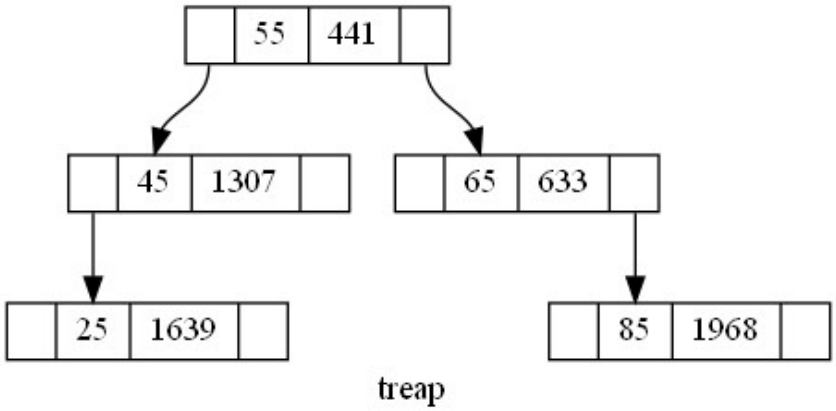
DS ASSIGNMENT 3

TREAP

Roll number : 214101022

INTRODUCTION

- Treap is the combination of binary search tree and binary heap.
- Every node of Treap maintain two values:
  1) key : Follows binary search tree ordering.
  2) priority : A randomly generated value which follows heap property.



treap

- Left values are key values which follows BST property.
- Right values are priorities which follows min-heap property.

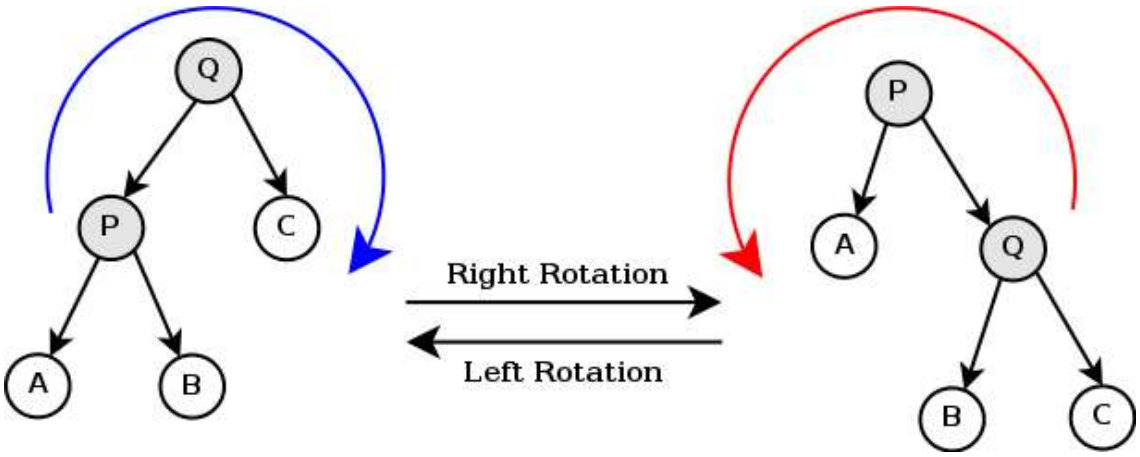
NODE STRUCTURE

Left pointer	Key value	Priority	Right pointer

- Left pointer points to left sub-tree root.
- Key value follows BST property.
- Priority follows min-heap property.
- Right pointer points to right sub-tree root.

ROTATIONS

- There are only two rotations in Treap. (Rotate right , Rotate left)
- Rotations in Treap depends on priority values.
- To keep the min-heap property, treap perform some rotations after insertion and deletion of nodes.



(1) Rotate right:

- If the priority of parent node is more than left child then we perform Rotate right operation.
- Function will create some temp pointers to point P and Q.
- In the above left hand side graph, if the priority of Q is more than the priority of P then it will rotate right and form the graph in the right hand side.
- Rearranging of pointers is given in the graph.

