Module-3

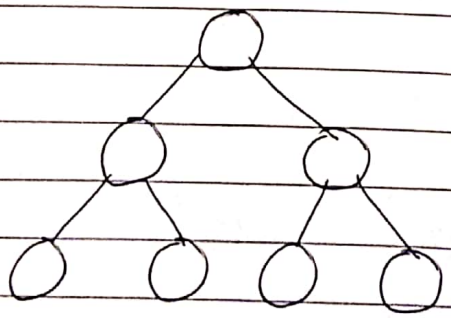
Binomial heap

→ Binomial heap is an extension of binary heap that provides union or merge operations together with other operations provided by binary heap.

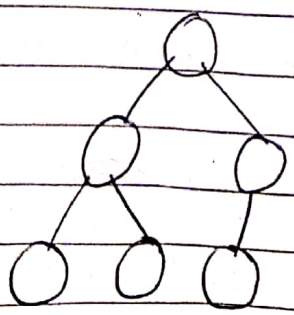
* Heap → A heap is a binary tree structure with the following properties:-
1) The tree is complete or nearly complete
2) The key value of each node is less than or greater than to the key values in each of its descendents.

→ A complete binary tree is a binary tree in which every level except the last level is completely filled, and all nodes are as left as possible.

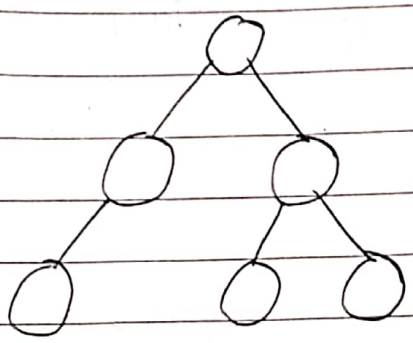
eg) Complete binary tree:-



Nearly complete binary tree



Not a complete binary tree:-



* Binary Heap →

→ A binary heap is a complete binary tree which satisfies the heap ordering property.

→ The heap ordering is of two types:-

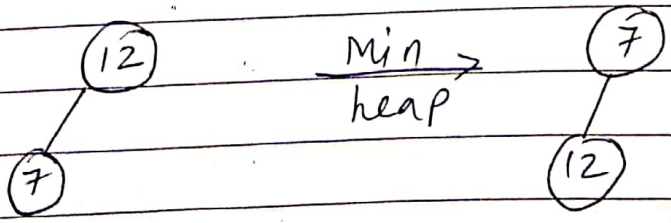
1) Min heap property
2) Max heap property

1) Min heap property →
The value of each node is greater than or equal to the value of its parent ie. the minimum value is at the root.

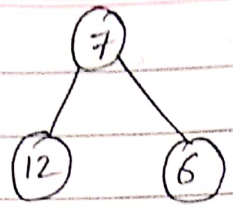
eg) Create minimum heap for the elements:-
12, 7, 6, 10, 8, 20

12 →

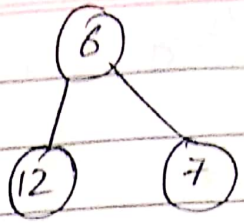
7 →



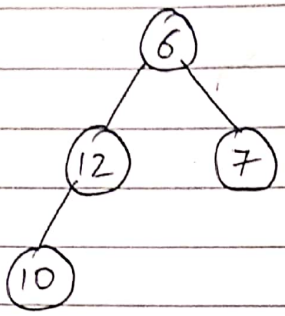
6 →



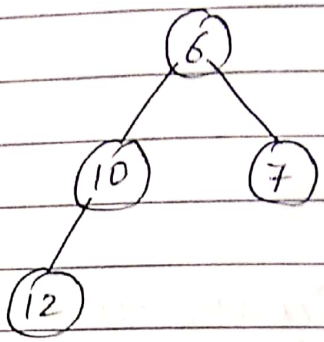
Min heap →



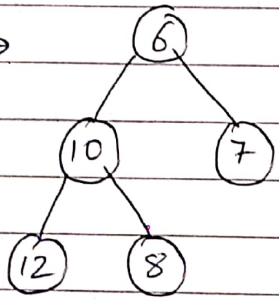
10 →



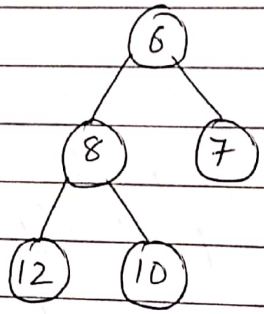
Min heap →



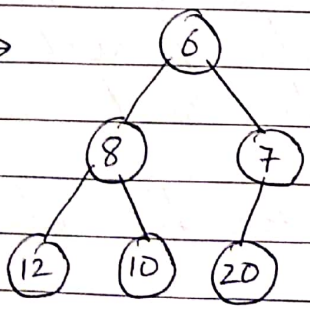
8 →



Min heap →



20 →

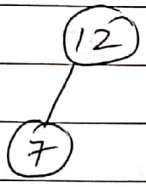


2) Max heap property →
The value of each node is less than or equal to the value of its parent ie. the maximum value is at the root node.

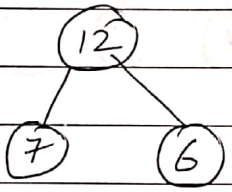
eg) Create maximum heap for the elements:-
12, 7, 6, 10, 8, 20

