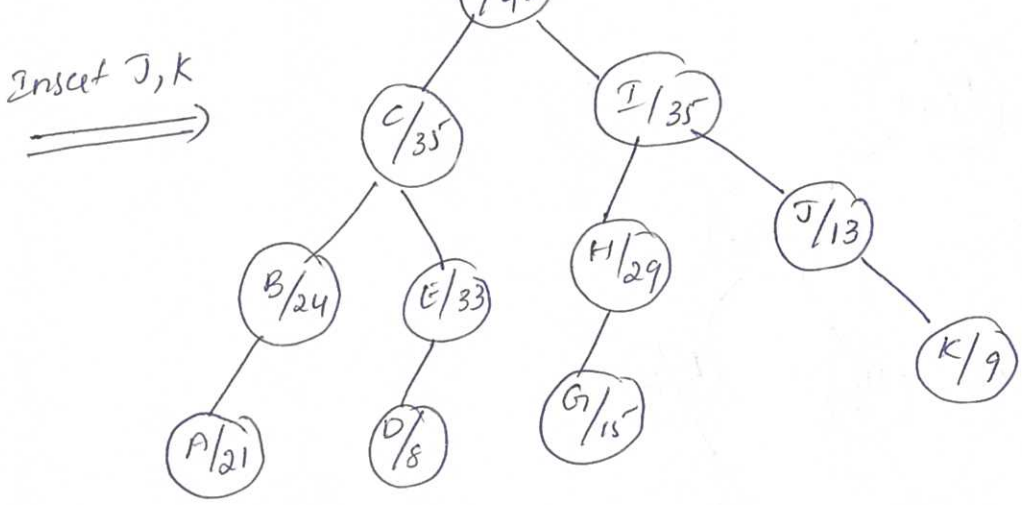
Insert I



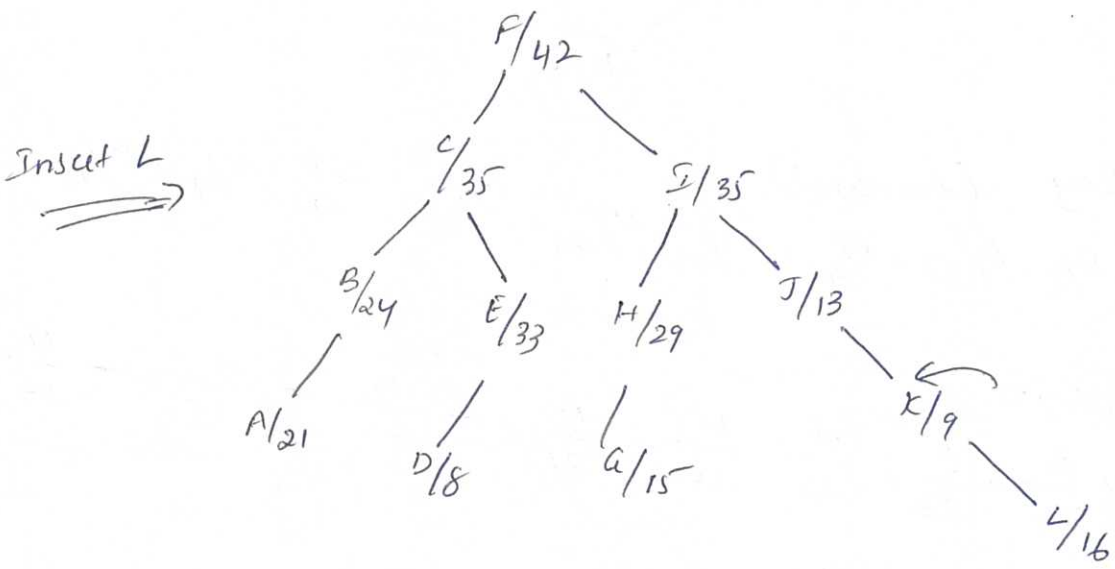
Left Rotate



Insert J, K



Insert L



Deletion operation in Treap

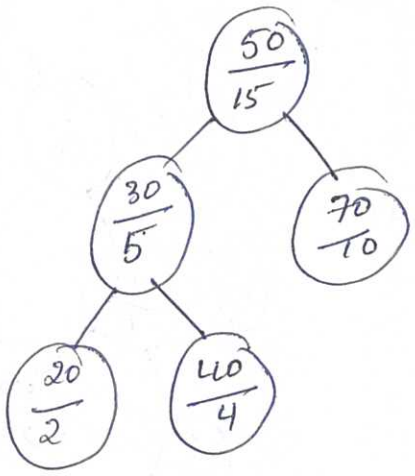
To delete an element first search for the element.

