CS 331: Algorithms and Complexity
Homework III

Trung Dang Nathan Mardanov Kevin Tian

Due date: October 20, 2025, end of day (11:59 PM), uploaded to Canvas.

Late policy: 15% off if submitted late, and 15% off for every further 24 hours before submission.

Please list all collaborators on the first page of your solutions.

When runtimes are unspecified, slower runtimes than the intended solution receive partial credit.

1 Problem 1

(**20 points**) We say that directed graph $G = (V, E)$ is *moderately connected* if for all pairs of distinct vertices $(u, v) \in V \times V$, either v is reachable from u, or u is reachable from v (not necessarily both). Give an algorithm that determines whether an input directed graph G is moderately connected.

2 Problem 2

(**20 points**) Let $G = (V, E)$ be a connected undirected graph with $n := |V|$ and $m := |E|$, and let $s \in V$ be a source vertex. For each $v \in V$, define $c(v)$ to be the number of distinct shortest paths between s and v. Give an $O(m + n)$-time algorithm that takes as input G and computes $c(v)$ for all $v \in V$. You may assume that adding arbitrary integers takes $O(1)$ time for this problem.

3 Problem 3

(**20 points**) Define the *width* of a path P to be the minimum edge weight along the path.[1] Let $G = (V, E, \mathbf{w})$ be a directed graph with $n := |V|$ and $m := |E|$, and let $s \in V$ be a source vertex. Assume all of V is reachable from s. For each $v \in V$, define $w(v)$ to be the maximum width of a path from s to v. Give an $O(m \log(n))$-time algorithm that takes as input G and computes $w(v)$ for all $v \in V \setminus \{s\}$. We consider the width of an empty path from s to itself to be ∞.

Your algorithm should be a very small modification of Dijkstra's algorithm (Section 3.1, Part V).

[1] Intuitively, you can think of each edge weight as specifying the diameter of a tube, so that the minimum edge weight along the path capacitates the width of the tubes, i.e., the largest amount of material that can pass.

4 Problem 4

(**20 points**) The country of Numerica, where the citizens are named $[n] := 1, 2, \ldots, n$, is having a massive 1-on-1 chess tournament. You are following along, and you are a big fan of citizen a. Currently, you know that each citizen $v \in [n]$ has a win count of w_v. You also know the values r_{uv} for every pair of citizens $(u, v) \in [n] \times [n]$, where r_{uv} is the number of remaining chess games between citizens u and v. Note that $r_{uv} = r_{vu}$ (i.e., r is symmetric), and $r_{uu} = 0$ for all $u \in [n]$. You want to see if your favorite player, a, can still win.

Formally, design an algorithm that takes as input a designated index $a \in [n]$, and the lists

$$R = \{r_{uv} \in \mathbb{Z}_{\geq 0}\}_{(u,v)\in[n]\times[n]}, \quad W = \{w_v \in \mathbb{Z}_{\geq 0}\}_{v\in[n]},$$

where R is given as a two-dimensional Array and W is given as a one-dimensional Array, and returns **True** or **False** depending on whether the remaining games can be played out in a way such that a finishes the tournament with strictly more wins than any other citizen in $[n]$.

For example, if $n = 3$, $a = 1$, $w_1 = w_2 = w_3 = 1$, and $r_{12} = r_{13} = r_{23} = 2$, we should return **True**, because a could win its 4 remaining games, and $2, 3$ could split their remaining games with each other. This would result in updated counts $w_1 = 5$, $w_2 = 2$, and $w_3 = 2$, so $a = 1$ would win.

For this problem, any runtime that is polynomial in n receives full credit.

5 Problem 5

(**20 points**) Complete the assignment at this link. This link is only accessible on your UT email.