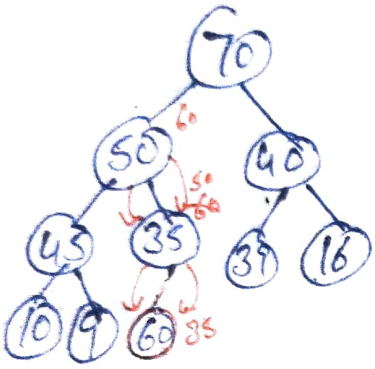
Heap

A tree based data structure and tree is a almost complete binary.

Max heap

For every node u, the value of the node is less than or equal to its parent.

$A[Parent(u)] \geq A[u]$

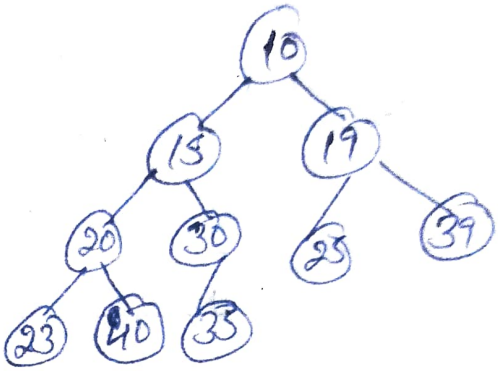


70	50	40	45	35	39	16	10	9	60
1	2	3	4	5	6	7	8	9	10

Min. heap

For every node u, the value of node is greater than or equal to its parent

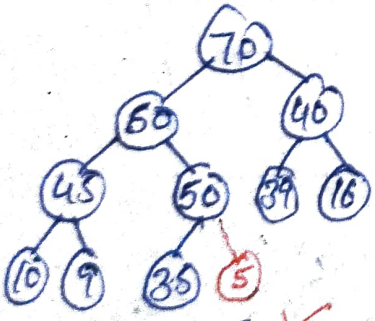
$A[Parent(u)] \leq A[u]$



10	15	19	20	30	25	39	23	40	35
1	2	3	4	5	6	7	8	9	10

Insert 60

Insertion is always at leaf.

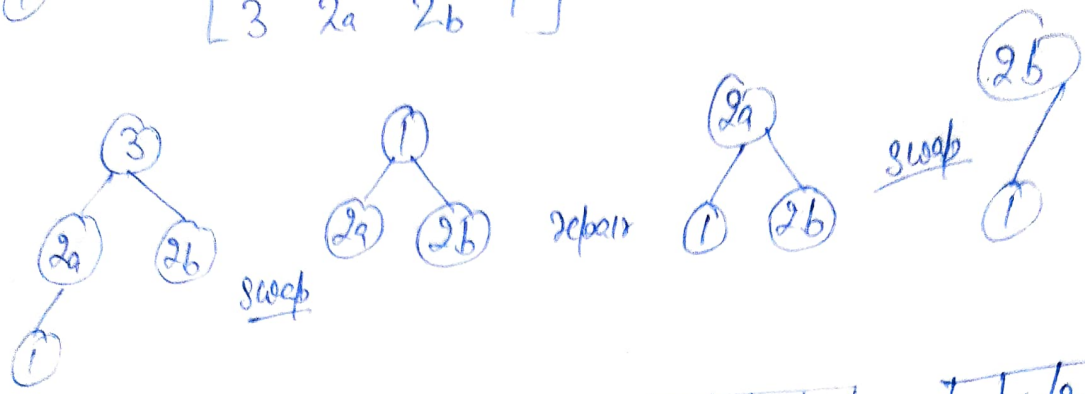


Insert 5

```
Insertheap (A, N, Value)
{
    n = n+1
    A[n] = value
    u = n
    while (u > 1)
    {
        Parent = ⌊u/2⌋
        if (A[Parent] < A[u])
        {
            swap (A[Parent], A[u])
            u = parent
        }
        else
        {
            return;
        }
    }
}
```

Stability of Heap Sort -

① [3 2a 2b 1]



