A Binary Heap is a Binary Tree with following properties.  
1) It’s a complete tree (All levels are completely filled except possibly the last level and the last level has all keys as left as possible). This property of Binary Heap makes them suitable to be stored in an array.

2) A Binary Heap is either Min Heap or Max Heap. In a Min Binary Heap, the key at root must be minimum among all keys present in Binary Heap. The same property must be recursively true for all nodes in Binary Tree. Max Binary Heap is similar to MinHeap.

10 10

/ \ / \

20 100 15 30

/ / \ / \

30 40 50 100 40

For Heap sort

**Worst Case Time Complexity :** O(n log n)

**Best Case Time Complexity :** O(n log n)

**Average Time Complexity :** O(n log n)

**Space Complexity :** O(n)

* Heap sort is not a Stable sort, and requires a constant space for sorting a list.
* Heap Sort is very fast and is widely used for sorting.

## Heap Sort Algorithm

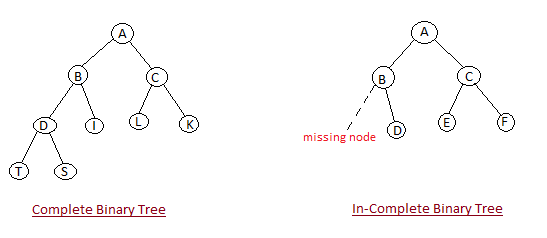
Heap Sort is one of the best sorting methods being in-place and with no quadratic worst-case scenarios. Heap sort algorithm is divided into two basic parts:

* Creating a Heap of the unsorted list.
* Then a sorted array is created by repeatedly removing the largest/smallest element from the heap, and inserting it into the array. The heap is reconstructed after each removal.

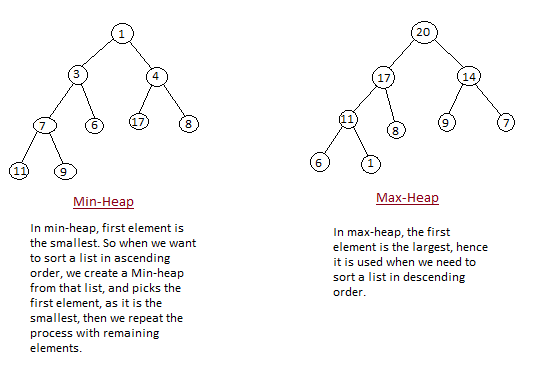
#### What is a Heap ?

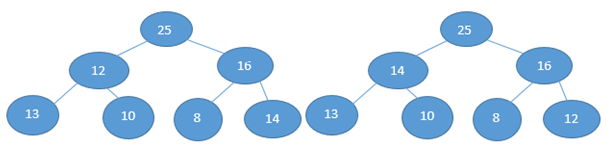
Heap is a special tree-based data structure that satisfies the following special heap properties:

1. **Shape Property:** Heap data structure is always a Complete Binary Tree, which means all levels of the tree are fully filled.



1. **Heap Property:** All nodes are either *[greater than or equal to]* or *[less than or equal to]* each of its children. If the parent nodes are greater than their children, heap is called a **Max-Heap**, and if the parent nodes are smaller than their child nodes, heap is called **Min-Heap**.





|  |  |
| --- | --- |
| It’s not a max heap as every node does not have a greater value Than its child node. Here 13 is leaf node and its parent node is having 12 so this is not a max heap. And the heap size will be 1. As only one node is following the max heap property Here that is 25. Node 16 is also following the max heap criteria but we won’t count it, once a node is failed the max heap criteria then there after we won’t consider the node for heap size. | It’s a max heap as every node have a greater value Than its child node. And the heap size will be 7 that is length of the array. As here all node is following the max heap property. **Leaf node is always a heap node.** |

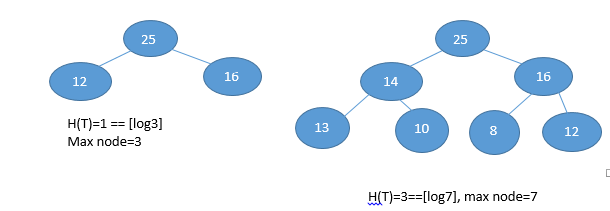
Heap is a complete binary tree because if it’s not a complete binary tree then there will be gap while putting the heap data into the array. As we take the heap data in index wise and put that data in array at the same index as it is in heap. Means in heap some data is at index 3 then in array also it should be at index 3. So if Heap is not a complete binary tree then we won’t find the successive index to put the data into the array so there will be some index gap in array on which index there is nothing in the heap. That’s why Heap is a complete binary tree.

