Assignment #3

Date Due: November 30, 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. (15 marks) Two children are collecting stamps of various values. They are meeting to send an envelope that needs exactly two stamps with a total value of v Write an efficient algorithm that helps the two children to find fast a pair of stamps one from each child that will allow them to send the envelope with a minimal cost.

   Compute its running time and compare with a brute-force solution.

2. (15 marks) Consider the following greedy strategy for finding a shortest path from vertex *start* to vertex *goal* in a given connected graph.

   (a) Initialize *path* to *start*.
   (b) Initialize *VisitedVertices* to the set $\{start\}$
   (c) Set *current* to *start*.
   (d) If $current = goal$, return *path* and *exit*. Otherwise, continue.
   (e) Find the edge $(current, v)$ of minimum weight such that v is adjacent to *current* and v is not in *VisitedVertices*.
   (f) Add v to *path*.
   (g) Add v to *VisitedVertices*.
   (h) Set *current* equal to v and go to step 2d.

Does this greedy strategy always find a shortest path from *start* to *goal*? Either explain intuitively why it works, or give a counter-example.

3. (20 marks) Design a dynamic programming algorithm for change-making problem.

4. (20 marks) **Shortest-Common-Super-sequence Algorithm**

   Given two sequences $X = \langle x_1, \dots, x_m \rangle$ and $Y = \langle y_1, \dots, y_m \rangle$, find the length L of the shortest possible sequence $Z = \langle z_1, \dots, z_L \rangle$ such that both X and Y are sub-sequences of Z. You are to give a dynamic programming algorithm which correctly solves this problem, by answering the questions below. You may **not** use the Longest-Common-Subsequence algorithm as part of your algorithm.

   (a) Define an array $C[i][j]$ to be used in your solution.

   (b) Give a recurrence (include initialization) which can be used to fill in your array. Justify your recurrence briefly.

   (c) Give an algorithm which fills in the array using your recurrence.

   (d) Show how to find the solution L to the problem from the array C.

   (e) Use the big-O notation to estimate the runtime of the algorithm.

5. (30 marks) Draw the recursion tree for the merge-sort procedure on an array of 32 elements. Explain why memoisation is ineffective in speeding up a good divide-and-conquer algorithm such as merge sort. Would memoization be effective in case of quick-sort?

6. Can we use BFS strategy to design a single-source shortest-path algorithm?

   (a) (20 marks) Explain your algorithm in case you can do it or explain why not, by giving strong arguments against this design including a counter example.

   (b) (10 marks) What about DFS?