due Sunday, 14 April 2019, at 11:00 PM

Note: If you choose to work with a partner on this homework, you need to register by 5:00 PM on April 7. Instructions for doing so will be given later, but you will need to follow them exactly. You are allowed to work with a partner in a different section.

Submissions for the written part of the homework, including your analysis of the programs should be submitted to GradeScope (these will be set up as two distinct assignments – `Homework 4` and `Program 4 Analysis`). The source files for your programs should be submitted to `Assignment 4 Programs` in Moodle. Other instructions are the same as with previous assignments.

[Total points for this part: 29]

1. [5 points] *Purpose: Understanding the structure of disjoint set union forests.*

   Do Exercise 21.3-4 on page 572.

   You are asked to revise the implementation of disjoint sets as forests so that it supports a PRINT-SET(x) function, which prints all the elements in the set containing x. Runtime must be O(1) per element printed.

   The naive solution – creating pointers that are reversals of the parent pointers – will not work for two reasons: (i) each node would have to keep track of all of its children, violating the condition that this be done with a single attribute for each node; and (ii) some elements would end up being printed more than once (the exercise does not explicitly forbid this, but we will require you to come up with a solution where each element is printed *exactly once*).

2. [12 points, 4 points each part] *Purpose: Understanding minimum spanning trees and edge contraction.*

   Do Problem 23-3 on pages 640-641 (bottleneck spanning tree).

   You are asked to come up with an algorithm for a variant of the minimum spanning tree problem. Instead of minimizing the total weight of the edges in a spanning tree you are asked to minimize the *maximum* weight of an edge. The three parts ask you to (a) prove that any minimum spanning tree is also a bottleneck spanning tree; (b) come up with a linear time algorithm that determines (yes or no) whether there is a spanning tree whose maximum weight edge has weight b; and (c) give a linear time algorithm for the bottleneck spanning tree problem using (b) and an algorithm described in another problem (no need to give details of that one).

3. [12 points] *Purpose: Understanding Dijkstra's shortest paths algorithm.*

   Suppose you have a weighted, undirected graph G with positive edge weights and a start vertex s. Come up with an algorithm that runs as fast as Dijkstra's and assigns a label usp[u] to every vertex u in G, so that usp[u] is `true` if and only if there is a *unique* shortest path from s to u. By definition usp[s] is `true`. Be sure to prove both the correctness and time bound of your algorithm.

   To get full credit, you need to (a) explain in English how your algorithm works; (b) give pseudocode (this may involve a modification of Dijkstra's algorithm); (c) illustrate how your algorithm works on a small (at most five vertices) but nontrivial (has cases where labels change) example; and (d) give a correctness proof (similar to that for Dijkstra's algorithm). Proof of the time bound should be simple if you did everything else correctly, but you need at least a sentence or two to address it.