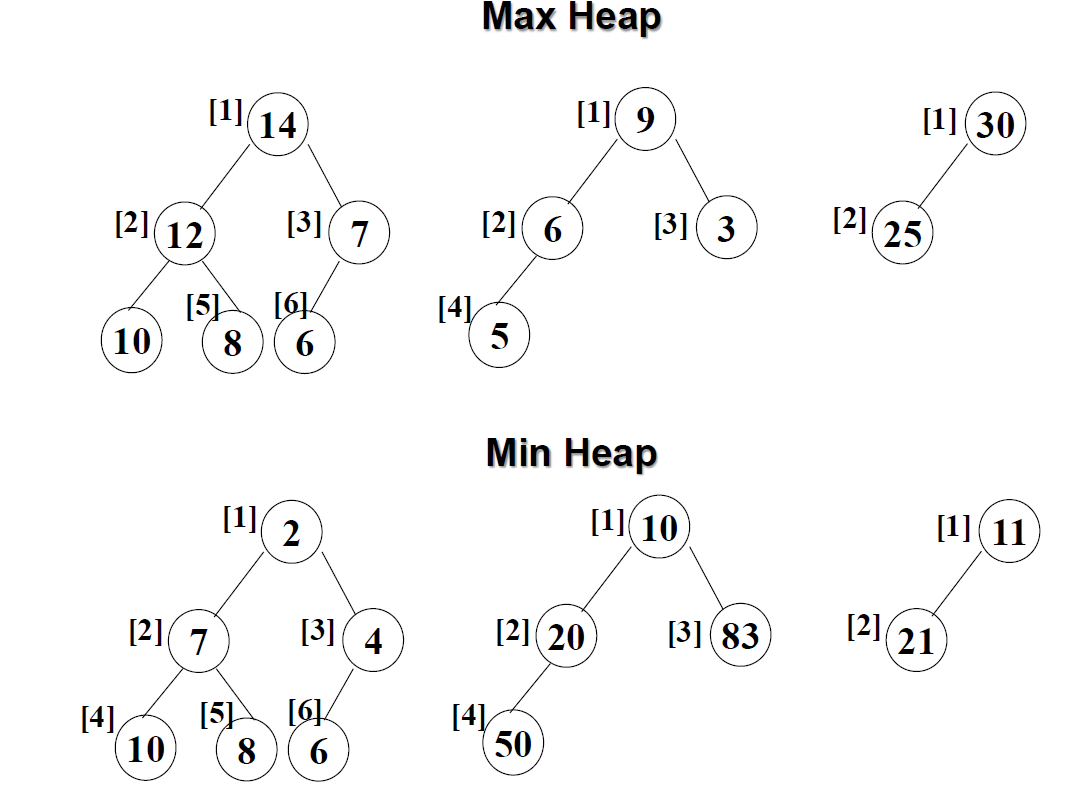
**HEAP**

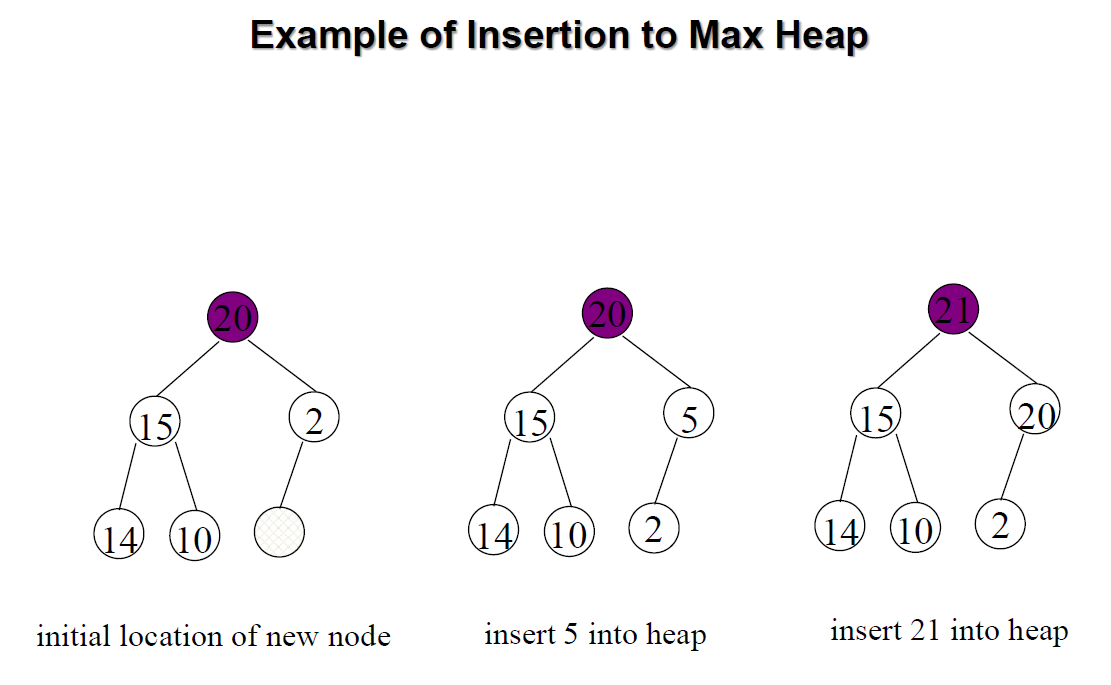
A *max tree* is a tree in which the key value in each node is no smaller than the key values in its children. A *max heap* is a complete binary tree that is also a max tree.

A *min tree* is a tree in which the key value in each node is no larger than the key values in its children. A *min heap* is a complete binary tree that is also a min tree.

