Operations on heap data structures—

1. Heapify
   ↓
   The process of creating heap from an array.

2. Insertion
   ↓
   Add an element in the existing heap.

3. Deletion
   ↓
   Deletes an element from the

existing heap.

4. Peek

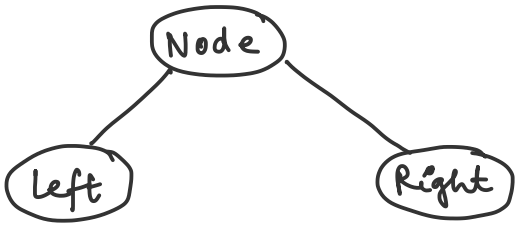
↓

To pick first element from the heap.

Implementation of heaps-

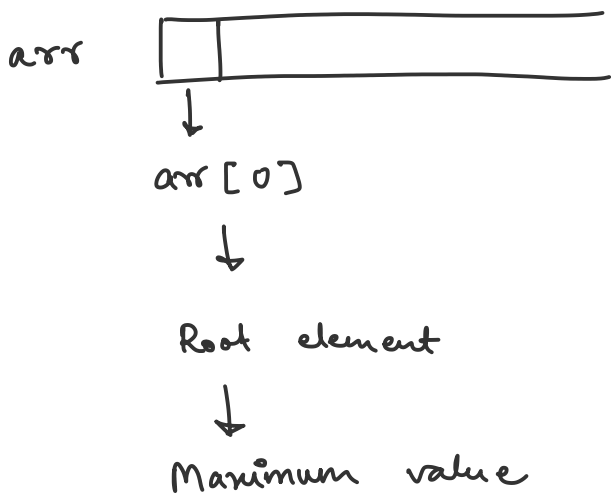
1. Tree method

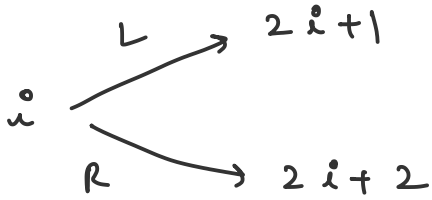
↳ Node

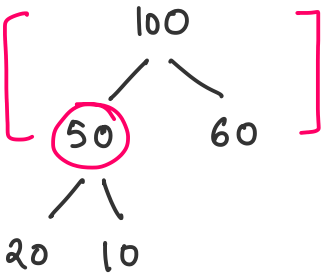


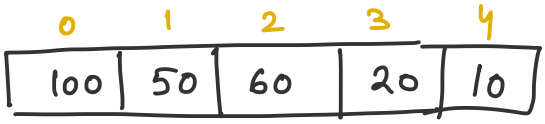
2. Arrays

Ex- Max heap









Root → At index 0

arr [0] = 100

Left child = 2 i + 1

= 2 * 0 + 1 = 1

Right child = 2 i + 2

= 2 * 0 + 2 = 2

For sub tree starting with 50

i = 1

left child = 2 i + 1

$= 2*1+1 = 3$

Right child $= 2i + 2$

$= 2*1 + 2 = 4$

Ex - Min Heap

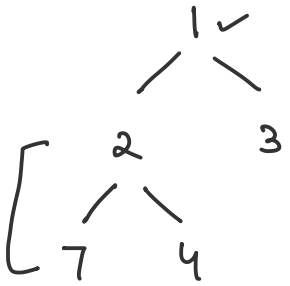
arr [0]

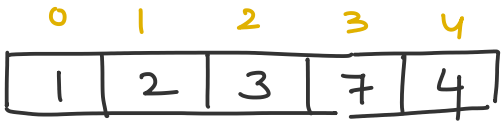
↓

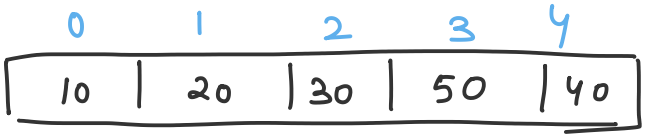
Root node

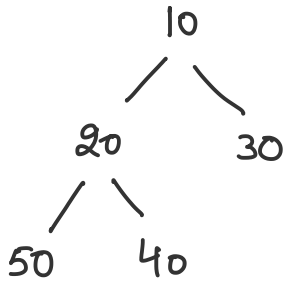
↓

Min value









$i = 1$

$L = 2i + 1$

$= 3$

$R = 2i + 2$

= 4

Insertion

Ex- Min Heap

Insert (15)

