Priority Queues is the kind of queue, in which element is deleted (or serviced) according to its priority.

So, each element has it's associated priority and element is serviced or deleted with highest or lowest priority depending upon the application.

Example:-

Consider a machine M, which can seirve one job at a time. The users of the machine are required to pay fixed amount per usage. However, time requirement of each of the user is

different. In order to maximize the no. of jobs serviced from the machine or minimize the length of waiting queue for the machine, one can use priority queue. This queue choose smallest job first (Smallest job means the one whose expected time requirement of the machine is the least.). Hence, a min-priority queue is required, in which the "priority" entity is nothing but the expected time requirement for finishing the job.

Observation :- To determine the order of jobs to be serviced, priority is used.

If we use linear structures to implement priority queue :-

Time Complexity

(Assume Array - implementation of priority queue)

Deletion of job (Job Servicing) : $O(n)$

Insertion of new job : $O(1)$

(Assume you keep the array sorted according to their priorities)

Deletion of job (Job Servicing) : $O(1)$

Insertion of new job : O(n)

Let's try to understand the implications if we used two-dimensional linear structure instead of one-dimensional structure (array) for the same problem.

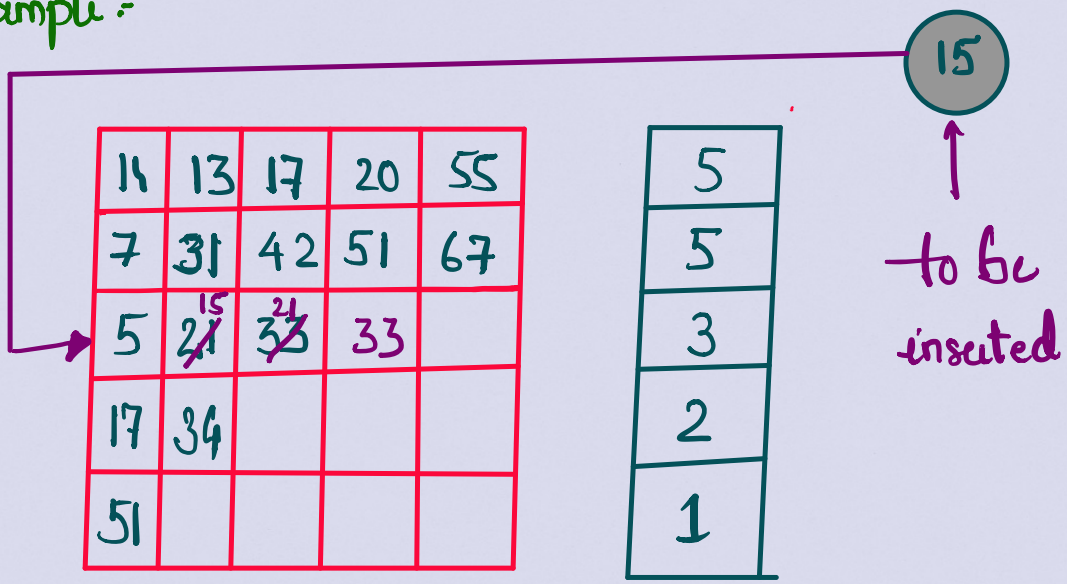
Let's assume there are 'N' jobs which needs to be serviced. All jobs have priorities associated with it.

We decide to service these 'N' jobs by maintaining a matrix of dimension $\sqrt{N} \times \sqrt{N}$

Insert :

1. Maintain the no. of elements in each row of $\sqrt{N} \times \sqrt{N}$ matrix.
2. Insert into first row that has free space.
3. Each row is maintained in sorted order.

Example :-



Time Complexity = $O(\sqrt{N})$

Delete (Delete an element with maximum priority)

- Maximum element can be found out by max { maximums of each row }

Time Complexity : $O(\sqrt{N})$

Total time for processing / servicing all the jobs = $O(\sqrt{N}\, N)$

Heaps

Heaps are generally used to implement the priority queues.

Binary heaps :

max - heap (max binary heap)

min - heap (min binary heap)

Binary heap :

A binary heap is a binary tree with two properties.

(i) Structure

Complete Binary Tree

Tree is filled from top to bottom and from left to right.

(ii) Value

Value of any node is no smaller than the value of its children.