12 →

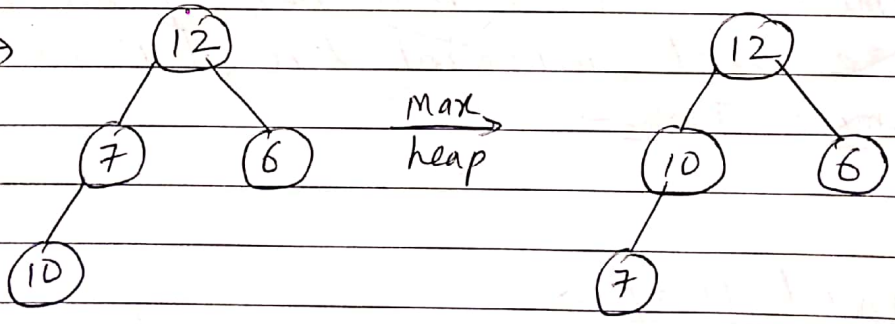
7 →



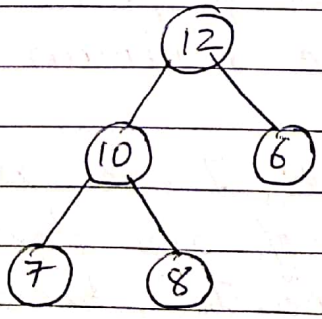
6 →

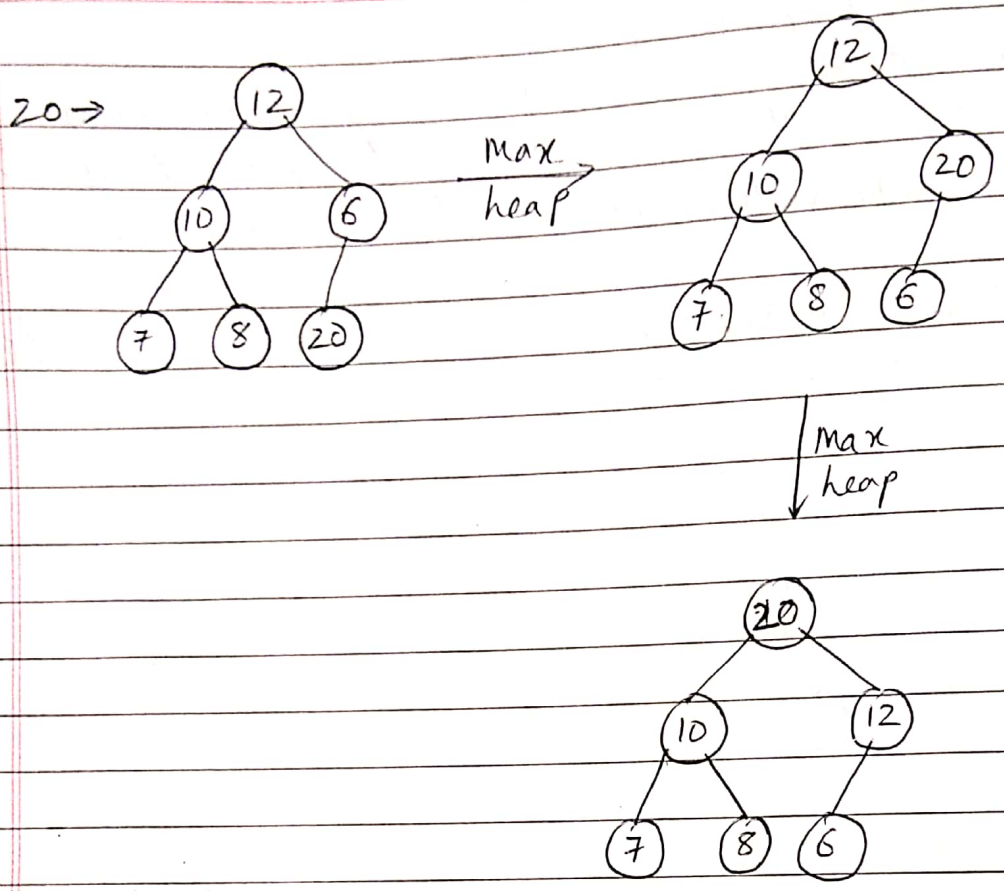


10 →



8 →





* ~~Binomial heap is a collection of binomial trees where each binomial tree follows~~
* ~~Binomial Tree~~ →

* Binomial Heap →
It is a collection of binomial trees that satisfies the following two properties:-
1) Each binomial tree is a minimum ordered heap.
2) ~~If~~ In a binomial heap H, there is at most one binomial tree of any degree. [ie in a binomial heap, no two binomial trees have same degrees].

Smita Attarde

* Binomial Tree →

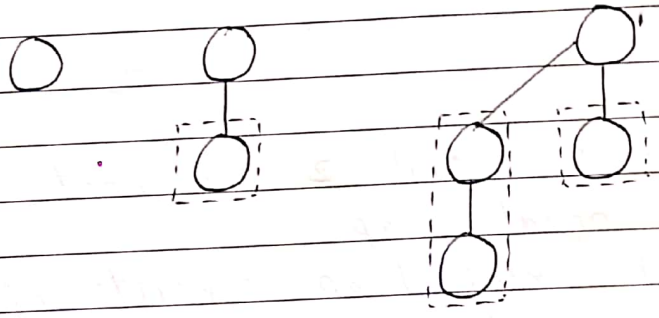
→ A binomial tree of order 0 has one node.

→ A binomial tree of order k has a root node, whose children are roots of binomial trees of order k-1, k-2, ... 2, 1, 0.

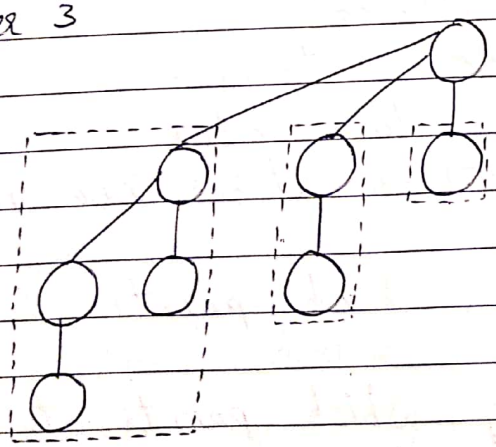
* properties of Binomial Tree →

1) It has exactly 2^k nodes.
2) Its depth is k.
3) The root has degree k and children of root are themselves binomial trees with order k-1, k-2, ... 2, 1, 0 from left to right.

order 0 1 2



order 3

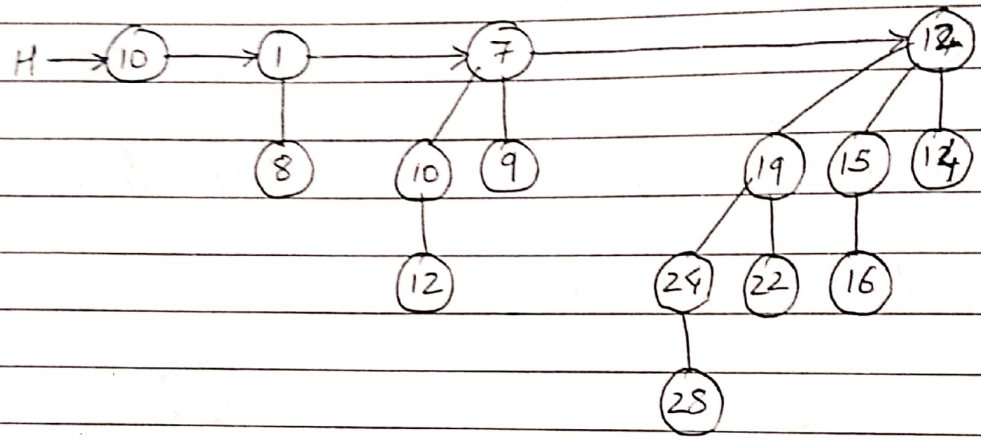


* Representing Binomial Heaps →

→ The roots of the binomial trees within a binomial heap are organized in a linked list, which is called as the root list.