Insert (22)

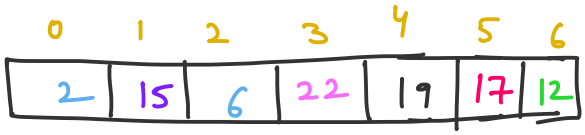
Insert (17)

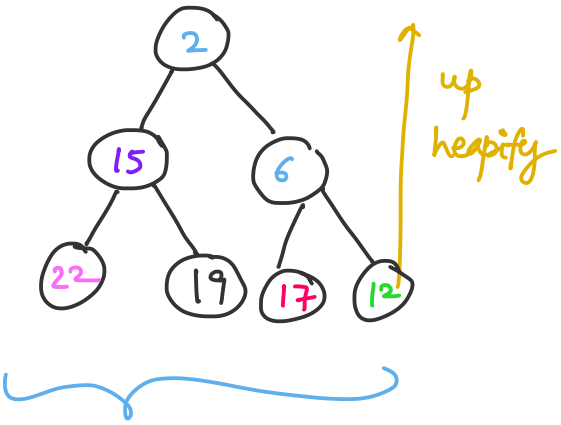
Insert (6)

Insert (19)

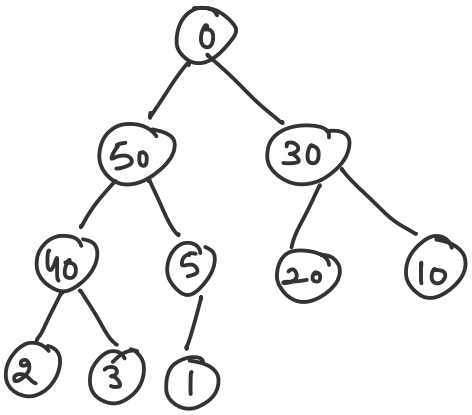
Insert (12)

Insert (2)





Eg-

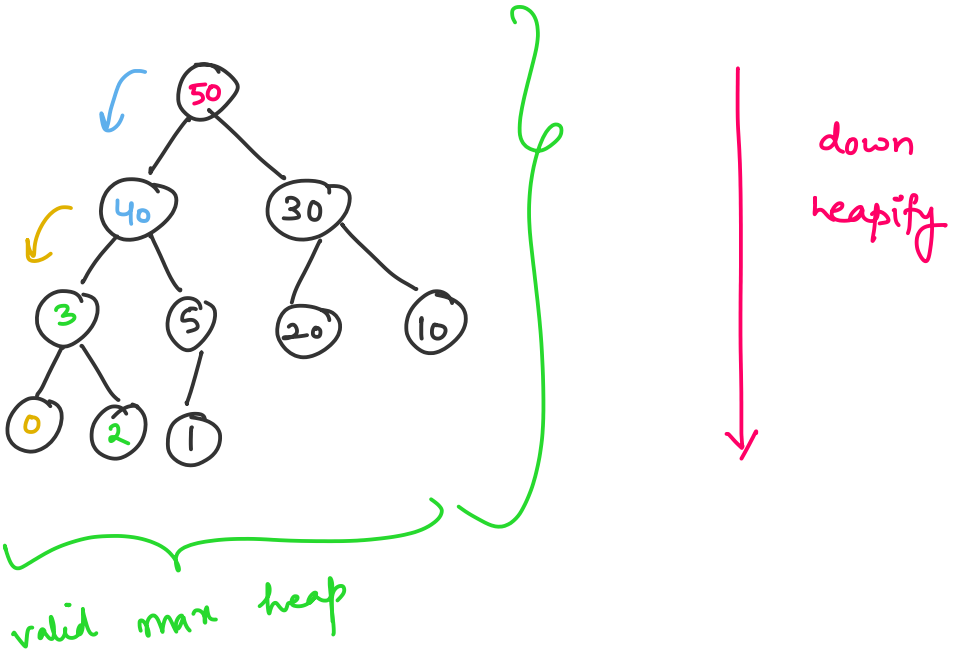


for a max heap to be valid,

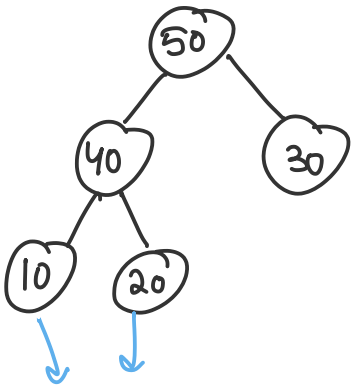
the priority of the parent node should be greater than the priority of the child node.

$$(\text{Priority})_{\text{parent}} > (\text{Priority})_{\text{child}}$$

Q Suppose left subtree and right subtree are max heaps but whole tree is not max heap then how to resolve it?