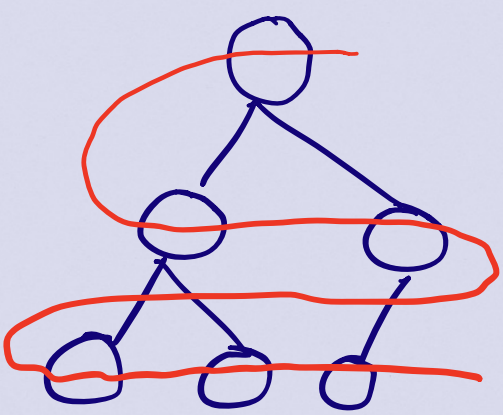
(Max-heap)

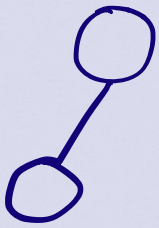
Value of any node is no larger than the value of its children.

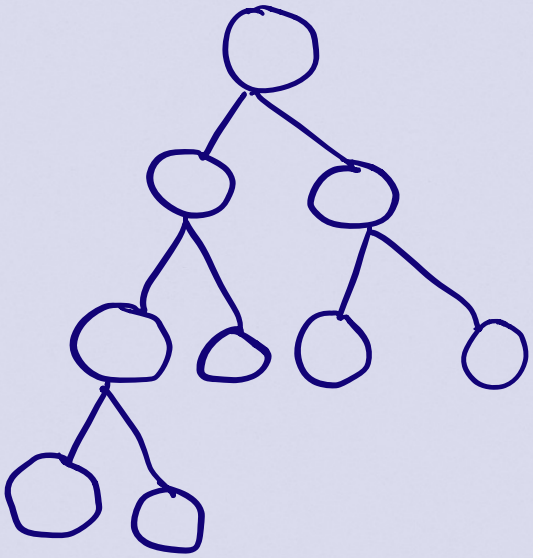
(Min-heap)

Examples :-

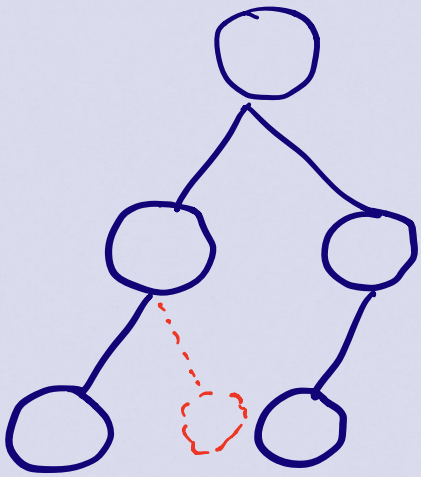
Valid heap Structure.

