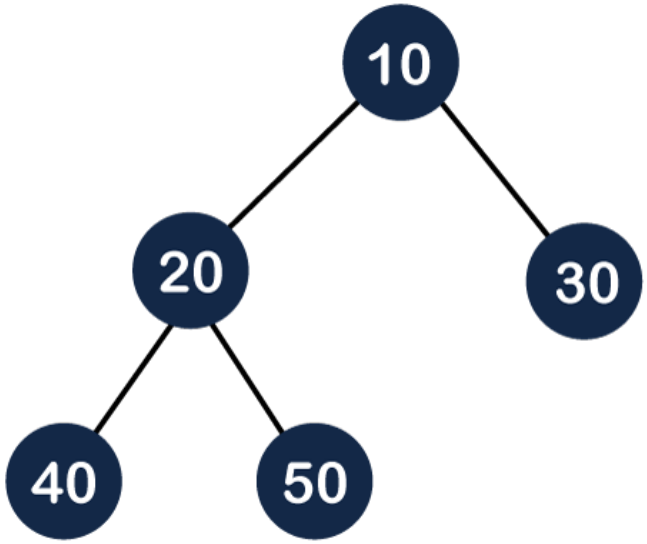
Data Structures and Algorithms

Lecture 33: Heap – Deletion, Heap Sort

Heap Data Structure

A Heap is a special Tree-based Data Structure in which the tree is a complete binary tree.

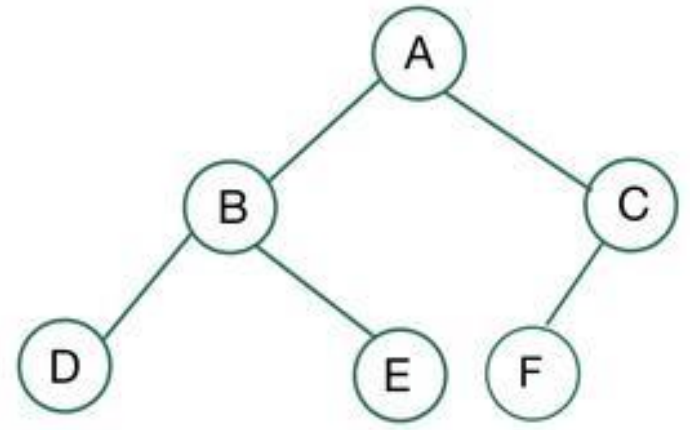


Heap Data Structure

Complete Binary Tree:

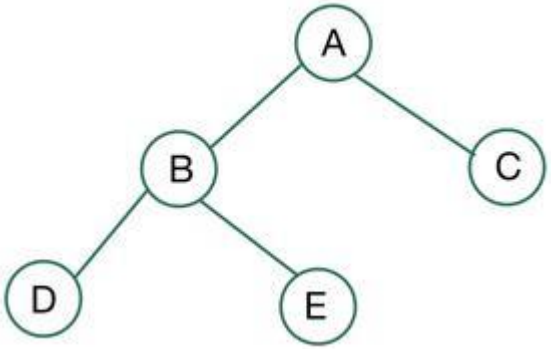
A binary tree is said to be a complete binary tree if all its levels, except possibly the last level, have the maximum number of possible nodes, and all the nodes in the last level appear as far left as possible.

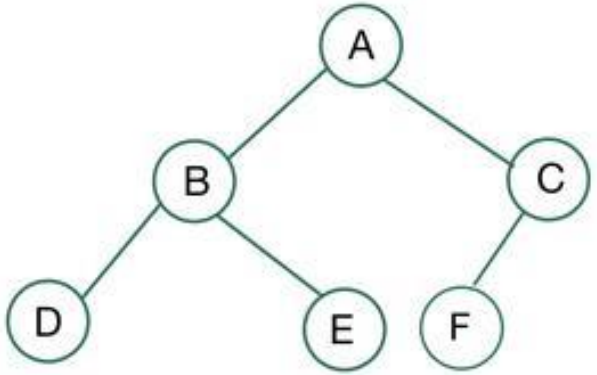
- The leftmost side of the leaf node must always be filled first.
- It isn't necessary for the last leaf node to have a right sibling.