→ The degrees of the roots strictly increase as we traverse the root list.

eg)



→ Here, 10, 1, 7 and 12 are root nodes of the binomial heap.

→ This binomial heap consists of 4 binomial trees of order 0, 1, 2 and 3.

→ There is at the most one binomial tree of a particular degree

→ Thus, each node contains :-

1) a field Key, to store value
2) a field degree which stores the number of children.
3) a pointer child, which points to the leftmost child.
4) a pointer sibling, which points to the right sibling
5) a pointer P, which points to the parent.

→ A given binomial heap H is accessed by the head node (H), which simply points to the first root of the list H.

* Operations on Binomial Heaps →

1) Creation of a new heap.
2) Search for the minimum key.
   [Extracting a node with minimum key value]
3) Merging two binomial heaps.
4) Insertion of a node.
5) Deletion of a node with minimum key.

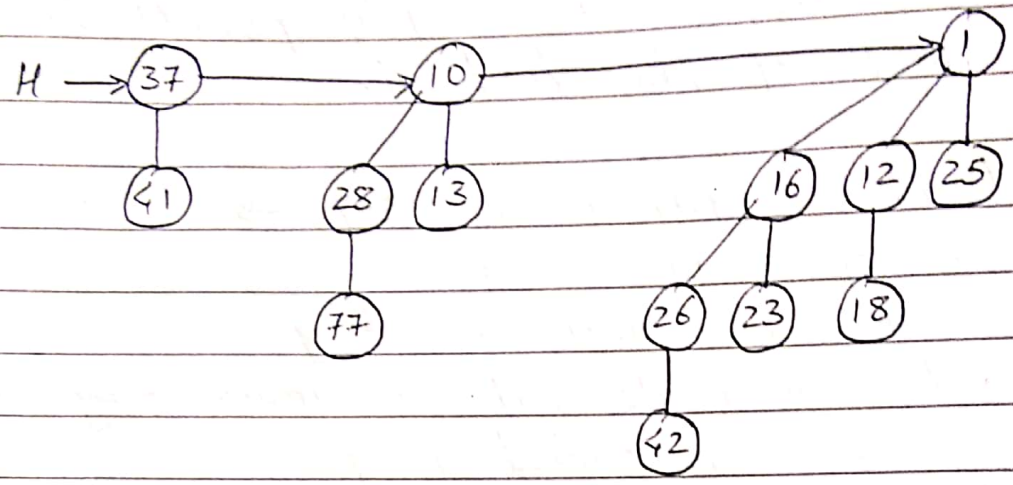
1) Creation of a new heap →

Creation of a binomial heap is same as inserting a new node in an empty binomial heap.

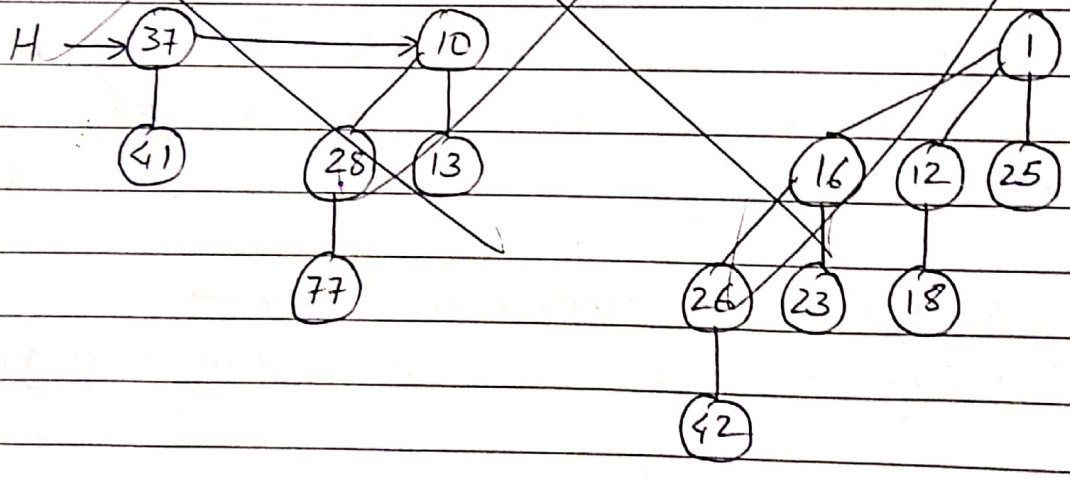
2) Search for the minimum key →
[Extracting a node with minimum key value] →

→ Since a binomial heap is a minimum heap-ordered, the minimum key resides in the list of root nodes.
→ To extract minimum element from the heap, find the minimum among roots of the binomial heap.
→ This can be easily done in O(log n) time, as there are just O(log n) trees whose roots are to be examined.

eg) Consider the binomial heap as-



Here, minimum value is 1. Hence, it is removed from the list of H.



→ Here, the root nodes of all the binomial trees are compared.
ie. 37, 10 and 1
→ Thus, the minimum element is 1.

Algorithm →

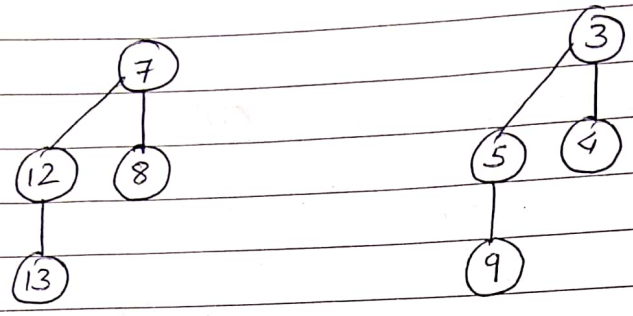
```
Binomial_heap_minimum(H)
{
    x = head(H)
    min = ∞

    while (x != NULL)
    {
        if (key(x) < min)
            min = key(x)

        x = sibling(x)
    }
}
```

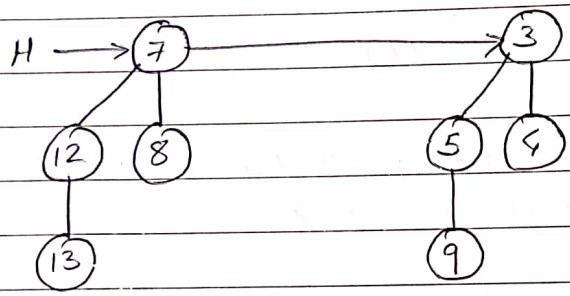
3) Merging two binomial heaps →

→ The merge operation repeatedly links the binomial trees whose roots have the same degree.