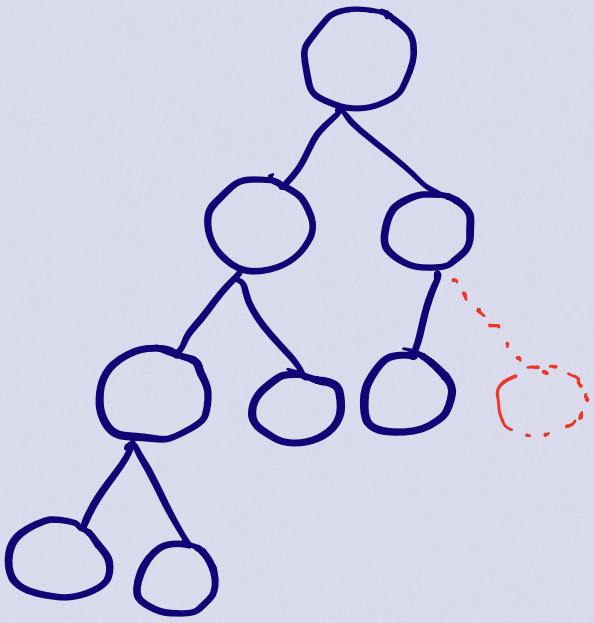






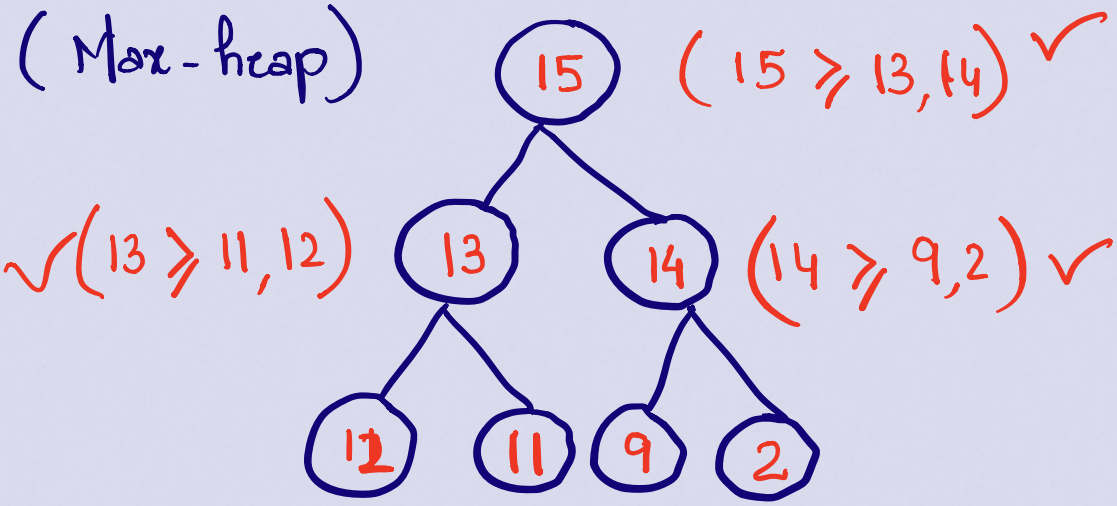
Invalid heap Structure



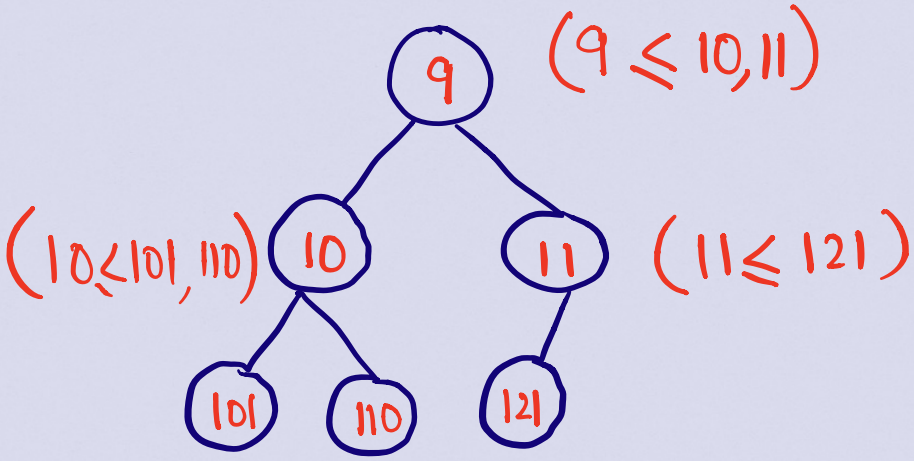


Valid Heap

(Max-heap)

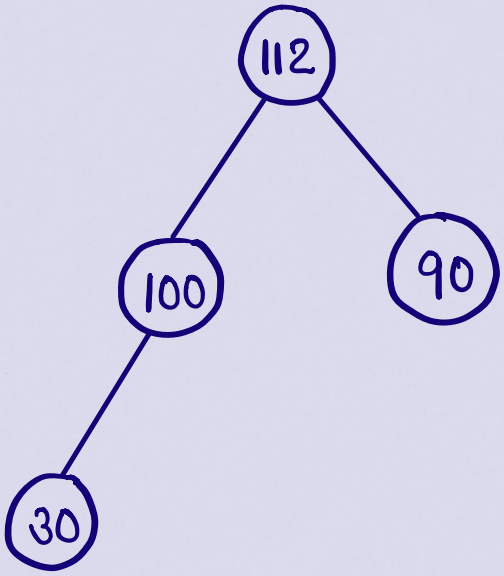


(Min-heap)

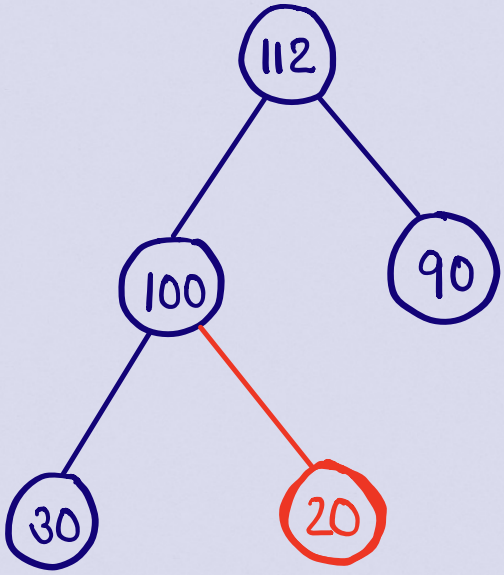


Operations on heap

(1) Inserting an element in heap.