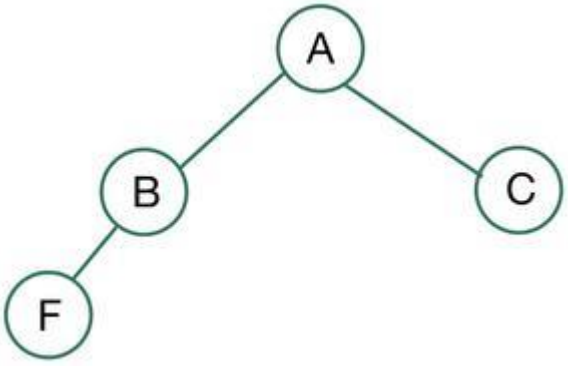


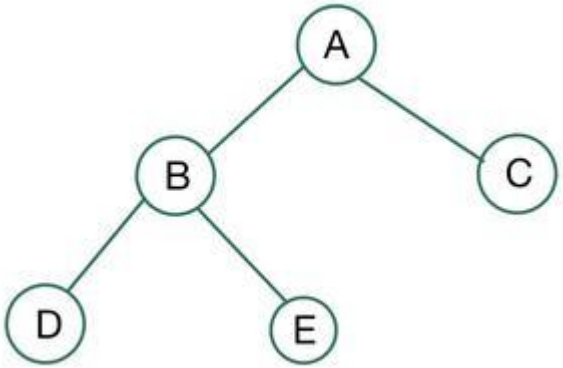
Heap Data Structure

Complete Binary Tree - Examples







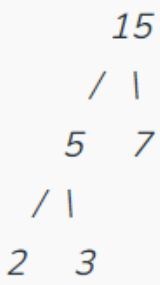


Heap – Deletion Operation

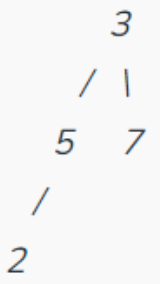
- If we delete the element from the heap it always deletes the root element of the tree and replaces it with the last element of the tree.
- Since we delete the root element from the heap it will distort the properties of the heap so we need to perform heapify operations so that it maintains the property of the heap.

Heap – Deletion Operation

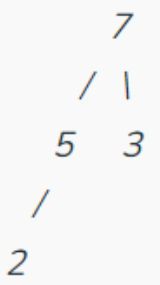
Assume initially heap(taking max-heap) is as follows



Now if we delete 15 into the heap it will be replaced by leaf node of the tree for temporary.



After heapify operation final heap will be look like this



Heap Sort

- Heapsort is a popular and efficient sorting algorithm. The concept of heap sort is to eliminate the elements one by one from the heap part of the list, and then insert them into the sorted part of the list.

- Heapsort is the in-place sorting algorithm. Recursively performs two main operations -
    1. Build a heap H, using the elements of array.
    2. Repeatedly delete the root element of the heap formed in 1st phase.

Heap Sort - Algorithm

- On a max heap, this process will sort the array in ascending order.
- On a min heap, this process will sort in descending order.

Here's the algorithm for heap sort:

Step 1: Build Heap. Build a heap from the input data. Build a max heap to sort in increasing order, and build a min heap to sort in decreasing order.

Step 2: Swap Root. Swap the root element with the last item of the heap.

Step 3: Reduce Heap Size. Reduce the heap size by 1.

Step 4: Re-Heapify. Heapify the remaining elements into a heap of the new heap size by calling heapify on the root node.

Step 5: Call Recursively. Repeat steps 2,3,4 as long as the heap size is greater than 2.