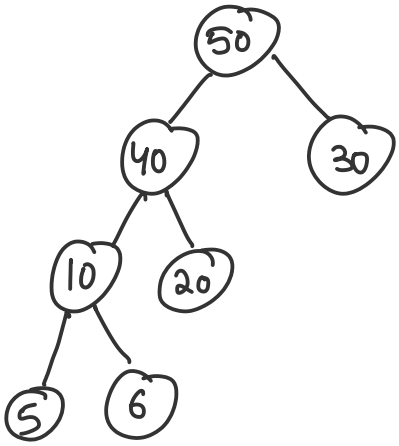


Deletion from heap

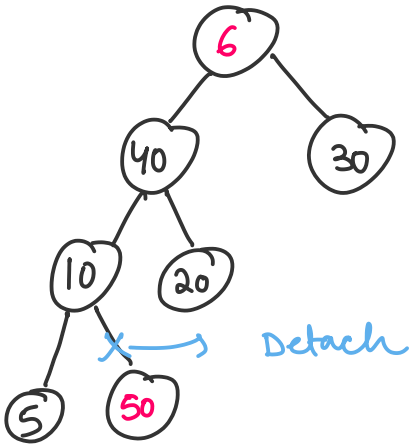


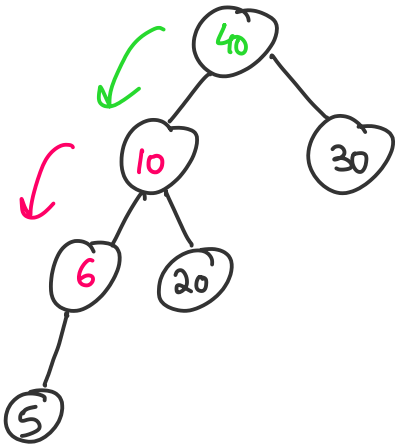
safest to remove because they don't hamper binary heap property

Remove root node = 50



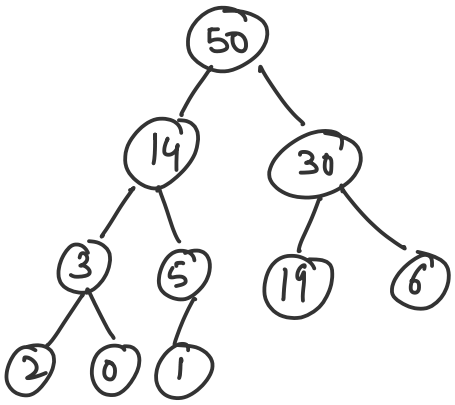
swap root node with last child node





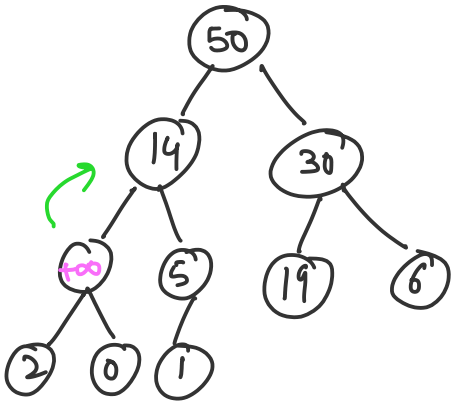
down heapify

Q How can we remove any element from the heap?

