CS 170 HW 5 (Optional)

Due 2019-09-02, at 10:00 pm
You may submit your solutions if you wish them to be graded, but they will be worth no points

1 Study Group

List the names and SIDs of the members in your study group.

2 Arbitrage

Shortest-path algorithms can also be applied to currency trading. Suppose we have \( n \) currencies \( C = \{c_1, c_2, \ldots, c_n\} \): e.g., dollars, Euros, bitcoins, dogecoins, etc. For any pair \( i, j \) of currencies, there is an exchange rate \( r_{i,j} \): you can buy \( r_{i,j} \) units of currency \( c_j \) at the price of one unit of currency \( c_i \). Assume that \( r_{i,i} = 1 \) and \( r_{i,j} \geq 0 \) for all \( i, j \).

The Foreign Exchange Market Organization (FEMO) has hired Oski, a CS170 alumnus, to make sure that it is not possible to generate a profit through a cycle of exchanges; that is, for any currency \( i \in C \), it is not possible to start with one unit of currency \( i \), perform a series of exchanges, and end with more than one unit of currency \( i \). (That is called arbitrage.)

More precisely, arbitrage is possible when there is a sequence of currencies \( c_{i_1}, \ldots, c_{i_k} \) such that \( r_{i_1,i_2} \cdot r_{i_2,i_3} \cdots r_{i_{k-1},i_k} \cdot r_{i_k,i_1} > 1 \). This means that by starting with one unit of currency \( c_{i_1} \) and then successively converting it to currencies \( c_{i_2}, c_{i_3}, \ldots, c_{i_k} \) and finally back to \( c_{i_1} \), you would end up with more than one unit of currency \( c_{i_1} \). Such anomalies last only a fraction of a minute on the currency exchange, but they provide an opportunity for profit.

We say that a set of exchange rates is arbitrage-free when there is no such sequence, i.e. it is not possible to profit by a series of exchanges.

(a) Give an efficient algorithm for the following problem: given a set of exchange rates \( r_{i,j} \) which is arbitrage-free, and two specific currencies \( s, t \), find the most advantageous sequence of currency exchanges for converting currency \( s \) into currency \( t \).

Hint: represent the currencies and rates by a graph whose edge weights are real numbers.

(b) Oski is fed up of manually checking exchange rates, and has asked you for help to write a computer program to do his job for him. Give an efficient algorithm for detecting the possibility of arbitrage. You may use the same graph representation as for part (a).

3 Picking a Favorite MST

Consider an undirected, weighted graph for which multiple MSTs are possible (we know this means the edge weights cannot be unique). You have a favorite MST, \( F \). Are you guaranteed that \( F \) is a possible output of Kruskal’s algorithm on this graph? How about Prim’s? In other words, is it always possible to “force” the MST algorithms to output \( F \) without changing the weights of the given graph? Justify your answer.
4 Finding MSTs by Deleting Edges

Consider the following algorithm to find the minimum spanning tree of an undirected, weighted graph \( G(V, E) \). For simplicity, you may assume that no two edges in \( G \) have the same weight.

\[
\text{procedure } \text{FindMST}(G(V, E)) \\
E' \leftarrow E \\
\text{for Each edge } e \text{ in } E \text{ in decreasing weight order do} \\
\text{if } G(V, E' - e) \text{ is connected then} \\
\quad E' \leftarrow E' - e \\
\text{return } E'
\]

Show that this algorithm outputs a minimum spanning tree of \( G \).

5 Unique Shortest Path

Shortest paths are not always unique: sometimes there are two or more different paths with the minimum possible length. Show how to solve the following problem in \( O(|V| + |E| \log |V|) \) time.

Input: An undirected graph \( G = (V, E) \); edge lengths \( l_e > 0 \); starting vertex \( s \in V \).

Output: A Boolean array \( \text{usp}[] \): for each node \( u \), the entry \( \text{usp}[u] \) should be \text{true} if and only if there is a unique shortest path from \( s \) to \( u \). (Note: \( \text{usp}[s] = \text{true} \).)

[Provide 3 part solution.]

6 Service scheduling

A server has \( n \) customers waiting to be served. Customer \( i \) requires \( t_i \) minutes to be served. If, for example, the customers were served in the order \( t_1, t_2, t_3, \ldots, t_n \), then the \( i \)-th customer would wait for \( t_1 + t_2 + \cdots + t_i \) minutes.

We want to minimize the total waiting time

\[
T = \sum_{i=1}^{n} (\text{time spent waiting by customer } i).
\]

Given the list of the \( t_i \)'s, give an efficient algorithm for computing the optimal order in which to serve the customers.