TREAP DOCUMENTATION

1. **INSERTION**

1) Create new node with key equals to x and value equals to a random value.

2) Perform standard BST insert.

3) A newly inserted node gets a random priority, so **Max-Heap** property may be violated.. Use rotations to make sure that inserted node's priority follows max heap property.

During insertion, we recursively traverse all ancestors of the inserted node.

a) If new node is inserted in left subtree and root of left subtree has higher priority, perform **right rotation.**

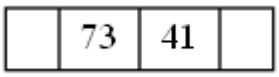
b) If new node is inserted in right subtree and root of right subtree has higher priority, perform **left rotation.**

EXAMPLE—

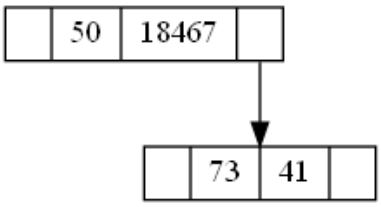
Node structure :

Left child	key	priority	Right child

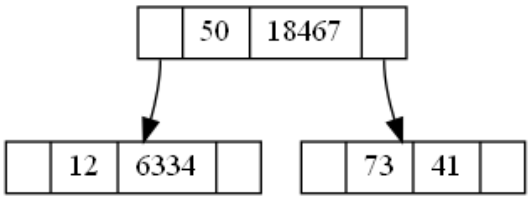
Insert>73



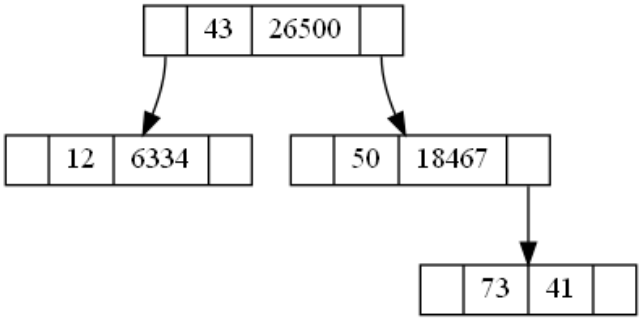
Insert>50



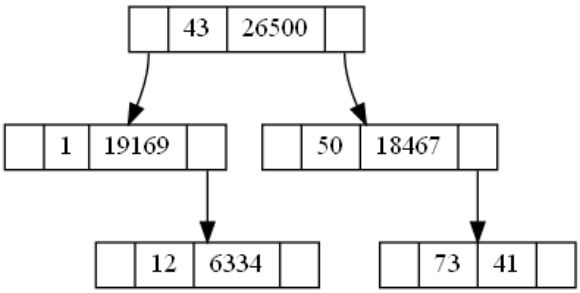
Insert>12



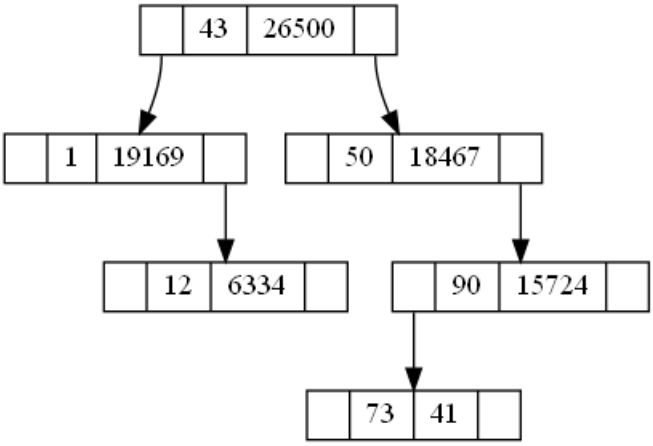
Insert>43



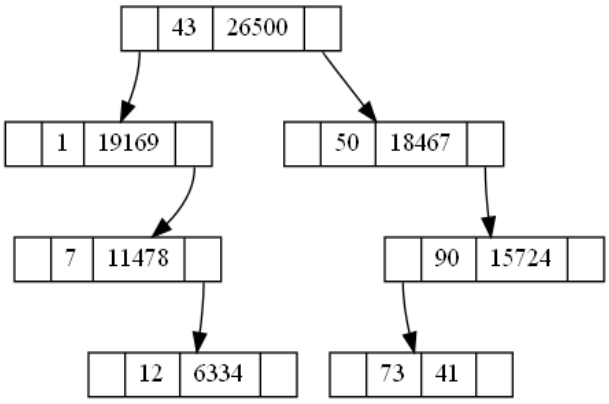
Insert>1



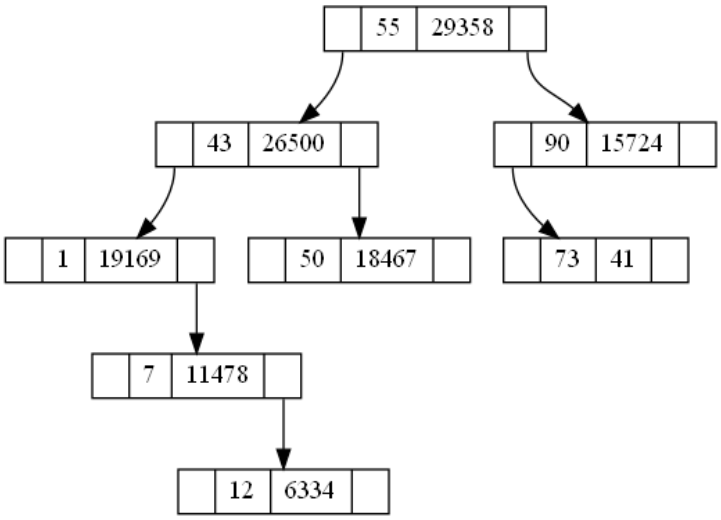
Insert>90



Insert>7



insert>55



2. <u>DELETION</u>

1) If node is a leaf, delete it.
2) If node has one child NULL and other as non-NULL, replace node with the non-empty child.
3) If node has both children as non-NULL, find max of left and right children.