Working of Heap sort Algorithm

Using the heap sort to sort the array:

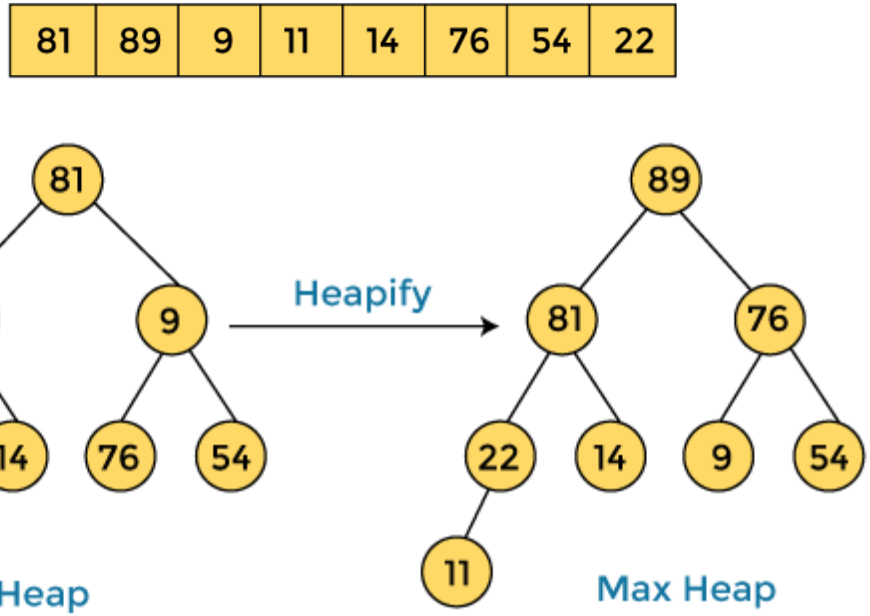
- After the heap formation using the **heapify** method, the sorting is done by:
- **Swapping** the root element with the last element of the array and decreasing the length of the heap array by one. (In heap representation, it is equivalent to swapping the root with the bottom-most and right-most leaf and then deleting the leaf.)
- **Restoring** heap properties (reheapification) after each deletion, where we need to apply the heapify method only on the root node. The subtree heaps will still have their heap properties intact at the beginning of the process.
- **Repeating** this process until every element in the array is sorted: Root removal, its storage in the position of the highest index value used by the heap, and heap length decrement.

Heap sort- Example

- The first step includes the creation of a heap by adjusting the elements of the array.
- After the creation of heap, now remove the root element of the heap repeatedly by shifting it to the end of the array, and then store the heap structure with the remaining elements.
- Let's take an unsorted array and try to sort it using heap sort.

