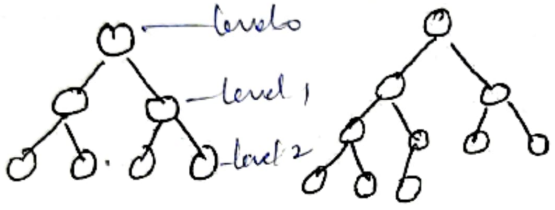
Heap Sort

Heap

Heap is a binary tree with 2 important Properties

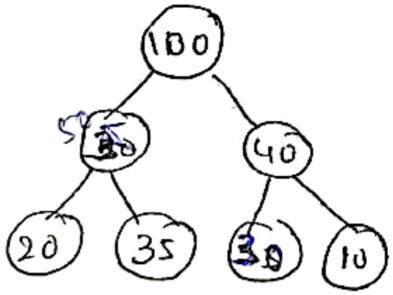
Structural property

(i) It Should be an almost complete binary tree. apart from last level, all other level should have 2^i nodes. and last level should be left filled.

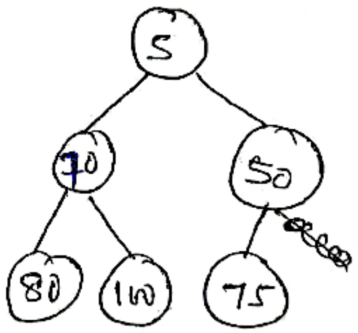


(ii) Parent dominantal property. The parent when compared should be more dominant to its children. (Max or Min heap)

Max heap.



Min heap



Methods of Constructing Heap

(i) Bottom up approach
(ii) Top down approach.

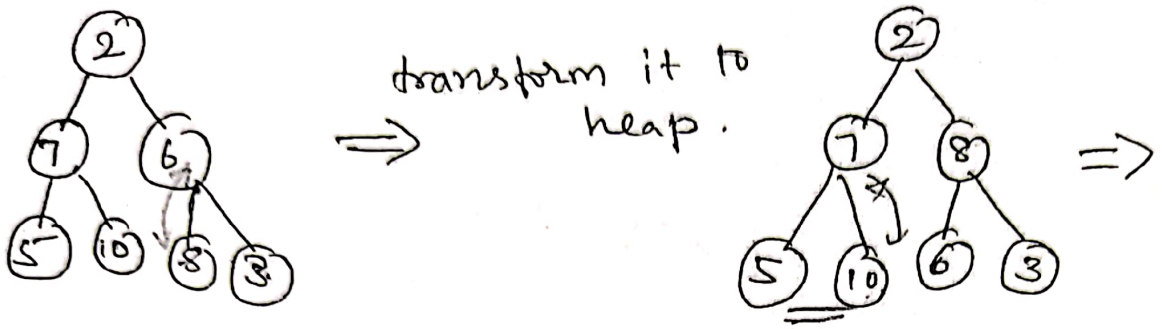
For node i,
left child will be at 2*i
right child will be at 2*i+1

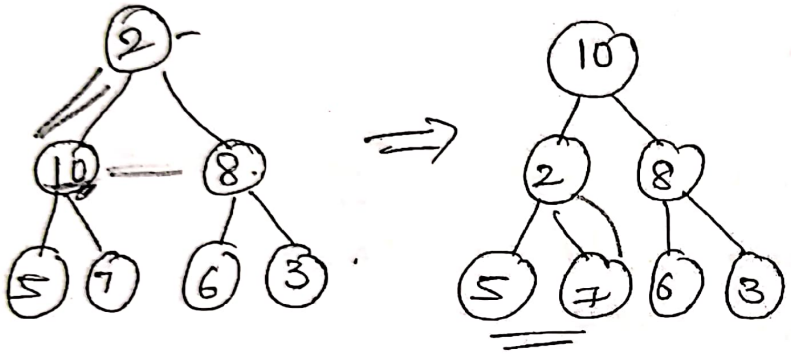
objective is to Construct max heap.