**Operations on heaps:**

* creation of an empty heap
* insertion of a new element into the heap;
* deletion of the largest element from the heap





**Insertion into a Max Heap**

void insert\_max\_heap(element item, int \*n)

{

int i;

if (HEAP\_FULL(\*n)) {

fprintf(stderr, “the heap is full.\n”);

exit(1);

}

i = ++(\*n);

while ((i!=1)&&(item.key>heap[i/2].key)) {

heap[i] = heap[i/2];

i /= 2;

}

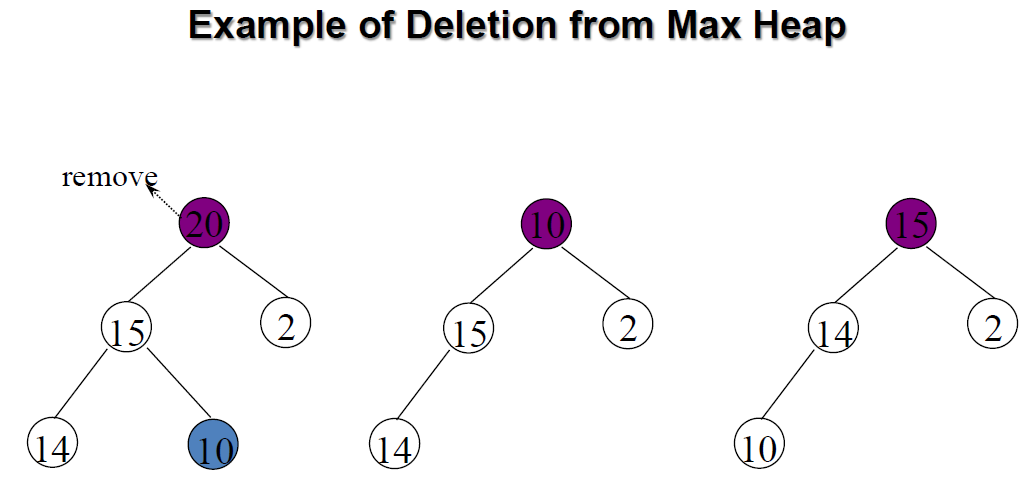
heap[i]= item;