→ As their root node is the smallest element within the tree, by comparing the two keys, the smaller of them is the minimum key and becomes the new root node. Then, the other tree becomes a subtree of the combined tree.

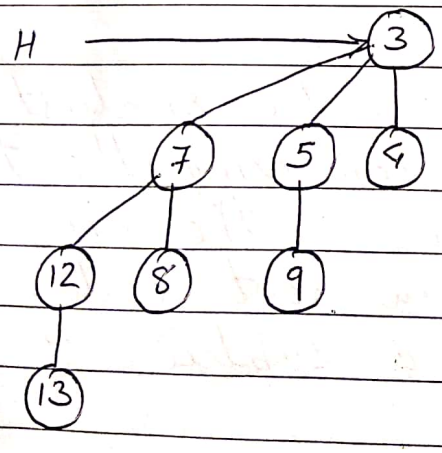
eg)



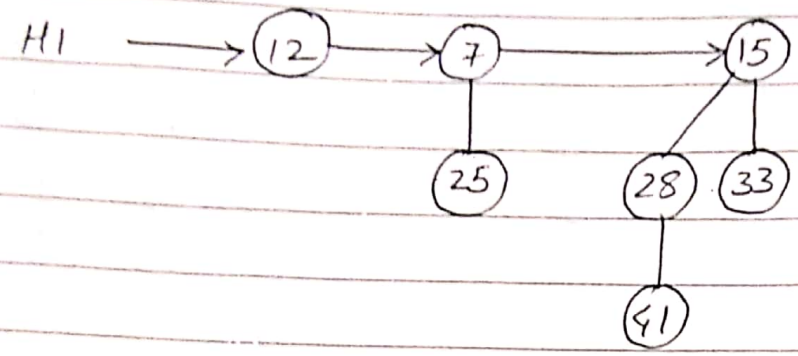


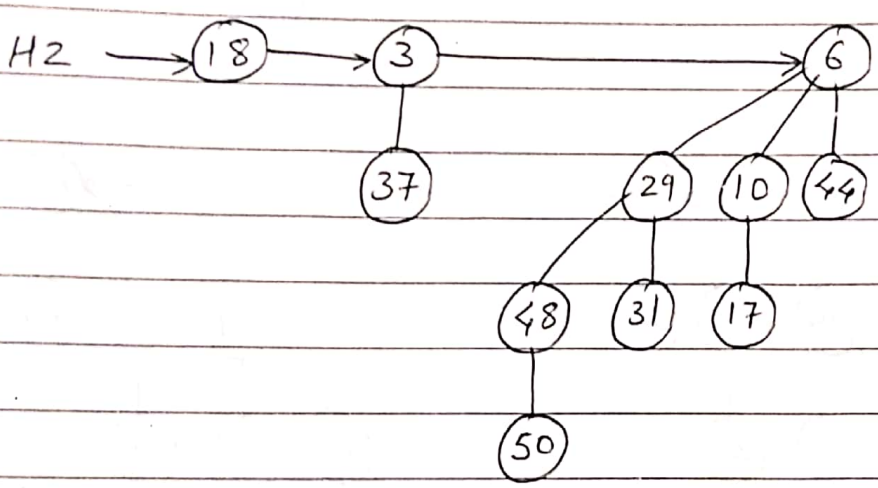
Here, both trees have same degree. Hence, they need to be merged.

Smallest of 7 and 3 is 3.
So 3 is the new root node and the first tree is attached to the second tree as its left child to form the merged tree.

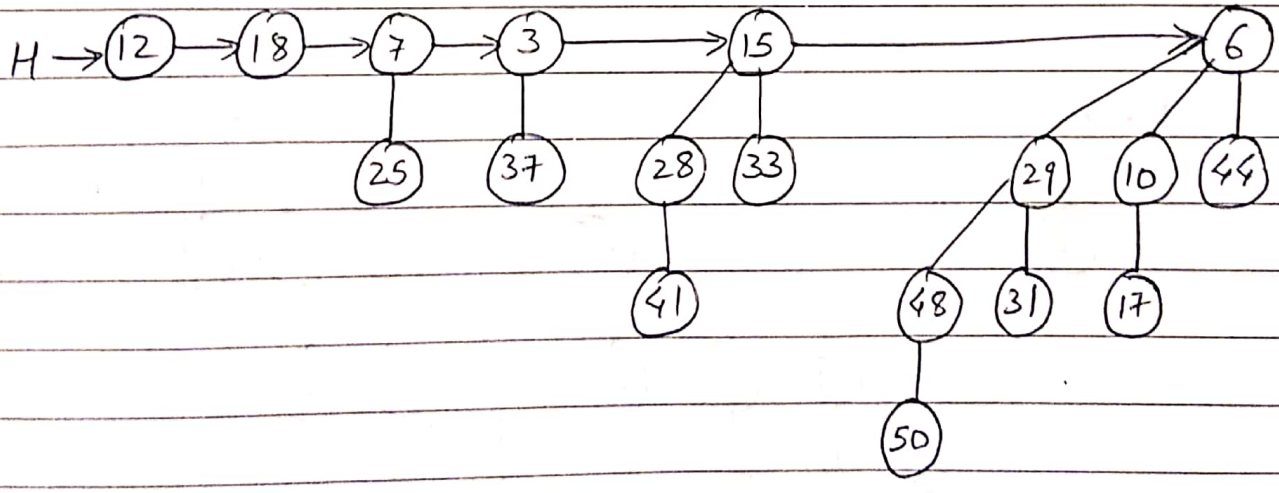


(g) Merge H1 and H2.

