CS 180 Summer 25 – Homework 2

Due Saturday, July 12, 11:59pm

- Please write your student ID and the names of anyone you collaborated with on the first page of your submission. Do not include your name on your assignment as the homework will be blind graded.
- You may use any theorem proven in class or in the textbook without proof. When referring to these theorems, write "as was proven in class" or "as was proven in the textbook" accordingly.
- **If you are asked to write an algorithm, you must prove its correctness.** If you are asked for a time complexity analysis, then you should give a reasonable big-O notation upper bound.
- The starred problems (Problems 3, 4, and 5) will be graded by correctness.
- All other problems (Problems 1 and 2) will be graded by completeness. You will receive full credit on these problems as long as a good-faith effort was made to solve them. This includes showing your work and providing proofs of your statements.

1. **Finding either a Topological Ordering or a Cycle (Problem 3.3 in the Textbook)**

   The algorithm for computing a topological ordering of a DAG repeatedly finds a node with no incoming edges and deletes it. This will eventually produce a topological ordering, provided that the input graph really is a DAG. But suppose that we're given an arbitrary graph that may or may not be a DAG. Extend the topological ordering algorithm so that, given an input directed graph G, it outputs one of two things:

   (a) a topological ordering, thus establishing that G is a DAG;

   (b) a cycle in G, thus establishing that G is not a DAG.

   Additionally, provide a time complexity analysis of your algorithm. The running time of your algorithm should be $O(n + m)$ for a directed graph with n nodes and m edges.

2. **Deleting a Node to Destroy all Paths (Problem 3.9 in the Textbook)**

   There's a natural intuition that two nodes that are far apart in a communication network—separated by many hops—have a more tenuous connection than two nodes that are close together. There are a number of algorithmic results that are based to some extent on different ways of making this notion precise. Here's one that involves the susceptibility of paths to the deletion of nodes.

   Suppose that an n-node undirected graph $G = (V, E)$ contains two nodes s and t such that the distance between s and t is strictly greater than $\frac{n}{2}$. Show that there must exist some node v, not equal to either s or t, such that deleting v from G destroys all s-t paths. (In other words, the graph obtained from G by deleting v contains no path from s to t.) Give an algorithm with running time $O(m + n)$ to find such a node v. Additionally, provide a time complexity analysis of your algorithm.

3. ***Butterfly Classification**
(Problem 3.4 in the Textbook)

Inspired by the example of that great Cornellian, Vladimir Nabokov, some of your friends have become amateur lepidopterists (they study butterflies). Often when they return from a trip with specimens of butterflies, it is very difficult for them to tell how many distinct species they've caught—thanks to the fact that many species look very similar to one another. One day they return with n butterflies, and they believe that each belongs to one of two different species, which we'll call A and B for purposes of this discussion. They'd like to divide the n specimens into two groups—those that belong to A and those that belong to B—but it's very hard for them to directly label any one specimen. So they decide to adopt the following approach.

For each pair of specimens i and j, they study them carefully side by side. If they're confident enough in their judgment, then they label the pair (i, j) either "same" (meaning they believe them both to come from the same species) or "different" (meaning they believe them to come from different species). They also have the option of rendering no judgment on a given pair, in which case we'll call the pair ambiguous.

So now they have the collection of n specimens, as well as a collection of m judgments (either "same" or "different") for the pairs that were not declared to be ambiguous. They'd like to know if this data is consistent with the idea that each butterfly is from one of species A or B. So more concretely, we'll declare the m judgments to be consistent if it is possible to label each specimen either A or B in such a way that for each pair (i, j) labeled "same," it is the case that i and j have the same label; and for each pair (i, j) labeled "different," it is the case that i and j have different labels. They're in the middle of tediously working out whether their judgments are consistent, when one of them realizes that you probably have an algorithm that would answer this question right away. Give an algorithm with running time $O(m + n)$ that determines whether the m judgments are consistent. Additionally, provide a time complexity analysis of your algorithm.

For this problem, you may assume the following:

- The graph $G = (V, E)$ where V is the set of n butterfly specimens and E is the set of m judgements (either "same" or "different") is a *connected* graph.
- Your input is given as a set of $2n$ linked lists $(L_{1,\mathsf{same}}, L_{1,\mathsf{diff}}, \ldots, L_{n,\mathsf{same}}, L_{n,\mathsf{diff}})$ where
  - $L_{i,\mathsf{same}}$ is a linked list containing all butterflies j such that (i, j) is labeled "same"
  - $L_{i,\mathsf{diff}}$ is a linked list containing all butterflies j such that (i, j) is labeled "different"

4. ***Number of Shortest Paths**
(Problem 3.10 in the Textbook)

A number of art museums around the country have been featuring work by an artist named Mark Lombardi (1951–2000), consisting of a set of intricately rendered graphs. Building on a great deal of research, these graphs encode the relationships among people involved in major political scandals over the past several decades: the nodes correspond to participants, and each edge indicates some type of relationship between a pair of participants. And so, if you peer closely enough at the drawings, you can trace out ominous-looking paths from a high-ranking U.S. government official, to a former business partner, to a bank in Switzerland, to a shadowy arms dealer.

Such pictures form striking examples of social networks, which have nodes representing people and organizations, and edges representing relationships of various kinds. And the short paths that abound in these networks have attracted considerable attention recently, as people ponder what they mean. In the case of Mark Lombardi's graphs, they hint at the short set of steps that can carry you from the reputable to the disreputable.

Of course, a single, spurious short path between nodes v and w in such a network may be more coincidental than anything else; a large number of short paths between v and w can be much more convincing. So in addition to the problem of computing a single shortest v-w path in a graph G, social networks researchers have looked at the problem of determining the number of shortest v-w paths.

This turns out to be a problem that can be solved efficiently. Suppose we are given an undirected graph $G = (V, E)$, and we identify two nodes v and w in G. Give an algorithm that computes the number of shortest v-w paths in G. (The algorithm should not list all the paths; just the number suffices.) Additionally, provide a time complexity analysis of your algorithm. The running time of your algorithm should be $O(m + n)$ for a graph with n nodes and m edges.

5. ***Births and Deaths**
(A Modification of Problem 3.12 in the Textbook)

You're helping a group of ethnographers analyze some oral history data they've collected by interviewing members of a village to learn about the lives of people who've lived there over the past two hundred years. From these interviews, they've learned about a set of n people (all of them now deceased), whom we'll denote P_1, P_2,... P_n. They've also collected facts about when these people lived relative to one another. Each fact has one of the following three forms:

- For some i and j, person P_i died before person P_j was born; or
- For some i and j, person P_i died before person P_j died; or
- For some i and j, the life spans of P_i and P_j overlapped at least partially.

Naturally, they're not sure that all these facts are correct; memories are not so good, and a lot of this was passed down by word of mouth. So what they'd like you to determine is whether the data they've collected is at least internally consistent, in the sense that there could have existed a set of people for which all the facts they've learned simultaneously hold.

Give an efficient algorithm to do this: either it should produce proposed dates of birth and death for each of the n people so that all the facts hold true, or it should report (correctly) that no such dates can exist—that is, the facts collected by the ethnographers are not internally consistent. Additionally, provide a time complexity analysis of your algorithm.