Heap sort

1. Create a tree
2. Heapify
3. Delete the element one by one

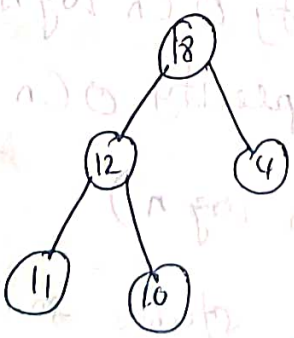
Heap is a complete binary tree or almost complete binary tree in which every parent node be either greater or lesser than its child node.

A Heap can be min heap or max heap.

Max heap :-

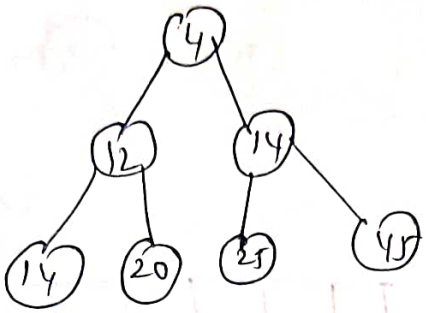
It is a tree in which a value of each node is greater than or equal to the value of child node.

Eg



Min heap :-

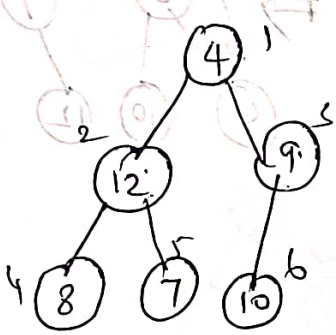
It is a tree in which the value of each node is less than or equal to the value of its child node.