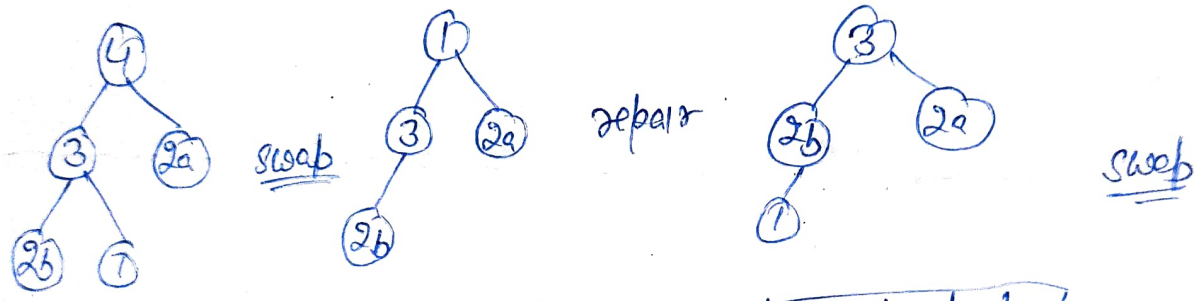
3	2a	2b	1

1	2a	2b	3

2a	1	2b	3

2b	1	2a	3

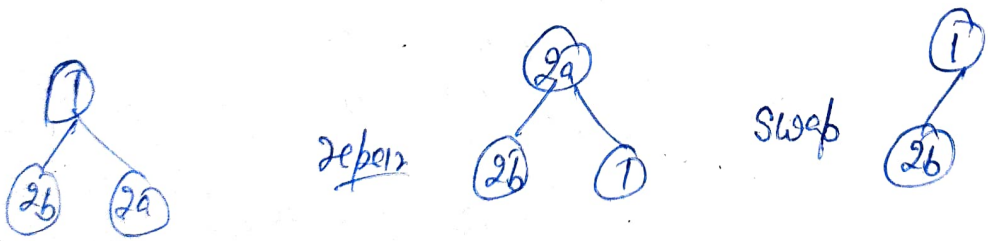
② [4, 3, 2a 2b, 1]



4	3	2a	2b	1

1	3	2a	2b	4

3	2b	2a	1	4



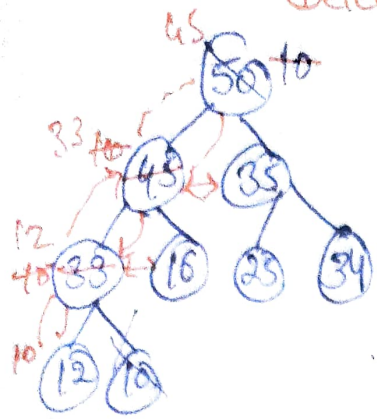
1	2b	2a	3	4

2a	2b	1	3	4

1	2b	2a	3	4

Deletion from Heap —

Delete the root



Delete Top to bottom.

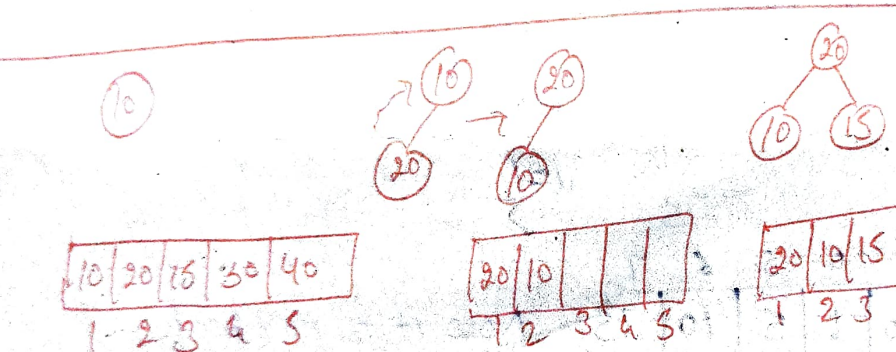
A	~~50~~	45	35	33	16	25	34	12	10
	1								58

