Design and Analysis of Algorithms

Lecture-10

Dharmendra Kumar (Associate Professor)
Department of Computer Science and Engineering
United College of Engineering and Research,
Prayagraj

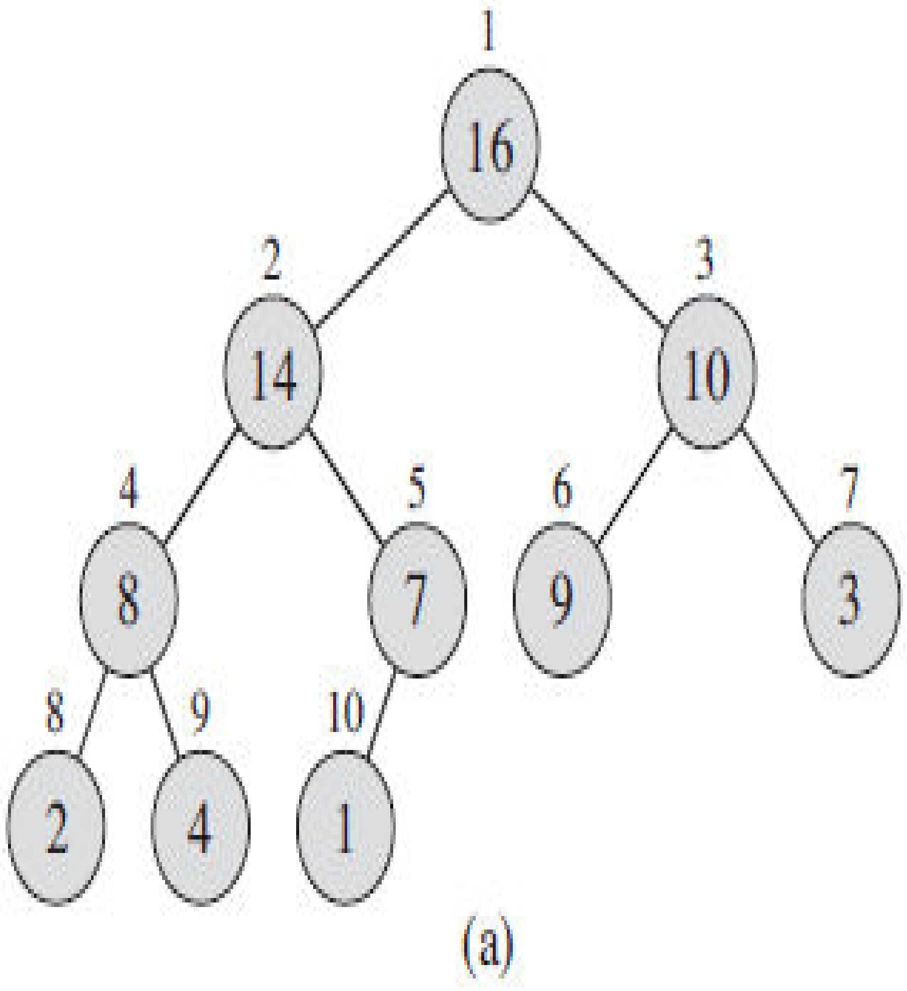
Heapsort

Heap

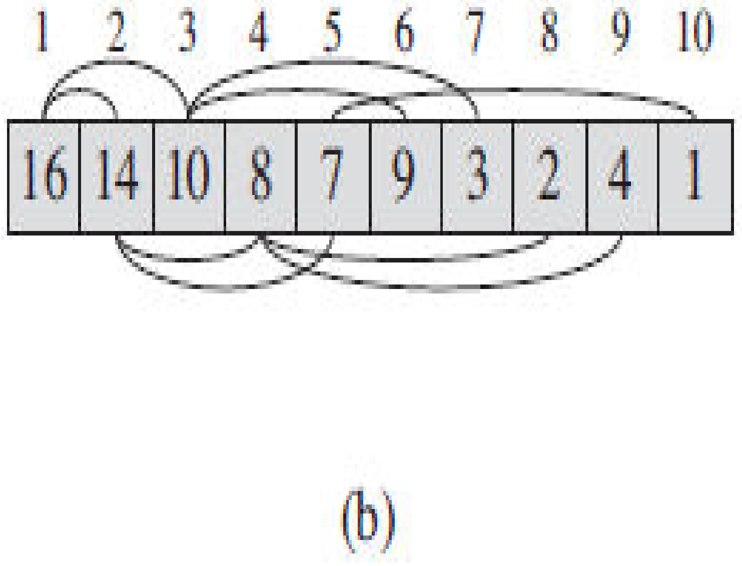
- The ***(binary) heap*** data structure is an array object that we can view as a nearly complete binary tree.

- Each node of the tree corresponds to an element of the array. The tree is completely filled on all levels except possibly the lowest, which is filled from the left up to a point.

Heapsort



Max-heap



Array

Heapsort

Index: If i the index of a node, then the index of parent and its child are the following:-

Parent(i) = $\lfloor i/2 \rfloor$

Left(i) = 2i

Right(i) = 2i+1

Note: Root node has always index 1 i.e. A[1] is root element.

