Algorithms and Data Structures № 5

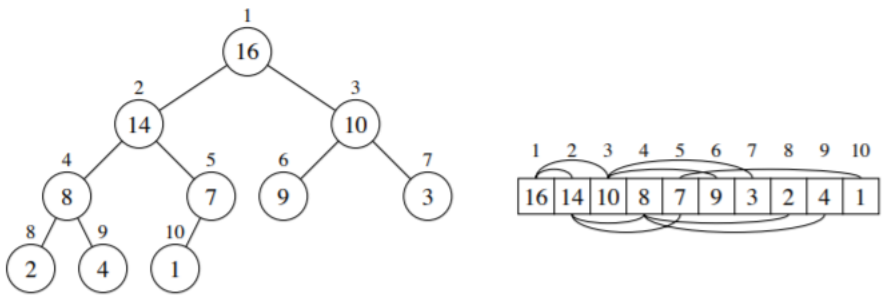
Introduction

Heap data structure: Heap is a nearly complete binary tree.

- Height of node = no. of edges on a longest simple path from the node down to a leaf.
- Height of heap = height of root = (lg n).

A heap can be stored as an array A.

- Root of tree is $A[1]$.
- Parent of $A[i] = A[i/2]$.
- Left child of $A[i] = A[2i]$.
- Right child of $A[i] = A[2i+1]$.

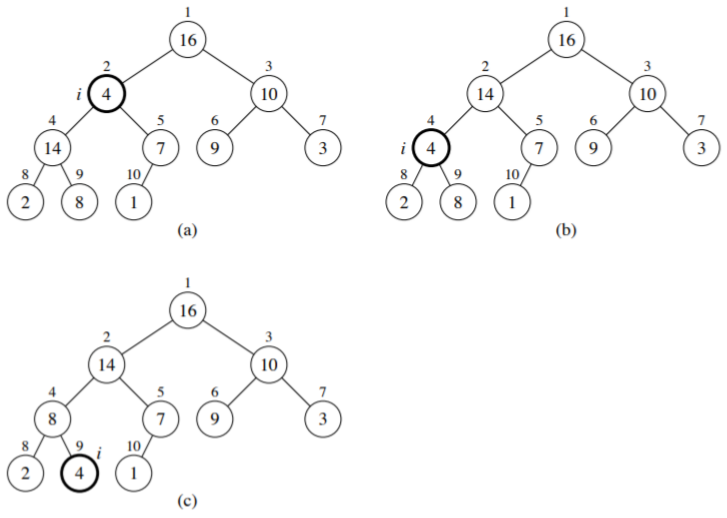


Maintaining the heap property: MAX-HEAPIFY is important for manipulating max-heaps. It is used to maintain the max-heap property.

- Before MAX-HEAPIFY, $A[i]$ may be smaller than its children.
- Assume left and right subtrees of i are max-heaps.
- After MAX-HEAPIFY, subtree rooted at i is a max-heap.

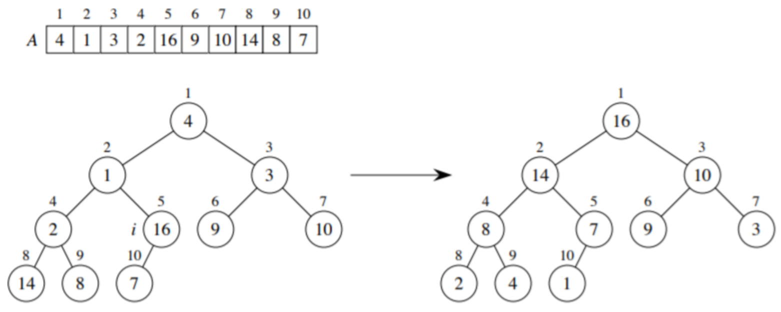
Time: O(lg n).

```
MAX-HEAPIFY(A,i, n)
l := LEFT(i)
r := RIGHT(i)
if l <= n and A[l] > A[i]
    then largest := l
    else largest := i
if r <= n and A[r] > A[largest]
    then largest := r
if largest <> i
    then
        exchange A[i] <-> A[largest]
        MAX-HEAPIFY(A,largest, n)
```



Building a heap: The following procedure, given an unordered array, will produce a max-heap.

```
BUILD-MAX-HEAP(A, n)
for i := n/2 downto 1
    do MAX-HEAPIFY(A,i, n)
```



In details it goes like this:

