CS5050 Advanced Algorithms

Fall Semester, 2017

Assignment 7: Graph Algorithms II

Due Date: 8:30 a.m., Wednesday, Dec. 6, 2017 (**at the beginning of CS5050 class**)

Note: In this assignment, we assume all input graphs are represented by adjacency lists.

1. In class we studied algorithms for computing shortest paths. Sometimes people are interesting in finding not only an arbitrary shortest path but also a shortest path with certain properties.

   Let's consider again the flight example given in Question 4 of the last assignment. Assume that saving money is very important to you (you need to pay tuition, you have kids to take care of, etc.), and so you want to find a shortest (i.e., cheapest) path regardless of how many edges it has. Even in this case, there is still something you can do to "optimize" your life. Indeed, it may be possible that there are multiple shortest paths from s to t (i.e., theses paths have different edges, but they have the same length). In this case, instead of returning an arbitrary shortest path, it would certainly be better to return a shortest path with as few edges as possible. This is the problem we are considering in this exercise. But now we are considering a general graph not necessarily a DAG. The problem is formally defined as follows.

   Given a directed graph $G = (V, E)$, let $n = |V|$ and $m = |E|$. Every edge (u, v) of G has a nonnegative weight $w(u, v) \geq 0$. Let s and t be two vertices in G. The *length* of a path in G is the sum of the weights of all edges in the path. A *shortest path* from s to t is a path with the minimum length among all paths in G from s to t. In class, we studied Dijkstra's algorithm for computing such a shortest path.

   It is possible that there are multiple different shortest paths from s to t in G (i.e., these paths may have different edges but their lengths are the same). In some applications, we may want to find a shortest path from s to t that has edges as few as possible. A path π in G from s to t is said to be **optimal** if (1) π is a shortest path from s to t, and (2) among all shortest paths from s to t, π has the minimum number of edges. Refer to Figure 1 for an example.

   Modify Dijkstra's algorithm to compute an optimal path from s to t. Your algorithm should be able to output the actual optimal path, but for simplicity, you only need to maintain the correct predecessor information $v.pre$ for each vertex v of the graph. Your algorithm should have the same time complexity as Dijkstra's algorithm. **(20 points)**

2. Let $G = (V, E)$ be an undirected connected graph, and each edge (u, v) has a positive weight $w(u, v) > 0$. Let s and t be two vertices of G. Let $\pi(s, t)$ denote a shortest path from s to t in G. Let T be a minimum spanning tree of G. Please answer the following questions and explain why you obtain your answers.

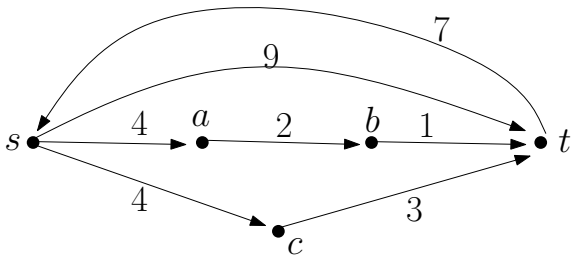

Figure 1: There are two shortest paths from s to t: $(s, a, b, t), (s, c, t)$. But only the second path is an optimal path because it has two edges while the first path has three edges.

(a) Suppose we increase the weight of every edge of G by a positive value $\delta > 0$. Then, is $\pi(s, t)$ still a shortest path from s to t? **(10 points)**

(b) Suppose we increase the weight of every edge of G by a positive value $\delta > 0$. Then, is T still a minimum spanning tree of G? **(10 points)**

Note: For each of the two questions, your answer should be either "Yes" or "Not necessary". Again, please explain why you obtain your answers.

3. Let $G = (V, E)$ be an undirected connected graph of n vertices and m edges. Suppose each edge of G has a color of either *blue* or *red*. Design an algorithm to find a spanning tree T of G such that T has as few red edges as possible. Your algorithm should run in $O((n + m) \log n)$ time. **(20 points)**

Total Points: 60