CPSC 365 / ECON 365: Algorithms	Yale University
Problem Set 3	
Out: February 8, 2022	Due: February 22, 2022 at 2:30 pm EST

Instructions

Write your solution to the following problems carefully. Submit the PDF of your solution via Gradescope. Please start your solution to every question on a new page. Make sure to assign the correct page in your document corresponding to each problem. We recommend writing your solution in Latex. Handwritten solutions are accepted if they are clearly legible. Write your name and SID in your answer to Problem 0; do not write them anywhere else in your solution.

Collaboration policy: You may collaborate and discuss these problems with other students. However, you must first make an honest effort to solve these problems by yourself. You may discuss hints and try to solve problems together. But you may not explicitly provide the solution to a problem to other students; nor receive the solution from them. You must write your solution independently, and make sure you understand your solution. You must list all your collaborators (anyone with whom you discussed any part of these problems). You may consult the course textbook and other references related to the class. However, you are forbidden from searching for these problems on the Internet. You must list any resources you consult (beyond the course textbook). You must also follow the Yale academic integrity policy.

0 Your Information

On the first page of your submission, include the following information (but do not include these anywhere else in your solution). Also certify that you have followed the collaboration policy. Your solution to the remainder of the problems should start on a new page.

(a) Your name.

(b) Your SID.

(c) A list your collaborators and any outside resources you consulted for this problem set. If none, write "None".

(d) Certify that you have followed the academic integrity and collaboration policy as written above.

(e) How many hours did you spend in this problem set?

1 Checking Connectivity

Let $G = (V, E)$ be an undirected graph with the additional property that every edge has a color, either red or blue. (There is no assumption on how the colors are organized; the graph may have all red edges, or all blue edges, or a mixture.) Let u and v be distinct vertices in G.

(a) Design an algorithm that decides whether or not there exists a path from u to v such that the path contains only red edges.

Justify the correctness of your algorithm and analyze the running time.

(*Hint:* You may use or modify any algorithm we have seen in class. You should describe your algorithm concisely and precisely, either in pseudocode or in words. Make sure your solution includes proof of correctness and running time analysis.)

(b) Design an efficient algorithm that decides whether or not there exists a path from u to v such that within the path, all red edges appear before all blue edges.

Justify the correctness of your algorithm and analyze the running time.

2 Road Trip

Suppose you are given a set of cities, along with the pattern of highways between them, in the form of an undirected graph $G = (V, E)$. Each stretch of highway $e \in E$ connects two of the cities, and you know its length in miles, $\ell_e > 0$. You want to get from city s to city t. There's one problem: your car can only hold enough gas to cover L miles. There are gas stations in each city, but not between cities. Therefore, you can only take a route if every one of its edges has length $\ell_e \leq L$.

(a) Given the limitation on your car's fuel tank capacity, show how to determine in linear time whether there is a feasible route from s to t.

(b) You are now planning to buy a new car, and you want to know the minimum fuel tank capacity that is needed to travel from s to t. Design an algorithm to do this. Justify the correctness of your algorithm and analyze the running time.

3 Counting Shortest Paths

Let $G = (V, E)$ be a weighted undirected graph with edge lengths $\ell_e > 0$. Let $s, t \in V$ be distinct vertices. Design an algorithm to *count* the total number of shortest paths from s to t.

Justify your answer and analyze the running time.

4 Spanning Tree with Leaves

Sometimes we want to construct light spanning trees under some constraints. Here is an example.

Design an algorithm to solve the following problem. Justify the correctness of your algorithm, and analyze the running time.

Input: An undirected graph $G = (V, E)$; edge weights $w_e > 0$; and a subset of vertices $U \subset V$.

Output: The lightest spanning tree in which the vertices of U are leaves.
(There can be other leaves in the tree.)

(*Note*: Recall a *leaf* is a vertex with degree 1. Lightest means with smallest total edge weights. The resulting tree is a spanning tree, but not necessarily a minimum spanning tree.)

5 Perfect Matching in a Tree

Recall a *perfect matching* in a graph is a collection of edges that touches every vertex exactly once.

Design an algorithm that takes as input a tree and determines whether it has a perfect matching.

Input: An undirected graph $G = (V, E)$ which is a tree, i.e. connected and has no cycles.

Output: Yes if it has a perfect matching; No otherwise.

Justify the correctness of your algorithm and analyze the running time.