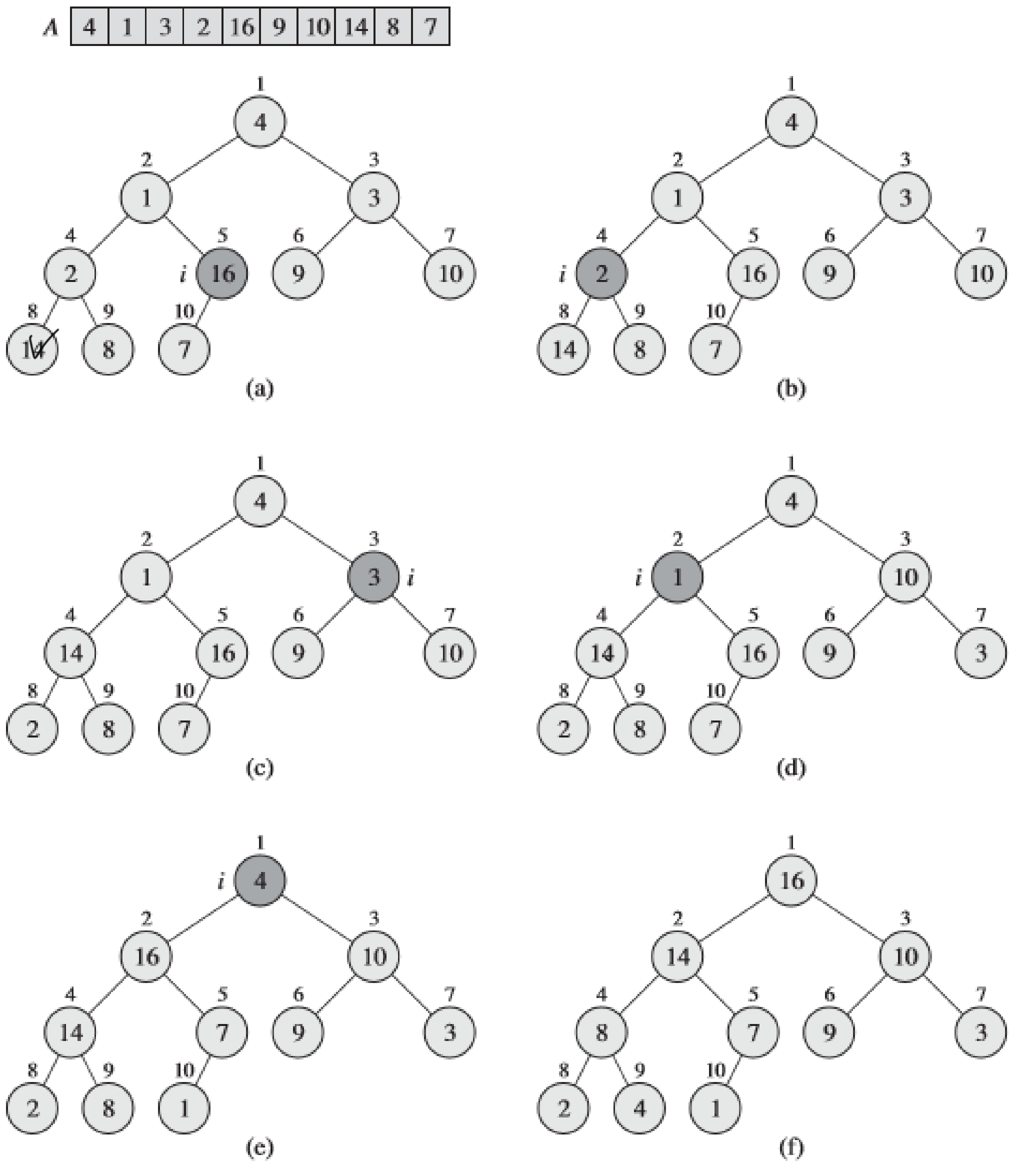


Figure 1: Heapifying a table

Heapsort algorithm: Given an input array, the heapsort algorithm acts as follows:

- Builds a max-heap from the array.
- Starting with the root (the maximum element), the algorithm places the maximum element into the correct place in the array by swapping it with the element in the last position in the array.
- ”Discard“ this last node (knowing that it is in its correct place) by decreasing the heap size, and calling MAX-HEAPIFY on the new (possibly incorrectly-placed) root.
- Repeat this ”discarding“ process until only one node (the smallest element) remains, and therefore is in the correct place in the array.

```
HEAPSORT(A, n)
BUILD-MAX-HEAP(A, n)
for i := n downto 2
    do
        exchange A[1] <-> A[i]
        MAX-HEAPIFY(A,1,i-1)
```

Heapsort:

- $O(nlgn)$ worst case – like merge sort.
- Sorts in place – like insertion sort.
- Combines the best of both algorithms.

Problem 1 (2 points)

What are the minimum and maximum numbers of elements in a heap of height h?

Problem 2 (2 points)

Show that an n-element heap has height $\lfloor lgn \rfloor$.

Problem 3 (6 points)

Illustrate the operation of HEAPSORT on the array

$$A = [5; 13; 2; 25; 7; 17; 20; 8; 4]$$

For extra credit

Challenging programming problems - not obligatory, but just for fun and extra credit :)

1. Solve the problem on Hackerrank: (+4 points):
   https://www.hackerrank.com/challenges/to-heap-or-not-to-heap/problem
2. Solve the problem on Hackerrank: (+4 points):
   https://www.hackerrank.com/challenges/ctci-find-the-running-median/problem