Heaps

A heap is a specialized tree-based data structure that satisfies the *heap property:* If A is a parent node of B then the key of node A is ordered with respect to the key of node B with the same ordering applying across the heap.

Heaps can be classified further as either a "max heap" or a "min heap". In a max heap, the keys of parent nodes are always greater than or equal to those of the children, in a min heap, they are less than or equal to those of the children.

A heap is not a sorted structure and can be regarded as partially ordered. There is no particular relationship among nodes on any given level, even among the siblings.

When a heap is a complete binary tree, it has a smallest possible height—a heap with N nodes always has log N height.

Heaps are used to implement the abstract data type Priority Queue. Heaps are also crucial in several efficient graph algorithms such as Dijkstra's algorithm, and in the sorting algorithm heapsort.

Heaps are usually implemented in an array, and do not require pointers between elements. The children of the node at position n would be at positions $2n$ and $2n+1$ in a one-based array.

Heaps support the following operations:

Heapify: create a heap out of given array of elements.

Find-max [or Find-min]: find the maximum item of a max-heap [or a minimum item of a min-heap].

Increase-key [or Decrease-key]: updating a key within a max [or min] heap, respectively and shift it up [or down] to be at the right place in the heap.

Insert: adding a new key to the heap and then shift it to be at the right place in the heap.

ExtractMin [or ExtractMax]: Returns the node of maximum value from a max heap [or minimum value from a min heap] after removing it from the heap.