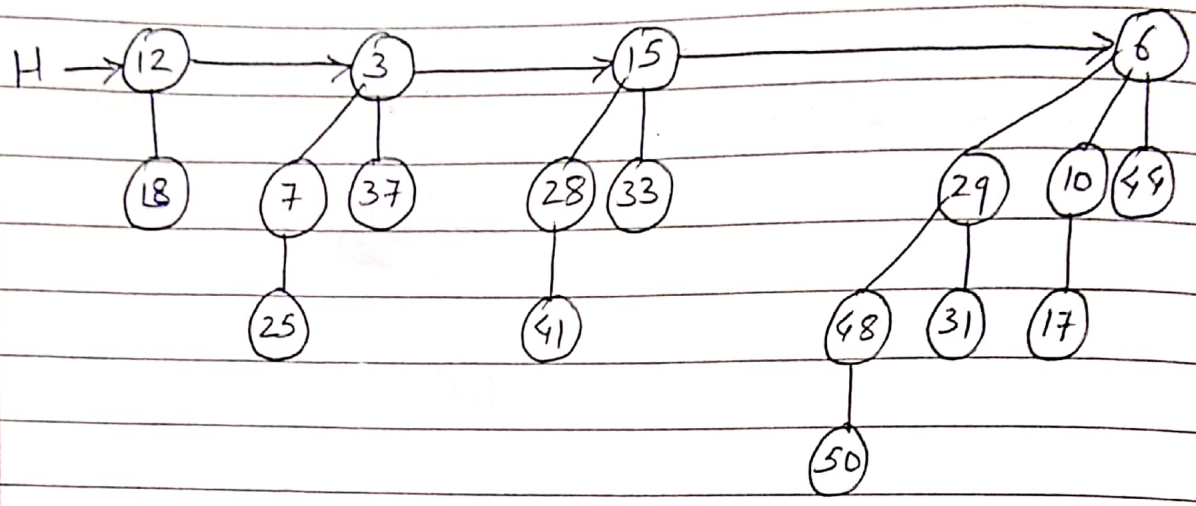


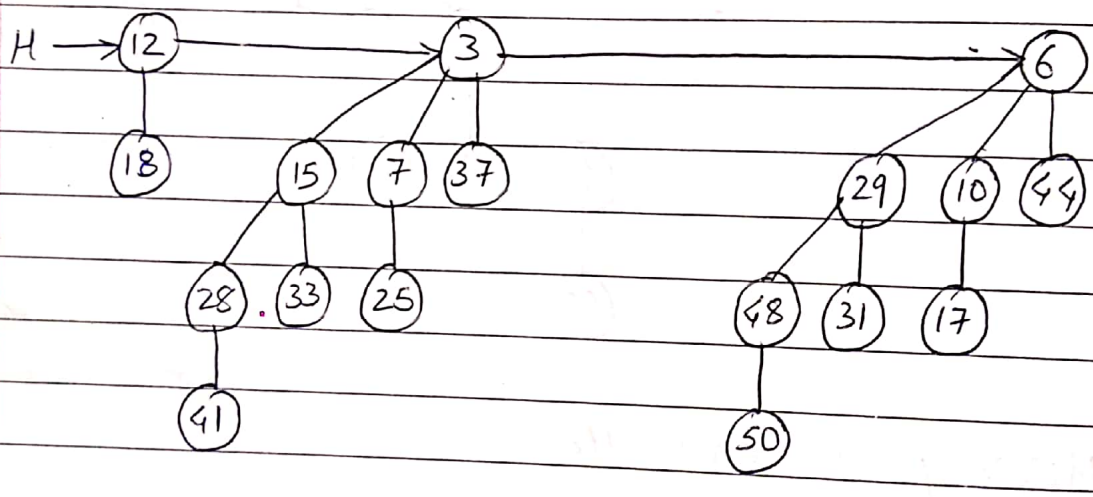


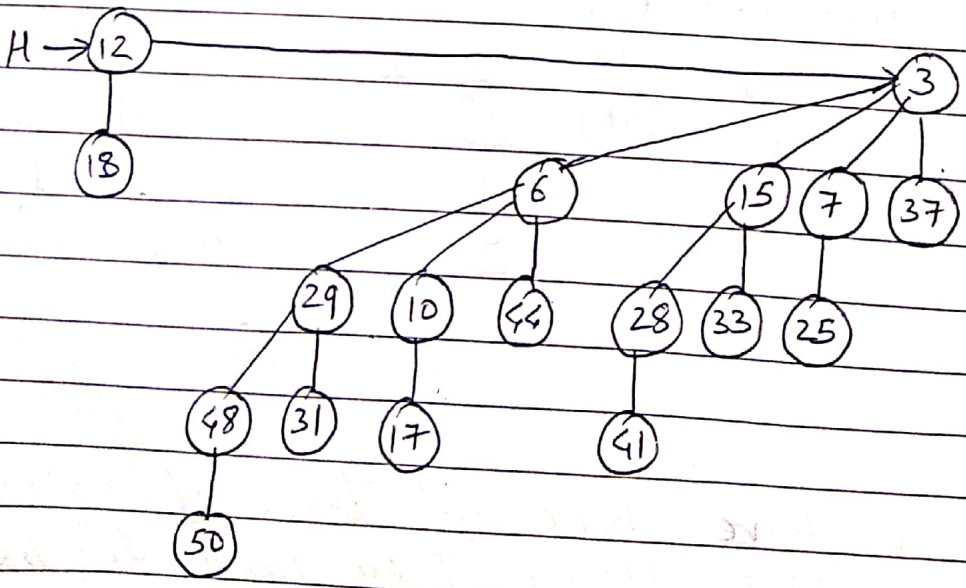
Merging H1 and H2.



Here, we have binomial trees of same degrees. ~~Here~~ Hence, they need to be merged.