Bottom up approach

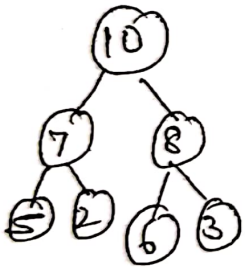
2 7 6 5 10 8 3

① Construct a Binary tree

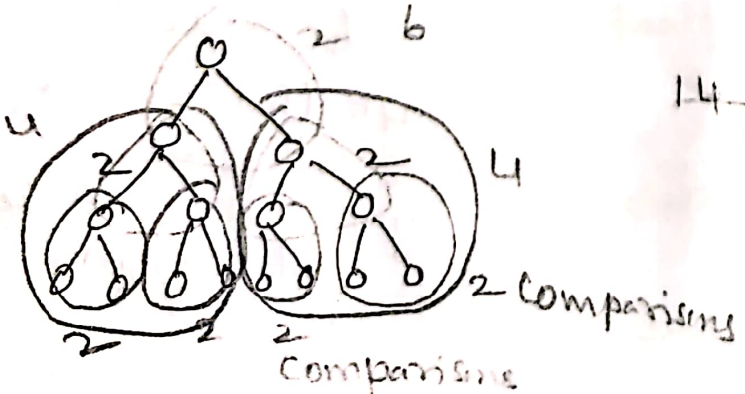




Take a non leaf node.
Compare its leaf nodes
then compare greatest leaf with root and then swap.



Max heap.



14 comparisons
Total no of nodes is 15
22 Comparisons

$2n - (\log_2 n + 1)$

n — discard

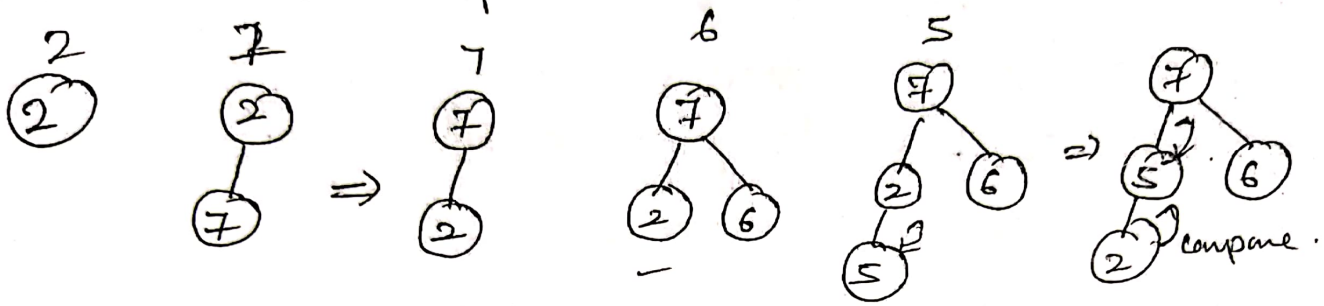
Left node is compared with right node, then whichever node is greater is compared with root node. ∴ It is 2 comparison.

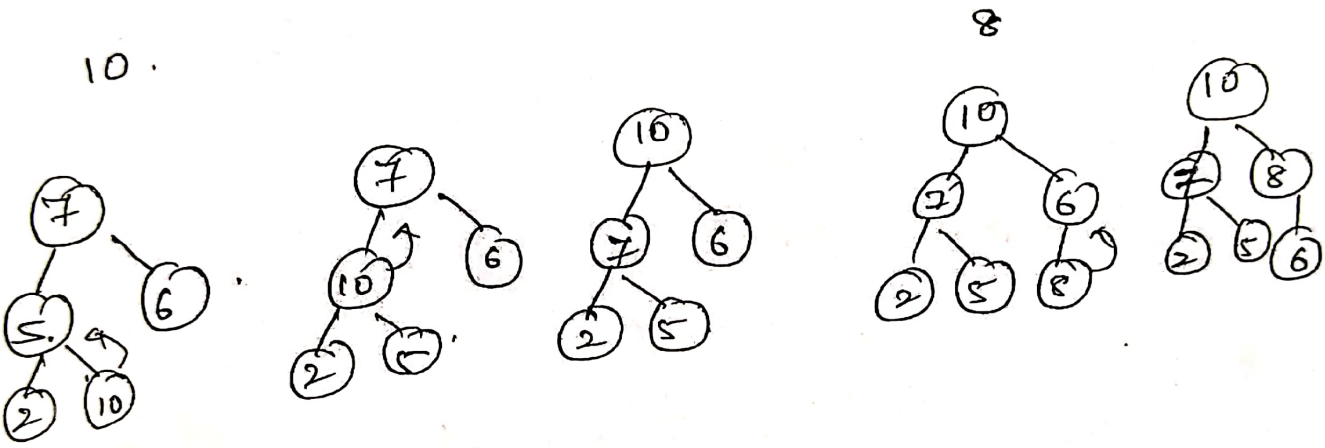
Time Taken — $O(n)$ to construct heap using bottom up