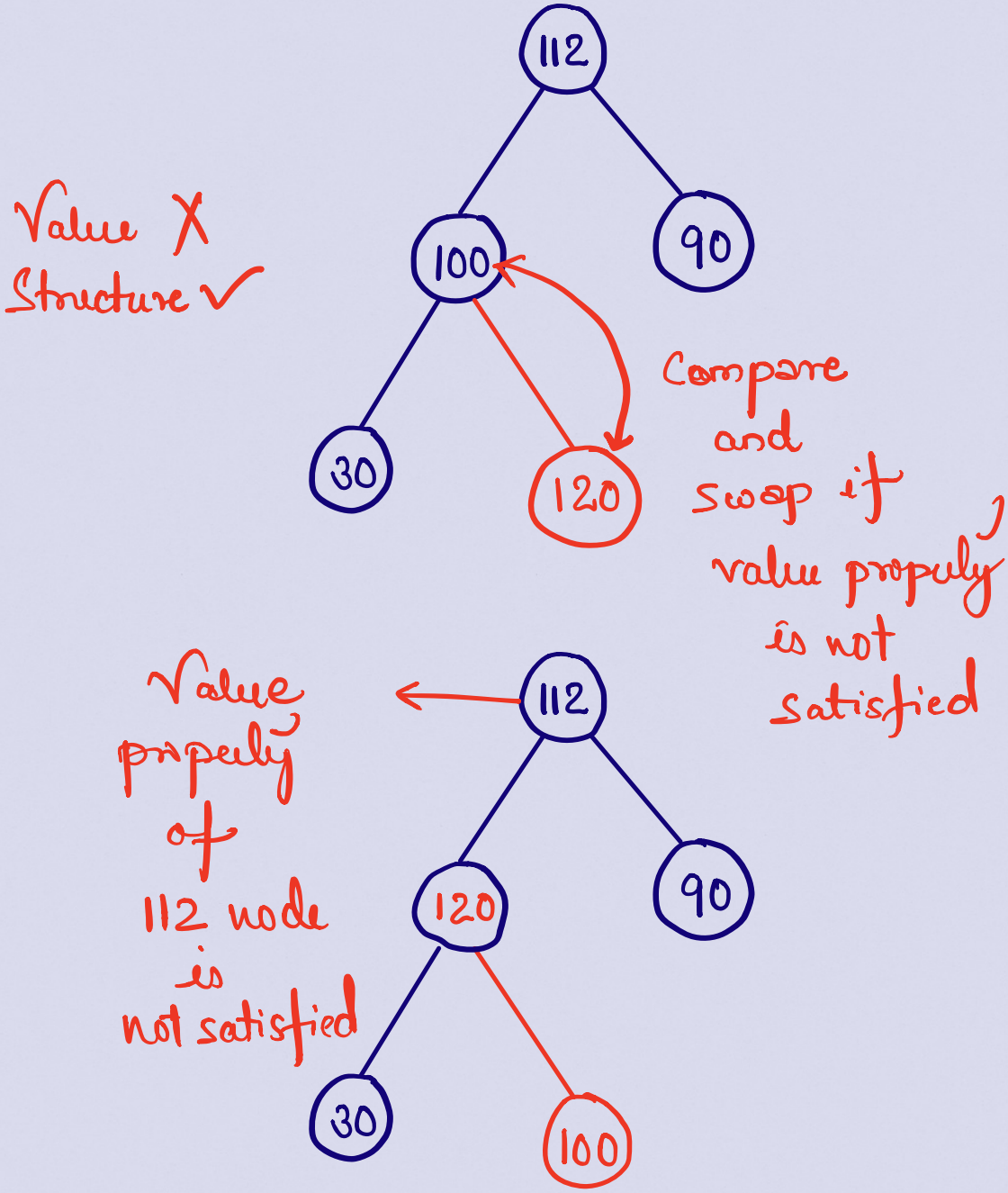
Let the new value to be inserted in the given heap is 20