45 33 35 12 16 25 34 12

$O(n^2)$

---

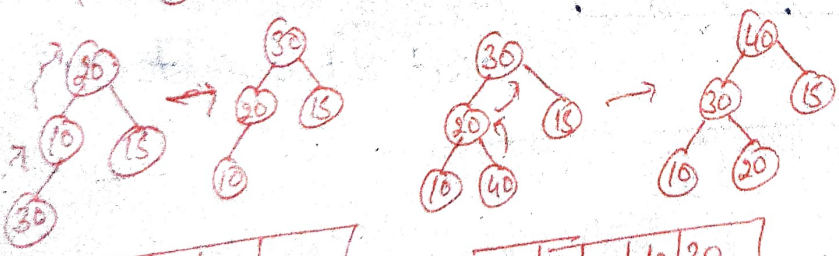
n log n



10	20	15	30	40
1	2	3	4	5

20	10			
1	2	3	4	5

20	10	15		
1	2	3	4	5



Delete

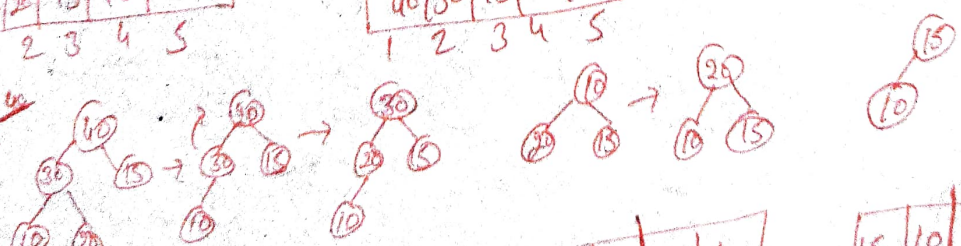
n log n

= 2 n log n

30	20	15	10	
1	2	3	4	5

40	30	15	10	20
1	2	3	4	5

Delete



30	20	15	10	40
1	2	3	4	5

20	10	15	30	40
1	2	3	4	5

15	10	20	30	40

10	15	20	30	40

① Suppose we are ~~for~~ sorting an array of eight integers using heapsort and we have just finished some heapify operations the array now looks like 16 14 15 10 12 27 28. How many heapify operations have been performed on the root of heap

Ans 2

② Which of the following sorting algorithms in its typical implementation gives best performance when applied on an array which is sorted or almost sorted (max 1 or 2 elements are out of place)?