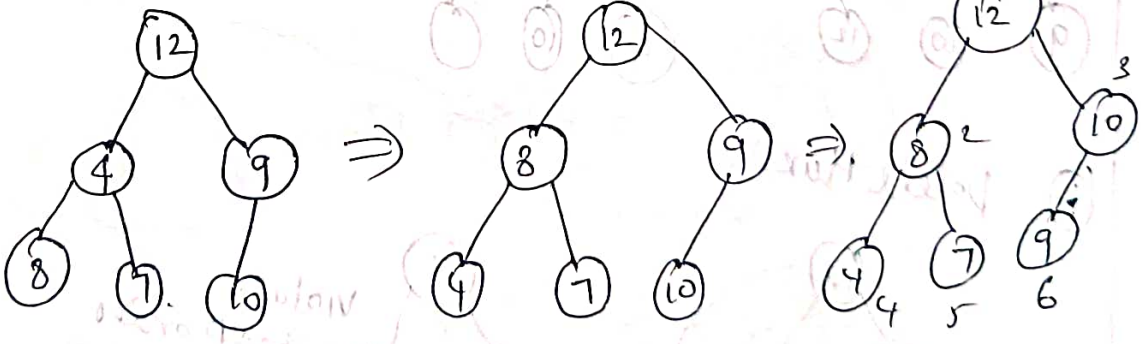
Sort the given elements using heap

Sort 4, 12, 9, 8, 7, 10

① create a tree

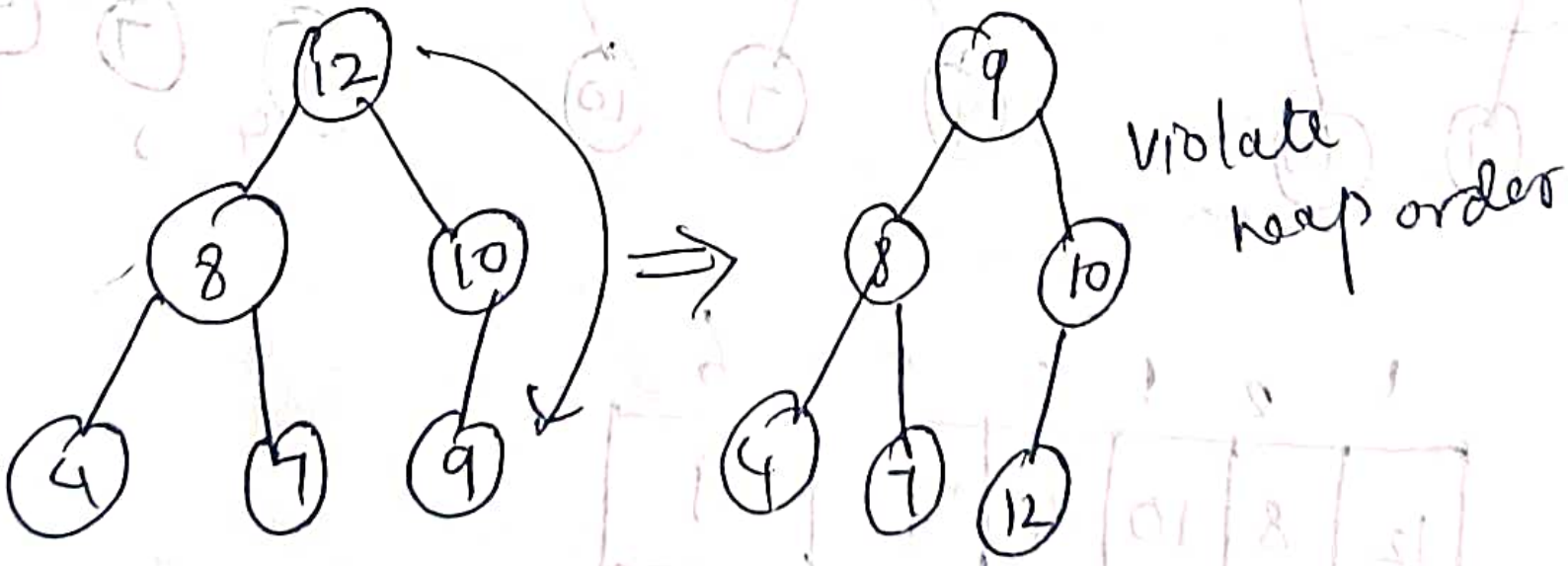


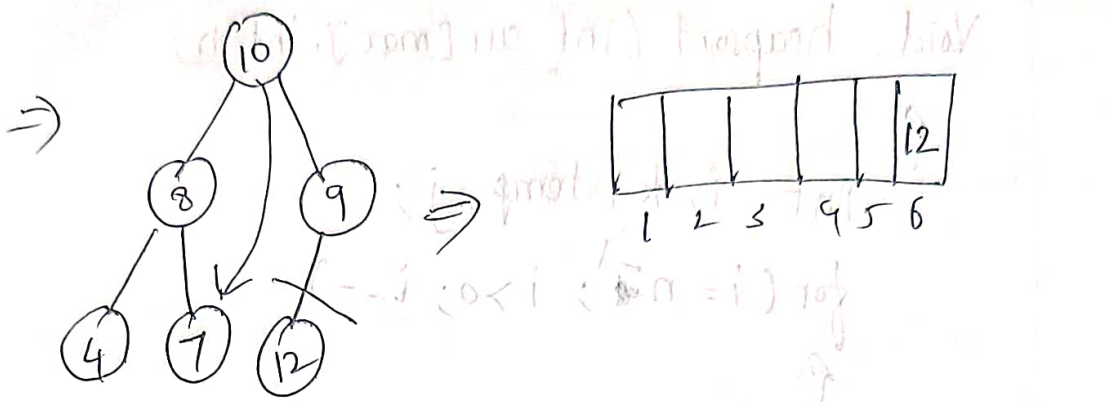
② Heapify as max heap

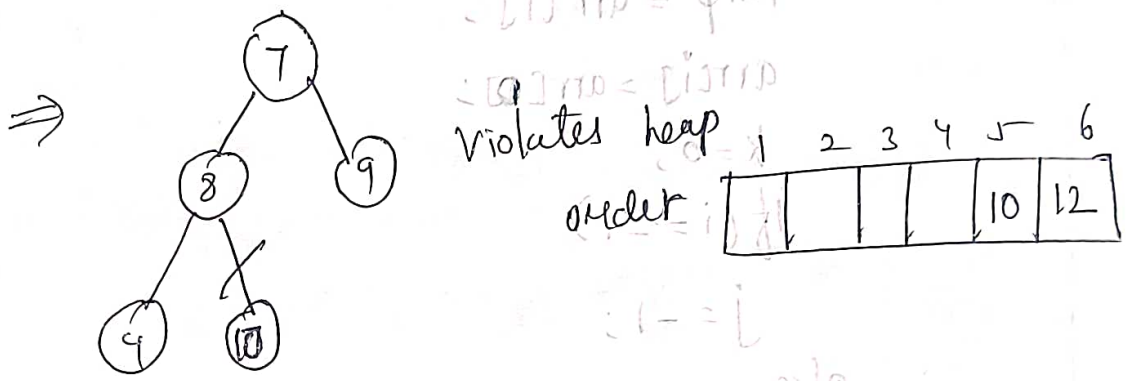


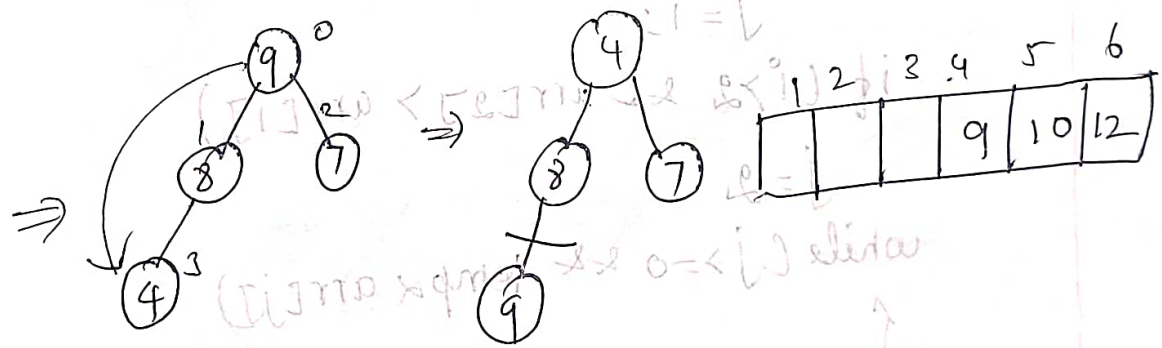
1	2	3	4	5	6
12	8	10	4	7	9

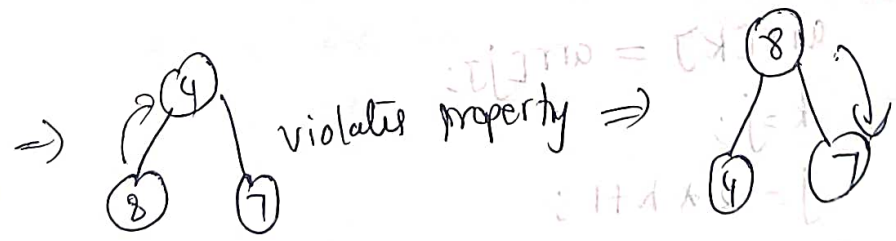
1. Delete Max

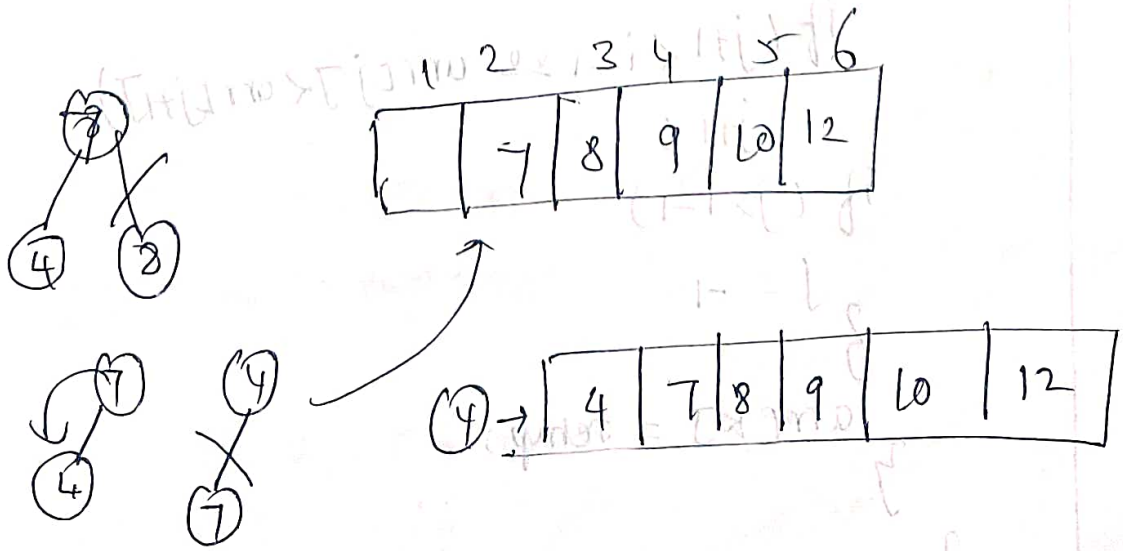



10
8
9
4
7
12
12
1 2 3 4 5 6

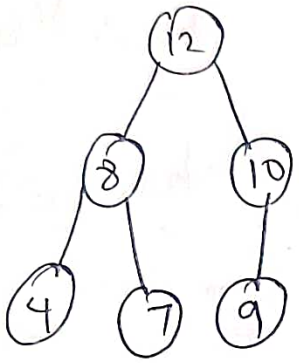

7
8
9
violates heap order
1 2 3 4 5 6
10
12


9
8
7
4
4
8
7
9
1 2 3 4 5 6
9
10
12


violates property
8
9
7


4
8
4
7
7
1 2 3 4 5 6
7