**If the array is in ascending order then it is min heap, and if the array is in descending order then it is max heap.**

**For max heap sort time complexity is O(nlogn) like merge sort.**

**If there is purpose to just a construct a heap then their time complexity will be O(n).**

**Height of a tree is maximum no of edges from that node to a leaf node. Height of the tree is equal to height of the root node.**

****

**So, the relation between no of node and height of the tree for complete binary tree is No of**

**node=2 h+1-1**

**And the height of the tree by using the node, H(T)=[logn], where n=no of nodes (its applicable for both complete binary tree as well as un-complete binary tree)**

**If there are a complete binary tree with n nodes, then leaf nodes will be starting from [n/2] to n.**

**After finding the leaf node, find the largest index non-leaf node and heapify them, means check the child node of the largest index non-leaf node and make the largest value as root node.**

**Suppose there is an array 1, 5, 6, 8, 12, 14, 16**

|  |  |
| --- | --- |
| Here leaf node will be [7/2] to 6th index.  Means 3rd index to 6th index there is leaf node.    Now we will find the largest non-leaf node.  That is 2nd index which has value 6.  Now we will check the child element of 2nd Node that is 14 and 16. So we will swap the Value of 6 with the maximum of its child element that is 16.  After swapping the value it will look like bellow. |  |
| Now till 2,3,4,5,6 index arrays is max heap.  Now we will find the second largest non-leaf node. That is 1st index node which has value 5. Now we will check the child element of 1st Node that is 8 and 12. So we will swap 5 with maximum of 8 and 12. |  |

|  |
| --- |
|  |

|  |  |  |
| --- | --- | --- |
| After swapping the value 5 and 12 it will look like this.  So till here 1,2,3,4,5,6th index it is max heap.  Now we will find the 3rd largest non-leaf node that is root node here and its value is 1.  Now we will find the child element of the root node That is 12 and 16. Now we will swap 1 with maximum of 12 and 16. |  | |
| After swapping the value 1 and 16 it will look like this.  Once we swapped the 1 with 16, 2nd index node that has value 1 is now not a max heap.  So we will check its child element that is 14 and 6. So we will swap 1 with maximum of 14 and 6.  After swapping 1 with 14 it will look  Like bellow.  Here till 0,1,2,3,4,5,6 it’s a max heap.  Now it’s a max heap. enjoy ☺  **If height of the tree is h, then no of nodes will be upper limit of [n/2] h +1.**  **If the height of tree is h then we can find that at any height how many maximum no of nodes are present.** | |  | |

Heap size will be the length of the array till it satisfies the heap condition. Suppose there is an array which doesn’t satisfies the heap condition then the heap size will be zero.

And all the leaf node is heap by default.

# Priority queues

Heapsort is an excellent algorithm, but a good implementation of quicksort usually beats it in practice. Nevertheless, the heap data structure itself has enormous utility. In this section, we present one of the most popular applications of a heap: its use as an efficient priority queue.

A ***priority queue*** is a data structure for maintaining a set *S* of elements, each with an associated value called a ***key***. A priority queue supports the following operations.

INSERT(*S*, *x*) inserts the element *x* into the set *S*. This operation could be written as *S* http://staff.ustc.edu.cn/%7Ecsli/graduate/algorithms/images/arrlt12.gif*S* http://staff.ustc.edu.cn/%7Ecsli/graduate/algorithms/images/wideu.gif*{*x*}.*

MAXIMUM(*S*) returns the element of *S* with the largest key.

EXTRACT-MAX(*S*) removes and returns the element of *S* with the largest key.

One application of priority queues is to schedule jobs on a shared computer. The priority queue keeps track of the jobs to be performed and their relative priorities. When a job is finished or interrupted, the highest-priority job is selected from those pending using EXTRACT-MAX. A new job can be added to the queue at any time using INSERT.

A priority queue can also be used in an event-driven simulator. The items in the queue are events to be simulated, each with an associated time of occurrence that serves as its key. The events must be simulated in order of their time of occurrence, because the simulation of an event can cause other events to be simulated in the future. For this application, it is natural to reverse the linear order of the priority queue and support the operations MINIMUM and EXTRACT-MIN instead of MAXIMUM and EXTRACT-MAX. The simulation program uses EXTRACT-MIN at each step to choose the next event to simulate. As new events are produced, they are inserted into the priority queue using INSERT.

Not surprisingly, we can use a heap to implement a priority queue. The operation HEAP-MAXIMUM returns the maximum heap element in http://staff.ustc.edu.cn/%7Ecsli/graduate/algorithms/images/bound.gif(1) time by simply returning the value *A*[1] in the heap.

