COMPSCI330 Design and Analysis of Algorithms
Assignment 2

Due Date: Friday, October 14, 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 (Currency System)**.** As we all know, US dollar is the official currency in United States. There are different bills of value $100, 50, 20, 10, 5, 2, 1$, and coins of value $1, 0.50, 0.25, 0.1, 0.05, 0.01$ (although you rarely see 2$ bills and 1$,0.50$ coins, they exist, see `https://en.wikipedia.org/wiki/United_States_dollar`). In this problem we consider the problem of buying something with as few number of bills/coins as possible.

A simple way to pay x dollars is to use the following Greedy-Pay algorithm:

```
Greedy-Pay(x)
   While x > 0
      Let u be the largest value bill/coin such that u <= x
      Pay the bill/coin with value u
      Let x = x - u
```

(a) (10 points) Prove that if you have unlimited number of bills/coins of all values, no matter what x is (as long as it is a multiple of 0.01) the Greedy-Pay algorithm minimizes the number of bills/coins used for the payment. (Change is not allowed here. For example, you cannot pay $99 by a $100 bill and get $1 back.)
(b) (10 points) Show an example for which the Greedy-Pay algorithm does not minimize the number of bills/coins used, when you have a limited number of bills/coins. (For example, you might not have a $2 bill or might only have 2 $20 bills.) Assume you have unlimited number of 1 cent coins so you have enough money to pay.

(Note: Your example should consist of an x - the amount you need to pay, and a set of bills/coins you have. You should be able to pay the amount x using fewer bills/coins than what Greedy-Pay uses, describe what bills you will use and what bills Greedy-Pay will use.)

Problem 2 (Physics Experiment)**.** (15 points) Professor X from Duke Physics Department is conducting a large experiment. This experiment has n crucial steps, and each step require one of Professor X's m Ph.D. students. However, these steps are very complicated. Each student only knows how to perform a subset of the steps. Professor X knows when the experiment switches hands from one student to another, there is a risk for failure due to limited communication. So he wants to finish the experiment with as few switches as possible. Since you are taking 330, Professor X hopes you can design an algorithm to help him.

Your input will be n, the number of steps, and m, the number of students. This is followed by m lists, where the i-th list contains all the steps that student i can perform.

You need to output a schedule: in particular, for each step you need to assign a student $a[i]$ to perform that step. Your schedule should minimize the number of switches (the number of steps where $a[i] \neq a[i-1]$).

As an example, suppose $n = 5$, $m = 3$, student 1 can do steps 1, 2, 5, student 2 can do steps 1, 3, 5 and student 3 can do steps 2, 3, 4. Then one valid solution is $a = [1, 3, 3, 3, 1]$. That is, student 1 will do step 1, student 3 will do steps 2, 3, 4, and then student 1 will finish step 5. The number of switch is 2. Notice that it is OK for student 1 to work on both 1 and 5.

Your algorithm should run in time polynomial in n, m.

Problem 3 (Elevator Optimization)**.** Company Y has a skyscraper and many hard-working employees. However, their skyscraper has only one slow elevator, and this has been causing a lot of waiting before people can actually start working.

The elevator starts on 1st floor. It takes 20 seconds to go up 1 floor, and takes an additional 30 seconds for each stop (therefore, it takes 80 seconds to go to 5th floor if you do not stop, and it takes 110 seconds if you stop at 3rd floor first). The employees are hard working so they don't mind taking the elevator to some floor, and then use the stairs. It takes them 30 seconds to go up or down 1 floor.

The building has n floors $(1, 2, 3, ..., n)$, and there are now m requests $1 \leq a_1 < a_2 < \cdots < a_m \leq n$ sorted in ascending order. These requests correspond to m employees who want to go to floors $a_1, a_2, ..., a_m$ respectively. You are going to schedule the elevator (which floors does it stop on) to try to make sure all of these employees can get to work within T seconds.

(a) (15 points) Suppose T is given. Design an algorithm that either outputs a set of floors that the elevator should stop on, such that all of these m employees can get to their floors within T seconds, or correctly claims that this is impossible. Your algorithm should run in time $O(m)$. (Hint: Intuitively we want to make as few stops as possible to get to the higher floors sooner.)

(b) (10 points) Design an algorithm that finds the smallest possible value of T for which all employees can get to their floors within T seconds. Your algorithm should run in time $O(m \log(n))$.

Problem 4 (Graph Traversal)**.** Recall that given a graph G, and a rooted tree T whose edges are a subset of the edges of the graph, a forward edge (blue edge in Figure 1) is an edge (u, v) of the graph where u is an ancestor of v (and u is not the direct parent of v). A cross edge (red edge in Figure 1) is an edge (u, v) of the graph such that u is not in the subtree rooted at v, and v is also not in the subtree rooted at u.

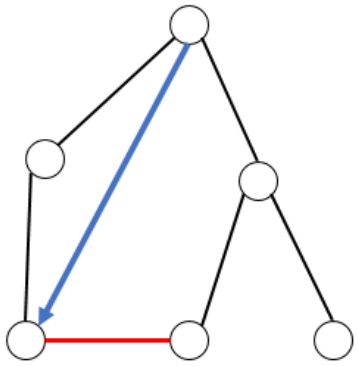

Figure 1: Black edges are tree edges, red edge is a cross edge and blue edge is a forward edge.

(a) (10 points) Prove that a DFS tree on an undirected graph does not have any cross edges.

(b) (10 points) Prove that a BFS tree does not have any forward edges.

Problem 5 (Store Locations)**.** There are n neighborhoods in North Carolina. You are given a map where neighborhoods are vertices and adjacent neighborhoods are connected by an edge. Suppose we would like to open some stores in North Carolina, so that everyone can either go to a store in his/her neighborhood, or a store in an adjacent neighborhood. However due to budget problems we can only open at most $n/2$ stores. Design an algorithm that finds at most $\lfloor n/2 \rfloor$ neighborhoods, such that if those neighborhoods have a store, everyone can find a store either in their neighborhood or in at least one of the adjacent neighborhoods.