Alexandria University
Faculty of Engineering
Computer and Systems Engineering
Department

CS 223 Data Structures 2
Assigned: Friday, February 27, 2015
Due: Thursday, March 9, 2015

Data Structures 2 - Lab 1
Implimenting Binary Heap & Sorting Techniques

The goal of this lab is to become familiar with the binary heap data structure as well as different sorting techniques.

1 Binary Heap

1.1 Introduction

The (binary) **heap** data structure is an array object that we can view as a nearly complete binary tree as shown in Figure 1. 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. An array A that represents a heap is an object with two attributes: A.length, which (as usual) gives the number of elements in the array, and A.heap-size, which represents how many elements in the heap are stored within array A.

There are two kinds of binary heaps: max-heaps and min-heaps. In both kinds, the values in the nodes satisfy a heap property, the specifics of which depend on the kind of heap. In a max-heap, the max-heap property is that for every node i other than the root,

$$A[parent[i]] \geq A[i] \tag{1}$$

that is, the value of a node is at most the value of its parent.

1.2 Requirements

In this assignment, you're required to implement some basic procedures and show how they could be used in a sorting algorithm:

- The MAX-HEAPIFY procedure, which runs in $O(\lg n)$ time, is the key to maintaining the max-heap property.
- The BUILD-MAX-HEAP procedure, which runs in linear time, produces a max-heap from an unordered input array.
- The HEAPSORT procedure, which runs in $O(n \lg n)$ time, sorts an array in place.
- The MAX-HEAP-INSERT, and HEAP-EXTRACT-MAX procedures, which run in $O(\lg n)$ time, allow the heap data structure to implement a priority queue.

You're required to implement the above procedures, pseudo code for the above procedures are explained in details in the textbook attached with this problem statement: "Cormen, Thomas H., et al. Introduction to algorithms. Vol. 2. Cambridge: MIT press, 2001". Feel free to select the programming language of your choice.

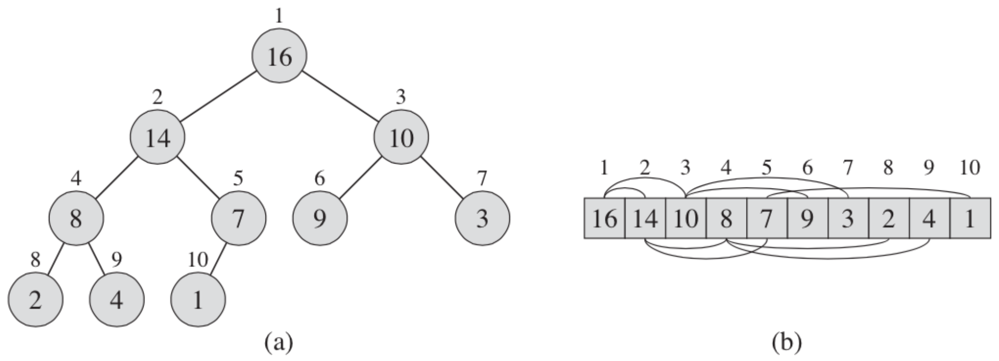

Figure 1: A max-heap viewed as (a) a binary tree and (b) an array. The number within the circle at each node in the tree is the value stored at that node. The number above a node is the corresponding index in the array. Above and below the array are lines showing parent-child relationships; parents are always to the left of their children. The tree has height three; the node at index 4 (with value 8) has height one. - figure from Introduction to Algorithms - 3rd Edition

2 Sorting Techniques

- You are required to implement the "heapsort" algorithm as an application for binary heaps. You're required to compare the running time performance of your algorithms against:
  - An $O(n^2)$ sorting algorithm such as Selection Sort, Bubble Sort, or Insertion sort.
  - An $O(n \lg n)$ sorting algorithm such as Merge Sort or Quick sort algorithm in the average case.[1].

  In addition to heapsort, select **one** of the sorting algorithms from each class mentioned above.
- To test your implementation and analyze the running time performance, generate a dataset of random numbers and plot the relationship between the execution time of the sorting algorithm versus the input size.

3 Bonus

The following parts could you considered as a bonus:

[1]In the worst case, the runtime for quick sort could be $O(n^2)$

- Implementing more sorting techniques (example: implementing both the quick sort and merge sort)
- Graphical Illustration for the operation of different sorting algorithms (Example: `http://www.sorting-algorithms.com/heap-sort`)

4 References

- Cormen, Thomas H., et al. Introduction to algorithms. Vol. 2. Cambridge: MIT press, 2001.
- Quick Sort: `http://en.wikipedia.org/wiki/Quicksort`
- Merge Sort: `http://en.wikipedia.org/wiki/Merge_sort`
- `http://www.sorting-algorithms.com/heap-sort`

Good Luck