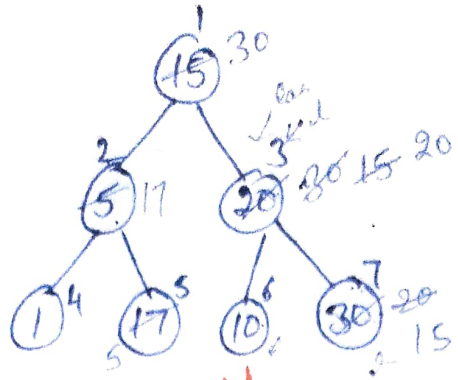
Quick

Merge

Heap

Insertion

Heapify



15	5	20	1	17	10	30
1	2	3	4	5	6	7

In a complete binary ~~tree~~ the leaf node would be from

$\lfloor \frac{n}{2} \rfloor + 1$ to n

So we will start heapify from $\frac{n}{2}$

1	2	3	4	5	6	7
~~15~~	~~5~~	~~20~~	1	~~17~~	10	~~30~~

So will start with $\frac{7}{2} = 3$

$i=3$

30 20

17 5 $i=2$

30 ~~15~~ $i=1$

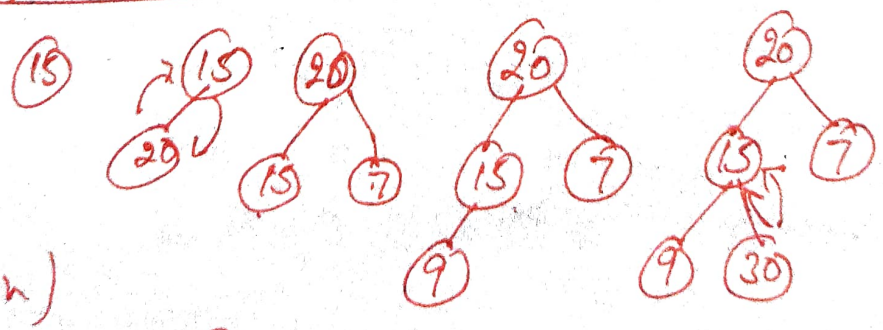
20 15

1	2	3	4	5	6	7
30	17	20	1	5	10	15

Now it is Max heap

Next step is delete root

Insertion | 15 | 20 | 7 | 9 | 30 |



$(n \log n)$

Del $(n \log n)$

$(2n \log n)$

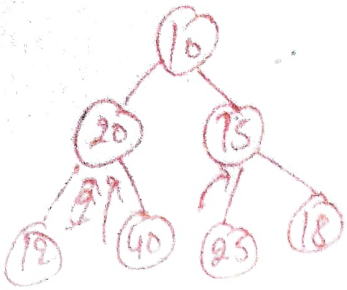
```
maxheapify (A, N, i)
{
    int largest = i;
    int l = 2*i;
    int r = (2*i) + 1;
    while(l<=n && A[l] > A[largest])
            largest = l;
    while(r<=n && A[r] > A[largest])
            largest = r;
    if(largest != i)
    {
        swap(A[largest], A[i]);
        Maxheapify(A, n, largest)
    }
}
```

```
heapsort (A, n)
{
    for(i=n/2; i>=1; i--)
    {
        Maxheapify(A, n, i)
    }
    for(i=n; i>=1; i--)
    {
        swap(A[1], A[i])
        Maxheapify(A, n, 1)
    }
}
```

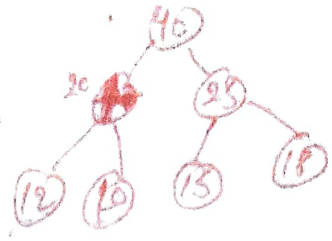
Build max heap

Deleting

Heapify



→



H

10	20	15	12	40	25	18
1	2	3	4	5	6	7

→

40	20	25	12	10	15	18

```
heapify (int a[], int n, int i)
{   int t,
    int largest = i,
    int l = 2*i+1,
    int r = 2*i+2,
    while(l<n && a[l] > a[largest])
            largest = l.
    while(r < n && a[r] > a[largest])
            largest = r,
    If (largest != i)
    {   swap (a[largest], a[i]),
        heapify (a, n, largest)
    }
}
```

