Lecture-29

Heap Sort

Heap sort is a comparison based sorting technique based on Binary Heap data structure. It is similar to selection sort where we first find the maximum element and place the maximum element at the end. We repeat the same process for remaining element.

What is Binary Heap?

Let us first define a Complete Binary Tree. A complete binary tree is a binary tree in which every level, except possibly the last, is completely filled, and all nodes are as far left as possible

A Binary Heap is a Complete Binary Tree where items are stored in a special order such that value in a parent node is greater(or smaller) than the values in its two children nodes. The former is called as max heap and the latter is called min heap. The heap can be represented by binary tree or array.

Why array based representation for Binary Heap?

Since a Binary Heap is a Complete Binary Tree, it can be easily represented as array and array based representation is space efficient. If the parent node is stored at index **I**, the left child can be calculated by 2 * **I** + 1 and right child by 2 * **I** + 2 (assuming the indexing starts at 0).

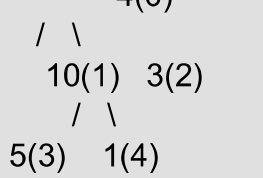
Heap Sort Algorithm for sorting in increasing order:

1. Build a max heap from the input data.
2. At this point, the largest item is stored at the root of the heap. Replace it with the last item of the heap followed by reducing the size of heap by 1. Finally, heapify the root of tree.
3. Repeat above steps while size of heap is greater than 1.

How to build the heap?

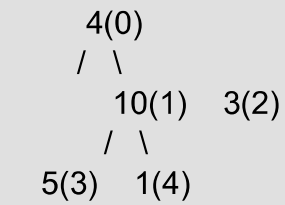
Heapify procedure can be applied to a node only if its children nodes are heapified. So the heapification must be performed in the bottom up order.
Lets understand with the help of an example:

Input data: 4, 10, 3, 5, 1

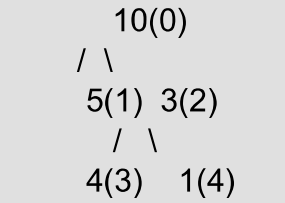


The numbers in bracket represent the indices in the array representation of data.

Applying heapify procedure to index 1:



Applying heapify procedure to index 0:



The heapify procedure calls itself recursively to build heap in top down manner.

Radix Sort

The lower bound for Comparison based sorting algorithm (Merge Sort, Heap Sort, Quick-Sort .. etc) is Ω(nLogn), i.e., they cannot do better than nLogn.

Counting sort is a linear time sorting algorithm that sort in O(n+k) time when elements are in range from 1 to k.

What if the elements are in range from 1 to n^2?

We can't use counting sort because counting sort will take $O(n^2)$ which is worse than comparison based sorting algorithms. Can we sort such an array in linear time? Radix Sort is the answer. The idea of Radix Sort is to do digit by digit sort starting from least significant digit to most significant digit. Radix sort uses counting sort as a subroutine to sort.