(2) Rotate left:

- If the priority of parent node is more than right child then we perform Rotate left operation.
- Function will create some temp pointers to point P and Q.
- In the above right hand side graph, if the priority of P is more than the priority of Q then it will rotate left and form the graph in the left hand side.
- Rearranging of pointers is given in the graph.

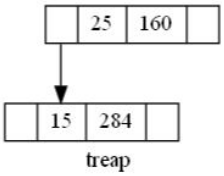
INSERTION

- Insertion happens in binary search tree fashion.
- If the key value is less than the current temp node value then it will move to left sub-tree.
- If the key value is more than the current temp node value then it will move to right sub-tree.
- New node will be inserted as leaf node.
- If the priority of the new node is less than its parent node then, rotation takes place.
- If the new node is left child of the parent then right rotation takes place.
- If the new node is right child of the parent then left rotation takes place.
- Rotations continues till parent nodes priority is less than ne nodes.
- Initially tree is empty. Lets add key values one by one and see how treap forms.

Example,

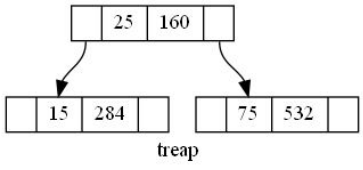
Lets insert 25 and 15 in empty tree.

```
Press the number to perform operation and press enter:
1. Insertion
2. Deletion
3. Search
4. Print treap
Any other number to exit

1
Enter the value you want to insert : 15
Key is inserted

Press the number to perform operation and press enter:
1. Insertion
2. Deletion
3. Search
4. Print treap
Any other number to exit

4
Image file is generated. Check the folder
```



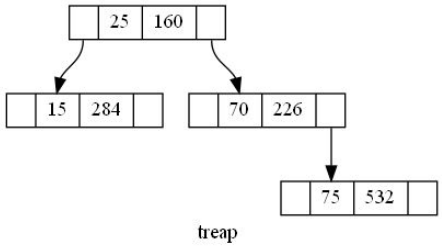
Insert 75

```
Press the number to perform operation and press enter:
1. Insertion
2. Deletion
3. Search
4. Print treap
Any other number to exit

1
Enter the value you want to insert : 75
Key is inserted

Press the number to perform operation and press enter:
1. Insertion
2. Deletion
3. Search
4. Print treap
Any other number to exit

4
Image file is generated. Check the folder
```



Insert 70

```
Press the number to perform operation and press enter:
1. Insertion
2. Deletion
3. Search
4. Print treap
Any other number to exit

1
Enter the value you want to insert : 70
Key is inserted

Press the number to perform operation and press enter:
1. Insertion
2. Deletion
3. Search
4. Print treap
Any other number to exit

4
Image file is generated. Check the folder
```

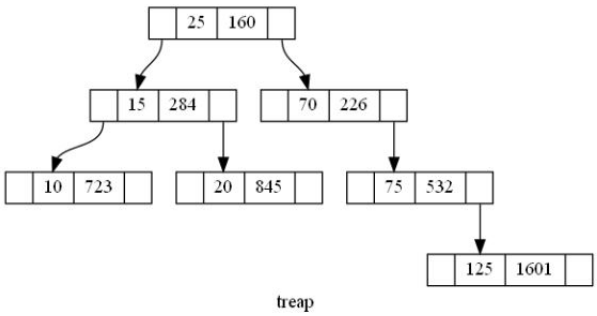


treap

We can right rotation here. 70 should be left of 75 as per BST property. But the priority value of 70 is less. That's why rotation takes place here.

Insert 125, 20, 10.

