CS 170 Homework 4

Due **9/27/2023, at 10:00 pm (grace period until 11:59pm)**

1 Study Group

List the names and SIDs of the members in your study group. If you have no collaborators, you must explicitly write "none".

2 Distant Descendants

You are given a tree $T = (V, E)$ with a designated root node r and a positive integer K. For each vertex v, let $s[v]$ be the number of descendants of v that are a distance of at least K from v. Describe an $O(|V|)$ algorithm to output $s[v]$ for every v. Give a 3-part solution.

3 Where's the Graph?

Each of the following problems can be solved with techniques taught in lecture. Construct a simple directed graph, write an algorithm for each problem by black-boxing algorithms taught in lecture, and analyze its runtime. You do not need to provide proofs of correctness.

(a) The CS 170 course staff is helping build a roadway system for PNPenguin's hometown in Antarctica. Since we have skill issues, we are only able to build the system using one-way roads between igloo homes. Before we begin construction, PNPenguin wants to evaluate the reliability of our design. To do this, they want to determine the number of *reliable* igloos; an igloo is reliable if you are able to leave the igloo along some road and then return to it along a path of other roads. However, PNPenguin isn't very good at algorithms, and they need your help.

Given our proposed roadway layout, design an efficient algorithm that determines the number of reliable igloos.

(b) There are p different species of Pokemon, all descended from the original Mew. For any species of Pokemon, Professor Juniper knows all of the species *directly* descended from it. She wants to write a program that answers queries about these Pokemon. The program would have two inputs: a and b, which represent two different species of Pokemon. Her program would then output one of three options in constant time (the time complexity cannot rely on p):

(1) a is descended from b.

(2) b is descended from a.

(3) a and b share a common ancestor, but neither are descended from each other.

Unfortunately, Professor Juniper's laptop is very old and its SSD drive only has enough space to store up to $O(p)$ pieces of data for the program. Give an algorithm that Professor Juniper's program could use to solve the problem above given the constraints.

Hint: Professor Juniper can run some algorithm on her data before all of her queries and store the outputs of the algorithm for her program; time taken for this precomputation is not considered in the query run time.

(c) Bob has n different boxes. He wants to send the famous "Blue Roses' Unicorn" figurine from his glass menagerie to his crush. To protect it, he will put it in a sequence of boxes. Each box has a weight w and size s; with advances in technology, some boxes have negative weight. A box a inside a box b cannot be more than 15% smaller than the size of box b; otherwise, the box will move, and the precious figurine will shatter. The figurine needs to be placed in the smallest box x of Bob's box collection.

Bob (and Bob's computer) can ask his digital home assistant Falexa to give him a list of all boxes less than 15% smaller than a given box c (i.e. all boxes that have size between 0.85 to 1 times that of c). Bob will need to pay postage for each unit of weight (for negative weights, the post office will pay Bob!). Find an algorithm that will find the lightest sequence of boxes that can fit in each other in linear time (in terms of the graph).

Hint: How can we create a graph knowing that no larger box can fit into a smaller box, and what property does this graph have?

4 Semiconnected DAG

A directed acyclic graph G is *semiconnected* if for any two vertices A and B, there is a path from A to B or a path from B to A. Show that G is semiconnected if and only if there is a directed path that visits all of the vertices of G. Make sure to prove both sides of the "if and only if" condition.

Hint: Is there a specific arrangement of the vertices that can help us solve this problem?

5 2-SAT

In the 2SAT problem, you are given a set of clauses, where each clause is the disjunction (OR) of two literals (a literal is a Boolean variable or the negation of a Boolean variable). You are looking for a way to assign a value true or false to each of the variables so that all clauses are satisfied – that is, there is at least one true literal in each clause. For example, here's an instance of 2SAT:

$$(x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (x_1 \vee x_2) \wedge (\overline{x_3} \vee x_4) \wedge (\overline{x_1} \vee x_4)$$

Recall that $\vee$ is the logical-OR operator and $\wedge$ is the logical-AND operator and $\overline{x}$ denotes the negation of the variable x. This instance has a satisfying assignment: set x_1, x_2, x_3, and x_4 to `true, false, false, and true`, respectively.

The purpose of this problem is to lead you to a way of solving 2SAT efficiently by reducing it to the problem of finding the strongly connected components of a directed graph. Given an instance I of 2SAT with n variables and m clauses, construct a directed graph $G_I = (V, E)$ as follows.

- G_I has $2n$ nodes: one for each variable and its negation.
- G_I has $2m$ edges: for each clause $(\alpha \vee \beta)$ of I (where α, β are literals), G_I has an edge from from $\overline{\alpha}$ to β, and one from the $\overline{\beta}$ to α.

Note that the clause $(\alpha \vee \beta)$ is equivalent to each of the implications $\overline{\alpha} \implies \beta$ and $\overline{\beta} \implies \alpha$. In this sense, G_I records all implications in I.

(a) Show that if G_I has a strongly connected component containing both x and $\overline{x}$ for some variable x, then I has no satisfying assignment.

(b) Now show the converse of (a): namely, that if none of G_I's strongly connected components contain both a literal and its negation, then the instance I must be satisfiable.

Hint: Pick a sink SCC of G_I. Assign variable values so that all literals in the sink are True. Why are we allowed to do this, and why doesn't it break any implications?

(c) Conclude that there is a linear-time algorithm for solving 2SAT. Provide the algorithm description and runtime analysis; proof of correctness is not required.

6 [Coding] DFS and Kosaraju's Algorithm

For this week's coding questions, we'll implement DFS and apply it to find the strongly connected components within a graph via Kosaraju's algorithm. There are two ways that you can access the notebook and complete the problems:

1. **On Local Machine**: `git clone` (or if you already cloned it, `git pull`) from the coding homework repo,

   `https://github.com/Berkeley-CS170/cs170-fa23-coding`

   and navigate to the `hw04` folder. Refer to the `README.md` for local setup instructions.

2. **On Datahub**: Click here and navigate to the `hw04` folder if you prefer to complete this question on Berkeley DataHub.

Notes:

- *Submission Instructions:* Please download your completed submission `.zip` file and submit it to the gradescope assignment titled "Homework 4 Coding Portion".
- *OH/HWP Instructions:* Designated coding course staff will provide conceptual and debugging help during office hours and homework parties.
- *Academic Honesty Guideline:* We realize that code for some of the algorithms we ask you to implement may be readily available online, but we strongly encourage you to not directly copy code from these sources. Instead, try to refer to the resources mentioned in the notebook and come up with code yourself. That being said, we **do acknowledge** that there may not be many different ways to code up particular algorithms and that your solution may be similar to other solutions available online.