Heap sort- Example

1. First, we have to construct a heap from the given array and convert it into max heap.

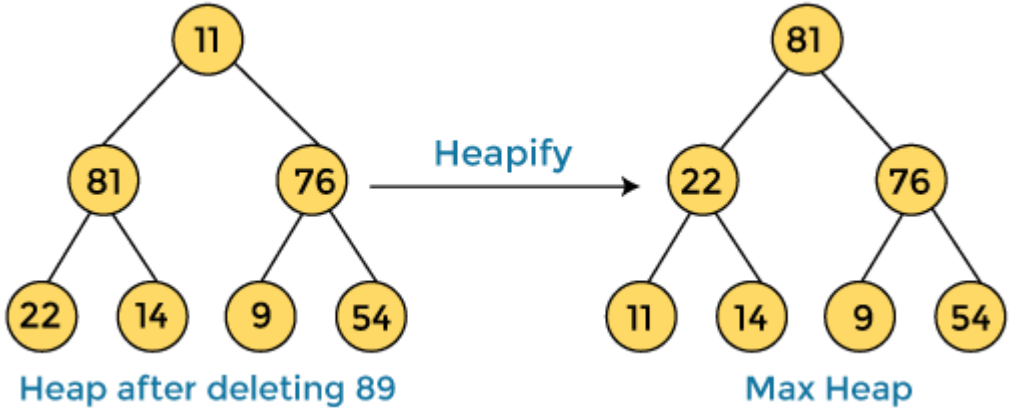


After converting the given heap into max heap, the array elements are -

89	81	76	22	14	9	54	11

Heap sort- Example

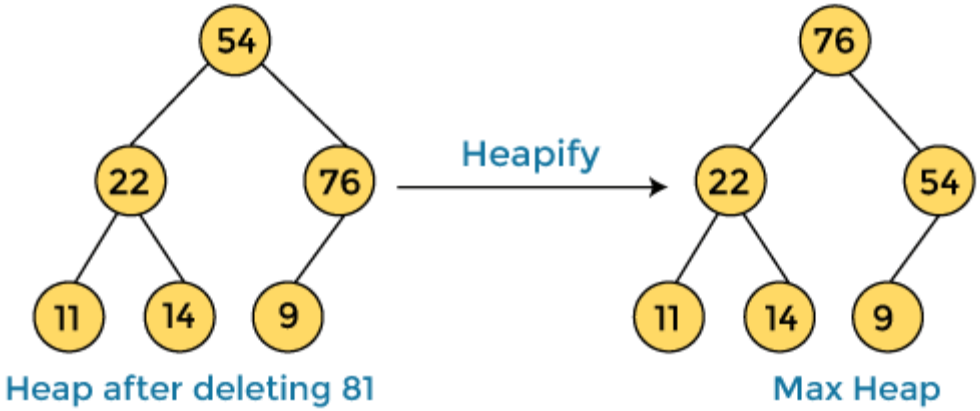
2. Next, we have to delete the root element (89) from the max heap. To delete this node, we have to swap it with the last node, i.e. (11). After deleting the root element, we again have to heapify it to convert it into max heap.



After swapping the array element 89 with 11, and converting the heap into max-heap, the elements of array are -

Heap sort- Example

3. In the next step, again, we have to delete the root element (81) from the max heap. To delete this node, we have to swap it with the last node, i.e. (54). After deleting the root element, we again have to heapify it to convert it into max heap.

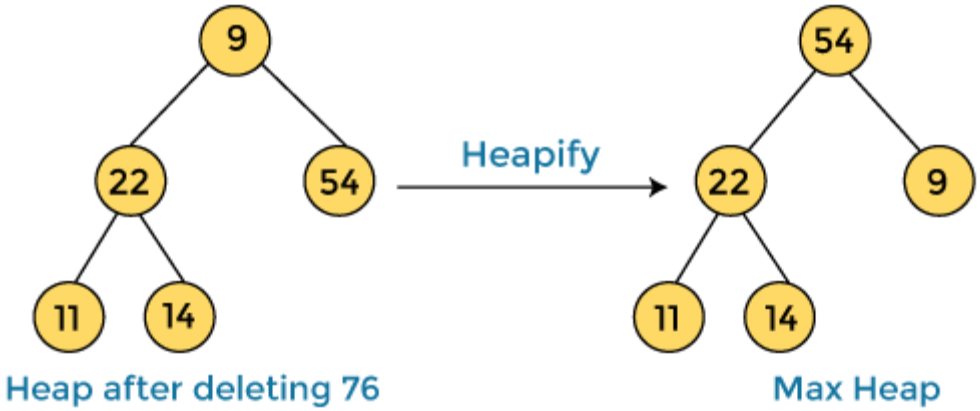


After swapping the array element 81 with 54 and converting the heap into max-heap, the elements of array are -

76	22	54	11	14	9	81	89

Heap sort- Example

4. In the next step, we have to delete the root element (76) from the max heap again. To delete this node, we have to swap it with the last node, i.e. (9). After deleting the root element, we again have to heapify it to convert it into max heap.