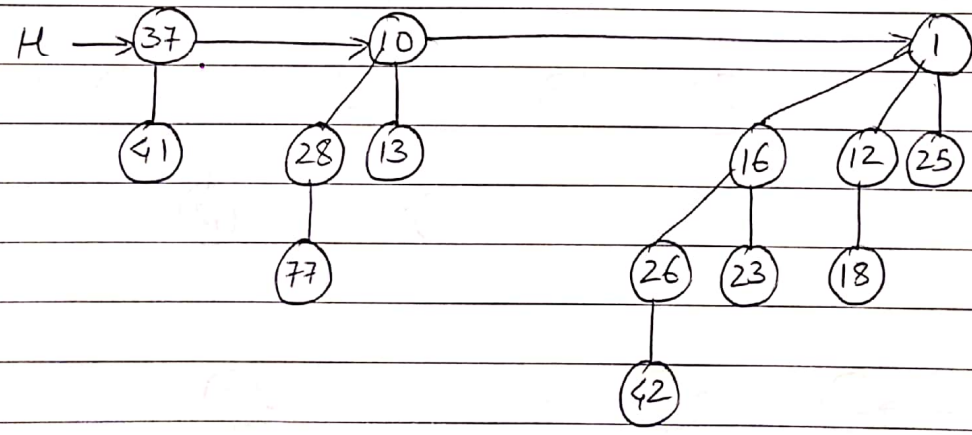




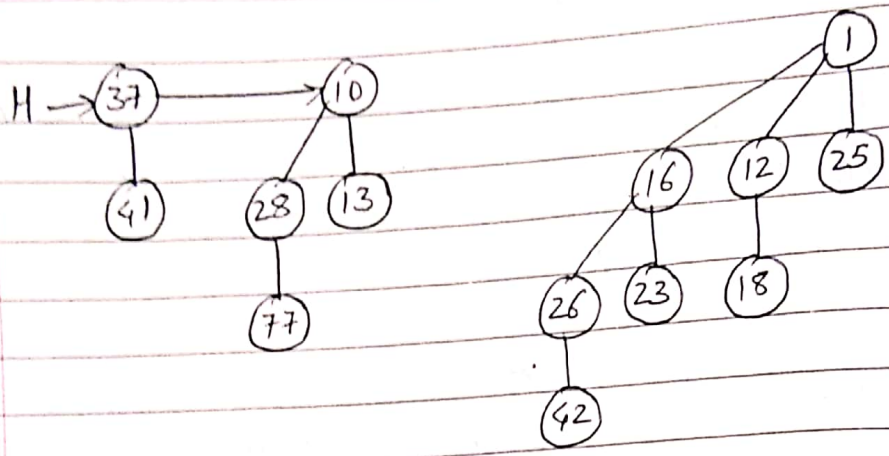
4) Insertion of a node →
Inserting a new node in a binomial heap can be done by simply creating a new heap containing only this element and then merging it with the original heap.

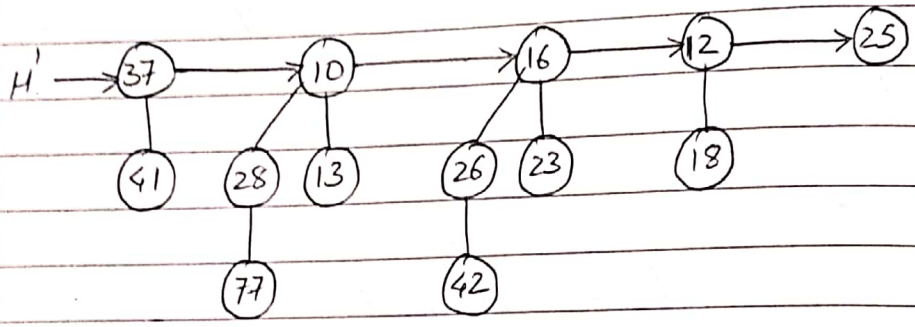
5) Deletion of a node with minimum key →
→ To delete a node with minimum key from the heap, first find the element, remove it from the binomial tree and obtain a list of its subtrees.
→ Then, transform this list of subtrees into a separate binomial heap by reordering them from smallest to largest order.
→ Then, merge this heap with the original heap.

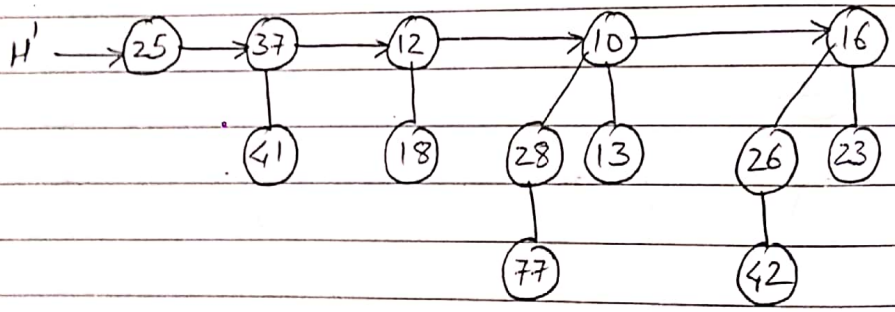
eg)

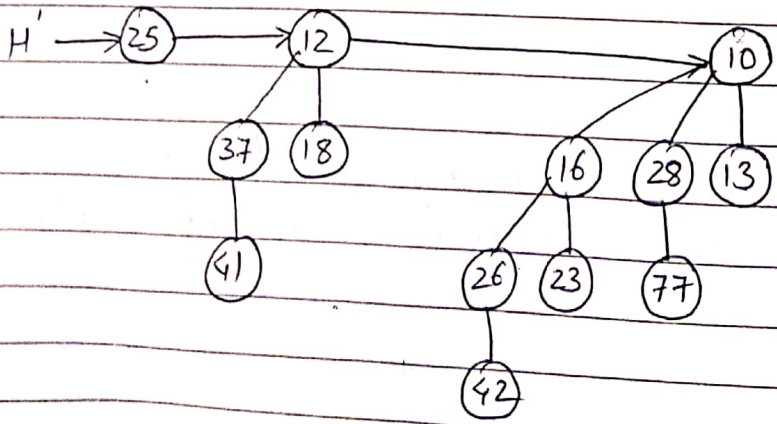


Minimum value among roots is 1.
Hence it is removed from the list of H.







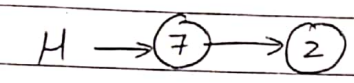


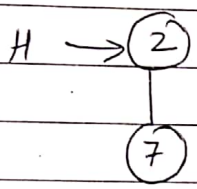
Example:-

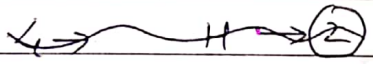
Let $A = \{7, 2, 4, 17, 1, 11, 6, 8, 15, 10, 20\}$

1) Draw a binomial heap whose keys are the elements of A.
2) Insert a new element with the key 5 into this heap.
3) To a binomial heap obtained this way, apply the operation of extracting the node with minimum key two times.

