Data Structures and Algorithms **Spring-Summer 2013/14**

Assignment 5

Mouna Kacimi, Werner Nutt, Camilo Thorne

Heap Sort and Quicksort

Instructions: Your assignment should represent your own effort. However, you are not expected to work alone. It is fine to discuss the exercises and try to find solutions together, but each student shall write down and submit his/her solutions separately. It is good academic standard to acknowledge collaborators, so if you worked together with other students, please list their names.
You can write up your answers by hand (provided your handwriting is legible) or use a word processing system like Latex or Word. Experience shows that Word is in general difficult to use for this kind of task.
For a programming task, your solution must contain (i) an explanation of your solution to the problem, (ii) the Java code, in a form that we can run it, (iii) instructions how to run it. Also put the source code into your solution document. For all programming tasks, it is not allowed to use any external libraries ("import") if not stated otherwise.
Please, include name and email address in your submission.

1. Heaps and Heapsort

Answer the following questions. Briefly explain your answers.

1. Suppose you are given an array of length n that represents a max-heap. Describe (if you like, in words) an algorithm that finds the smallest element in this array/heap. What is the running time?

2. What is the running time of Heapsort on an array of length n that is already sorted in increasing order?

3. What is the running time of Heapsort on an array of length n that is already sorted in decreasing order?

4. Is there an initial configuration of an array of length n that will result in linear running time of Heapsort?

(10 Points)

2. Sorting algorithm comparison

In this exercise you are asked to:

- Implement in Java the *Quicksort* algorithm for arrays of integers.
- Implement in Java the *Hybrid Quicksort* algorithm for arrays of integers, a variant of Quicksort that depends on an additional parameter k. The algorithm works as follows:
  - if the size of the portion of the array to be sorted is greater than k, it recursively calls the Hybrid Quicksort method with the same k;
  - if the size of the portion of the array to be sorted is less than or equals to k, it sorts that portion by using the Insertion Sort algorithm.
- Empirically compare these two sorting algorithms for different values of k (for instance, $k = \{1, 2, 5, 10, 25, 50, \ldots\}$) and different array sizes. Find out for which values of k and for which array sizes Hybrid Quicksort outperforms the standard Quicksort.

Provide:

1. an analysis for which value of k and for which array sizes the standard Quicksort algorithm starts to outperform the hybrid one;
2. a discussion and possible explanation for the obtained running times.

Background: Many implementations of recursive sorting algorithms, like Quicksort or Mergesort, actually implement such a hybrid version.

Hint: Write Java code to make the tries and take the measurements, rather than doing it manually with pencil and paper.
Consider also to increase the memory size if needed, using the virtual machine option -XX:AggressiveHeap.

(20 Points)

Submission: Until Tue, 15 April 2014, 8:30 pm, to

`dsa-submissions AT inf DOT unibz DOT it.`