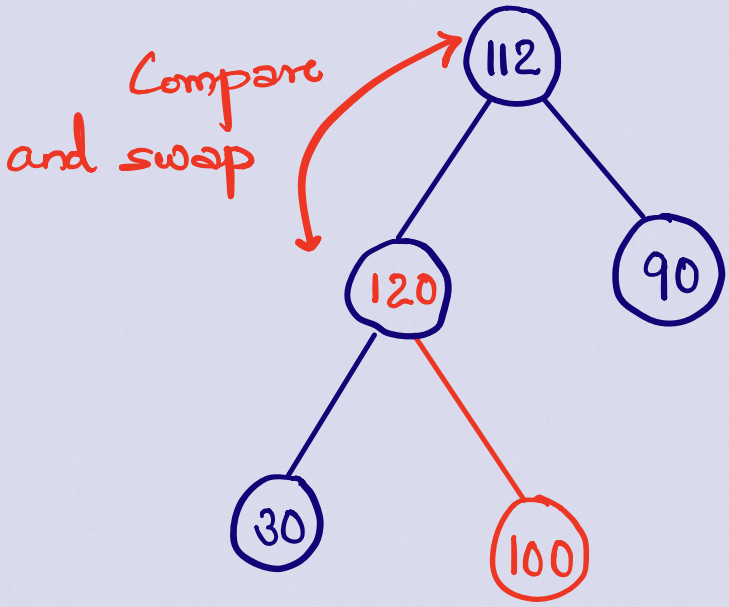


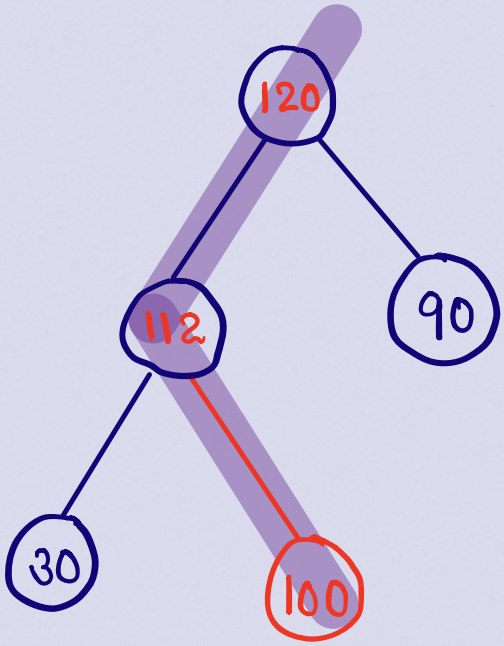
Structure ✓

Value ✓

Let the value to be inserted is 120




Compare
and swap
112
120
90
30
100


120
112
90
30
100

Insertion

1. The position of new job is fixed by the structure property of the heap.

2. We need to restore the heap property, i.e, (remaining value property is restored along the path to root.

Complexity of Insertion:

We need to walk up from the leaf node to the root node (at worse)

