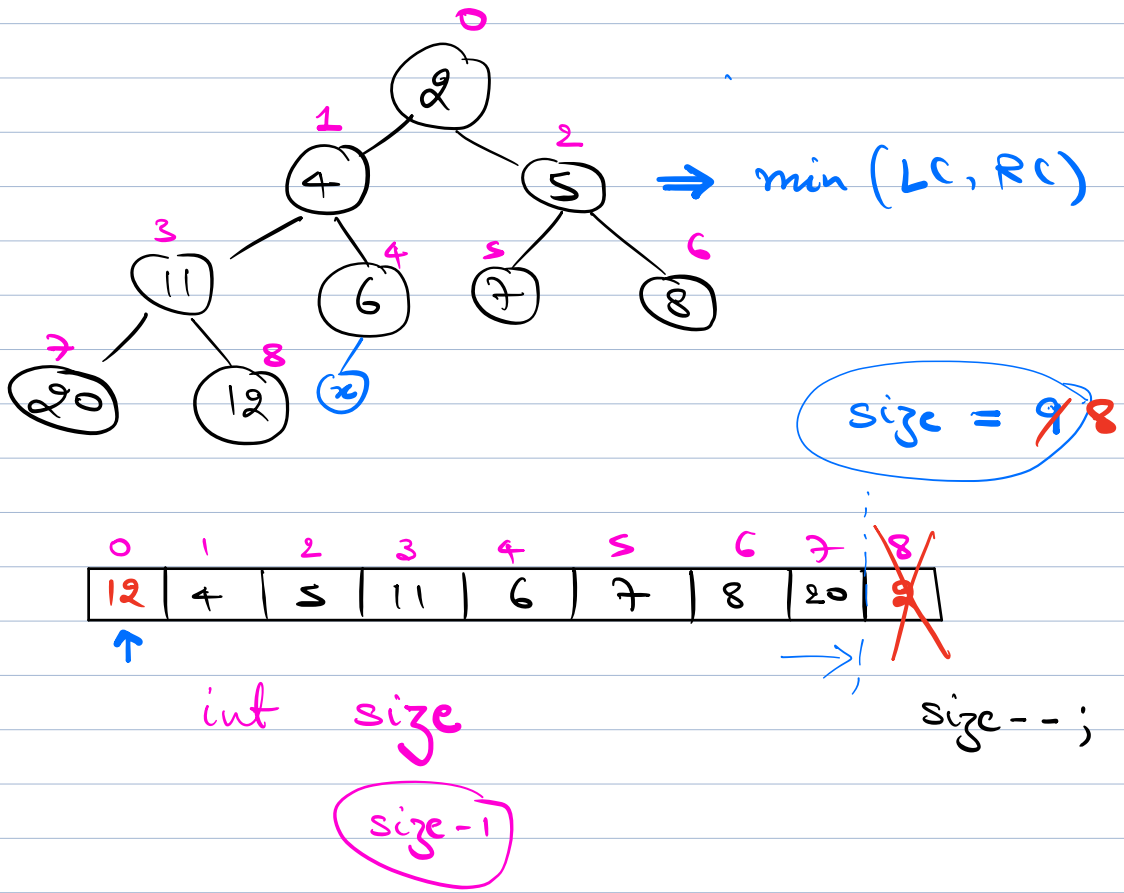
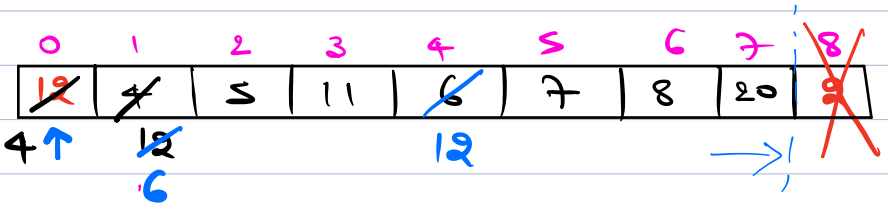


Code

size

```
int deleteMin ( arr ) {
    swap ( size-1 , 0 );
                ↓
              last
    int min = arr [size-1];
    size --;
    downHeapify ( arr , 0 );
    return min;
```



idx	lc (2i+1)	rc (2i+2)
0	1	2
1	3	4
4	9	10

[0, size-1]

[0, 7]

Code

```
void downHeapify (arr, idx) {
    int lc = 2*(idx)+1
    if (lc >= size) {
        return;
    }
    int rc = 2*idx + 2
    if (rc >= size) {
```

```
        if (arr[idx] > arr[lc]) {
            swap (idx, lc);
        }
        return;
    }

    if (arr[idx] <= arr[lc] &&
        arr[idx] <= arr[rc]) {
        return;
    }

    if (arr[lc] <= arr[rc]) {
        swap (idx, lc);
        idx = lc;
        downHeapify (arr, idx);
        return;
    }
    else {
        swap (idx, rc);
        downHeapify (arr, rc);
        return;
    }
  }
}
```

Build Heap

N Elements. array.

