Data Structures - Assignment no. 7

Remarks:

- **Please write your exercises in pen, or in clearly visible pencil. Please write very clearly.**
- **For every question where you are required to write pseudo-code, also explain your solution in words.**

1. Suppose that the splits at every level of quicksort are in proportion $1 - \alpha$ to α, where $0 < \alpha < 1/2$ ia a constant. Show that the minimum depth of a leaf in the recursion tree is approximately - $\log n / \log \alpha$ and the maximum depth is approximately - $\log n / \log(1 - \alpha)$. (Don't worry about integer round-off).

2. During the running procedure of $RANDOMIZED - QUICKSORT$, how many calls are made to the random-number generator $RANDOM$ in the worst case? How about in the best case? Give your answer in terms of Θ-notation.

3. In this question we discuss a model called the "extended comparison model", which is like the comparison model, except that you are allowed 5 types of questions: (i) "$a = b$?", (ii) "$a < b$?", (iii) "$a > b$?", (iv) "$a < b + 100$?", (v) "$a > b + 100$?". Prove a lower bound of $\Omega(n \log n)$ for sorting an array of size n in the extended comparison model.

4. (a) You are given two arrays, A and B, each of size n. Give an algorithm that returns an array C of size n, such that $C[i]$ is equal to the number of elements of A that are less or equal to $B[i]$. The algorithm should run in time $O(n \log n)$. Describe the algorithm and explain why the running time is $O(n \log n)$. You do not have to give pseudo-code.

   (b) Prove a lower bound of $\Omega(n \log n)$ for this problem in the comparison model.
   Hint: You can prove this lower bound directly. However, it is easier to give a reduction. To do this, you should: (i) Prove that if you can solve this problem in time $f(n)$ then you can sort an array of size n in time $O(f(n) + n)$; (ii) Deduce from this that if you can solve this problem in time $f(n)$ then $f(n) = \Omega(n \log n)$.

5. (a) You are given an array of size n, which contains $\log n$ distinct elements, each of them occurring exactly $\frac{n}{\log n}$ times. Give an algorithm that sorts this array in time $O(n \log \log n)$. Describe the algorithm and explain why the running time is $O(n \log \log n)$. You do not have to give pseudo-code.

   (b) Prove a lower bound of $\Omega(n \log \log n)$ in the comparison model for this problem.
   Hint: Work like the lower bound that you have seen in class. First prove that there must be at least $(n!) / \left((n/\log n)!\right)^{\log n}$ leaves in the comparison tree. Then use Stirling's approximation of $n!$ to prove that the depth of the tree is $\Omega(n \log \log n)$. When using Stirling's approximation, it is enough to use that:

   $$n! = \Theta\left(\sqrt{n}\left(\frac{n}{e}\right)^n\right)$$

   You can use this approximation without proving it.

6. You are given an array of size n. In this array there exists an index j such that for every $i < j$, $A[i] < A[i+1]$ and for every $i > j$, $A[i] > A[i+1]$.

   (a) Describe an efficient algorithm to find index j.

   (b) Prove that $\Omega(\log n)$ is the lower bound on the number of required comparisons in the worst case.