Heap-size: Heap-size is equal to the number of elements in the heap.

Height of a node: The height of a node in a heap is the number of edges on the longest simple downward path from the node to a leaf.

Height of heap: The height of the heap is equal to the height of its root.

Heapsort

Types of heap

There are two kinds of binary heaps:

(1) max-heaps (2) min-heaps

Max-heap: The heap is said to be max-heap if it satisfy the **max-heap property**.

The **max-heap property** is that the value at the parent node is always greater than or equal to value at its children.

Min-heap: The heap is said to be min-heap if it satisfy the **min-heap property**.

The **min-heap property** is that the value at the parent node is always less than or equal to value at its children.

Heapsort

Heap sort algorithm consists of the following two sub-algorithms.

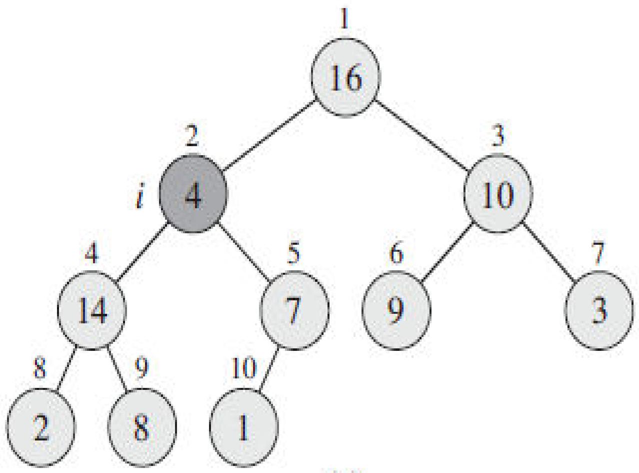
(1) Max-Heapify: It is used to maintain the max-heap property.

(2) Build-Max-Heap: It is used to construct a max-heap for the given set of elements.

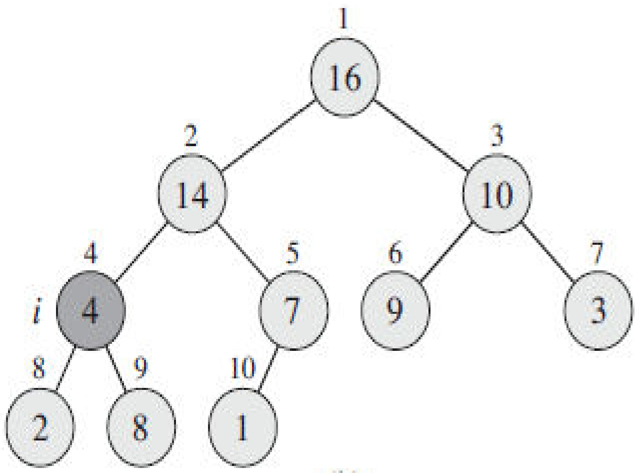
Heapsort

Max-Heapify Algorithm

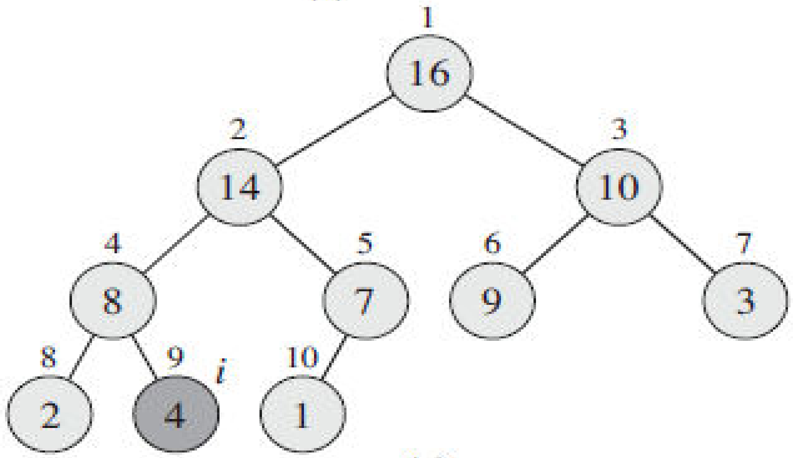
Action done by max-heapify algorithm is shown in the following figures:-



(a)



(b)



(c)

Heapsort

Max-Heapify Algorithm

```
MAX-HEAPIFY(A, i)
 1  l = LEFT(i)
 2  r = RIGHT(i)
 3  if l ≤ A.heap-size and A[l] > A[i]
 4          largest = l
 5  else largest = i
 6  if r ≤ A.heap-size and A[r] > A[largest]
 7          largest = r
 8  if largest ≠ i
 9          exchange A[i] with A[largest]
10          MAX-HEAPIFY(A, largest)
```

Heapsort

<u>Time complexity of Max-Heapify Algorithm</u>

The running time of max-heapify is determined by the following recurrence relation:-

$$T(n) \leq T(2n/3) + \theta(1)$$

Here n is the size of the sub-tree rooted at node i.

Using master theorem, the solution of this recurrence relation is

$$\mathbf{T(n) = \theta(lg\ n)}$$