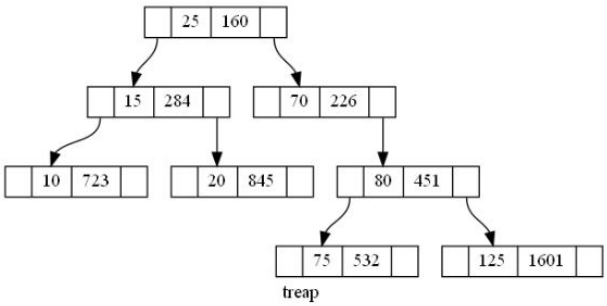
```
Press the number to perform operation and press enter:
1. Insertion
2. Deletion
3. Search
4. Print treap
Any other number to exit

1
Enter the value you want to insert : 10
Key is inserted

Press the number to perform operation and press enter:
1. Insertion
2. Deletion
3. Search
4. Print treap
Any other number to exit

4
Image file is generated. Check the folder
```



treap

Insert 80

```
Press the number to perform operation and press enter:
1. Insertion
2. Deletion
3. Search
4. Print treap
Any other number to exit

1
Enter the value you want to insert : 80
Key is inserted

Press the number to perform operation and press enter:
1. Insertion
2. Deletion
3. Search
4. Print treap
Any other number to exit

4
Image file is generated. Check the folder
```

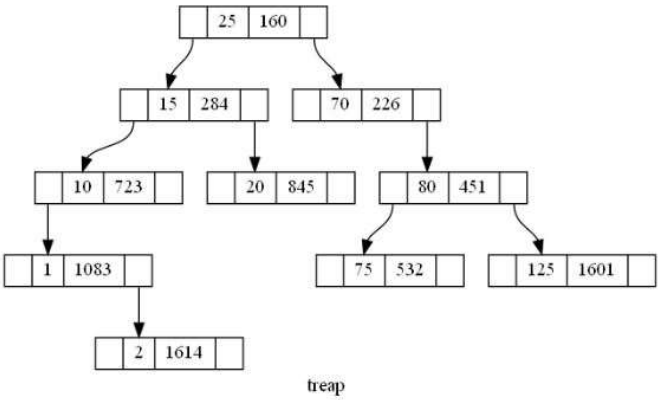


treap

We can right and left rotation here. 80 should be left of 125 as per BST property. But the priority value of 80 is less. That's why rotation takes place here. 125 goes right of 80. Then 75 goes left of 80.

Insert 2, 1

```
Press the number to perform operation and press enter:
1. Insertion
2. Deletion
3. Search
4. Print treap
Any other number to exit

1
Enter the value you want to insert : 1
Key is inserted

Press the number to perform operation and press enter:
1. Insertion
2. Deletion
3. Search
4. Print treap
Any other number to exit

4
Image file is generated. Check the folder

Press the number to perform operation and press enter:
```

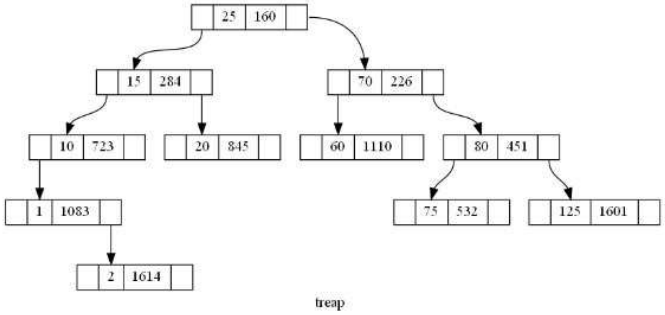


treap

Right rotation takes place here at 1. As 1 is inserted after 2 it should be at left of 2. But because of the priority value rotation takes place