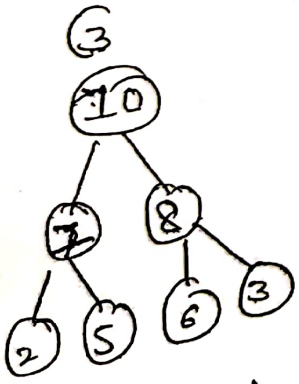
Top down approach

2 7 6 5 10 8 3.

Construct max heap.







max heap.

Top down approach — $\log n$.
is the efficiency.

For complete binary tree — height of the tree is $\log n$.

Heap Sort.

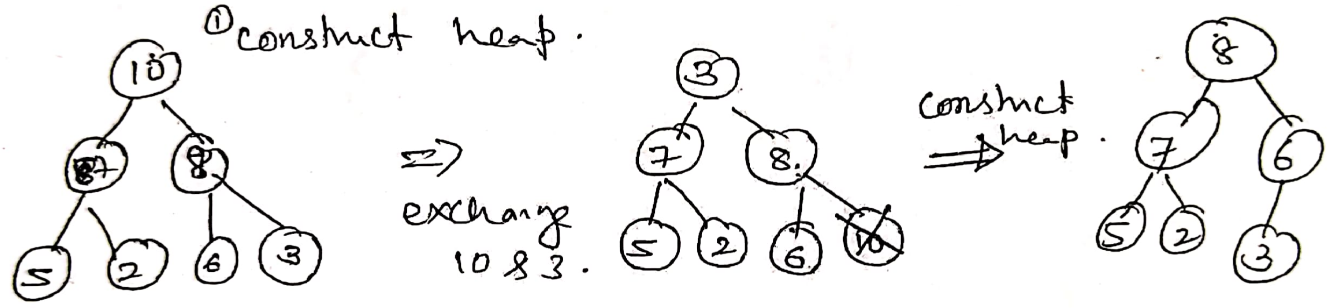
Procedure to perform Heap Sort

1. Construct a max heap for the given set of elements (n)
2. Exchange the first and last element, reduce the size of heap by 1 & construct heap for remaining (n-1) elements
3. Repeat step 2 until only one element is left ~~out~~ out.

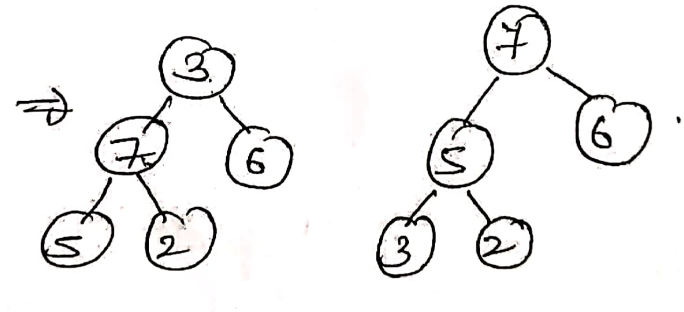
Example

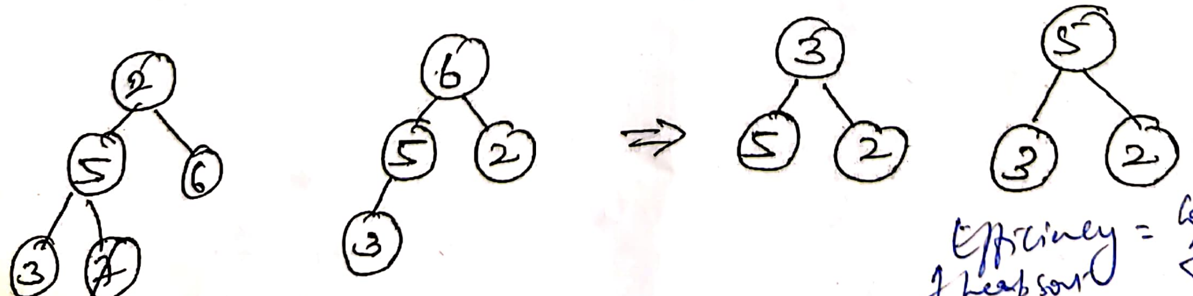
2 7 6 5 10 8 3

① construct heap.