#### **How Heap Sort Works**

Initially on receiving an unsorted list, the first step in heap sort is to create a Heap data structure (Max-Heap or Min-Heap). Once heap is built, the first element of the Heap is either largest or smallest (depending upon Max-Heap or Min-Heap), so we put the first element of the heap in our array. Then we again make heap using the remaining elements, to again pick the first element of the heap and put it into the array. We keep on doing the same repeatedly until we have the complete sorted list in our array.

When you add a new node to a heap, you add it to the rightmost unoccupied leaf. And after adding that node if the heap is violating the heap rule then we heapify the heap.

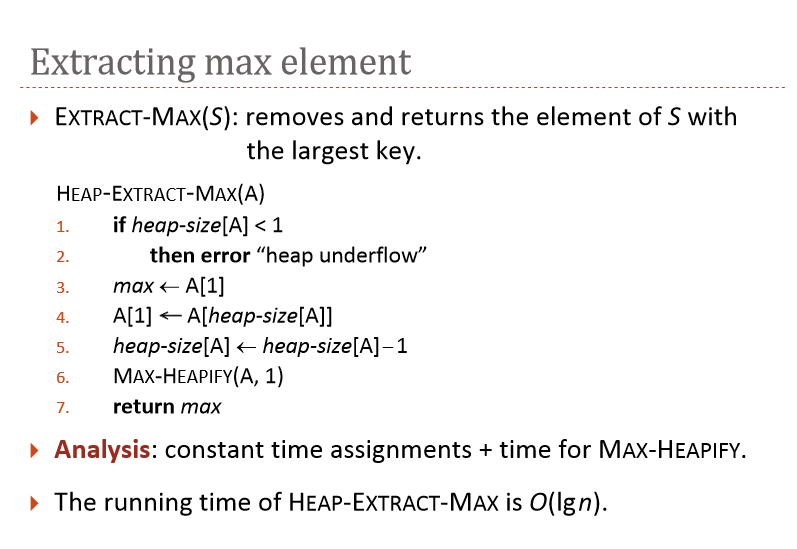
While removing the heap data, we remove the root data and fill the root node with last element of the heap. And after doing this if the heap is violating the heap rule then we heapify the heap.

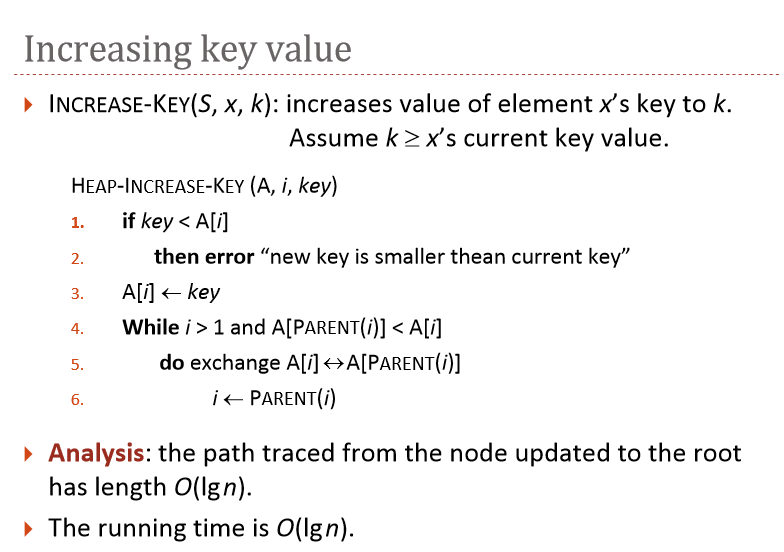
|  |  |
| --- | --- |
| **Adding a new element to the heap**  http://upload.wikimedia.org/wikipedia/commons/thumb/a/ac/Heap_add_step1.svg/150px-Heap_add_step1.svg.png  x=15 is the new element we need to insert  http://upload.wikimedia.org/wikipedia/commons/thumb/1/16/Heap_add_step2.svg/150px-Heap_add_step2.svg.png  Compare with the parent and swap. Continue until the parent is lesser than the child.  http://upload.wikimedia.org/wikipedia/commons/thumb/5/51/Heap_add_step3.svg/150px-Heap_add_step3.svg.png  Finally, the root gets replaced with the new element added. This is up-bubbling which is carried out by out heap-insert procedure. | **Deleting an element:** This operation is similar to Heap-extract max routine we saw.    In the above diagram, we remove a max element 15 from the heap. after we remove “15”, we replace last element “4” as the root.  http://upload.wikimedia.org/wikipedia/commons/thumb/e/ee/Heap_remove_step1.svg/150px-Heap_remove_step1.svg.png  Now, it violates the heap property. Do max-heapify to maintain the heap property.  http://upload.wikimedia.org/wikipedia/commons/thumb/2/22/Heap_remove_step2.svg/150px-Heap_remove_step2.svg.png |