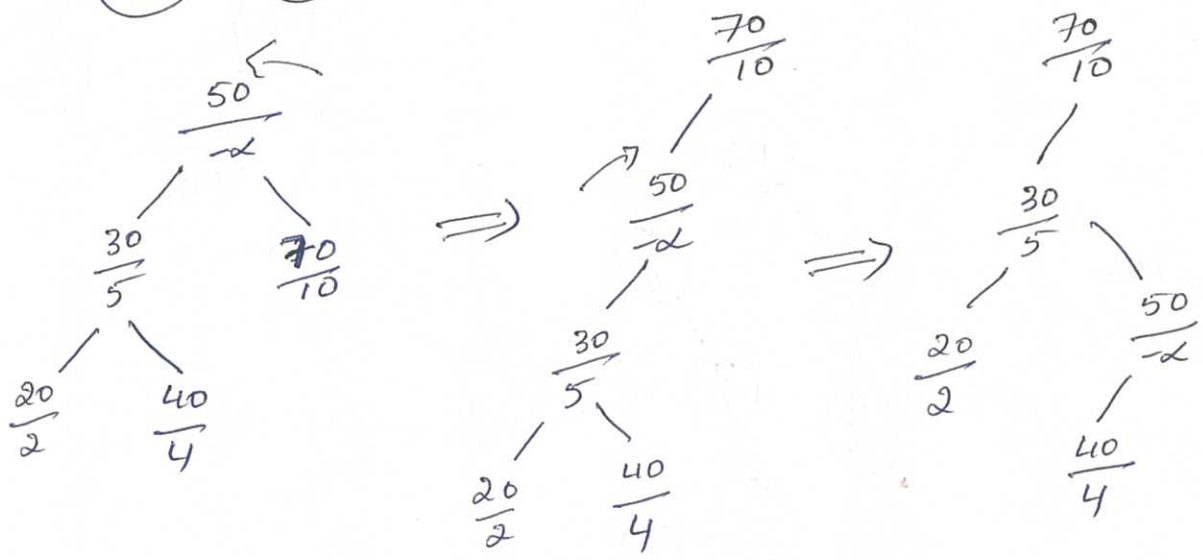
- If the element is not found, deletion is not possible.
- If element is found

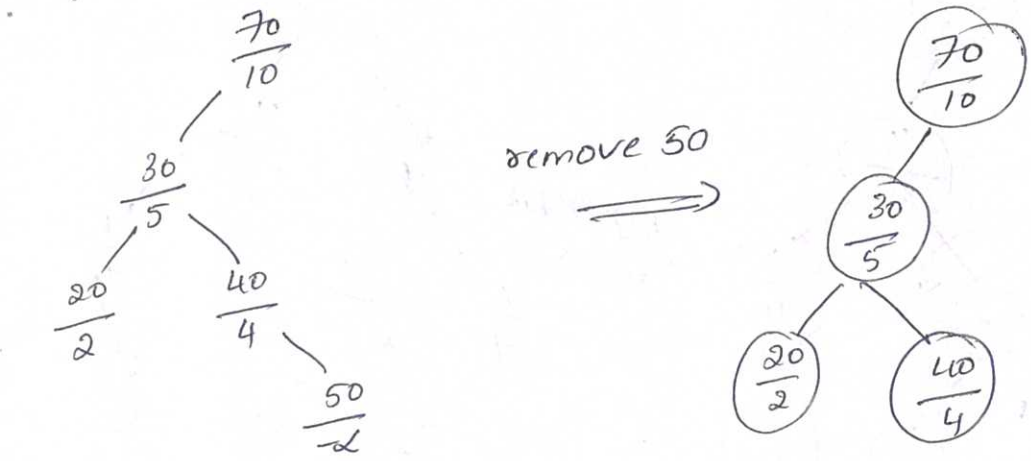
case 1: If the node is leaf, delete it.

case 2: otherwise replace the node's priority with $-\infty$ (on max Treap) or $+\infty$ (on min Treap) and perform appropriate rotations to bring the node down to a leaf. Go to case 1.