1 2 3 4 5 6 7.
2·3 5 6 7 8 10



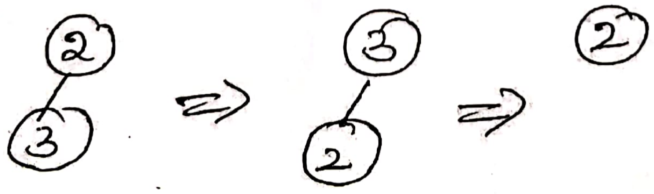


Efficiency of heap sort = Construction of heap + Exchange 1st & last element

$\Theta(n)$ using bottom up approach

$(n-1)\log n$ after exchange we need to heapify, for which we need to do comparisons, which depends on no. of comps on the tree height i.e. $\log n$.

⇒ ∴ $n\log n$ by neglecting constant



$(n-1)\log n$

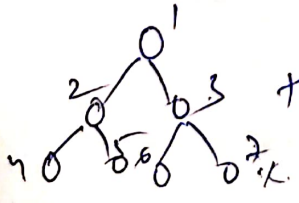
$\Rightarrow n\log n$

2 3 5 6 7 8 10 Sorted.

Algorithm

Algorithm Heapify(H[1..n])
// Constructs a max heap using bottom up approach
// input: An array H of n elements

for $i \leftarrow \lfloor n/2 \rfloor$ down to 1 do (no of iterations depends on no. of non leaf nodes)
    $k \leftarrow i$
    $v \leftarrow H[k]$ - Intial node
    heap $\leftarrow$ FALSE
    while not heap and $2k \leq n$ do
        $j \leftarrow 2k$
        if $j < n$
            if $H[j] < H[j+1]$ → Compare left & right child of non-leaf node
                $j \leftarrow j+1$
        if $v \geq H[j]$ (Compare non leaf node with its greater child if non leaf node is already greater than it is already a leaf)
            heap $\leftarrow$ TRUE
        else
            $H[k] \leftarrow H[j]$ ← (if not exchange non leaf node & with its greater child)
            $k \leftarrow j$
        $H[k] \leftarrow v$



If $n = 7$ (no of nodes)
then no of non leaf nodes $= \lfloor \frac{n}{2} \rfloor$
$= \lfloor \frac{7}{2} \rfloor = \lfloor 3.5 \rfloor = 3$

If position of a node is i
then position of its left child is at $2*i$
& position of its right child is at $2*i+1$.

Program

```
#define MAX 100
int main()
{
    int i, n, a[max];
    Pf("Enter value % N");
    Sf("%d", &n);
    for(i=1; i<=n; i++)
        Sf("%d", &a[i]);
    heapify(a,n);
    heapsort(a,n);
    Pf('Sorted elements are");
    for(    )
    Pf(" %d", a[i]);
    } return(0);
```

```
Void heapify(int a[max], int n)
{
    int i,j,k,v, flag;
    for(i= n/2; i>=1; i--)
    {
        k=i;
        v= a[k];
        flag = 0;
        while(!flag && 2*k <=n)
```

```
{
  j = 2*k;
  if (j < n){
      if (a[j] < a[j+1])
        j = j+1
    }
  if( v > a[j] )
    flag = 1;
  else
  {
    a[k] = a[j];
    k = j;
  }
}
a[k] = v;
}
}
```

```
heapsort(int a[MAX], int n)
{
  int i, temp;
  for(i = n; i > 1; i--)
  { temp = a[1];
    a[1] = a[i];
    a[i] = temp;
    heapify(a, i-1);
  }
}
```

Examples

① 4 3 7 1 8 5

② 10, 3, 76, 34, 23, 32.

③ 12, 3, 9, 14, 10, 18, 8, 23

④ 2, 3, 7, 1, 8, 5, 6

⑤ 9, 22, 7, 8, 15, 25.