Insert 60

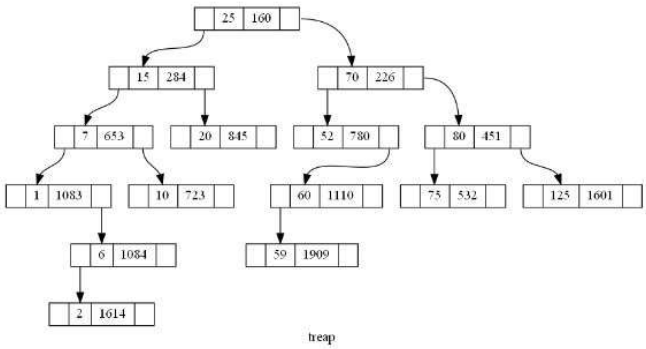
```
Press the number to perform operation and press enter:
1. Insertion
2. Deletion
3. Search
4. Print treap
Any other number to exit

1
Enter the value you want to insert : 60
Key is inserted

Press the number to perform operation and press enter:
1. Insertion
2. Deletion
3. Search
4. Print treap
Any other number to exit

4
Image file is generated. Check the folder
```



treap

Insert 52, 59, 7, 6. This is our final treap after inserting 15 keys.

```
Press the number to perform operation and press enter:
1. Insertion
2. Deletion
3. Search
4. Print treap
Any other number to exit

1
Enter the value you want to insert : 6
Key is inserted

Press the number to perform operation and press enter:
1. Insertion
2. Deletion
3. Search
4. Print treap
Any other number to exit

4
Image file is generated. Check the folder
```



treap

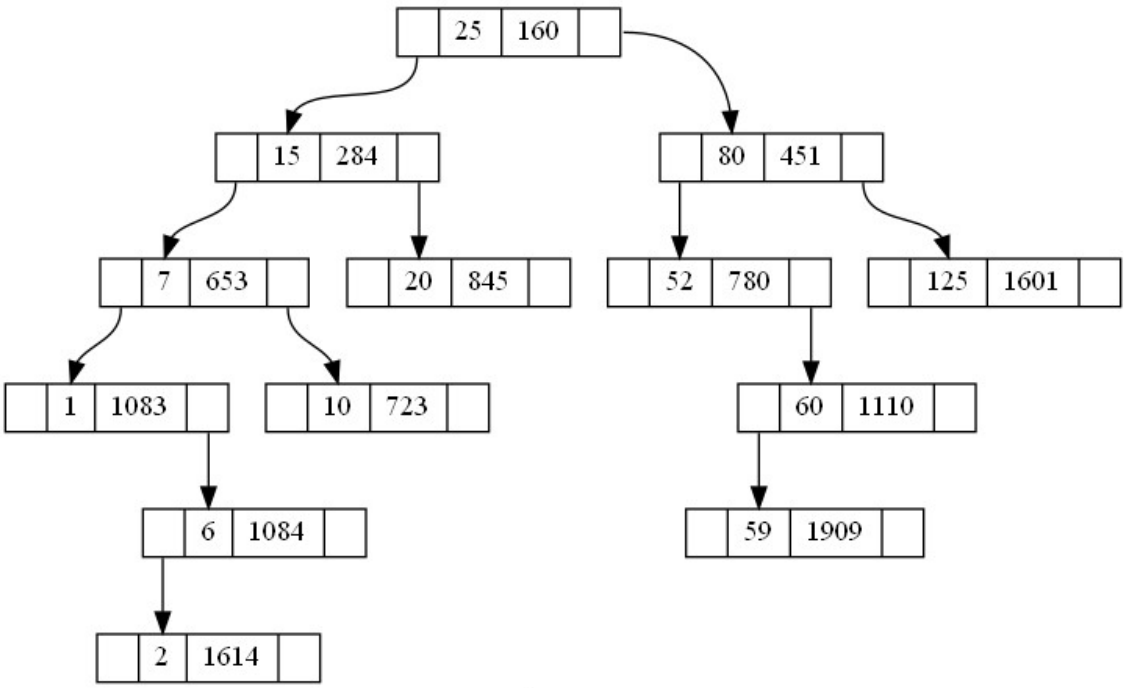
DELETION

- We need to traverse from root to key value node.
- If the key value is less than the current temp node value then it will move to left sub-tree.
- If the key value is more than the current temp node value then it will move to right sub-tree.
- After reaching to the key value node there will be 3 cases in deletion.

