Homework 4

ECE 345 Algorithms and Data Structures
Winter Semester, 2015

Due: Mar 24 at 12noon in the ECE345 dropbox.

- All page numbers are from 2009 3rd edition of Cormen, Leiserson, Rivest and Stein.
- For each algorithm you are asked to design you should give a detailed *description* of the idea, proof of algorithm correctness, termination, analysis of time and space complexity. If not, your answer will be incomplete and you will miss credit. You are allowed to refer to pages in the textbook.
- Do not write C code! When asked to describe an algorithm give analytical pseudocode.
- Staple your homework properly. Use a stapler; do not use glue or other weird material to put it together. If you are missing pages, we are not responsible for it but you are!
- Write *clearly*, if we cannot understand what you write you may not get credit for the question. Be as formal as possible in your answers. Don't forget to include your name(s) and student number(s) on the front page !

---

1. **[Graphs, 30 points]** You are developing an upcoming social network website. The network is represented as a graph where each user is a node. If u and v are friend, there is an edge between u and v (friendship is undirected). Also, there is an interest rating $I(u, v)$ based on how much u pays attention to v (for example, $I(u, v)$ can be calculated based on how many times a day u views v's profile or comments on v's wall). While friendship is undirected, interest rating is not, i.e. $I(u, v) \neq I(v, u)$. Also, all interest rating falls in the range (0,1). A user u is connected to a user v, if there exists a path from u to v through some mutual friends, i.e. $u = u_0$ has a friend u_1, who has a friend $u_2, \cdots$, who has a friend $u_k = v$. Define a strength of such a connection for a path p between u and v to be:

$$S(p) = \prod_{i=0}^{k-1} (I(u_i, u_{i+1}))$$

   (a) Assume that users u and v are connected. Find the connection p from u to v with the largest strength $S(p)$.

   (b) Now you want add a friend suggestion feature. For an user s you want to suggest friends for s; however, you do not want to suggest all the users that s can connect to because your network has reached million of users (and would be computationally intensive to do so). Instead, you will only suggest users v such that s can connect to v with k or less mutual friends. For each node v suggested, you also return the connection p that has the largest strength from all paths that connects s to v with k or less mutual friends. Note that the connection p might not be the globally largest strength path between s and another vertex because of the added restriction of using k or less mutual friends. Assume that the network has $|V|$ users and $|E|$ friend pairs. Describe your algorithm in pseudocode and analyze its runtime.

2. **[Shortest Path, 20 points]**

   (a) Prove that Dijkstra's algorithm does not generate the shortest path when there exist negative weight edges, even if there is no non-positive weight cycle in the directed graph.

   (b) Give an algorithm that detects a negative weight cycle in a strongly connected directed graph. Analyze its complexity and formally prove its correctness.

3. **[Minimum Spanning Tree, 20 points]**

   (a) Let the weight of a spanning tree be the weight of the maximum weight edge in the tree (the weight is **not** the sum of all weights of all edges in the tree). Give an algorithm to compute a minimum weight spanning tree under this new definition of weight of a spanning tree. Analyze its complexity.

   (b) Given a graph G and a minimum spanning tree T of G, suppose that we decrease the weight of the one of the edges that is not in T. Give an algorithm in pseudocode that finds the minimum spanning tree of the modified graph. Analyze its complexity and formally prove its correctness.

4. **[Topological Sort, 20 points]** A directed graph G is said to be semi-connected if for any pair of vertexes u, v, u is connected to v or v is connected to u. Give an algorithm to find out whether G is semi-connected or not and analyze its complexity.