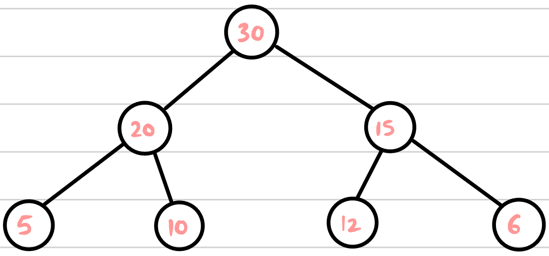
HEAP

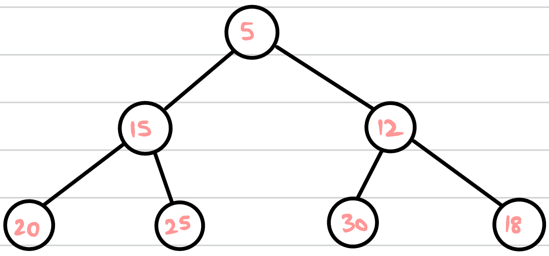
↳ It is a complete binary tree

- what is a heap?
- Insert in a heap
- Deleting from heap
- Heap Sort
- Heapify
- Priority Queues

Max Heap



Min Heap



30	20	15	5	10	12	6			
1	2	3	4	5	6	7	8	9	10

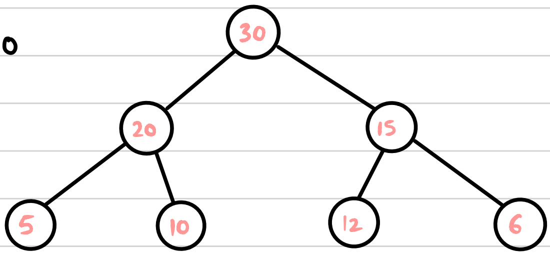
- For complete binary tree, there should not be any free space between the elements of array

Node at index i
Left child at index 2* i
Right child at index 2* i + 1

- Every node should have an element greater than or equal to all its decendants (duplicates can be there)

