CS 6363.003 Midterm 2

April 22, 2021

Please read the following instructions carefully before you begin.

- The exam has 4 problems, some of which have multiple parts. Please start the solution to each problem on a new page.
- The exam is meant to take around 1 hour and 15 minutes, but you have some extra time to prepare the solutions for submission to eLearning and work through any related technical issues. You must finish uploading your solutions within 2 hours of beginning the exam or by 5:59am CST on Friday, April 23, *whichever comes first*. You may upload as many versions of your solutions as you would like, but you can only submit them once. Your solutions will be submitted automatically 2 hours after you begin the exam to prevent you from uploading but forgetting to actually submit your solutions.
- If you cannot successfully upload your solutions to eLearning before the end of the 2 hour time limit, email a copy to Kyle as soon as you possibly can.
- It is highly recommended that you take the exam between 9am CST and ***3:30pm*** CST on Thursday, April 22nd. Kyle will be actively watching for questions via email during this time. He will also try to answer emailed questions at other times during the exam period, but he may be slower to respond, especially if he is asleep! Feel free to ask for clarification on any of the problems.
- Questions are not necessarily given in order of difficulty, so read through them all before you begin writing.
- This exam is closed book. No notes or calculators are permitted.
- If asked to describe an algorithm, you should state your algorithm clearly (preferably with pseudocode) and briefly explain its asymptotic running time in big-O notation in terms of the input size. Each individual problem will specify whether or not you must justify correctness.
- DON'T PANIC

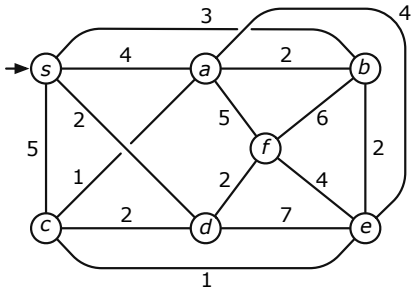
1. ***Clearly*** indicate the edges of the following spanning trees of the weighted graph pictured below by, say, drawing a new copy of the graph with vertex labels and edge weights intact. Some of these subproblems have more than one correct answer.
   Each part is worth 2.5 points out of 10.

   (a) A depth-first spanning tree rooted at s

   (b) A breadth-first spanning tree rooted at s

   (c) A shortest paths tree rooted at s

   (d) A minimum spanning tree



2. Suppose you are a shopkeeper living in a country with n different types of coins, with values $1 = c[1] < c[2] < \cdots < c[n]$. (In the U.S., for example, $n = 6$ and the values are 1, 5, 10, 25, 50, and 100.) Not wanting to burden their pockets, whenever you give a customer change, you always use the smallest possible number of coins.

   (a) **(3 out of 10)** In the United States, there is a simple greedy algorithm that always results in the smallest number of coins when giving a customer C units of change: give the largest coin i such that $c[i] \leq C$, and then recursively give $C - c[i]$ units of change with the same algorithm.

   However, your country's coin system may not be so nice. Show there is a set of coin values for which the greedy algorithm does *not* always give the smallest possible number of coins. Please state both the coin values and the amount of change C for which the greedy algorithm is not optimal. *[Hint: Kyle can think of a system with only three different coins. If your solution uses more than three coins, that's fine too.]*

   (b) **(7 out of 10)** Suppose your country's government decides to impose a currency system where the coin denominations are consecutive powers $b^0, b^1, b^2, \ldots, b^k$ for some integer $b \geq 2$. Prove the greedy algorithm described in part (a) does make optimal change in this currency system. *[Hint: Do an exchange argument. It may help to observe that the value of each consecutive coin is an integer multiple of the value for the coin before it.]*

3. (a) **(3 out of 10)** Let $G = (V, E)$ be a connected undirected graph with real edge weights $w : E \to \mathbb{R}$. Briefly describe and analyze a fast algorithm to compute a *maximum* weight spanning tree of G. You may assume edge weights are distinct. You do not need to justify correctness of your algorithm. *[Hint: Reduce to a similar problem we saw in class.]*

(b) **(7 out of 10)** Let $G = (V, E)$ be a connected undirected graph with *positive* edge weights $w : E \to \mathbb{R}_{>0}$. A ***feedback edge set*** of G is a subset F of its edges such that every cycle in G contains at least one edge in F. In other words, removing every edge in F turns the graph G into a forest. Briefly describe and analyze a fast algorithm to compute a minimum weight feedback edge set of G. You may assume edge weights are distinct, and you may assume your solution to part (a) is correct. You do not need to justify correctness of your algorithm.

4. Suppose we are given a directed acyclic graph $G = (V, E)$ with real edge lengths $\ell : E \to \mathbb{R}$ such that G has a unique source s and a unique sink t. In other words, s is the only vertex with no predecessor and t is the only vertex with no successor. We have also been given a non-negative integer k and told that some vertices of G have been marked *important*. For simplicity, we may assume neither s nor t have been marked important.

   Our goal is to design an algorithm to find the maximum length over all paths from s to t that contain at least k important vertices.

   (a) **(5 out of 10)** For any vertex $v \in V$ and any non-negative integer r, let $MaxLength(v, r)$ denote the maximum length over all paths from v to t that contain at least r important vertices. If no such path exists, let $MaxLength(v, r) := -\infty$.

   Give a recursive definition of $MaxLength(v, r)$. You do not need to justify correctness of your definition.

   (b) **(5 out of 10)** Describe and analyze an efficient dynamic programming algorithm to compute the maximum length over all paths from s to t that contain at least k important vertices. If no such path exists, your algorithm should return $-\infty$.

   For your algorithm, it suffices to describe

   - the set of subproblems (choices for v and r) it will solve
   - the evaluation order in which it solves them, and
   - the value it will return.

   You do not need to provide further explanation or justification beyond explaining these items. Your running time should be expressed in terms of V, E, and/or k.

   Unless you gave a recursive definition for which there is no efficient memoization method, you may assume your answer to part (a) is correct when designing your algorithm.