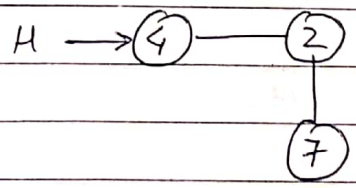
Soln →



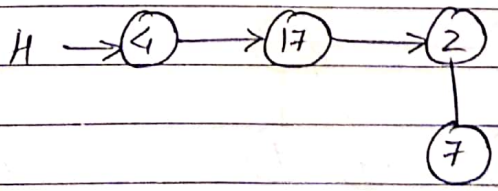


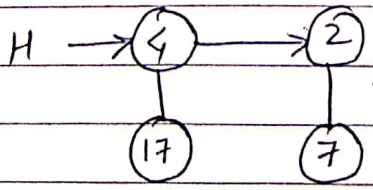


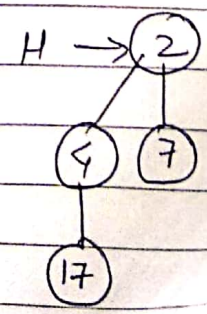
4



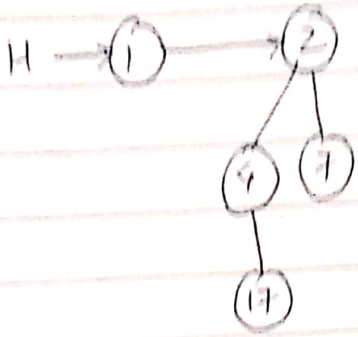
17



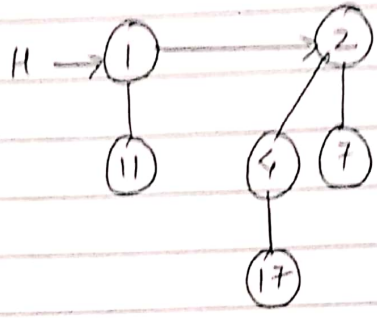




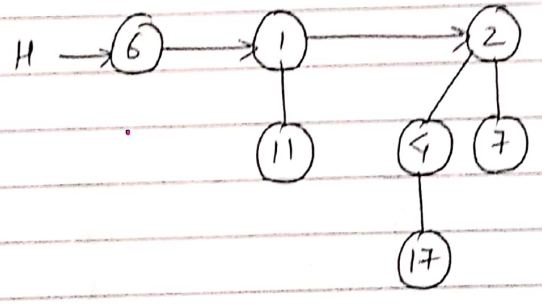
1 →



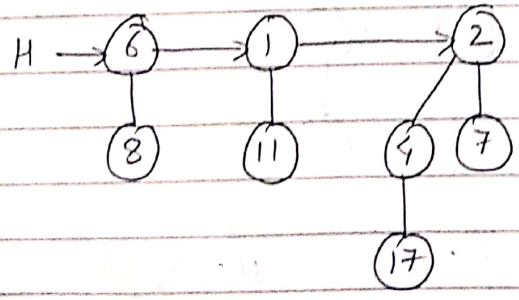
11 →



6 →

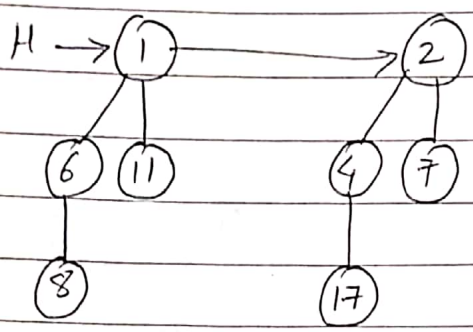


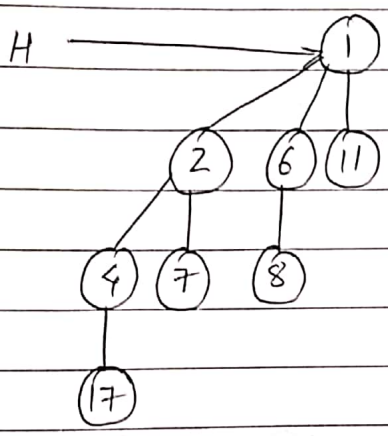
8 →



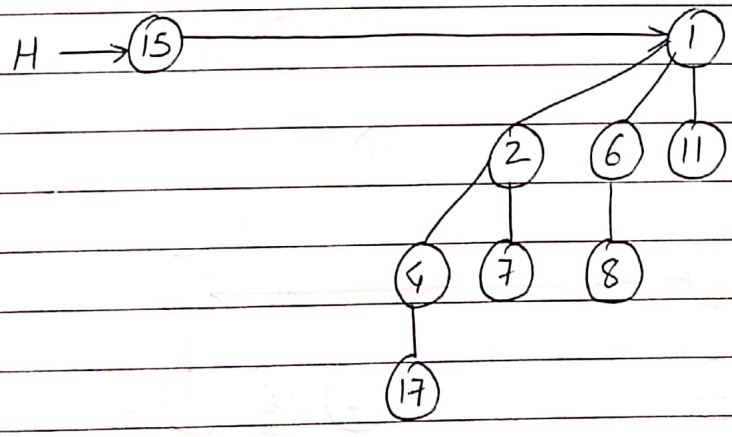
