HW 5

Due on **Wednesday, October 4, 2023 at 9pm**

CS 473: Algorithms, Fall 2023 Version: **1.2**

Submissions instructions: As in previous homework.

13 (100 PTS.) Not again.

A new virus had spread into a community with n people $C = \{c_1, \ldots, c_n\}$. You know that exactly k people (but not their identity) are sick, but unfortunately, you have only t testing kits (conceptually, think about t as being very small, like 2 or 3). Fortunately, you can apply the test on a group of people (but then, you can not distinguish between the members of the group). Specifically, if you apply the test on a group $G \subseteq \{c_1, \ldots, c_n\}$, you get back that either somebody in the group is sick, or that all the people in G are virus free (yey!).

13.A. (60 PTS.) Describe a randomized algorithm, that performs such t group tests, and output a set of citizens that are all not sick (for certain!). Your algorithm should return a set that is in expectation as large as possible. Provide an ***exact bound*** on the expected size of the set output by your algorithm, as a function of n, k and t.

13.B. (10 PTS.) What is the expected size of the set for $t = 1$, $t = 2$, $t = 3$, $t = k$, say for $k = 20$? (Numerical calculations of the values are fine here, as long as your formula is correct, naturally.)

13.C. (30 PTS.) What is the minimum value of t for which your algorithm would output the (exactly) $n - k$ citizens in the community that are not sick (say with probability $\geq 1/2$)? Provide an asymptotic bound as a function of n and k.

14 (100 PTS.) Monotone path.

Let G be an undirected graph with n vertices and m edges. Given a numbering $\chi : \mathsf{V}(\mathsf{G}) \rightarrow [\![k]\!]$ of the vertices, a cycle $v_1, v_2, \ldots, v_k, v_1$ is a ***monotone*** path if $v_i v_{i+1} \in \mathsf{E}(\mathsf{G})$, for all i, and $\chi(v_i) = i$, for all i.

14.A. (50 PTS.) Describe an algorithm, countPaths, as fast as possible, that given G and a numbering of the vertices (with parameter k), outputs the number of distinct monotone paths in G. What is the running time of your algorithm? Shortly describe how to modify your algorithm to a new algorithm computePath, that outputs such a path if it exists.

14.B. (50 PTS.) Let H be a directed graph with n vertices and m edges. Let k be a parameter. For a given k, describe a **randomized** algorithm that uses computePath (directly) as a black box, only a "few" times, and outputs a *simple* path in H of length exactly k (you can safely assume that such a path exists), and does it with probability at least (say) $1/2$. How many times does your algorithm calls computePath, as a function of k? What is the running time of your algorithm overall? Prove that the algorithm indeed outputs the desired cycle with the desired probability.

(As a reminder, a simple path of length k is a path k vertices, such that no vertex on the path appears more than once.)

(Hint: How to generate a coloring?)

15 (100 PTS.) Shortest path.

15.A. (40 PTS.) You are given an undirected graph G with n vertices and m edges, with positive weights on the edges. Given a subset $S \subseteq \mathsf{V}(\mathsf{G})$, describe an algorithm nn(G, S), as fast as possible, that computes for every vertex $v \in \mathsf{V}(\mathsf{G})$, its nearest neighbor in S. That is, for every vertex $v \in V$, the algorithm computes the pair $(\ell(v), \mathsf{nn}(v))$, where

$$\ell(v) = \min_{s \in S} d_G(s, v) \qquad \text{and} \qquad n(v) = \arg\min_{s \in S} d_G(s, v),$$

where $d_G(s, v)$ is the length of the shortest path in G between v and s. (It would also be useful for the algorithm to store. What is the running time of your algorithm?

(Hint: Modify Dijkstra.)

15.B. (60 PTS.) Given G, S (as above) and a parameter k, describe a randomized algorithm, as fast as possible, that computes for every vertex $v \in \mathsf{V}(\mathsf{G})$, the k closest vertices for it in S. Your algorithm must work by calling the procedure nn from the previous part a “small” number of times (and this is the only part of your algorithm that compute shortest paths). How many times does your algorithm has to call nn so that the results returned by the algorithm are correct with probability $\geq 1 - 1/n^{10}$?