

# CS381 Homework 6

Out: Nov. 21, Tuesday

Due: Dec. 1, Friday before 4 pm.

Submit to Paula Perkins (or the receptionist) at LWSN 1123.

Submissions will not be accepted afterwards.

## Instructions:

**Read the course policy (including academic dishonesty) in the course webpage at <http://www.cs.purdue.edu/homes/gopal/cs381>.**

Text refers to the Introduction to Algorithms (second edition) book.

**Justify your answers. Show appropriate work.**

**Reading:** Chapters 22 and 23 of Text.

## Problem 1

Let  $G = (V, E)$  be a directed graph. We call  $G$  to be *mildly-connected* if for all pairs of vertices  $u, v \in V$ , there is a path either from  $u$  to  $v$  **or** from  $v$  to  $u$  (or possibly both). Give an  $O(|V| + |E|)$  (i.e., linear) time algorithm to determine if a given directed graph is mildly-connected.

(Hint: First find the component graph  $G' = (V', E')$  of  $G$  (i.e., the vertices of  $G'$  correspond to the strongly connected components and there is a directed edge between two such vertices  $u', v' \in V'$  if there a directed edge between some vertex of the component (corresponding to)  $u'$  and some vertex of component (corresponding to)  $v'$ ). (You should show how to construct the component graph in linear time.) Note that the component graph is a DAG (directed acyclic graph) (why?). What property should this DAG have for  $G$  to be mildly connected? Topological sort will prove useful now.)

## Problem 2

We are given a *connected undirected* graph  $G = (V, E)$  with positive edge weights. Assume that all the edge weights are *distinct*. The operation of

*contracting* an edge  $e = (u, v)$  in  $G$  involves the following operations: (1) remove the vertices  $u$  and  $v$  and introduce a new vertex  $x$ ; (2) for each edge  $e'$  having an endpoint in either  $u$  or  $v$  do: replace the endpoint  $u$  or  $v$  by  $x$ , i.e., if  $e' = (w, u)$  (or  $e' = (w, v)$ ) it is replaced by the edge  $e' = (w, x)$ . (Note that we can use the same labels on the edges, as they were before contraction, for easy identification.)

Consider the following algorithm for finding a minimum spanning tree:

1. **repeat** until  $G$  has one vertex
2.       **for** each vertex  $v \in G$  do
3.             Select the minimum-weight edge incident on  $v$ .
4.             Contract all the selected edges chosen in Step 3.
5.             Eliminate self-loops.
6.             Eliminate all but the lowest-weight edge among each set of multiple edges (i.e., edges that go across the same pair of vertices).
7. **endrepeat**
8. Output all the edges selected in Step 3.

- a) Run the above algorithm on the graph of figure 23.1 of Text (page 562). Show the main steps.
- b) Show that the minimum spanning tree of  $G$  is unique if all the edge weights are distinct. (Hint: One way to show this is proof by contradiction: assume two distinct minimum spanning trees and get a contradiction.)
- c) Show that the above algorithm correctly outputs the edges of the minimum spanning tree. (Hint: Show that the algorithm adds edges to the minimum spanning tree according to the cut rule.)
- d) Give a tight (up to a constant factor) upper bound on the number of iterations (one iteration is execution of Steps 2-6) of the algorithm.
- e) Show how to implement one iteration of the algorithm in  $O(|V| + |E|)$  time. (Hint: Use depth-first search.) Using your answer to part (d), what is the (overall) running time of the algorithm?
- f) Suppose the edge weights are not distinct. Will the algorithm still work? If your answer is "no", then show how to fix it.