CS 344: Design and Analysis of Computer Algorithms **Rutgers: Fall 2019**

Homework #4

November 14, 2019

Name: FIRST LAST *Extension: Yes/No*

Problem 1. You are given a tree T on n nodes and each node a of T is given some points, denoted by $Points[a]$. The goal in this problem is to choose a subset of the nodes that maximize the total number of points achieved while ensuring that no two chosen nodes are neighbors to each other. Design an $O(n)$ time algorithm for this problem that outputs the *value* of maximum number of points we can achieve.

You may assume that the input is specified in the following way: (1) we have a pointer r to the root of the tree (we can root the tree arbitrarily); and (2) for each node a of the tree, we have a linked-list to the child-nodes of this tree in $Child[a]$ ($Child[a] = NULL$ for every leaf-node a). **(25 points)**

Solution. Solution to Problem 1 goes here.

Problem 2. You are in charge of providing access to a library for every citizen of a far far away land. A long time ago, this land had a collection of (bidirectional) roads between different pairs of cities. However, over time these roads all got destroyed one way or another. The ruler of this land now asks for your help to determine which of these roads to repair and where to build the libraries: the objective is that after this, every city should either contain a library or the people in this city should be able to travel to another city that contains a library using these repaired roads. You are also told that repairing a road has a cost of R, and building a library has a cost L.

Design an algorithm that given n cities, the specification of m (damaged) roads that used to connect two of these cities together, and the parameters R and L, outputs the list of roads that needs to be repaired plus the name of the cities in which a library should be built, while minimizing the total cost. **(25 points)**

Solution. Solution to Problem 2 goes here.

Problem 3. Suppose you are given a directed acyclic graph $G(V, E)$ with a unique source s and a unique sink t. Any vertex $v \notin \{s, t\}$ is called an (s, t)-*cut vertex* in G if *every* path from s to t passes through v; in other words, removing v from the graph makes t unreachable from s.

(a) Prove that a vertex v is a (s, t)-cut vertex if and only if in any topological sorting of vertices of G, no edge connects a vertex with order less than v to a vertex with order higher than v. **(10 points)**

Solution.

Objective: To prove that no edge connects a vertex with order less than v to a vertex with order higher than v, where vertex v is a (s, t)-cut vertex

Proof of Correctness:

- We will use contradiction to prove this statement.
- First of all, to understand the objective, it basically says that for given order n of the vertex v, all vertices less than order n will have no edges to vertices with order greater than n. **Hypothesis:** there exists edge that connects a vertex with order less than v to a vertex with order higher than v

---

(b) Use part (a) to design an $O(n+m)$ time algorithm for finding *all* (s,t)-cut vertices in G. **(15 points)**

Solution. Solution to Problem 3 - part (b) goes here.

Problem 4. Use graph reductions to design an algorithm for each of the following two problems.

(a) Given a vertex s in a directed graph $G(V,E)$, find the length of the shortest *cycle* that starts from the vertex s. **(12.5 points)**

Solution. Solution to Problem 4 - part (a) goes here.

(b) Given a connected undirected graph $G(V,E)$ with distinct weights w_e on each edge $e \in E$, output a *maximum* spanning tree of G. A maximum spanning tree of G is a spanning tree of G with maximum total weight of edges. **(12.5 points)**

Solution. Solution to Problem 4 - part (b) goes here.

Challenge Yourself. A *cut-vertex* in an undirected connected graph $G(V,E)$ is any vertex v such that removing v (and all its edges) from G makes the graph disconnected. Design an $O(n+m)$ time algorithm for finding all cut-vertices of a given graph with n vertices and m edges.