CS 341, Fall 2025 T. Brown, É. Schost

ASSIGNMENT 4

DUE: Mon Nov 17, 11:59 PM. DO NOT COPY. ACKNOWLEDGE YOUR SOURCES.

Please read `http://www.student.cs.uwaterloo.ca/~cs341` for general instructions and policies.

Note: All logarithms default to base 2 (i.e., $\log x$ is defined as $\log_2 x$). **Note:** "Giving" an algorithm means doing the four parts (i)–(iv) as described on the course web page. **Proofs are required unless we explicitly state otherwise.** If you think we made a mistake and a proof is not required, feel free to ask. You may use the **unit cost model unless otherwise stated**.

1. [15 marks] **Solving a System of Linear Inequalities.** You are given a system Σ of m linear inequalities over n unknowns $x_1, \ldots, x_n$. Each inequality is of the form:

   $$x_i - x_j \leq a_{i,j}$$

   where $a_{i,j}$ is an integer constant. Note that $a_{i,j}$ is not necessarily defined for all (i, j). Thus, the $a_{i,j}$ integers that *are* defined are given to you in an array $A[1..n]$ where *each entry* $A[i]$ is itself an array. More specifically, $A[i]$ is an array of entries of the form $[j, a_{i,j}]$. In other words, $A[i]$ contains $a_{i,j}$ for all j such that $a_{i,j}$ is actually defined in Σ.

   Give an algorithm that decides if there is a solution to these inequalities, and if so, finds one. The output is thus: NULL if there is no solution, and an array of n integers containing the (solved) values of $x_1 ... x_n$ otherwise.

   Hint: build a weighted directed graph G such that G has no negative weight cycle if and only if there is a solution to the system Σ.

2. [15 marks] **Verifying Treeness, Spanning and Minimality.** You are given a connected weighted undirected graph $G = (V, E)$, with vertices $V = \{1, \ldots, n\}$, and m edges. As input, you can assume an adjacency list representation $A[1..n]$. Each $A[i]$ is itself an array, with all entries of the form $[j, w_{i,j}]$, with $w_{i,j}$ the weight of edge $\{i, j\}$. You can assume that all $w_{i,j}$ are distinct, and that the entries of $A[i]$ are sorted in increasing order of j.

   You are also given an array $T[1..n-1]$, whose entries are in principle *meant to* represent some edges of G, but you **may not** assume its entries actually represent edges of G. All you may assume is that each entry of T is of the form (u, v), with $u \neq v$ integers in $1, \ldots, n$. The goal of this problem is to verify whether T is the minimum spanning tree of G.[1]

   (a) Prove the following general property: for any cycle C in G, the edge e with maximum weight on C does not belong to the minimal spanning tree of G.

   (b) Give an algorithm that determines whether T is a **spanning tree** (not necessarily minimal). If so, return an array $P[1..n]$ where for all i, $P[i] = [j, w_{i,j}]$, with j the parent of i in T. For this to make sense, assume that *if T is a spanning tree*, then we choose the vertex 1 as root (and set $P[1] = [1, \bullet]$). If T is not a spanning tree, you can return NULL. For full credit, this should be done in time $O(n \log n)$.

[1] Warning: we will not obtain a better worst-case bound than Kruskal's algorithm (in any part). Still, we hope to design an algorithm that would be able, in practice, to detect *early on* that T is not an MST.

(c) (bonus question, +4 marks) Assume T is a **spanning tree**. For any vertices $u \neq v$ in G, there is a unique path $P_{u,v}$ connecting u to v in T; we are interested in accessing efficiently the maximum weight $w^T_{u,v}$ of the edges along this path, that is, $w^T_{u,v} = \max_{e \in P_{u,v}} w_e$. Describe a data structure S such that

- given P from the previous question, S can be created in time $O(n \log n)$
- given S, u and v, you can obtain $w^T_{u,v}$ in time $O(\log n)$.

Note that S should contain information about *all pairs* of u, v so that you can answer queries about *any* u, v pair in $O(\log n)$ time.

(d) For this part, assume that you already have a complete solution to part (c), meaning you have access to a data structure that lets you find $w^T_{u,v}$ in $O(\log n)$ time for any u, v. Assume that T is a spanning tree. Give a simple criterion involving the weights $w_{u,v}$ and $w^T_{u,v}$ to test if T is minimal. Give an algorithm that decides if T is minimal, and give a big-O on the worst-case runtime.

3. [10 marks] **Squid-like game.** Input: a 2D array $A[1..k, 1..k]$ of positive integers, a start location $s = (s_i, s_j)$ and a finish location $f = (f_i, f_j)$.

   The array represents a $k \times k$ grid of breakable floor tiles that you can stand on. You start on tile s at time 0, and it takes 1 unit of time to jump to a neighbouring tile (up, down, left, right).[2] You want to travel to tile f.

   Unfortunately, as time ticks forward, the floor tiles are being smashed! Fortunately, you know, for each tile, the exact time when it will be smashed. Those times are exactly the contents of the array A. That is, tile i, j is smashed at time $A[i, j]$. The tile you are trying to move to, f, is a safe tile (smashed at time ∞).

   You cannot occupy a smashed tile, but you can leave a tile at the same time as it is being smashed. More specifically, if you are on tile i, j at time $t = A[i, j] - 1$ (meaning the tile will be smashed in the next time step), then you can safely move to a neighbouring tile, arriving at time $t + 1$, as long as that neighbouring tile will be smashed at time $t + 2$ or later.

   Output: True/false. Can you travel safely from s to f?

   Runtime $O(k^2)$ for full marks.

[2] It doesn't *really* matter which direction you consider +/- i or +/- j to be, but let's say -i is up, +i is down, -j is left and +j is right. I.e., the same as: `for (i = 1..k) { for (j = 1..k) print` $A[i, j]$ `} print newline`.