}

**Complexity:**

2k-1=n ==> k=log2(n+1)

O(log2n)



**Deletion from a Max Heap**

element delete\_max\_heap(int \*n)

{

int parent, child;

element item, temp;

if (HEAP\_EMPTY(\*n)) {

fprintf(stderr, “The heap is empty\n”);

exit(1);

}

/\* save value of the element with the

highest key \*/

item = heap[1];

/\* use last element in heap to adjust heap \*/

temp = heap[(\*n)--];

parent = 1;

child = 2;

while (child <= \*n) {

/\* find the larger child of the current

parent \*/

if ((child < \*n)&&(heap[child].key<heap[child+1].key))

child++;

if (temp.key >= heap[child].key) break;

/\* move to the next lower level \*/

heap[parent] = heap[child];

child \*= 2;

}

heap[parent] = temp;

return item;

}

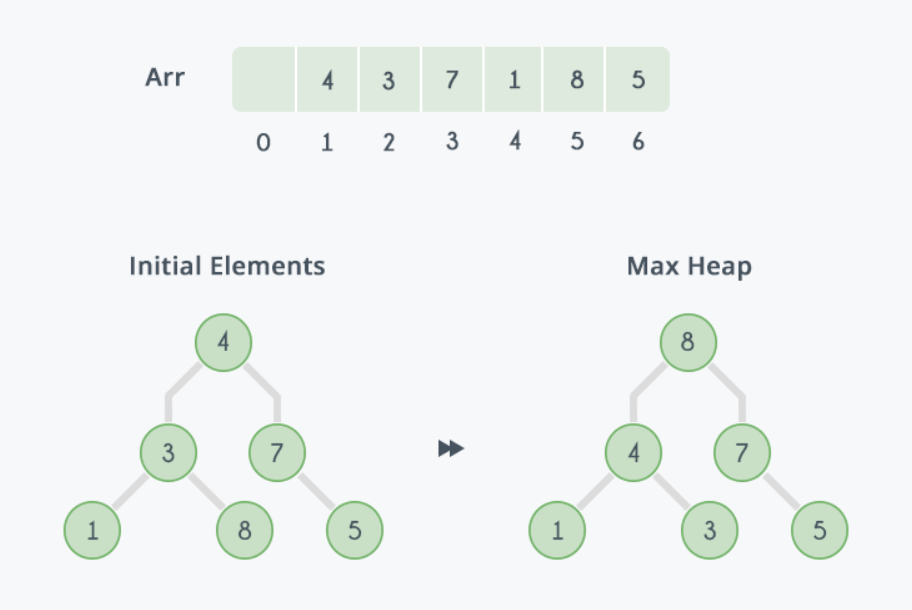
**Heap Sort**

Heaps can be used in sorting an array. In max-heaps, maximum element will always be at the root. Heap Sort uses this property of heap to sort the array.

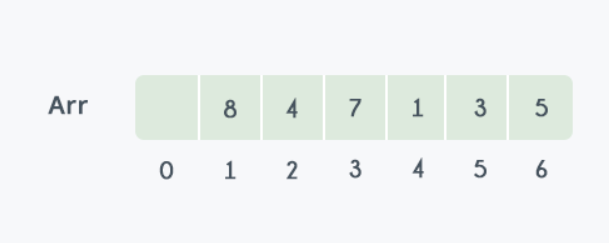
Consider an array Arr which is to be sorted using Heap Sort.

* Initially build a max heap of elements in Arr.
* The root element, that is Arr[1], will contain maximum element of Arr. After that, swap this element with the last element of Arr and heapify the max heap excluding the last element which is already in its correct position and then decrease the length of heap by one.
* Repeat the step 2, until all the elements are in their correct position.