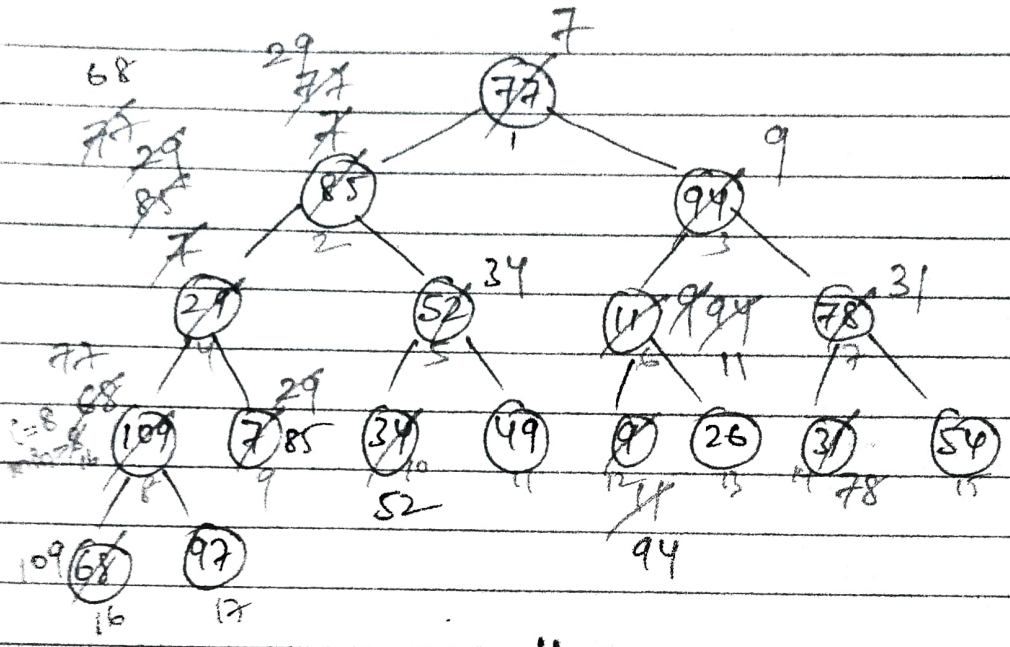
```
heapsort (int a[], int n)
{   int i, t,
    for(i = n/2, i>=0, i--)
        heapify (a, n, i),
    for(i = n-1, i>=0, i--)
    {   t = a[0],
        a[0] = a[i],
        a[i] = t
        heapify (a, i, 0)
    }
}
```

```
int main(    )
{   int a[50], i, n,
    printf("enter elements")
    scanf ("%d", &n),
    for(i=0, i<n, i++)
    scanf ("%d", &a[i]),
    heapsort (a, n)
    Printf("Sorted Array"),
    for (i=0, i<n, i++)
    printf("%d", a[i]),
}
```

Date 28/11/12

77, 85, 94, 29, 52, 11, 78, 109, 7, 34, 49, 9, 26, 31, 54, 68, 97

(1) Create ACBT



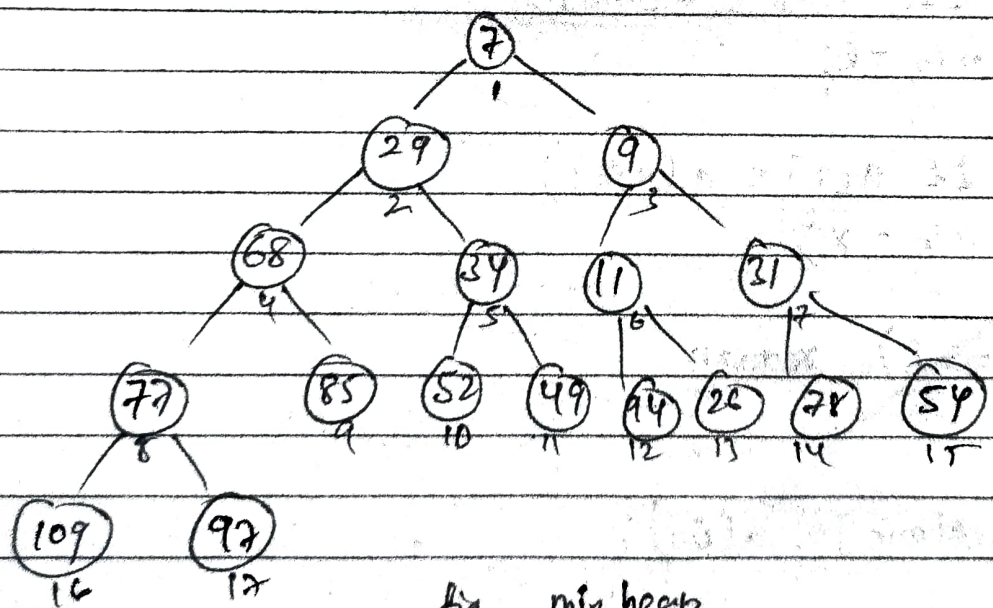


fig min heap

min-heapify

worst case = $O(\log n)$

best case = $O(1)$

1
2
4
8
16