Height of Complete Binary Tree

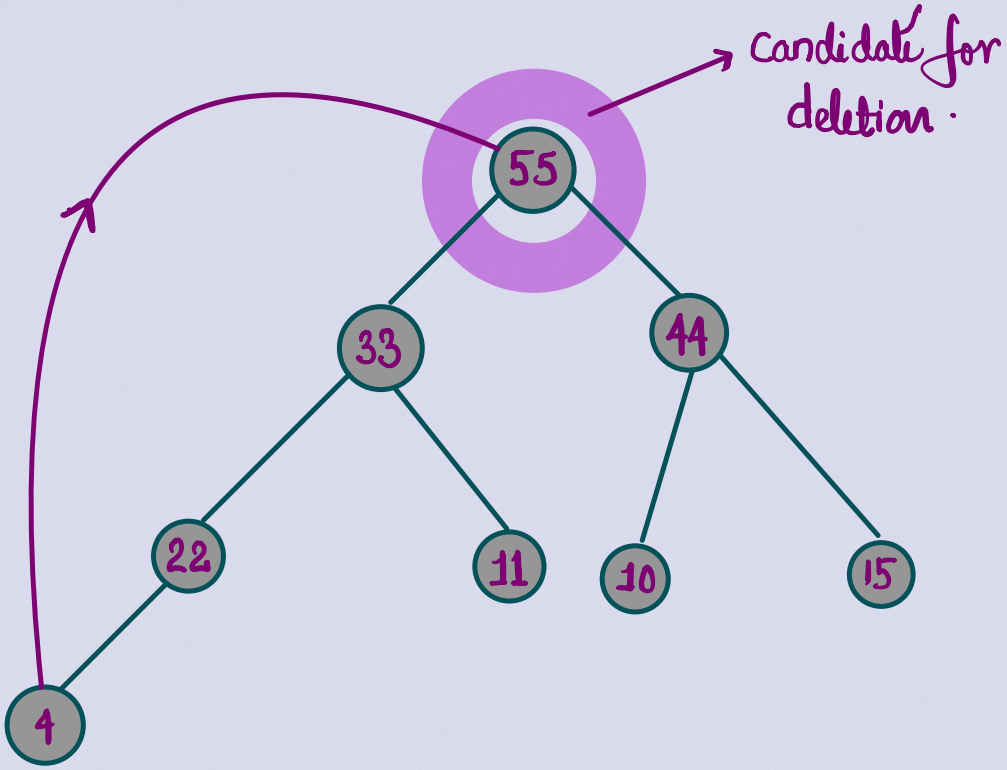
$O(\log|N|)$

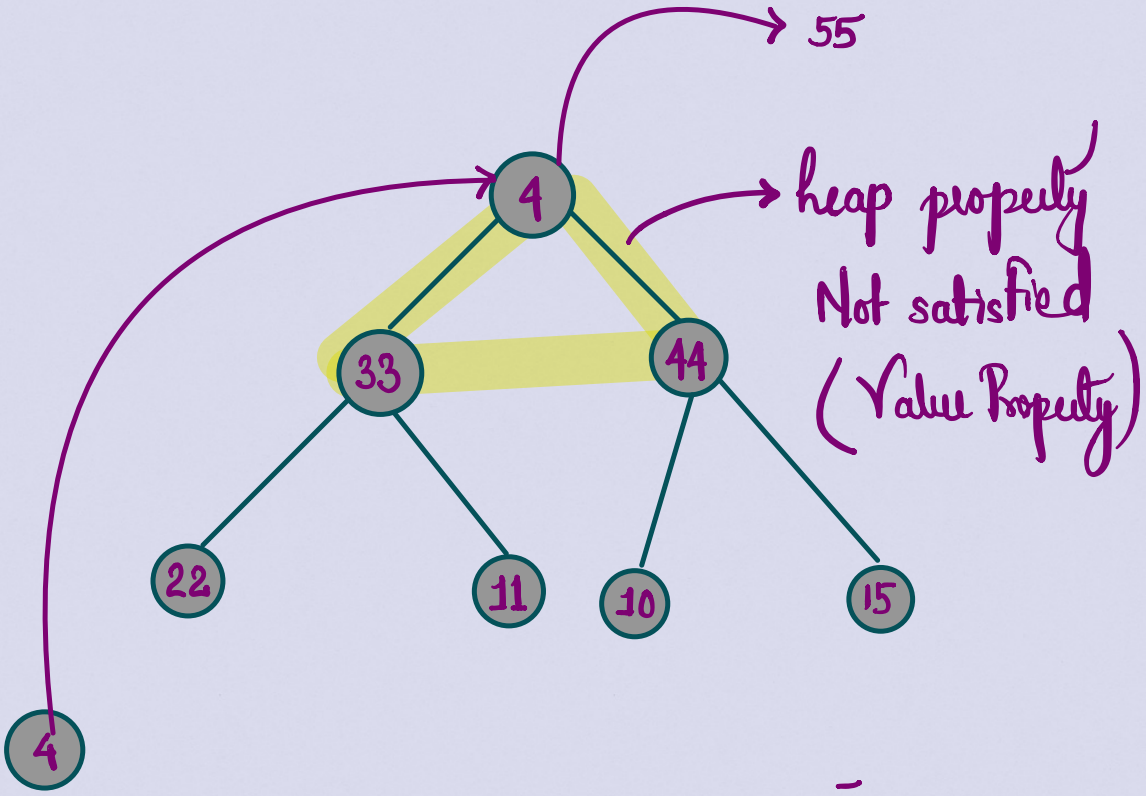
Delete-Max operation for Binary Max-heap.

