Assignment #3

Date Due: November 15, 2017

Total: 100 marks

For each of the algorithms designed you have to

- compute its time complexity,
- prove that the alleged time complexity corresponds to that of your algorithm,
- prove that your algorithm is optimal or at least give an argument to support your design.
- You should specify for each algorithm designed the programming technique used (brute force approach should be also mentioned). Missing the name of the programming technique may have as a result marks deducted.

In case you design a sub-optimal algorithm you may loose marks, depending on how slow is your algorithm, compared to an optimal one.

1. (20 marks) The entries in a phone book have multiple keys: last name, first name, address, and telephone number. They are sorted first by last name, then by first name, and finally by address. Design a variation of radix sort[1] for sorting the entries in a phone book, assuming that you have available a procedure that sorts records on a single key[2]. The running time of your algorithm should be of the same order as the running time of the procedure that sorts on a single key. (Beware: the procedure sorting using just one key may not be stable.)

2. (15 marks) Write an algorithm that receives as input two n-element arrays A and B of real strict positive numbers and a value v. The algorithm returns 1 if there are i and j such that $v = A[i] * B[j]$ and 0 otherwise.

3. (10 marks) Assume we have n real numbers . Design an algorithm to find the pair (x, y) for which the distance between x and y is minimal. Compute its time complexity.

4. (10 marks) Write an algorithm for the union of two n-element sets A and B of real numbers. Sets are represented by sorted arrays.

5. (10 marks) Repeat exercise 4 for intersection.

   Write the algorithm and prove its efficiency.

[1] You have comparator functions for names addresses and phone numbers

[2] if you give as input an array or records it is able to sort it using just one key

6. (20 marks) Is it possible to implement quick sort using binary search trees lists? What would be the time complexity? What about linked lists? Discussion.

7. (20 marks) Two thieves are carrying n gold coins in two bins, in a stolen car. The physical characteristics of the the gold coins are the same, but while the coins in the first bin are all the same, the ones in the second bin are different from the ones in the first bin. At one corner, the driver suddenly brakes and the coins are all mixed up. They have a device that can be applied to two coins and tells whether they are different or not. It is known in advance that most of the coins (more then 50%) are from the first bin. Find the optimal algorithm that the two thieves should apply in order tor put the coins back into the bins as fast as possible, to hide the first bin before the police will arrive. How many comparisons are necessary, in the worst case, to find at least one coin from the first bin?

   (Beware: it is possible that two coins are identical, but do not belong to the first bin.)