CSOR W4231 (sec. 001, 002, H02): Analysis of Algorithms Nov 4

Homework 5: Due on Nov 19 by 12:01am

Instructors: *Alex Andoni, Cliff Stein*

Instructions. Please follow the homework policy and submission instructions in the Course Information handout. A few highlights to remember:

- follow the collaboration and academic honesty policies;
- write your name, uni, and collaborators on the top page;
- submission is via GradeScope (you are encouraged to use LaTeX, using the provided template);
- if you don't know how to solve a particular part of a problem, just write "*empty*", for which you will get 20% of the points of that part (in contrast, note that non-sensical text may get 0%).

Note that, for each bullet item of a problem, you can use the previous bullets as "black box", even if you didn't manage to solve them. Similarly, you can use anything from the lectures as black-box.

Problem 1

Consider an undirected graph G with n vertices (named 1 through n) and e edges. The graph is given in adjacency list format. Your task is to produce n new lists, where the i-th list contains the vertices that are reachable from vertex i. Argue that your algorithm has an optimal running time (in the sense of Θ notation), as a function of n and e.

Problem 2

An undirected graph $G = (N, E)$ is called *bipartite* if its set N of nodes can be partitioned into two subsets N_1, N_2 ($N_1 \cap N_2 = \emptyset$ and $N_1 \cup N_2 = N$) so that every edge connects a node of N_1 with a node of N_2.

(a) Prove that if a graph contains a cycle of odd length then it is not bipartite.

(b) Give a $O(n + e)$-time algorithm that determines whether a given graph is bipartite, where $n = |N|$ is the number of nodes and $e = |E|$ is the number of edges; the graph is given by its adjacency list representation. If the graph is bipartite, then the algorithm should compute a bipartition of the nodes according to the above definition (i.e., compute the sets N_1, N_2). If the graph is not bipartite then the algorithm should output a cycle of odd length.

Problem 3

We are given a set V of n variables $\{x_1, x_2, \ldots, x_n\}$ and a set C of m weak and strict inequalities between the variables, i.e., inequalities of the form $x_i \leq x_j$ or $x_i < x_j$. The set C of inequalities is called consistent over the positive integers $Z^+ = \{1, 2, 3, \ldots\}$ iff there is an assignment of positive integer values to the variables that satisfies all the inequalities. For example, the set $\{x_1 \leq x_3, x_2 < x_1\}$ is consistent, whereas $\{x_1 \leq x_3, x_2 < x_1, x_3 < x_2\}$ is not consistent.

(a) Give an efficient algorithm to determine whether the set C of inequalities is consistent over the positive integers. State precisely the asymptotic running time of your algorithm in terms of n and m.

(b) If the set of inequalities has a solution, then it has a unique minimum solution, i.e., a solution in which every variable has the minimum value among all possible solutions. Give an efficient algorithm to compute the minimum solution.

Both parts have $O(n + m)$ solutions. Hint: Construct a suitable graph and use appropriate algorithms.

Problem 4

(a) Prove that if the edges weights in a graph are distinct, then the graph has a unique minimum spanning tree.

(b) Now assume that the edge weights are not necessarily distinct. Suppose that you are given a graph G and a partition of nodes $R, N \setminus R$. Suppose the edge $(u, v) \in (R, N \setminus R)$ is an edge crossing the partition. Prove that if the edge (u, v) is one of the minimum-weight edges across the partition, then there exists a minimum spanning tree that contains edge (u, v).

(c) Consider a new divide-and-conquer algorithm for computing minimum spanning trees, which goes as follows. Given a graph $G = (V, E)$, partition the set V of vertices into two sets V_1 and V_2 such that $|V_1|$ and $|V_2|$ differ by at most 1. Let E_1 be the set of edges that are incident only on vertices in V_1, and let E_2 be the set of edges that are incident only on vertices in V_2. Recursively solve a minimum-spanning-tree problem on each of the two subgraphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. Finally, select the minimum-weight edge in E that crosses the cut (V_1, V_2), and use this edge to unite the resulting two minimum spanning trees into a single spanning tree.

Either argue that the algorithm correctly computes a minimum spanning tree of G, or provide an example for which the algorithm fails.

Problem 5

(a) Suppose that you had a graph $G = (V, E, w)$ with a source s and one negative cost edge (u, v) Give an algorithm that computes the shortest path from source s to v.

(b) Arbitrage is the use of discrepancies in currency exchange rates to transform one unit of a currency into more than one unit of the same currency. For example, suppose that 1 U.S. dollar buys 64 Indian rupees, 1 Indian rupee buys 1.8 Japanese yen, and 1 Japanese yen buys 0.009 U.S. dollars. Then, by converting currencies, a trader can start with 1 U.S. dollar and buy $64 \times 1.8 \times 0.009 = 1.0368$ U.S. dollars, thus turning a profit of 3.68 percent.

Suppose that you are given n currencies $c_1, c_2, \ldots, c_n$ and an $n \times n$ table R of exchange rates, such that one unit of currency c_i buys $T[i, j]$ units of currency c_j.

Give an efficient algorithm to determine whether or not there exists a sequence of currencies $< c_{i_1}, c_{i_2}, \ldots, c_{i_k} >$ such that

$$T[i_1, i_2] \cdot T[i_2, i_3] \cdots T[i_{k-1}, i_k] \cdot T[i_k, i_1] > 1 \ .$$

Analyze the running time of your algorithm. Your algorithm should return a sequence if one exists. What is the running time of your algorithm?