8
9
10
12
4
7
8
9
10
12

```
Void heapsort (int arr[maxJ, int n)
{
  int i, k, temp, j;
  for (i = n-1; i>0; i--)
  {
    temp = arr[i];
    arr[i] = arr[0];
    k=0;
    if(i==1)
      j = -1;
    else
      j = 1;
  if (i>2 && arr[2] > arr[1])
    j=2
  while (j>=0 && temp< arr[j])
  {
    arr[k] = arr[j];
    k=j;
    j= 2*k+1;
    if (j+1 < i-1 && arr[j]< arr[j+1])
      j++;
    if (j> i-1)
      j = -1
  }
  arr[k] = temp;
}
}
```



0	1	2	3	4	5
12	8	10	4	7	9

① $i = n-1 = 5 \quad 5 > 0$

temp = arr[5]

temp = 9

arr[5] = arr[0] = 12

if (i == 1) false

else

j = 1

$10 > 8$

5 > 2 && arr[2] > arr[1] (True)

j = 2

arr[

2 >= 0 && 9 < arr[2] (True

2 >= 0 && 9 < 10

arr[0] = arr[j]

arr[0] = 10

k = 2

j = 2 * k + 1

j = 2 * 2 + 1

j = 5

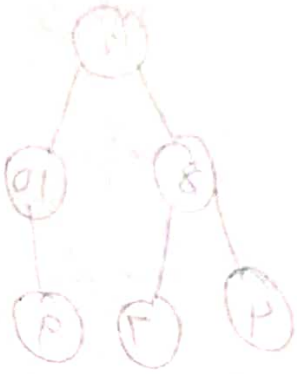
j + 1 6 < 5 - 1 false

if (5>4)

j = -1

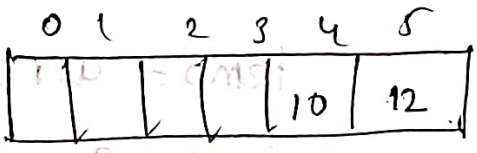
-1 >= 0 false.

arr[0] = 9



(2) i = 4

temp = arr[4] = 7



arr[4] = arr[0]

arr[4] = 10

k = 0

i == 1 false

else

j = 1

9 7 8

4>2 && arr[2] > arr[1] false True

j = 2 2<4 True

4>=0 && 7 < arr[2]

arr[0] = arr[2]

arr[0] = 9

k = 2

j = 2 * k + 1

j = 2 * 2 + 1

j = 5

6 < 3 false

5 > 3

j = -1

arr[2] = 7

Time complexity

Best case - O(n log n)

Worst case - O(n log n)

Average case - O(n log n)