- Case 1: If the current which we are going to delete is leaf node, then we can just delete node and make parent nodes pointer null.
- Case 2: If the current which we are going to delete have only one child, then we can just delete node and join parent nodes pointer to delete nodes child.
- Case 3: If the current which we are going to delete have both, then we need to perform some rotation here. If left child's priority value is less than right child's priority value, then perform rotate right else perform rotate left. Keep doing it till we reach to case 1 or 2.
- Rotation won't affect BST property.

Example,

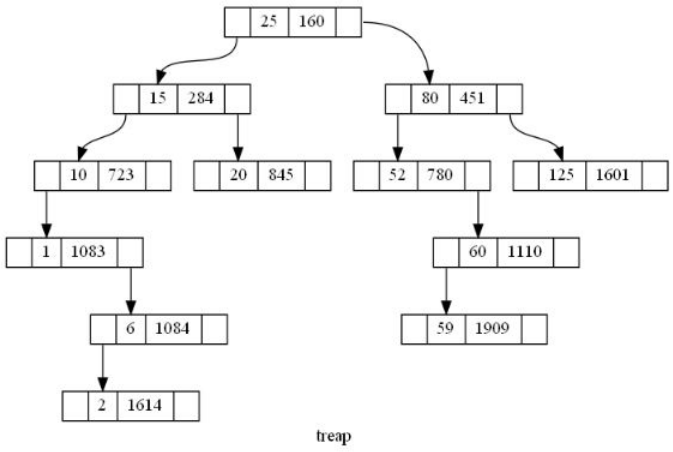
Initial treap



treap

Delete 7 form above treap

```
Press the number to perform operation and press enter:
1. Insertion
2. Deletion
3. Search
4. Print treap
Any other number to exit

2
Enter the value you want to delete : 7
Key is deleted

Press the number to perform operation and press enter:
1. Insertion
2. Deletion
3. Search
4. Print treap
Any other number to exit

4
Image file is generated. Check the folder

Press the number to perform operation and press enter:
1. Insertion
2. Deletion
```

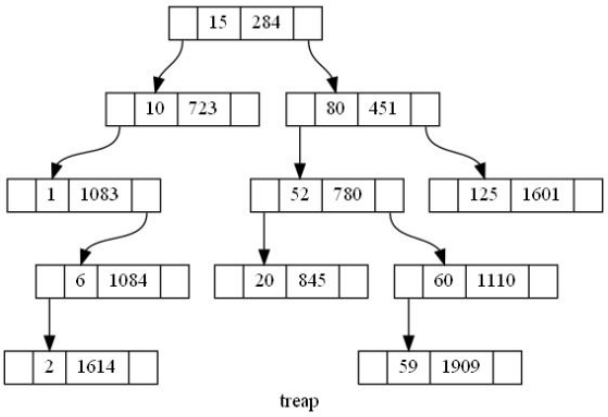


treap

Left rotation will takes place here. 7 will become left of 10. As we reach to case2 here, right subtree of 7 will become left child of 10. Then we will delete 7.

Delete 25 now

```
Press the number to perform operation and press enter:
1. Insertion
2. Deletion
3. Search
4. Print treap
Any other number to exit

2
Enter the value you want to delete : 25
Key is deleted

Press the number to perform operation and press enter:
1. Insertion
2. Deletion
3. Search
4. Print treap
Any other number to exit

4
Image file is generated. Check the folder

Press the number to perform operation and press enter:
1. Insertion
2. Deletion
3. Search
```



treap

Three rotations will takes place here. First right, then left, then again left. Now we will reach to case 2. 20 will become left child of 52 and then we will delete 25.