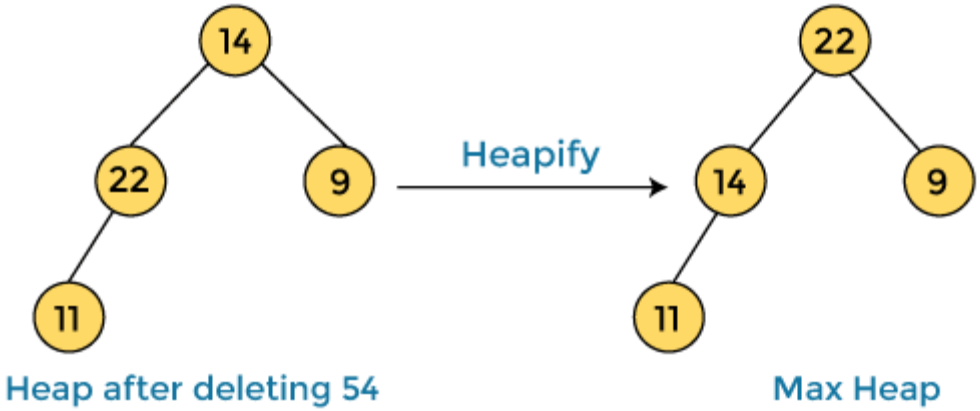


After swapping the array element 76 with 9 and converting the heap into max-heap, the elements of array are -

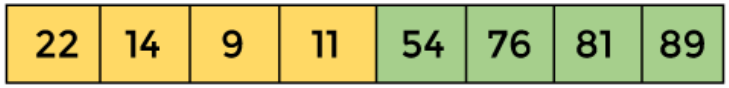
54	22	9	11	14	76	81	89

Heap sort- Example

5. In the next step, again we have to delete the root element (54) from the max heap. To delete this node, we have to swap it with the last node, i.e. (14). After deleting the root element, we again have to heapify it to convert it into max heap.

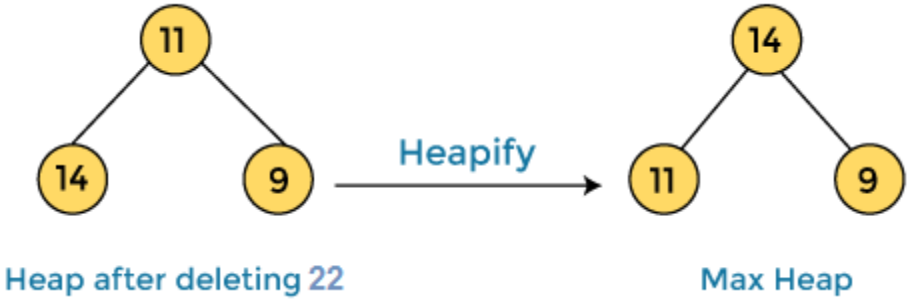


After swapping the array element 54 with 14 and converting the heap into max-heap, the elements of array are -



Heap sort- Example

6. In the next step, again we have to delete the root element (22) from the max heap. To delete this node, we have to swap it with the last node, i.e. (11). After deleting the root element, we again have to heapify it to convert it into max heap.



After swapping the array element 22 with 11 and converting the heap into max-heap, the elements of array are -

14	11	9	22	54	76	81	89

Heap sort- Example

8. In the next step, again we have to delete the root element (14) from the max heap. To delete this node, we have to swap it with the last node, i.e. (9). After deleting the root element, we again have to heapify it to convert it into max heap.

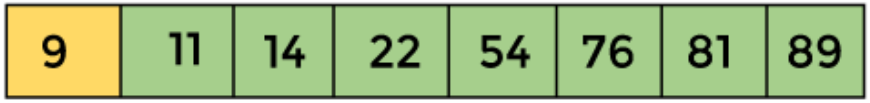
After swapping the array element 14 with 9 and converting the heap into max-heap, the elements of array are -

11	9	14	22	54	76	81	89

Heap sort- Example

9. In the next step, again we have to delete the root element (11) from the max heap. To delete this node, we have to swap it with the last node, i.e. (9). After deleting the root element, we again have to heapify it to convert it into max heap.

After swapping the array element 11 with 9, the elements of array are -



Heap sort- Example

10. Now, heap has only one element left. After deleting it, heap will be empty.

After completion of sorting, the array elements are -

9	11	14	22	54	76	81	89

Applications of Heap Sort

- Heap sort has limited usage since algorithms like merge sort and quicksort are better in practice. We extensively use heaps for problems like getting the largest or smallest elements in an array, sorting an almost sorted array, etc.

Some key applications of Heap sort include:

- Implementation of priority queues
- Security systems
- Embedded systems (for example, Linux Kernel)