CS 170 Homework 4

Due **Friday 9/27/2024, 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, explicitly write "none".

2 Sparsity and SCCs

Call a directed graph G *weakly-connected* if, upon making all edges bidirectional (i.e. turning the directed graph into an undirected graph), the graph is connected.

(a) Argue that every weakly-connected directed graph G with n vertices and $n-1$ edges must have n strongly connected components.

(b) Consider a weakly-connected directed graph G with $n \geq 3$ vertices and n directed edges such that for every vertex, v, there are exactly two directed edges incident to v. For each k from 1 to n, describe conditions upon which there are k strongly-connected components.

Hint: this graph structure is very special. If $n = 5$, is it possible for there to be exactly two strongly connected components?

(c) For the graph described part (b), describe a *simple* and efficient algorithm using *only one* DFS or BFS traversal that determines the number of SCCs G has.

Note: you may *not* use the SCC-finding algorithm from lecture or another black-box SCC algorithm such as Tarjan's algorithm.

(d) Now, consider any weakly-connected directed graph G' with $n \geq 3$ vertices and n directed edges. Modify your algorithm from part (c) to efficiently count the number of SCCs G' has (still using at most one DFS or BFS traversal).

3 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 (in fact, you've already done the bulk of the proof!)

4 Road Trip

The CS 170 staff are preparing to drive from Berkeley to Chicago to attend a computer science conference!

Assume that the roads from Berkeley to Chicago can be expressed as a directed weighted graph $G = (V, E, c)$, where each $v \in V$ represents a city, each $(u, v) \in E$ represents a road from city u to city v, and $c(u, v)$ represents the cost of gas required to drive from city u to city v. Each road takes exactly one day to traverse, and after driving across road (u, v), the staff will stop in city v overnight to rest.

Unfortunately, there are too many course staff, so they will have to rent two separate cars for the trip, and each city along the way can only accommodate one group of staff per night. Both cars start in Berkeley and depart on the same day.

Additionally, each group has to pay r dollars per day in car rental fees. They are allowed to spend a day in any city without driving, but they still have to pay the rental fee for that day. Once a group arrives in Chicago, the group will return the car and does not have to pay any more rental fees.

Design an efficient algorithm to find the route that minimizes the total cost of the trip, including gas and rental fees. (You may assume that no matter the route, the staff will always arrive in Chicago before the conference starts.)

Provide a 3-part solution with runtime in terms of $n = |V|$, $m = |E|$.

5 Shortest Path with Clusters

Sometimes, we can exploit the structure of a graph to come up with faster shortest-path algorithms, as we saw in class with Dijkstra's algorithm and the Shortest Path in DAGs algorithms.

In this problem, consider a graph $G = (V, E, w)$ with possibly negative edge weights and the guarantee that every strongly connected component of G has at most k vertices for some fixed $k \in \mathbb{N}$.

Design an algorithm to find the shortest path from a source vertex s to a target vertex t in G that runs in time asymptotically faster than $O(|V| \cdot |E|)$ when k is asymptotically smaller than $|V|$, and does not run slower than $O(|V| \cdot |E|)$ if $k = \Theta(|V|)$.

Give a 3-part solution.

6 Arbitrage

Shortest-path algorithms can also be applied to currency trading. Suppose we have n currencies $C = \{c_1, c_2, \ldots, c_n\}$: e.g., dollars, Euros, bitcoins, dogecoins, etc. For any pair of currencies c_i, c_j, there is an exchange rate $r_{i,j}$: you can buy $r_{i,j}$ units of currency c_j at the price of one unit of currency c_i. Assume that $r_{i,i} = 1$ and $r_{i,j} \geq 0$ for all i, j.

The Foreign Exchange Market Organization (FEMO) has hired Oski, a CS170 alumnus, to make sure that it is not possible to generate a profit through a cycle of exchanges; that is, for any currency $i \in C$, it is not possible to start with one unit of currency i, perform a series of exchanges, and end with more than one unit of currency i. (That is called *arbitrage*.)

More precisely, arbitrage is possible when there is a sequence of currencies $c_{i_1}, \ldots, c_{i_k}$ such that $r_{i_1,i_2} \cdot r_{i_2,i_3} \cdot \cdots \cdot r_{i_{k-1},i_k} \cdot r_{i_k,i_1} > 1$. This means that by starting with one unit of currency c_{i_1} and then successively converting it to currencies $c_{i_2}, c_{i_3}, \ldots, c_{i_k}$ and finally back to c_{i_1}, you would end up with more than one unit of currency c_{i_1}. Such anomalies last only a fraction of a minute on the currency exchange, but they provide an opportunity for profit.

We say that a set of exchange rates is arbitrage-free when there is no such sequence, i.e. it is not possible to profit by a series of exchanges.

(a) Give an efficient algorithm for the following problem: given a set of exchange rates $(r_{i,j})_{i,j \in n}$ which is *arbitrage-free*, and two specific currencies a, b, find the most profitable sequence of currency exchanges for converting currency a into currency b. That is, if you have a fixed amount of currency a, output a sequence of exchanges that gets you the maximum amount of currency b.

Hint 1: represent the currencies and rates by a graph whose edge weights are real numbers.

For part (a), give a 3-part solution.

(b) Oski is fed up of manually checking exchange rates, and has asked you for help to write a computer program to do his job for him. Give an efficient algorithm for detecting the possibility of arbitrage, and state the runtime of your algorithm. *Proof of correctness is not required.*