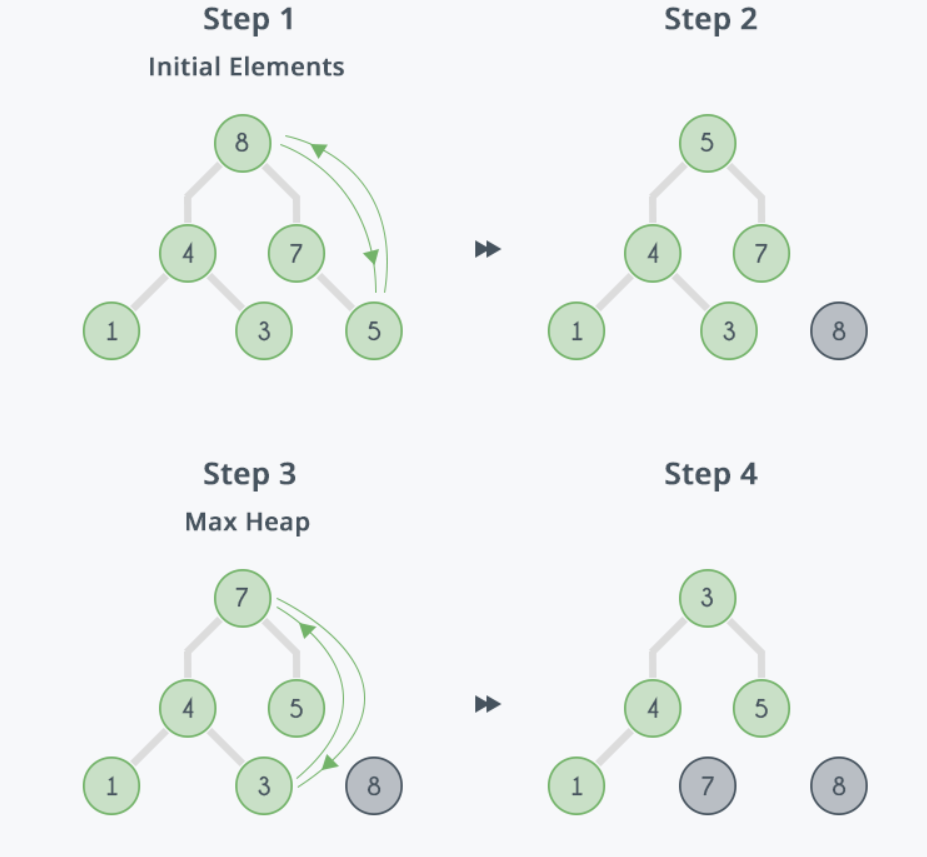
**Example:**  
In the diagram below, initially there is an unsorted array Arr having 6 elements and then max-heap will be built.

