6331 - Algorithms, Autumn 2016, CSE, OSU
Homework 9
Instructor: Anastasios Sidiropoulos

Problem 1. Let G be an undirected graph, with weights on the edges.

(a) Let $s \in V(G)$, and let T be a shortest-paths tree with root s. Is it true that T must also be a minimum spanning tree? Give a proof that justifies your answer.

(b) Let $s \in V(G)$, and let T be a minimum spanning tree. Is it true that T must also be a shortest-paths tree with root the given vertex s? Give a proof that justifies your answer.

Problem 2. Let G be an edge-weighted graph. Let T be a minimum spanning tree T of G. Suppose that we decrease the weight of some edge $e = \{u, v\}$ not in T. Let G' be the resulting edge-weighted graph. Let P be the unique path between u and v in T. Let C be the cycle formed by the union of P with e. Let e' be an edge in C of maximum weight. Let $T' = (T \cup \{e\}) \setminus \{e'\}$. Prove, or disprove that T' is a minimum spanning tree of G'.

Problem 3: Robot navigation. Consider a robot that moves inside a room, represented by a 2-dimensional $n \times n$ square grid. Every location of the grid is indexed by a pair of integers (i, j), where $i, j \in \{1, \ldots, n\}$. Location (i, j) may either be empty, or it might contain an obstacle. The robot is allowed to move only in empty locations. Apart from its position, the state of the robot also consists of its orientation, which can be either North, South, West, or East. At every step, the robot can either move by one position forward in its current orientation, or it can change its orientation by 90°. For example, if the robot has orientation North, and is located in position (i, j), then in one step it can do one of the following: Move to position $(i, j + 1)$, and maintain orientation North, or stay in position (i, j), and change its orientation to either East, or West.

(a) Suppose that initially the robot is positioned at $(1, 1)$ and has orientation North. The goal is to reach position (n, n) in any orientation. Design an algorithm that computes a routing for the robot, with a minimum number of steps. Your algorithm should run in time polynomial in n.

(b) Suppose that there are k robots in the room. At each step every robot moves simultaneously. Two robots cannot occupy the same location and they cannot pass through each other. You are given the starting position and orientation of each robot and its destination. Design an algorithm that computes the shortest sequence of steps that move all the robots to their respective destinations. The running time of your algorithm should be $n^{O(k)}$.

Hint: Express the above problems as shortest-path computations in some graph.

Problem 4: Finding a cheap flight. Let $G = (V, E)$ be a directed graph, where V is a set of cities, and E represents all possible flights between the cities in V.

For every edge $\{u, v\} \in E$, you are given the duration of a direct flight from u to v, denoted by $d(u, v)$, which is an integer. For example, if you are at city u at time t, and you take a direct flight to v, departing at time $t' \geq t$, then you arrive at v at time $t' + d(u, v)$.

For every $\{u,v\} \in E$, you are given a timetable of all available direct flights from u to v, for some interval $\{0,\dots,T\}$, where $T > 0$ is an integer. That is, for any $\{u,v\} \in E$, you are given a list of pairs of integers $((t_{u,v,1}, c_{u,v,1}),\dots,(t_{u,v,k}, c_{u,v,k}))$, where the pair $(t_{u,v,i}, c_{u,v,i})$ denotes the fact that there is a direct flight from u to v that departs at time $t_{u,v,i}$, and costs $c_{u,v,i}$ dollars.

Design an algorithm that given a pair of cities $u, v \in V$, computes the cheapest possible route that starts at u at time 0, and ends at v at time at most T. The running time of your algorithm should be polynomial in $|V|$, and T.

Hint: Express the above problem as shortest-path computations in some graph.

Problem 5: How to become rich. Let $G = (V,E)$ be a directed weighted graph, that represents a set of possible financial transactions. More precisely, the set of vertices V corresponds to a set of currencies, and for any two currencies $u, v \in V$, there is a directed edge $(u,v) \in E$ with weight $\alpha \geq 0$ if we can sell x units of currency u for $\alpha \cdot x$ units of currency v.

(a) Give a polynomial-time algorithm which decides whether there exists a currency $s \in V$, such that starting with 1 unit of currency s, after performing some sequence of transactions, we can obtain strictly more than 1 unit of currency s. Hint: Use the Bellman-Ford algorithm.

(b) Modify your algorithm so that it outputs the above sequence of transactions.