1) Take an Empty Heap

Perform N insertions.

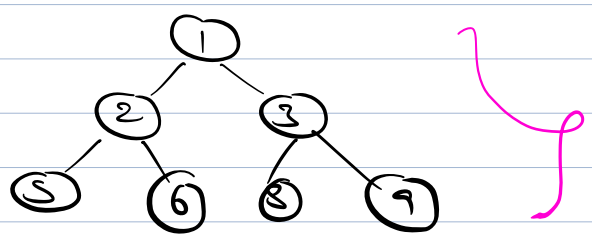
T.C. = $O(N \log N)$

S.C. = $O(N)$

2) Sort the given array.

2, 8, 9, 1, 3, 5, 6

1, 2, 3, 5, 6, 8, 9



T.C. = $O(N \log N)$

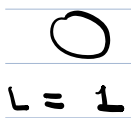
S.C. = $O(1)$ → Sorting the I/P array.

(3)

Complete BT with N nodes.

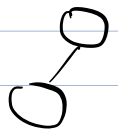
No. of leaf nodes ???

N = 1



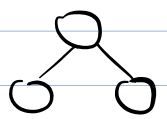
L = 1

N = 2



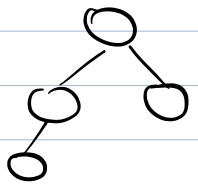
L = 1

N = 3



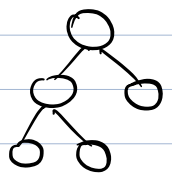
L = 2

N = 4



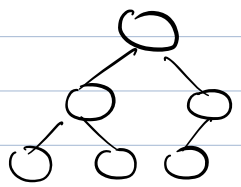
L = 2

N = 5



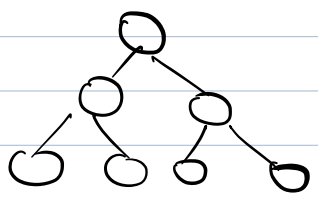
L = 2

N = 6



L = 3

N = 7



L = 4

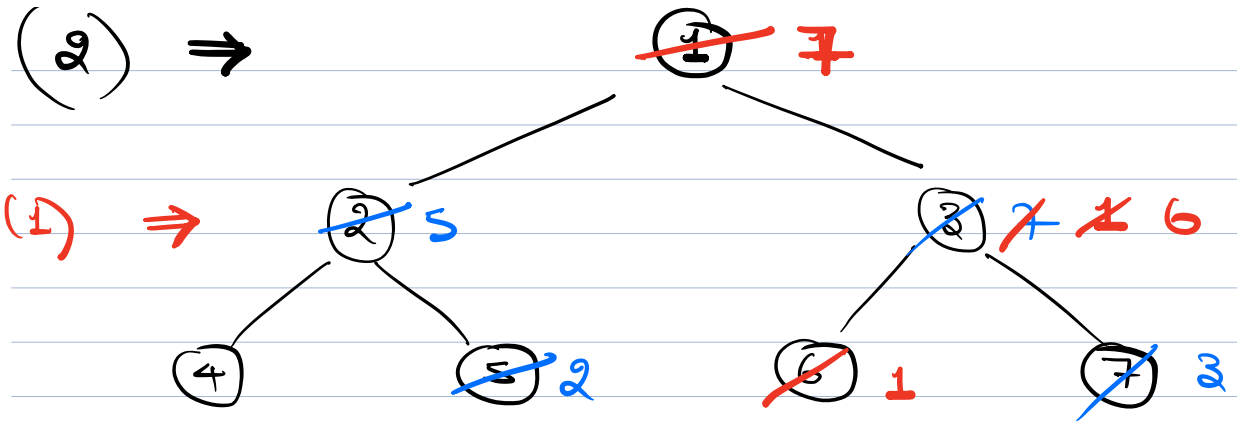
Leaf nodes = $\left(\frac{N+1}{2}\right)$ or $\lceil \frac{N}{2} \rceil$

Leaf nodes.