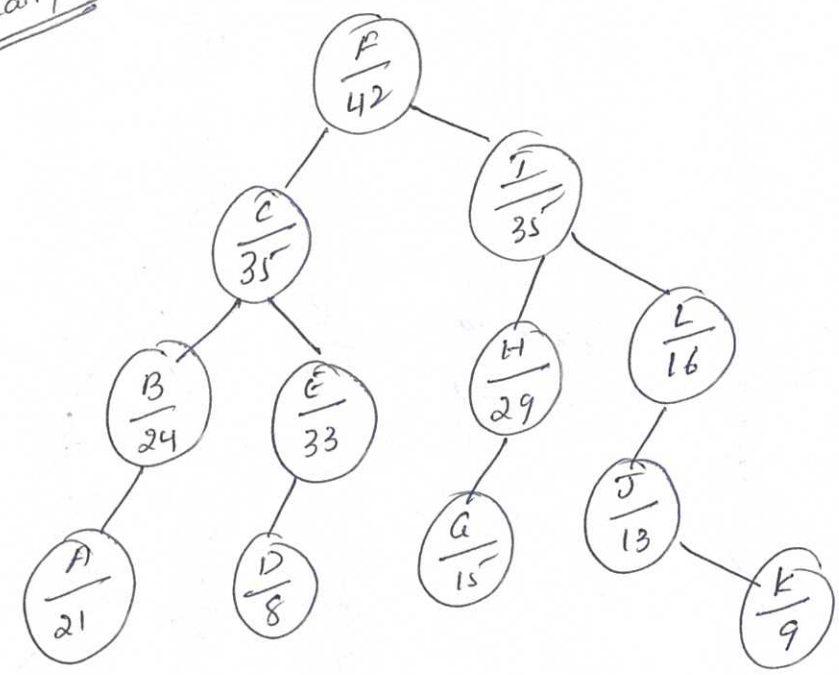
Ex: Delete 50 from the following max Treap.