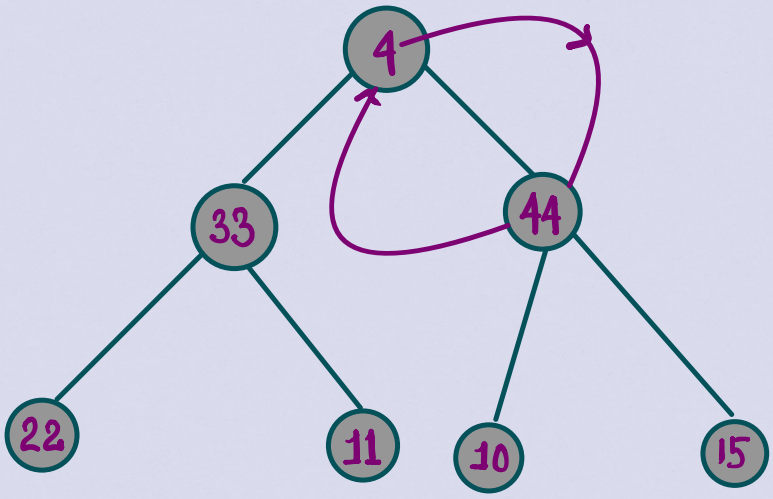
Delete-Min operation for Binary Min-heap.

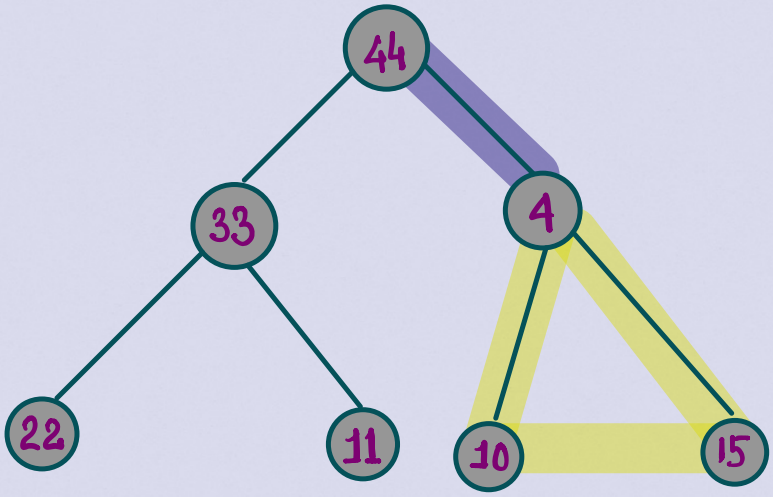
Maximum value at the root } Max-heap

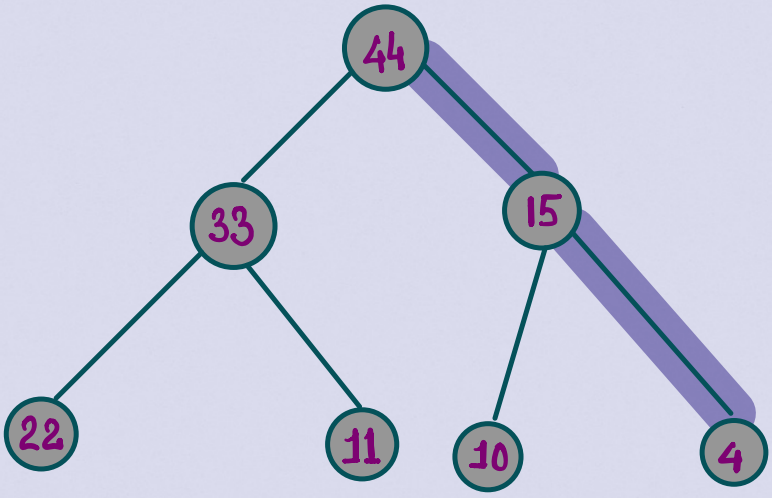
Minimum value at the root } Min-heap




55
4
heap property
Not satisfied
(Value Property)
33
44
22
11
10
15
4


4
33
44
22
11
10
15


44
33
4
22
11
10
15


44
33
15
22
11
10
4

Restore the heap property from root downwards towards leaf.

root $\longrightarrow$ leaf.

Time Complexity = $O(\log|N|)$