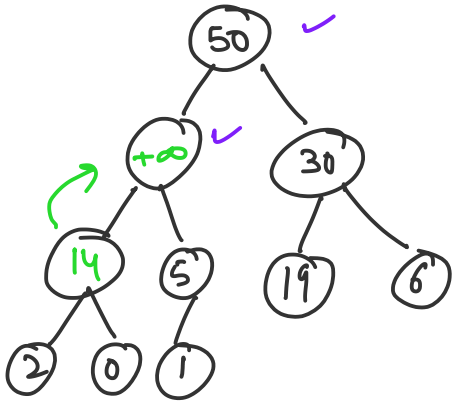


Remove 3

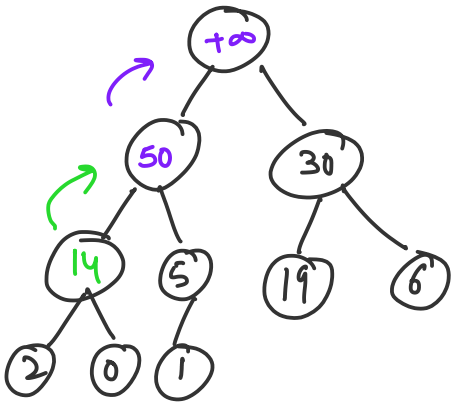
Replace the element to be removed with $+\infty$.



Swap $+\infty$ with 14



swap 50 with $+\infty$



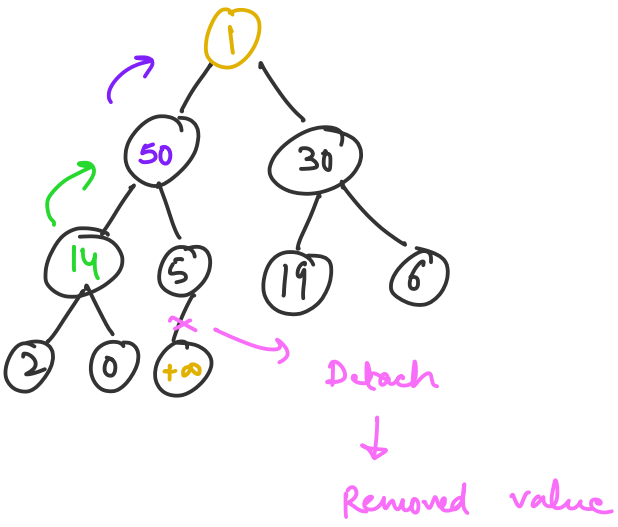
3 was replaced with $+\infty$

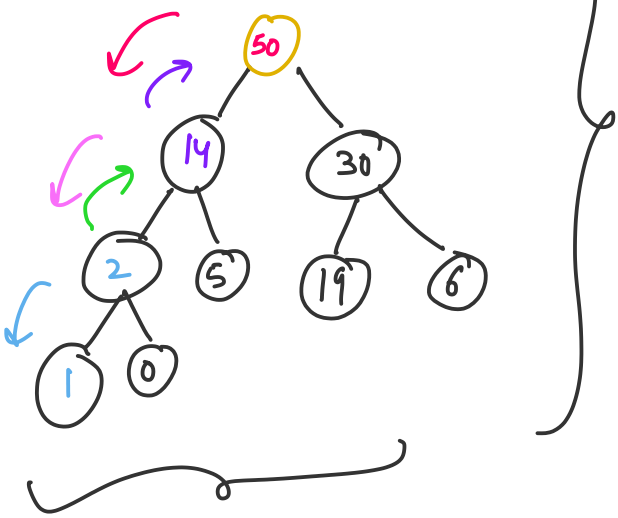
+∞ is at root node.

Remove root node.

↓

Swap it with the last child node.





Valid max heap

↓

up heapify

+

down heapify

Heap sort algorithm

↓

Comparison based sorting technique.

Steps -

1. Build heap from the given array.
2. Repeat the following steps until the heap contains only one element -

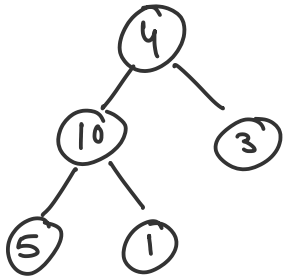
(i) Swap the root element with the last element of the heap.

(ii) Remove last element from the heap.

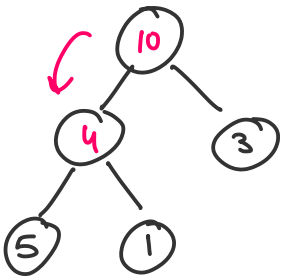
(iii) Heapify the remaining elements in the heap.