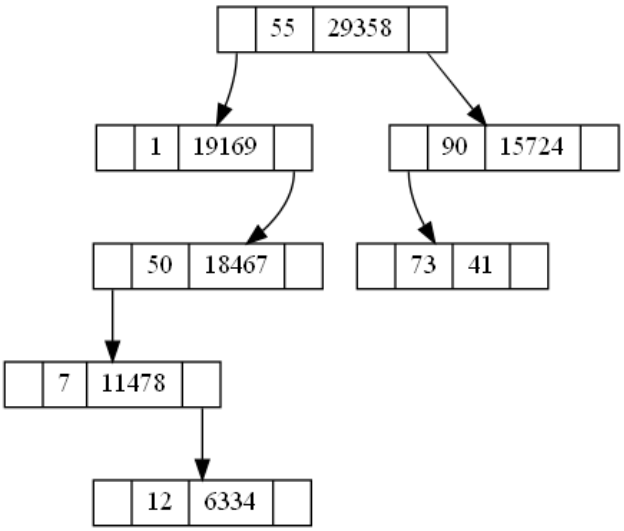
a) If priority of right child is greater, **perform left rotation at node**

b) If priority of left child is greater, **perform right rotation at node.**

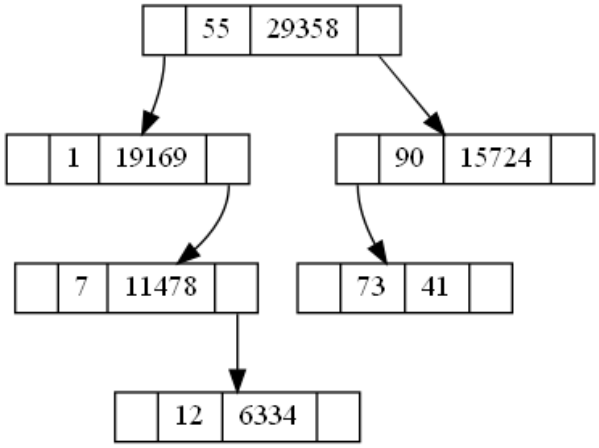
step 3 will move the node to down so that we end up with either case 1 or case 2.

EXAMPLE—

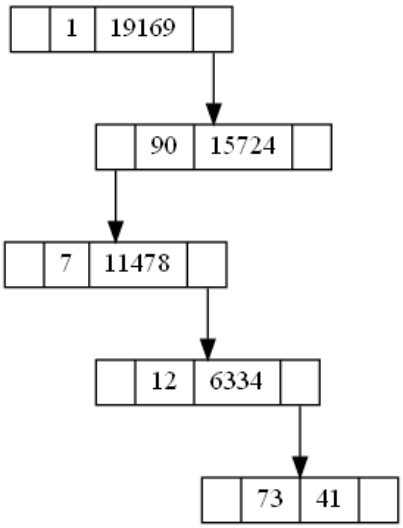
Delete>43



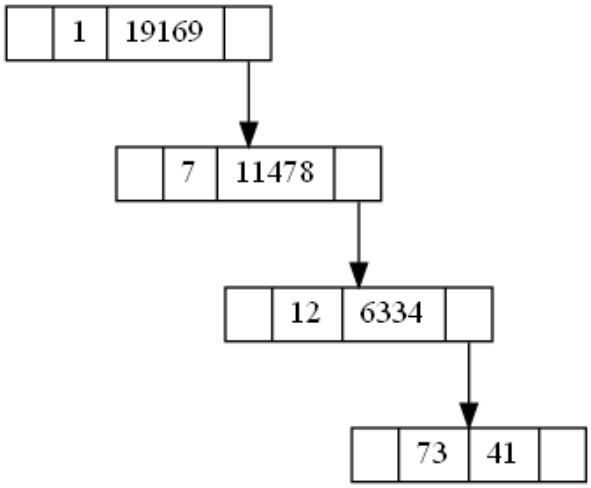
Delete>50



Delete>55



Delete>90



3. **SEARCH**

Same as BST search. Priority is not considered for search.

1.If tree is emty return false

2.It is a simple search operation in which the key will be compared to value of nodes if key == node_value is true then return true.

3.If key > node_value then go to right child

4.If key < node_value then go to left child

EXAMPLE—

Search in : 73 50 12 43 1 90 7 (above insertion example)

```
Operations on Treap
1.Insert Element
2.Delete Element
3.Print Treap
4.Search
5.Quit
Enter your choice : 4

----------------------------
Enter element to search inside treap :43

Element found

----------------------------

Operations on Treap
1.Insert Element
2.Delete Element
3.Print Treap
4.Search
5.Quit
Enter your choice : 4

----------------------------
Enter element to search inside treap :100

Element not found

----------------------------
```

4. **PRINT TREAP**

   In this function using the PREORDER TRAVERSAL AND GRAPHVIZ the treap is printed using TREAP.gv file in TREAP_Tree.png image file.