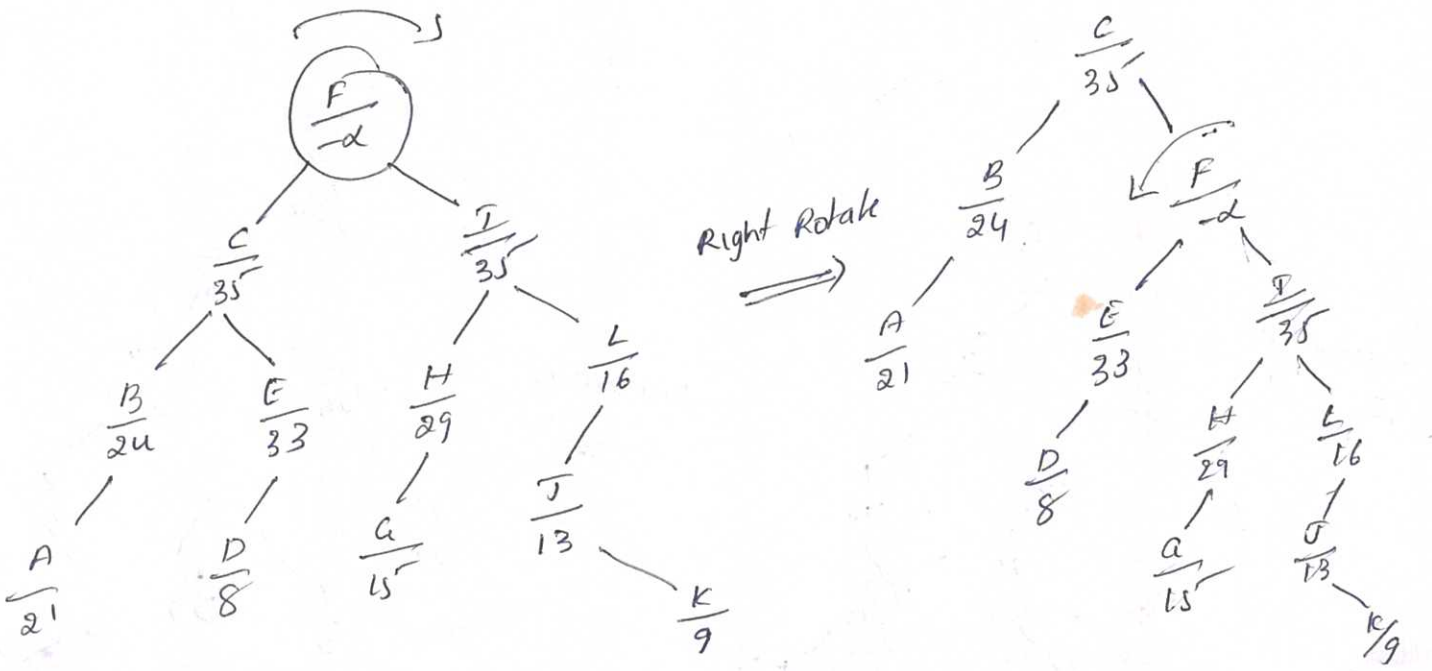


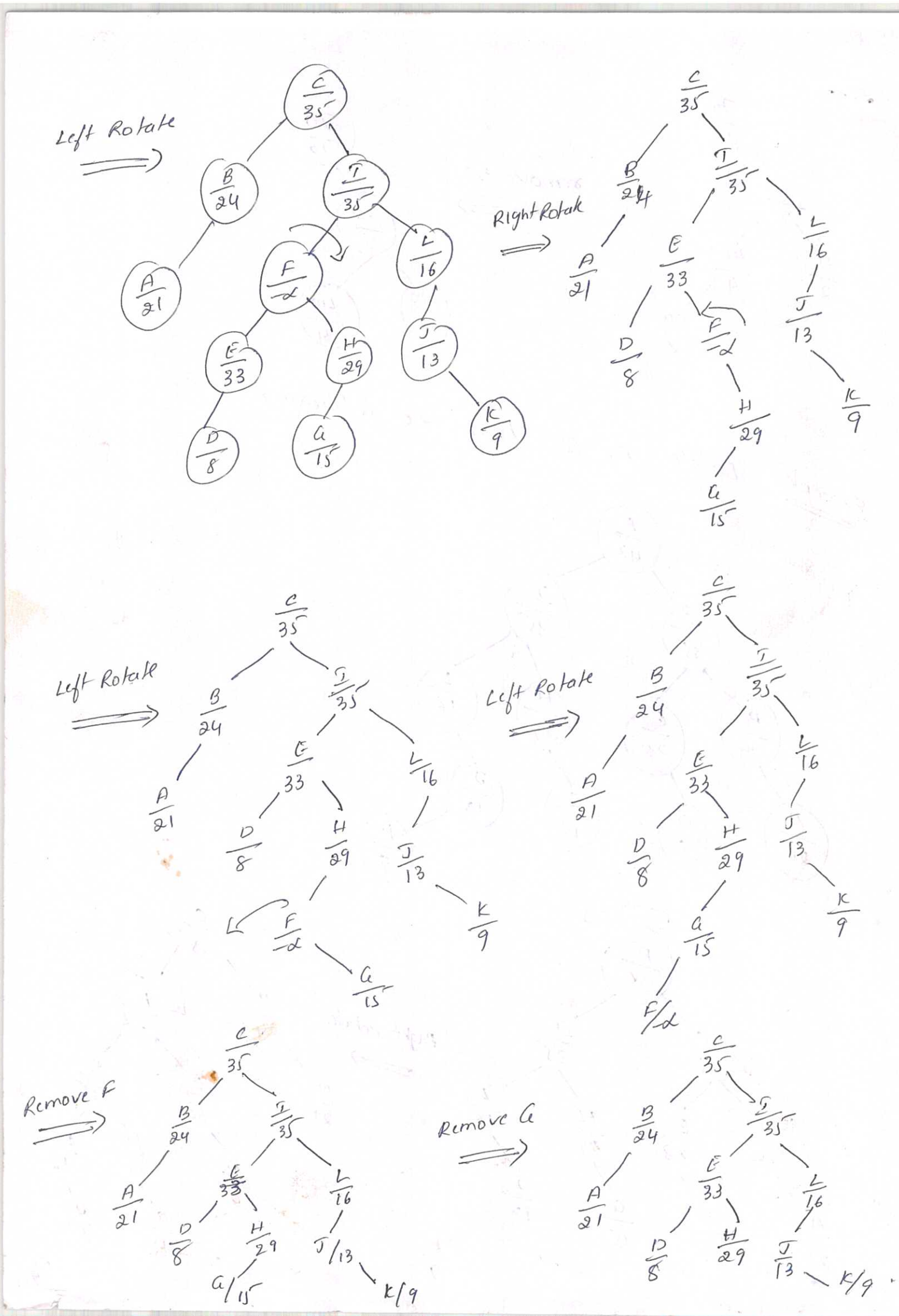




Example Delete F, A from the following Tree.






Left Rotate
C 35
B 24
I 35
A 21
F α
L 16
E 33
H 29
J 13
D 8
G 15
K 9
Right Rotate
C 35
B 24
I 35
A 21
E 33
L 16
D 8
F α
J 13
H 29
K 9
G 15
Left Rotate
C 35
B 24
I 35
A 21
E 33
L 16
D 8
H 29
J 13
F α
K 9
G 15
Left Rotate
C 35
B 24
I 35
A 21
E 33
L 16
D 8
H 29
J 13
K 9
G 15
F α
Remove F
C 35
B 24
I 35
A 21
E 33
L 16
D 8
H 29
J 13
G 15
K 9
Remove G
C 35
B 24
I 35
A 21
E 33
L 16
D 8
H 29
J 13
K 9
F α