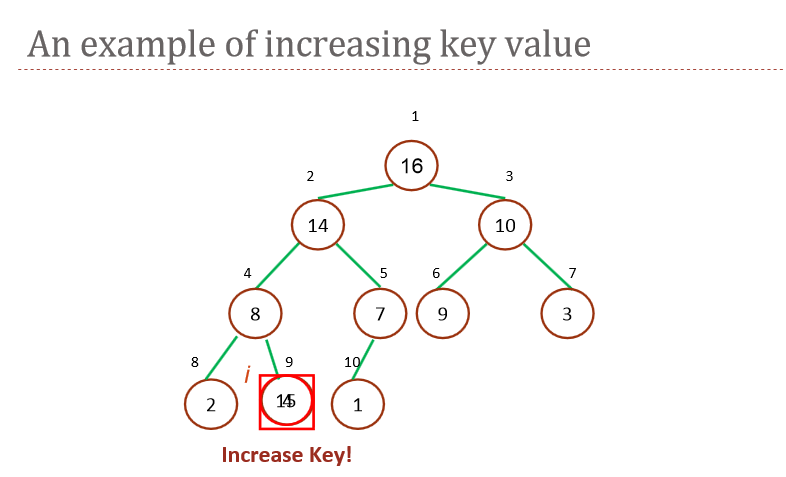
**Finding the maximum element**

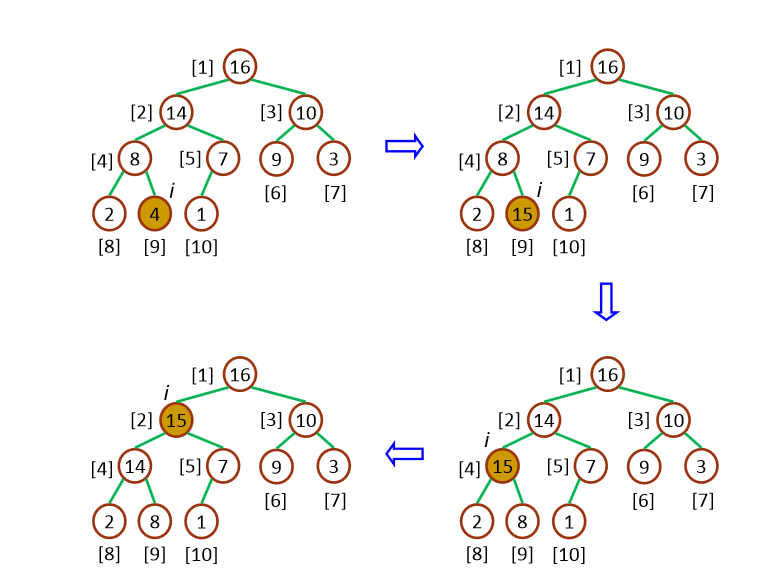
MAXIMUM(S): returns the element of S with the largest key. Getting the maximum element is easy: it’s the root.

The running time of H EAP ‐ MAXIMUM is Θ(1)

****



****



**applications-of-heap-data-structure**

Heap Data Structure is generally taught with Heapsort. Heapsort algorithm has limited uses because Quicksort is better in practice. Nevertheless, the Heap data structure itself is enormously used. Following are some uses other than Heapsort.

*Priority Queues:*Priority queues can be efficiently implemented using Binary Heap because it supports insert(), delete() and extractmax(), decreaseKey() operations in O(logn) time. Binomoial Heap and Fibonacci Heap are variations of Binary Heap. These variations perform union also in O(logn) time which is a O(n) operation in Binary Heap. Heap Implemented priority queues are used in Graph algorithms like [Prim’s Algorithm](http://en.wikipedia.org/wiki/Prim%27s_algorithm) and [Dijkstra’s algorithm](http://en.wikipedia.org/wiki/Dijkstra%27s_algorithm).

*Order statistics:* The Heap data structure can be used to efficiently find the kth smallest (or largest) element in an array.

The main application of [Binary Heap](http://geeksquiz.com/binary-heap/)is as implement priority queue. Binomial Heap is an extension of [Binary Heap](http://geeksquiz.com/binary-heap/)that provides faster union or merge operation together with other operations provided by Binary Heap.