CS 457, Data Structures and Algorithms I
Sixth Problem Set

December 2, 2019

Due on December 6 at 11:59 P.M. Collaboration is not allowed. Contact Daniel and me for questions.

For full credit, for any algorithms that you propose, also provide a (high level) justification regarding their worst-case running time bounds, and a brief explanation regarding their correctness. Throughout the problem set n denotes the number of vertices of a graph and m denotes the number of edges.

1. (20 pts) Your textbook defines the *connected components* of undirected graphs and the *strongly connected components* of directed graphs (see pages 1170-1171). Adding edges to a graph increases the connectivity and may lead to a decrease in the number of such components. i) If you add an edge to an undirected graph G, what is the largest possible drop in the number of connected components of G that this can possibly lead to? Similarly, ii) if G is a directed graph and you add a directed edge to it, what is the largest possible drop in the number of strongly connected components of G that this can lead to? Provide both upper and lower bound arguments to support your claims. For instance, if you claim that the largest possible drop is $n/4$, then provide an example where the drop is $n/4$ and an argument why it can be no more than $n/4$ in general.

2. (25 pts) You are given a weighted graph $G = (V, E)$ as well as a minimum spanning tree T of G. Then, a new edge e is added to G and you want to compute a minimum spanning tree for the new graph $G' = (V, E \cup \{e\})$. Provide an algorithm NEW-MST(G, T, e) that computes a minimum spanning tree T' for G' in time $O(n)$. Clearly, T' could be computed from scratch using Prim's or Kruskal's algorithm, or one could use algorithms covered in recitation, but all of these solutions would require time $\omega(n)$ which is not good enough.

3. (25 pts) You are given a directed graph $G = (V, E)$ along with a function $w : E \to \mathbb{R}_+$ that assigns positive weights to all the edges. This graph may contain directed cycles and your goal is to check if such cycles exist in G and, if they do, to return one with the smallest possible weight. The running time of your algorithm should be $O(n^3)$ or $O(n^2 m)$ (whichever one you prefer). You can use any algorithm from Chapter 24 as a subroutine.