INSERTING IN HEAP

Element to insert = 40

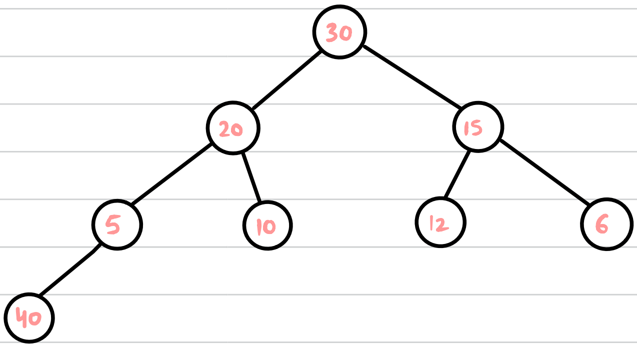


30	20	15	5	10	12	6			
1	2	3	4	5	6	7	8	9	10

Step 1 : Insert element at next free index in array

30	20	15	5	10	12	6	40		
1	2	3	4	5	6	7	8	9	10
							↑		

Step 2 : Insert the element in the tree

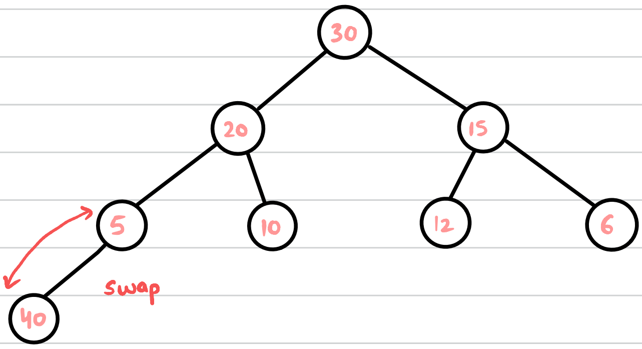


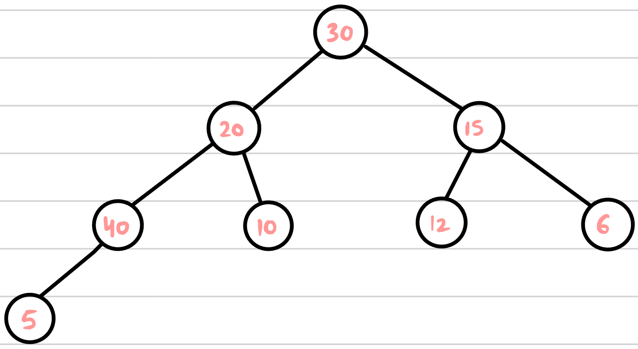
Parent of 40 = $\frac{8}{2}$ = 4

30	20	15	5	10	12	6	40		
1	2	3	4	5	6	7	8	9	10
			↑						

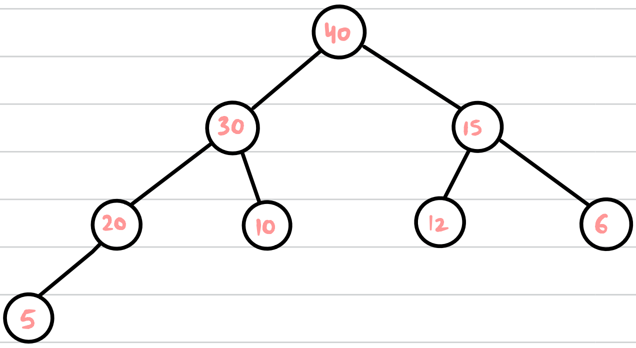
Step 3 : Now tree does not satisfy max heap condition

Compare 40 with its parent / ancestor, if node is found greater than ancestor, swap.





Repeat until condition is satisfied



Check this condition in array also. Compare node with parent $\left(\frac{i}{2}\right)$, if found greater than parent, replace

40	30	15	20	10	12	6	5		
1	2	3	4	5	6	7	8	9	10

PROGRAM

```
void Insert (int A[], int n)          O(logn)
{
        int temp, i=n;
        temp = A[n];

        while( i>1 && temp > A[i/2] )
        {
                A[i] = A[i/2]);
                i= i/2;
        }
        A[i] = temp;
}

void createheap()  O(nlogn)      1 element - logn
{                                n elements - nlogn
        int A[] = {0,10,20,30,25,5,40,35};
        int i;
                  ┌→ because first element is the heap and next element onwards,
        for (i=2; i<=7; i++)     insertion in heap is done.
                  Insert (A,i);
}
```

Heapify : Faster method to implement creation of binary heap ← Separate Topics

Element Priority

Elements → 6, 8, 3, 10, 15, 2, 9, 17, 5, 8

- Element is itself a priority
- Smaller number higher priority

1. Insert in same order O(1) O(logn)
   Delete max priority by searching it O(n) O(logn)] After implementing using heap

   Heap is used to implement priority queues

DELETION FROM HEAP

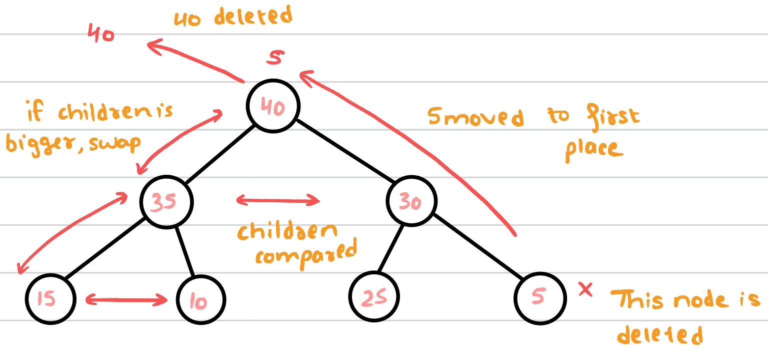
Only the root node can be deleted from heap i.e the first node

```
void Delete (int A[], int n)
{
        int x, i, j ;
        x = A[n];
        A[1] = A[n]      // Smallest or last node is copied at first place
        i = 1 ; j = 2*i ;  // i is at index 1 and j is at its left child

        while (j<n-1)
        {
            if ( A[j+1] > A[j])   // Right child is greater than left child
                j = j+1 ;

            if ( A[i] < A[j])  // Parent is smaller than child
            {
                swap( A[i], A[j]);
                i = j;
                j = 2*j ;
            }
            else
                break;

            A[n] = x ;
}
```



HEAP SORT

While deleting, store the deleted element at vacant position

Through this, you will get sorted array in ascending order after deleting all the elements

40	35	30	15	10	25	5			
1	2	3	4	5	6	7	8	9	10

5	35	30	15	10	25				
1	2	3	4	5	6	7	8	9	10

x = 40

35	15	30	5	10	25	40			
1	2	3	4	5	6	7	8	9	10

STEPS

nlogn • Create heap of 'n' elements

nlogn • Delete 'n' elements 1 by 1

O(nlogn)