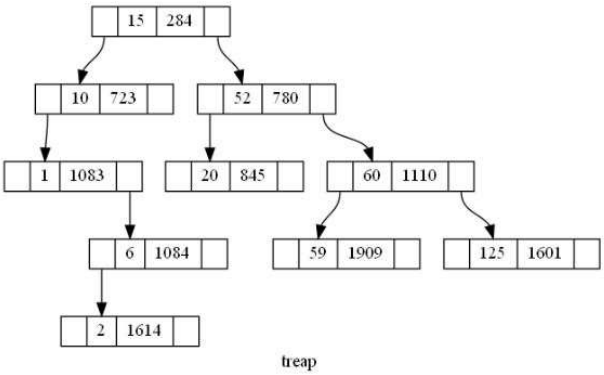
Right rotation will make 15 as root of the treap. 25 will be right chid of root.

Left rotation will make 25 as left child of 80.

2nd left rotation will make 25 as left child of 52.

Delete 80 now.

```
Press the number to perform operation and press enter:
1. Insertion
2. Deletion
3. Search
4. Print treap
Any other number to exit

2
Enter the value you want to delete : 80
Key is deleted

Press the number to perform operation and press enter:
1. Insertion
2. Deletion
3. Search
4. Print treap
Any other number to exit

4
Image file is generated. Check the folder

Press the number to perform operation and press enter:
1. Insertion
2. Deletion
```

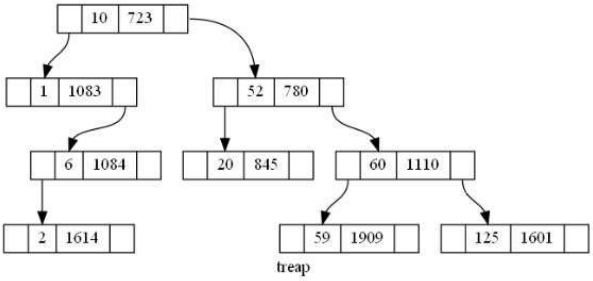


treap

Two right rotations takes place here.

Delete 15

```
Press the number to perform operation and press enter:
1. Insertion
2. Deletion
3. Search
4. Print treap
Any other number to exit

2
Enter the value you want to delete : 15
Key is deleted

Press the number to perform operation and press enter:
1. Insertion
2. Deletion
3. Search
4. Print treap
Any other number to exit

4
Image file is generated. Check the folder
```

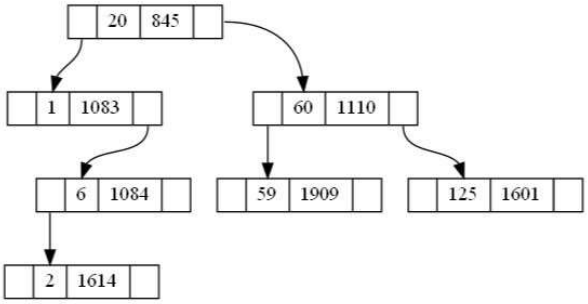


treap

First right rotation then left rotation.

Delete 10, 52

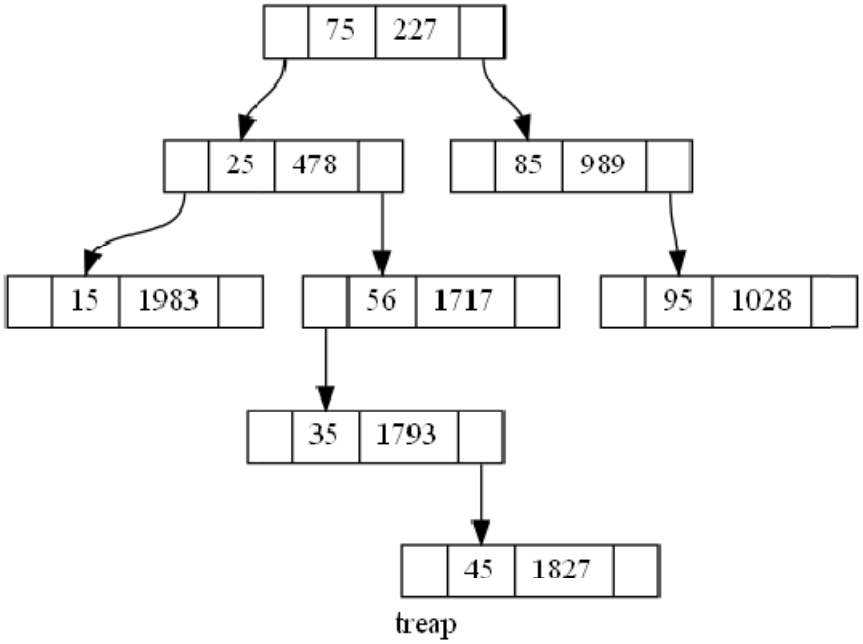
```
Press the number to perform operation and press enter:
1. Insertion
2. Deletion
3. Search
4. Print treap
Any other number to exit

2
Enter the value you want to delete : 52
Key is deleted

Press the number to perform operation and press enter:
1. Insertion
2. Deletion
3. Search
4. Print treap
Any other number to exit

4
Image file is generated. Check the folder
```