Eg-

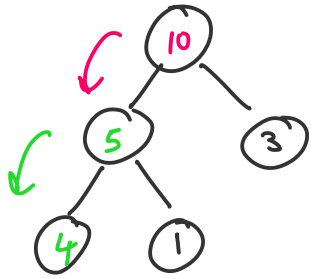
arr[] = {4, 10, 3, 5, 1}



↓ convert into max heap

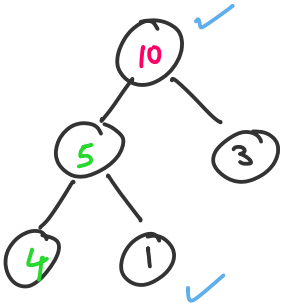


↓



Max heap

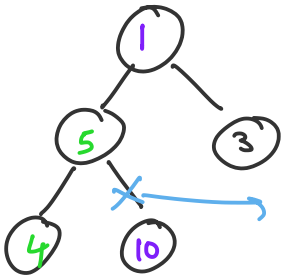
Perform heap sort



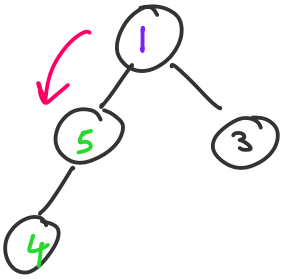
Max element = 10

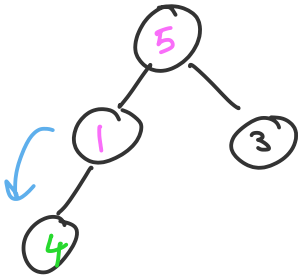
↓

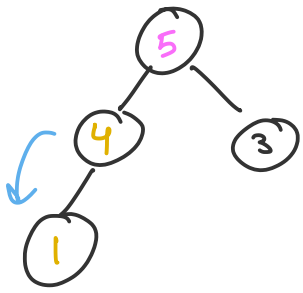
Remove it



Detach







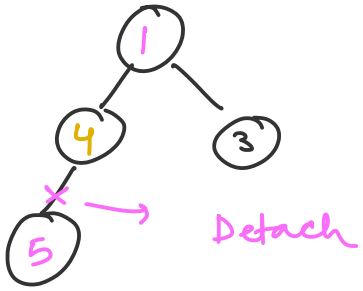
Max element = 5

↓

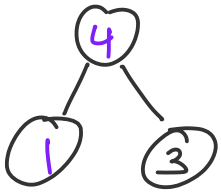
Remove it

10	5

↓



↓



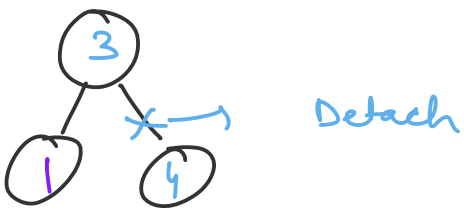
} Max heap

Max element = 4

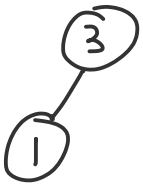
↓

Remove it

10	5	4



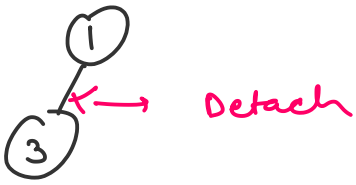
↓

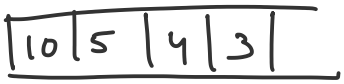


Max value = 3

↓

Remove it

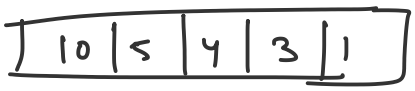




↓

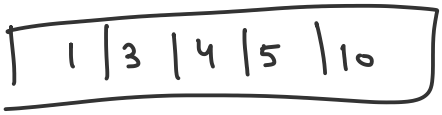
↓

Remove



↓

Reverse the order

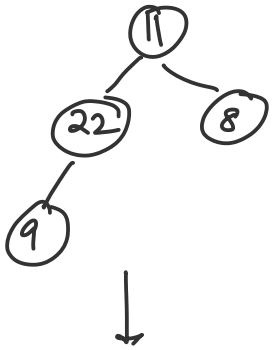


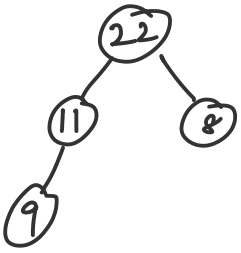
Kth largest elements

11, 2, 1, 3, 6, 5, 9, 22, 4, 8

$k = 4$

11, 22, 8, 9 — Max heap → 4





Max heap

(1.) Max heap of size k

(2.) Iterate over the array to check if we have any larger value.

(3.) Max heap obtained.

(4) Remove + Heapify