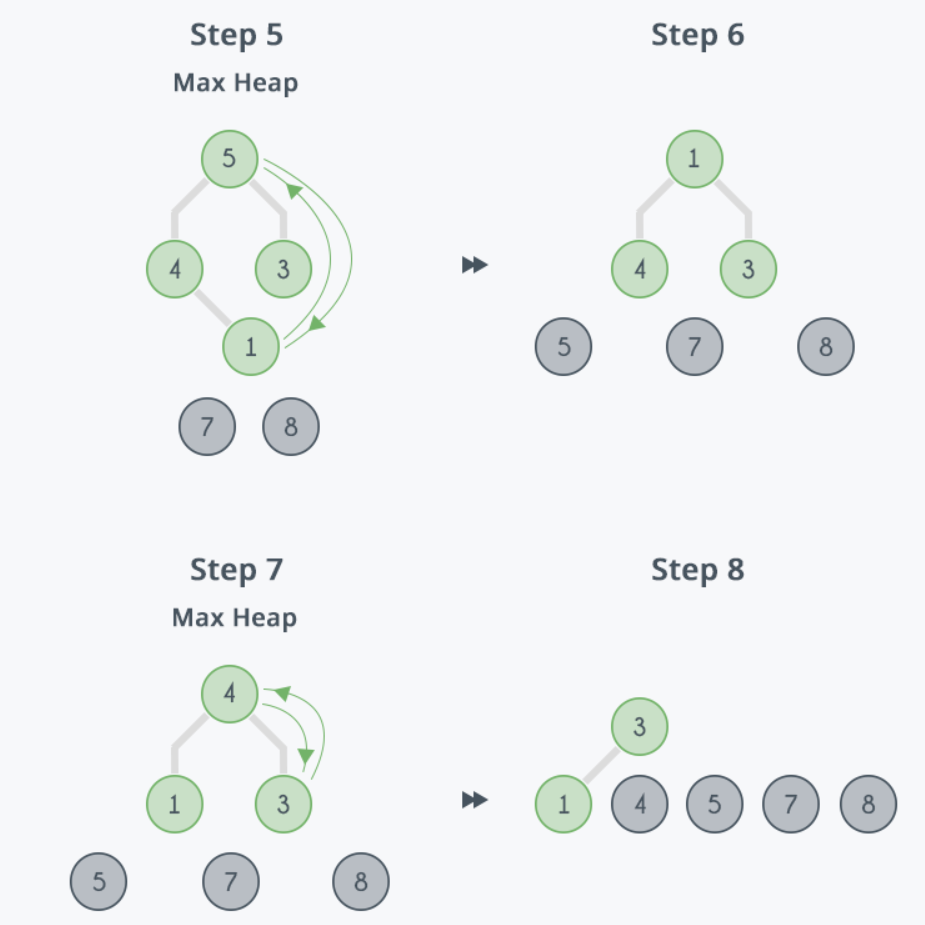


After building max-heap, the elements in the array Arr will be:



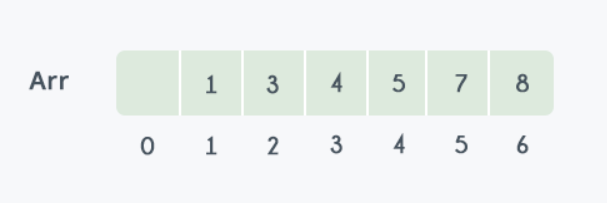
Step 1: 8 is swapped with 5.  
Step 2: 8 is disconnected from heap as 8 is in correct position now and.  
Step 3: Max-heap is created and 7 is swapped with 3.  
Step 4: 7 is disconnected from heap.  
Step 5: Max heap is created and 5 is swapped with 1.  
Step 6: 5 is disconnected from heap.  
Step 7: Max heap is created and 4 is swapped with 3.  
Step 8: 4 is disconnected from heap.  
Step 9: Max heap is created and 3 is swapped with 1.  
Step 10: 3 is disconnected.







After all the steps, we will get a sorted array.



**RELEVANT READING MATERIAL AND REFERENCES:**

**Source Notes:**

1. <https://www.slideshare.net/suyashbhardwaj1/trees-binary-search-tree-avl-tree-in-data-structures>
2. <https://www.hackerearth.com/practice/algorithms/sorting/heap-sort/tutorial/>

**Lecture Video:**

1. <https://www.youtube.com/watch?v=NEtwJASLU8Q>
2. <https://www.youtube.com/watch?v=0Jae4ApidS4>

**Online Notes:**

1. <http://www.crectirupati.com/sites/default/files/lecture_notes/ds%20ln.pdf>
2. <http://www.vssut.ac.in/lecture_notes/lecture1428550942.pdf>

**Text Book Reading:**

1. Cormen, Leiserson, Rivest, Stein, “*Introduction to Algorithms*”, Prentice Hall of India, 3rd edition 2012. problem, Graph coloring.
2. Lipschutz, S., “*Data Structures, Schaum's Outline Series*”, Tata McGraw Hill.

**Online Book Reference:**

1. <https://www.edutechlearners.com/download/books/DS.pdf>

**In addition: PPT can be also be given.**