Smita Attarde

Smita Attarde

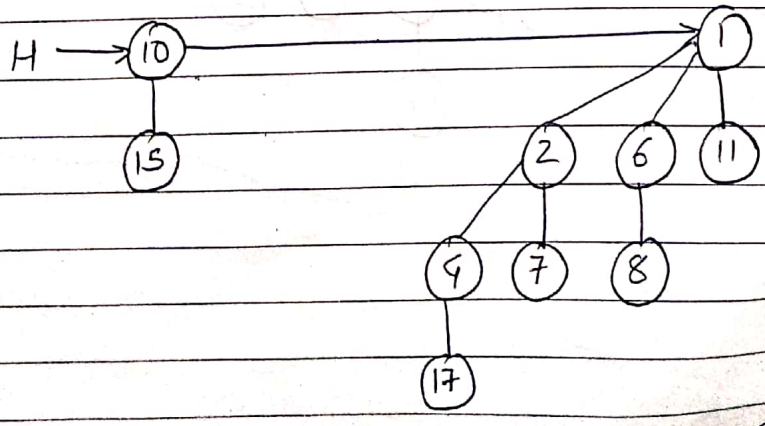




15→

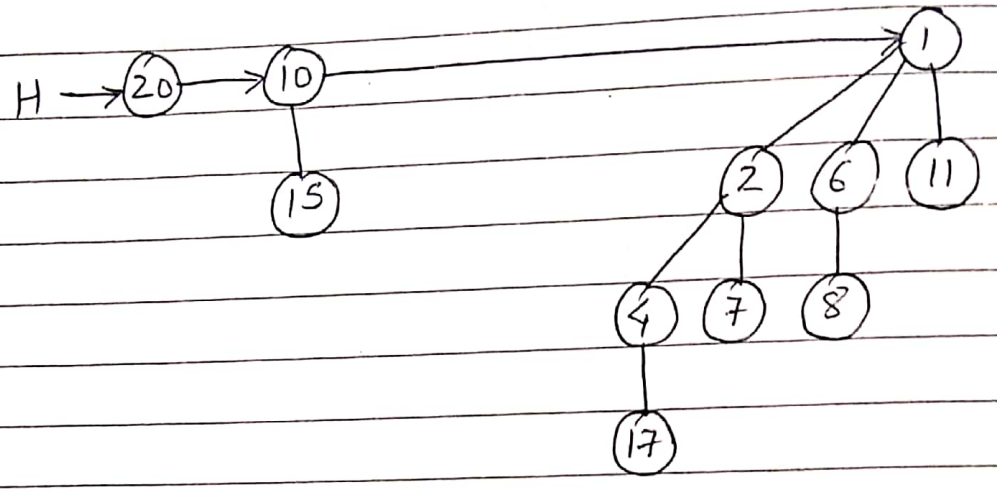


10→

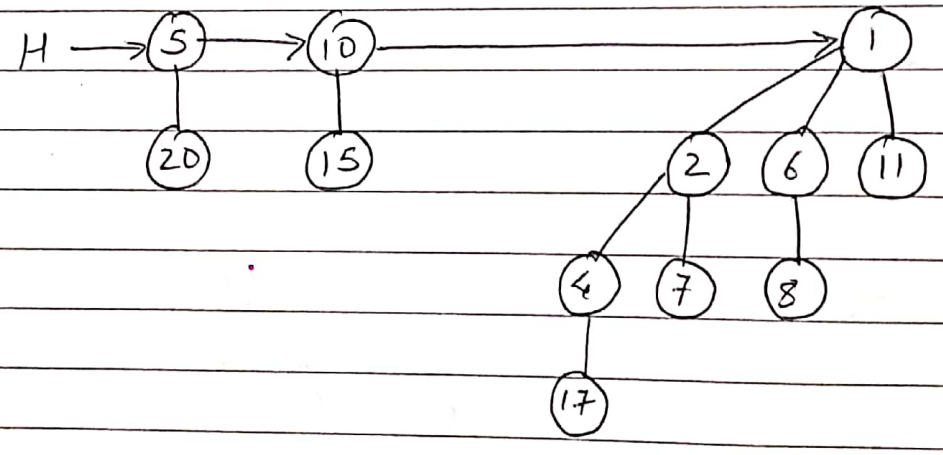


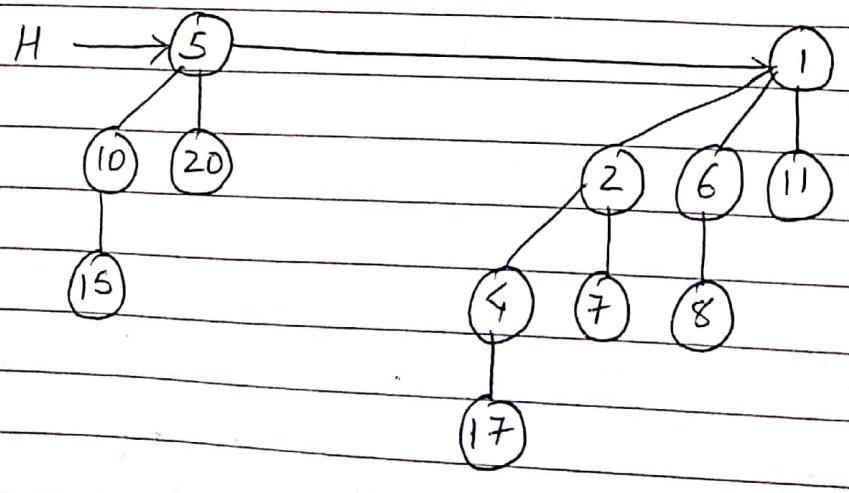
Smita Attarde

20 →

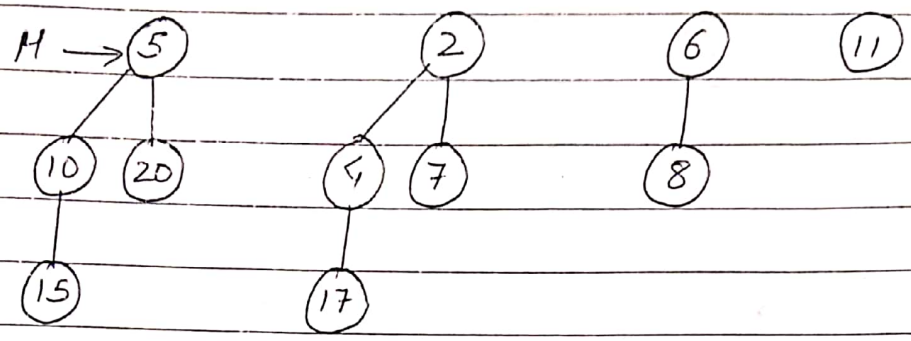


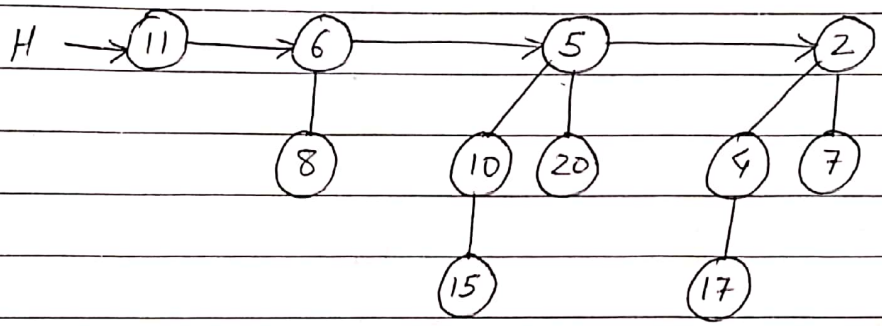
2) Insert 5 →

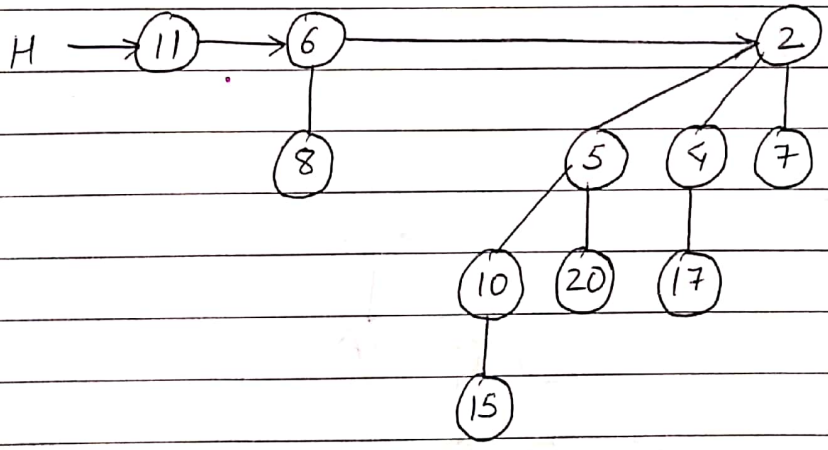




3) Extract node with minimum key →
Here, minimum key is 1







4) Extract node with minimum key →
Here, minimum key is 2.