CS41 Homework 5

This homework is due at 11:59pm, October 13. This is a 12 point assignment. Write your solution using \LaTeX. Submit this homework using github as a file called hw5.tex. For this homework, you will work with a partner. It’s ok to discuss approaches with others at a high level, but most of your discussions should just be with your lab partner. The only exception to this rule is work you’ve done with a lab partner while in lab. In this case, note who you’ve worked with and what parts were solved during lab. If there are questions about academic integrity, please visit the section on Academic Integrity on the course website.

0. Before final submission, make sure to fill out the README file.

1. **The cut property** (adapted from CLRS 23.2-1)
   In class we saw the cut property, which stated that for any nonempty subset $S \subseteq V$ of vertices, the edge $e = (u, v)$ of minimal weight such that $u \in S$ and $v \in V \setminus S$ is in every minimal spanning tree of the given graph.

   There may be more than one minimum spanning tree of a graph. The cut property is worded very carefully: the edge $e$ is in every minimum spanning tree.

   However, if edge weights are not distinct, then there might be two edges which are tied: both have the smallest weight.

   (a) Rewrite the cut property so that it covers the case where edge weights are not distinct.

   (b) Kruskal’s algorithm can return different spanning trees for the same graph $G$ depending on how ties are broken when the edges are ordered by weight. Show that for each minimum spanning tree $T$ of $G$, there is a way to sort the edges of $G$ in Kruskal’s algorithm so that the algorithm returns $T$.

2. **Changing edge costs** (Kleinberg and Tardos, 4.2)
   For each of the following two statements, decide whether it is true or false. If it is true, give a short explanation. If it is false, give a counterexample.

   (a) Suppose we are given an instance of the Minimum Spanning Tree (MST) problem on a graph $G = (V, E)$, with edge costs that are all positive and distinct. Let $T$ be a minimum spanning tree for this instance. Now, suppose we replace each cost $c_e$ by its square $c_e^2$, thereby creating a new instance of the problem with the same graph but different costs. TRUE or FALSE? $T$ must still be a minimum spanning tree for this new instance.

   (b) Suppose we are given an instance of the Shortest $s \leadsto t$ Path Problem on a directed graph $G = (V, E)$, with edge costs that are all positive and distinct. Let $P$ be a a minimum-cost $s \leadsto t$ path for this instance. Now, suppose we replace each edge cost $c_e$ by its square $c_e^2$, thereby creating a new instance of the problem with the same graph but different costs. TRUE or FALSE? $P$ must still be a minimum $s \leadsto t$ path for this new instance.

3. **Making change** (foreign edition)
   Consider the problem of making change for $n$ cents out of the fewest number of coins. Assume that $n$ and the coin values are positive integers (cents).
(a) Describe a greedy algorithm to solve the problem using the EU coin denominations of 50 cents, 20 cents, 10 cents, 5 cents, 2 cents, and 1 cent. Prove your algorithm is optimal.

(b) Suppose the country of Algorithmland uses denominations that are powers of $c$ for some integer $c$. This country uses $k + 1$ denominations of $c^0, c^1, \ldots, c^k$. Show that your greedy algorithm works in Algorithmland as well.

(c) Design a currency system of your choosing with at least three coin denominations such that a greedy algorithm does not yield a minimum number of coins for some amount of $m$ cents. Assume that one of your denominations has value one, so a solution exists for all values of $m$.

4. **Can this graph exist?** (Kleinberg and Tardos, 4.29)

   The degree of a node in an undirected graph is the number of edges it has. (For example, a node with only one neighbor has degree 1, and the root of a binary tree has degree 2.)

   Given a list of $n$ natural numbers $d_1, d_2, \ldots, d_n$, show how to efficiently decide whether there exists an undirected graph $G = (V, E)$ whose node degrees are precisely the numbers $d_1, d_2, \ldots, d_n$. (That is, if $V = \{v_1, v_2, \ldots, v_n\}$ then the degree of $v_i$ should be exactly $d_i$.) $G$ should not contain multiple edges between the same pair of nodes, or “loop” edges where both endpoints are the same node. Prove that your algorithm is correct.

5. **Summer camp triathlon** (Kleinberg and Tardos, 4.6)

   Your friend is working as a camp counselor, and is in charge of organizing activities for a set of junior-high-school-age campers. One of the plans is the following mini-triathlon exercise: each contestant must swim 20 laps of a pool, then bike 10 miles, then run 3 miles. The plan is to send the contestants out in a staggered fashion, via the following rule: the contestants must use the pool one at a time. (In other words, first one contestant swims the 20 laps, gets out, and starts biking. As soon as this person is out of the pool, a second contestant begins swimming the 20 laps; as soon as the second person is out and starts biking, a third contestant begins swimming, and so on.)

   Each contestant has a projected swimming time (the expected time it will take them to complete the 20 laps), a projected biking time (the expected time it will take them to complete the 10 miles of bicycling), and a projected running time (the expected time it will take them to complete the 3 miles of running). Your friend wants to decide on a schedule for the triathlon: an order in which to sequence the starts of the contestants. Let’s say that the completion time of a schedule is the earliest time at which all contestants will be finished with all three legs of the triathlon, assuming they each spend exactly their projected swimming, biking, and running times on the three parts. (Again, note that participants can bike and run simultaneously, but at most one person can be in the pool at any time.) What’s the best order for sending people out, if one wants the whole competition to be over as early as possible? More precisely, give an efficient algorithm that produces a schedule whose completion time is as small as possible.

6. **Extra credit.** One reason why the United States has the coin denominations it has is to make it easier to make change using a small number of coins. However, is the US set of coin denominations the optimal set of denominations?
Perhaps some other set of denominations is better. In this problem, design and analyze an algorithm to determine the optimal set of coin denominations. Given an integer \( N \), say the 
change efficiency of a set of denominations of coins as the average number of coins required to make change, where the average is taken over all \( 1 \leq m \leq N \).

For example, using the standard US coin denominations \{1, 5, 10, 25\}, the coin efficiency given \( N = 5 \) is 2.2, since it takes \( m \) pennies to make change for values \( 1 \leq m \leq 4 \), and one nickel to make change for value \( m = 5 \).

Design and analyze an efficient algorithm that takes as input positive integers \( N \) and \( k \) and returns a set of \( k \) denominations that minimizes the coin efficiency for \( N \).