COMPSCI330 Design and Analysis of Algorithms
Assignment 3

Due Date: Tuesday, Nov 1, 2016

Guidelines

- **Describing Algorithms** If you are asked to provide an algorithm, you should clearly define each step of the procedure, establish its correctness, and then analyze its overall running time. There is no need to write pseudo-code; an unambiguous description of your algorithm in plain text will suffice. If the running time of your algorithm is worse than the suggested running time, you might receive partial credits.
- **Typesetting and Submission** Please submit each problem as an *individual* pdf file for the correct problem on Sakai. LaTeX is preferred, but answers typed with other software and converted to pdf is also accepted. Please make sure you submit to the correct problem, and your file can be opened by standard pdf reader. **Handwritten answers or pdf files that cannot be opened will not be graded.**
- **Timing** Please start early. The problems are difficult and they can take hours to solve. The time you spend on finding the proof can be much longer than the time to write. If you submit within one week of the deadline you will get half credit. Any submission after that will not receive any credit.
- **Collaboration Policy** Please check this page for the collaboration policy. You are **not** allowed to discuss homework problems in groups of more than 3 students. **Failure to adhere to these guidelines will be promptly reported to the relevant authority without exception.**

Problem 1 (Public Transit)**.** (20 points) Alice wants to go to Durham downtown by the public transit system. She has a transit map of the area, which in this problem is abstracted as a graph. Alice is at a vertex s and she is going to vertex t. The edges on the graph represents possible bus schedules. Each directed edge $e = (u, v)$ is described by three numbers: $l(e) \geq 0$ is the amount of time to get from u to v by bus; $t(e) \geq 0$ is the first time of the bus; $int(e) > 0$ is the interval of the bus. Starting at time $t(e)$, there will be a bus from u to v every $int(e)$ minutes (so the bus leaves u at times $t(e), t(e) + int(e), t(e) + 2int(e), ...$). In this problem you need to design an algorithm to help Alice figure out what is the fastest time to get to t. Alice leaves s at time 0 and she does not care about how many transfers to make. The graph is connected. If Alice arrives some vertex u at time t, and at the same time there is a bus leaving, Alice will be able to catch the bus.

Your algorithm should run in time $O(m + n \log n)$.

Problem 2 (Generic MST)**.** We consider the following general algorithm for Minimum Spanning Tree:

```
Initialize set F = empty
For i = 1 to n - 1
    Choose some cut S that agrees with the connected components of F
    Choose edge e to be one of the minimum cost edge in cut(S)
    Add e to F
```

(a) (10 points) Show that this algorithm always finds a Minimum Spanning Tree (you can use the Key Theorem we proved in class).
(b) (10 points) Show that for any Minimum Spanning Tree T, and any ordering of the edges $(e_1, e_2, ..., e_{n-1})$ in tree T, there is a way to choose S and e in the iterations, so that the edge picked in i-th iteration is exactly e_i.

Problem 3 (Cooking Recipe)**.** Bob wants to become a chef so he is practicing his cooking skills. He tries to follow a recipe book but the book is poorly written. The book only specifies the relative times between different steps (e.g. you need to preheat the oven at least 30 minutes before preparing the turkey). More concretely, there are n steps that Bob needs to do. If we use x_i to denote the exact time to start working on the i-th step, the recipe specifies inequalities about differences of x_i's (e.g. $x_{preheat} - x_{turkey} \leq -30$). All of the inequalities are of the form $x_i - x_j \leq c_{i,j}$. You need to design an algorithm to help Bob find a schedule for the cooking (a set of values x_i's that satisfy all the inequalities).
(a) (6 points) Prove that in a graph, let $d(u)$ be the length of shortest path from s to u, then for any (directed) edge (u, v)

$$d(v) \leq d(u) + l(u, v).$$

(b) (6 points) Design an algorithm for Bob (find a set of valid x_i's that satisfy all inequalities). If there are m inequalities, your algorithm should run in time $O(nm)$. (For this problem, you only need to prove if your algorithm returns a solution, then it is correct. However, you will only get full credit if your algorithm only fails when there is no solution.)
(c) (8 points) Explain what happens to your algorithm when there is no solution to the system of inequalities. (This is part of the correctness proof for (b). In this part you need to prove your algorithm succeeds if and only if the system of inequalities has a solution.)

Problem 4 (Classroom Assignment)**.** (20 points) Let's continue to talk about the classroom assignment problem in Lecture 12 (bipartite matching). Recall we talked about the fact that an augmenting path is like a way to reschedule the room: suppose Professor A is teaching a course that do not currently have a room, and the only room that can fit A's course is room X; however, currently room X is taken by Professor B's course. One way to construct an augmenting path of length 3 is to ask Professor B to find a new room for his course (say room Y is available and fits B's course). If this can be done then we can let Professor A use room X and Professor B use room Y.

You are helping with the classroom schedules. However, Professors are easily confused by complicated requests (e.g. "Hi Professor C I would like to swap your class to room Z because Professor B would like to swap to your current room Y, because Professor A needs to use Professor B's current room X." This is just an augmenting path of length 5.), you cannot afford to use an augmenting path of length equal to or more than $2k + 1$ (where k is a given integer).

Show that if the bipartite graph between courses and classrooms have a matching of size r, and you have a matching M such that there are no augmenting paths of length smaller than $2k + 1$, then the size of the matching M is at least $\frac{rk}{k+1}$.

Problem 5 (Alternative Link)**.** (20 points) Suppose you have a graph G and a Minimum Spanning Tree T. These are already in memory, stored in whatever form (adjacency list/array) you want. Now there is a new edge $e = (u, v)$ added to G with weight $w(u, v)$. Design an algorithm to update the tree T to be a Minimum Spanning Tree for $G \cup \{e\}$. Your algorithm should run in time $O(n)$.