Heaps

A heap is a complete binary tree that satisfies the heap property. There are two types of heaps, the max heap and the min heap.

Binary Heap

Binary Heap

Wojciech Muła [CC BY-SA 1.0]

The heap property

The heap property says that is the value of Parent is either greater than or equal to (in a max heap) or less than or equal to (in a min heap) the value of the Child.

A heap is described in memory using linear arrays in a sequential manner.

Max Heap

In a max heap, the key present at the root is the largest in the heap and all the values below this are less than this value.

Max Heap

Max Heap

Ermishin [CC BY-SA 3.0]

Min Heap

In a min heap, the key present at the root is the smallest in the heap and all the values below this are greater than this value.

Min Heap

Min Heap

Vikingstad at English Wikipedia [Publidomain]

Heap Operations

The following operations can be performed on a heap data structure:

Insertion

A new element is always inserted at the last child of the original heap. The new element may now violate the heap property that a heap must satisfy.

Therefore, an operation known as reheapify upward is

performed on the heap. The aim of the reheapify operation is to compare the new value inserted with its parent's value.

If the value is greater (in a max heap) or smaller (in a a min heap) it is swapped with its parent. The process is then continued from the parent node recursively until the heap property is satisfied or the root node is hit.

Deletion

An element is always deleted from the root of the heap. But deleting an element will leave a hole in the heap, which disturbs the requirement that the heap must be a complete binary tree. To fill this hole, the last node in the heap is swapped to this place. This causes the heap to violate the heap property.

Similar to inserting an element, an operation known as reheapify downward is performed on the heap. The value of the root node is first replaced with the largest (or smallest) value amongst its children. Then the down heap property is repeated from this child again recursively until we hit the leaf node.

Summing up the algorithm:

Replace the root node's value with the last node's value.
Delete the last node.
Sink down the new root node's value so that the heap again satisfies the heap property.
Finding Maximum/Minimum
Finding the node which has maximum or minimum value is easy due to the heap property and is one of the advantages of using a heap.

Since all the elements below it are smaller (or larger in a min-heap), it will be always the root node. This can be accessed in constant time.

Application in Programming
Heapsort: This is one of the best in-place sorting methods with no quadratic worst-case scenarios. This is because the minimum or maximum element is always the root of the heap.
Implementing priority queues: As the highest (or lowest)

priority element is always stored at the root of the heap, they could be accessed quickly.

Selection algorithms: A heap allows access to the min or max element in constant time, and other selections (such as median or kth-element) can be done in sub-linear time on data that is in a heap.

Graph algorithms: By using heaps as internal traversal data structures, run times can be reduced by polynomial order.