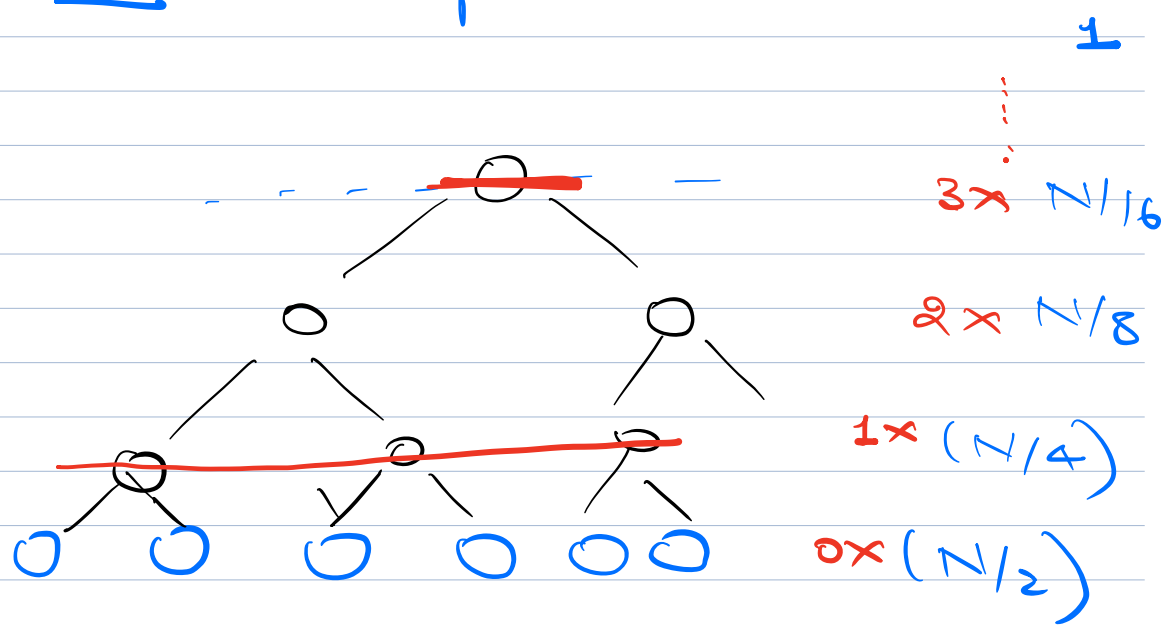
I/P : 1, 2, 3, 4, 5, 6, 7

Create a max Heap. N = 8

LN = 4



N nodes (Perfect)



#swaps

$$= 0 \times \frac{N}{2} + 1 \times \frac{N}{4} + 2 \times \frac{N}{8} + 3 \times \frac{N}{16} \cdots \infty$$

$$= \sum_{i=0}^{\infty} i \times \frac{N}{2^{(i+1)}}$$

$$\frac{N}{2}\left(\underbrace{\sum_{i=0}^{\log N}\frac{i}{2^i}}_{2}\right)$$

$$\frac{N}{2}\times 2 = N$$

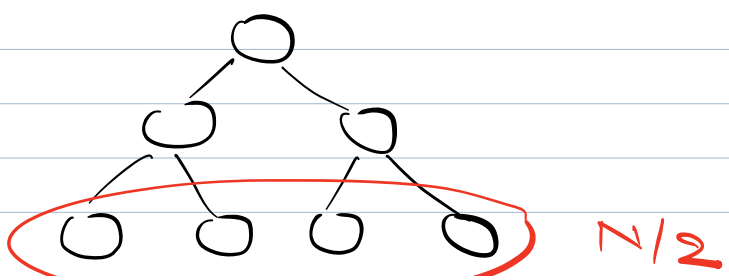
T.C. = $O(N)$

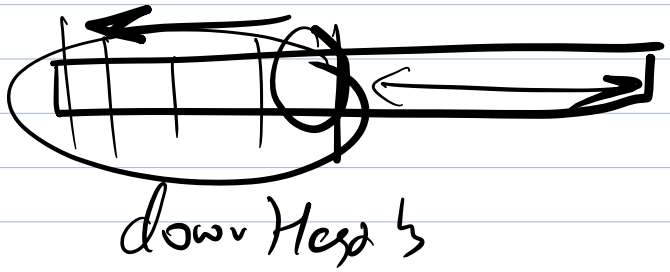
N

( ) ( )

N/2

```
for (i = N/2 ; i >= 0; i--) {
    doownHepify (arr, i);
}
```




down Head