Homework 3

Winston (Hanting) Zhang

Friday, September 23, by 23:00

General Instructions. The following assignment is meant to be challenging, and we anticipate that it will take most of you at least 10–15 hours to complete, so please allow yourself plenty of time to work on it. (This assignment is probably somewhere between Homework 1 and Homework 2 in difficulty.) We highly recommend reading the entire assignment right away — you never know when inspiration will strike. Please provide a formal mathematical proof for all your claims, and present runtime guarantees for your algorithms using asymptotic (big-$O/\Omega/\Theta$) notation, unless stated otherwise. You may assume that all basic arithmetic operations (multiplication, subtraction, division, comparison, etc.) take constant time.

Collaboration. Please carefully check the collaboration policy on the course website. When in doubt, ask an instructor.

Consulting outside sources. Please carefully check the policy regarding outside sources on the course website. Again, when in doubt, ask an instructor.

Submission. Homework submission will be through the Gradescope system. Instructions and links have been provided through the course website and Piazza. The only accepted format is PDF. Starting with this homework, only typed solutions will be accepted, and we highly recommend producing your solutions using LaTeX (the text markup language we are also using for this assignment).

Recommended practice problems (do not hand in): KT Problems 4.1, 4.2, 4.19, 4.20, 4.22, 4.26 (for extra challenge).

Problem 1. [8+7=15]

Consider an undirected connected graph $G = (V, E)$ with edge costs $c_e > 0$ for $e \in E$ which are all distinct.

1. Let $E' \subseteq E$ be defined as the following set of edges: for each node v, E' contains the cheapest of all edges incident on v, i.e., the cheapest edge that has v as one of its endpoints. Is the graph (V, E') connected? Is it acyclic? For both questions, provide a proof or a counter-example with explanations.

2. Consider the following outline for an algorithm, which starts with an empty set T of edges: Let E' contain the cheapest edge out of each connected component of (V, T). Add E' to T, and repeat until (V, T) is connected. Show that this algorithm outputs a minimum spanning tree of G, and can be implemented in time $O(m \log n)$.

Hints:

- Each iteration of this algorithm can be viewed as applying the operation from part (a) on a "contracted graph", where each connected component of (V, T) corresponds to a node.
- If you want, you are welcome to use the efficient implementation of the Union-Find data structure which supports Find and Union in logarithmic time, as described in Chapter 4.6 of the KT book and briefly outlined in class. However, this implementation is not necessary, and focusing on it may mislead you from the basic idea.
- We recommend thinking about how many iterations it will take until (V, T) is connected.

Problem 2. [15]

We frequently motivate the shortest path problem, and Dijkstra's Algorithm for solving it, by considering transportation networks, such as driving from one place to another. Other transportation networks are rail or flight networks. Those are a little different from road networks, in that the edges (e.g., trains) are only available at certain times, rather than all the time with a given length.

Specifically, you are given a directed graph $G = (V, E)$ in which each directed edge $e = (u, v, t_u, t_v)$ corresponds to a train connection that goes from u to v, leaves at time t_u, and arrives at time t_v. Of course, there can now be multiple edges from u to v. You know that for each such edge, $t_v > t_u$, since the train cannot arrive before it leaves.[1] You know nothing else about the arrival and departure times; in particular, there could be two trains from u to v with $t'_u > t_u$ and $t'_v < t_v$. For example, these could be an express train and a scenic local train, where the express train leaves later and arrives earlier. You are also given a start node s, start time T, and destination node d. Your goal is to compute the earliest that you can arrive at d when starting at s at time T. You can assume that changing trains takes 0 time, so if you arrive at node u at time t, you can board a train that leaves u at time t.

Give and analyze (i.e., prove correct and analyze the running time) an algorithm that solves this problem in time $O(m \log n)$.

Hint: Obviously, this question is closely related to Dijkstra's Algorithm, which we did not cover in class. We highly recommend that you review the algorithm and — more importantly — its analysis in Section 4.4 of the textbook before attempting this question.

Problem 3. [0]

Chocolate Problem: 2 chocolate bars

Reminder: If you solve a chocolate problem (which you can do in groups of size up to 3), please e-mail David with the solution — do not submit it on Gradescope. Also, feel free to list preferences or dietary restrictions for/against particular types of chocolate.

Exercise 4.31 in the textbook. Notice that Part (b) is really the interesting thing here — Part (a) is basically a slightly harder regular problem.

[1] We are talking about trains here, not DeLoreans.