Assignment 1

COMP 5703 - Fall 2014

1 Instructions

This assignment consists of 10 problems. This is due at the start of the class on October 19, 2015. Please write clearly and answer questions precisely. As a thumb rule, the answer should be limited to < 2 written pages, with ample spacing between lines and in margins, per question. Always start a new question on a new page, starting with Question 1, followed by Question 2, ..., Question n. Please cite all the references (including web-sites, names of friends, etc.) which you have used/consulted as the source of information for each of the questions. BTW, when a question asks you to design an algorithm - it **requires** you to (1) Clearly spell out the **steps** of your algorithm in pseudo code (2) **Prove** that your algorithm is correct and (3) **Analyze** the running time. By default a graph $G = (V, E)$ is simple, undirected and connected.

2 Problems

1. Consider the Proposal Algorithm that was discussed in the class (and is on the course-notes/web-page as well). The input consists of a list of n men M and n women W, where each man has an ordered list of n women (a permutation) whom he will like to marry. Similarly, each woman has an ordered list (a permutation) of n men whom she will like to marry. Output is a set of n pairs $\mathcal{M}$ forming a stable perfect matching, where each pair in $\mathcal{M}$ consists of a man and a woman. Moreover, each man is present in a unique pair, and each woman is present in a unique pair in $\mathcal{M}$. Prove that the output of this algorithm is always the same and is independent of in which order the 'unmarried men' are chosen in the while loop. That is, show that any execution of this algorithm will always result in the stable matching corresponding to $\mathcal{M} = \langle m, \mathsf{best}(m) \rangle$, $\forall m \in M$.

2. Suppose that there are more men than women, i.e. $|M| > |W|$. As before, each men has a permutation of women according to his preference list, and similarly each women has a permutation of men corresponding to her preference list. Define what will be the notion of stable matching in this case, and design an algorithm for this new definition of stable matching.

3. We assumed that for each man (or woman), the preference list consists of a permutation of women (men) where there is a strict order of preferences. The woman appearing 1st on the list is most preferred and then the next one and then the next one etc.. Suppose that the preference list of man still consists of a permutation, but the ordering is not strict. In other words, lets say that in a configuration of $|M| = |W| = 7$, the preference list of $m_1 = w_3, \{w_4, w_1, w_2\}, w_7, \{w_5, w_6\}$, meaning that w_3 is most preferred, followed by $\{w_4, w_1, w_2\}$,

each having the same preference, followed by w_7, followed by $\{w_5, w_6\}$ having equal preference. Think of what will the notion of stability in this case, what will be an algorithm for stable matching. [I am expecting some ideas here and not a full solution - as this is bit non-trivial and one of the seminar talks is supposed to address this problem]

4. Running time of many divide-and-conquer algorithms are expressed by a recurrence relation. Consider the following recurrence relation

$$T(n) = T(xn) + T((1-x)n) + cn$$

expressed in terms of x and n where x is a constant in the range $0 < x < 1$ and n is the size of the problem, typically very large. Note that $c > 0$ is a fixed constant. (BTW, $T(n)$ represents the time to solve the problem of size n. It is solved recursively by partitioning it in two subproblems, one of size xn and other of size $(1-x)n$, and the time required to partition the problem and merging the solution of the subproblems is cn.) Is the asymptotic complexity the same when $x = 0.5, 0.1$ and 0.001? What happens to the constants hidden in the $O()$ notation. (Some of you may be tempted to use Master's Theorem directly - best is to ignore the statement of Master's Theorem, and concentrate on its proof. Purpose of this exercise is to learn the proof of Master's theorem, as well as remind you of recurrence relations.)

5. Assume all edges in a weighted simple connected graph $G = (V, E)$ have distinct cost.

   (a) Show that the edge with the maximum weight in any cycle in G cannot be any minimum spanning tree of G.

   (b) Consider an edge $e = (uv)$ which is not in a spanning tree T of G. (Note that T may not be a MST.) Consider the unique path, $\pi(u, v)$, between nodes u and v in T. Prove the following. If the weight of each edge on $\pi(u, v)$ is less than the weight of e then e cannot occur in any minimum spanning tree of G.

6. We have seen examples of weighted undirected graphs which may have multiple minimum spanning trees. Prove that if all edge weights are distinct (no two edges have the same weight), there is always a unique minimum spanning tree in a graph.

7. Suppose you are given n-points in the plane. We can define a complete graph on these points, where the weight of an edge $e = (u, v)$, is Euclidean distance between u and v. We need to partition these points into k non-empty clusters, for some $n > k > 0$. The property that this clustering should satisfy is that the minimum distance between any two clusters is maximized. (The distance between two clusters A and B is defined to be the minimum among the distances between pair of points, where one point is from cluster A and the other from cluster B.) Show that the connected components obtained after running Kruskal's MST algorithm till it finds all but the last $k-1$ (most expensive) edges of MST produces an optimal clustering. You need to recall how Kruskal's algorithm work - but do not worry much about its complexity analysis.

8. This problem requires you to review depth-first search traversal in graphs. Show that in a depth-first search of a graph G, if we output a left parenthesis '(' when a node is accessed for the first time and output a right parenthesis ')' when a node is accessed for the last time,

then resulting parenthesization (or bracketing sequence) is proper. Each left '(' is properly matched with each right ')'. (BTW, this property is basis for several graph algorithms, e.g. Euler Tour technique - which we will see during the course.)

9. An Euler tour of an undirected simple connected graph $G = (V, E)$ is a path that starts and ends at the same vertex and uses each edge exactly ones. A graph G has an Euler tour if and only if the degree of each vertex is even. Assume that G is represented using an adjacency list representation. Provide an algorithm running in $O(|V| + |E|)$ time that (a) checks whether G has an Euler tour (b) if it has an Euler tour, it outputs one. Please present your algorithm in terms of Steps (pseudo-code) - not like a C-program! Provide correctness and complexity analysis etc.

10. Look at the list of A.M. Turing award winners - this is like the Nobel prize in CS. Identify at least three award winners who have worked in the field of Algorithms and/or Data Structures. Give some reasoning for your choice. For each of them, list two of their main contributions (i.e. publications), and state in your own words what is their main contribution. [I am not expecting more than two paragraphs per researcher.]