treap

SEARCH

- Searching in Treap is same as searching binary search tree.
- If the key value is less than the current temp node value then it will move to left sub-tree.
- If the key value is more than the current temp node value then it will move to right sub-tree.
- If the current node is null then key is not present in tree, return false.
- If key is equal to node key value then return true.



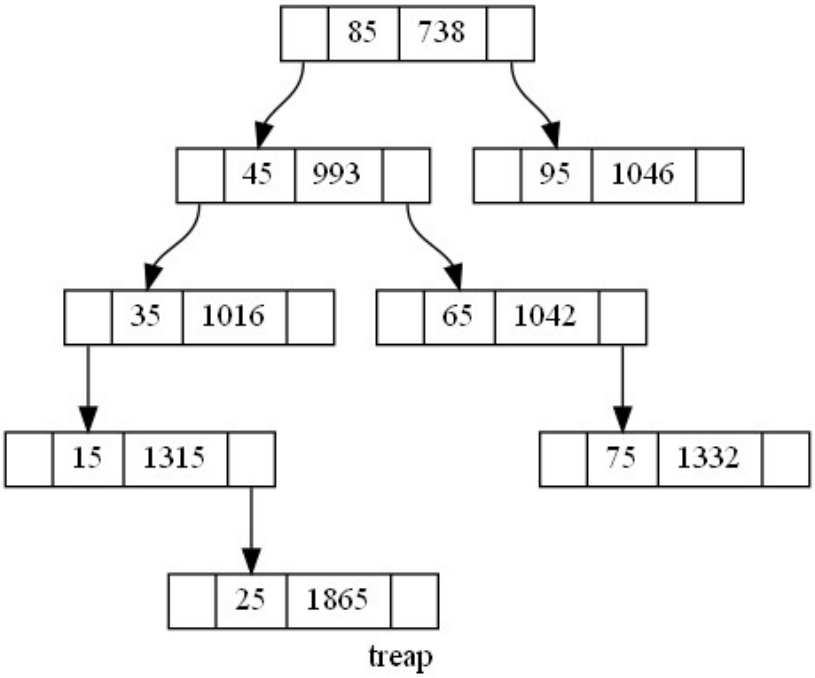
treap

```
Press the number to perform operation and press enter:
1. Insertion
2. Deletion
3. Search
4. Print treap
Any other number to exit

3
Enter the value you want to search : 35
Key is present in tree

Press the number to perform operation and press enter:
1. Insertion
2. Deletion
3. Search
4. Print treap
Any other number to exit

3
Enter the value you want to search : 20
Key is not present in tree
```

PRINT

- For the print operation press 4 and enter.
- A dot file will generate in same folder.
- An image.png file will be generated directly.
- System call for image.dot file: "dot -Tpng image.dot -o image.png".

```
Press the number to perform operation and press enter:
1. Insertion
2. Deletion
3. Search
4. Print treap
Any other number to exit

4
Image file is generated. Check the folder

Press the number to perform operation and press enter:
1. Insertion
2. Deletion
3. Search
4. Print treap
Any other number to exit
```



treap