CS2800 Design and Analysis of Algorithms
Problem Set #4 February 13, 2023

1. Give a linear-time algorithm to find an odd cycle in a digraph.

2. Fake News Now wants to spread a rumor among the people of Gullible Town which has n inhabitants. If a member of Gullible Town hears a rumor, he/she will spread it to all his/her friends. By snooping around on social media, Fake News Now knows the friends of each of the inhabitants of Gullible Town. Since you are in the R& D wing of Fake News Now, you are tasked with finding the smallest number of people to whom you should say the rumor so that finally at some point, everyone in Gullible Town will know it. Design and analyze an algorithm to perform this task.

3. Given an undirected graph G, design an algorithm to check if it is possible to orient the edges of the graph such that the constructed digraph becomes strongly connected.

   Hint: You might want to recollect the concept of bridges that you learnt in your graph theory course.

4. Suppose that we are given a DAG G with a unique source s and a unique sink t. A vertex $v \notin \{s, t\}$ is said to be a *cut vertex* if every path from s to t in G goes via v.

   (a) Design an $O(|V| + |E|)$-time algorithm to find all cut vertices in G.

   For this, start by doing a topological sort of G. Now, show that a vertex v is a cut vertex iff there is no edge $(u, w) \in G$ such that u occurs before v in the topological order and w occurs after v in the topological order.

   First, use the observation above (without proof) to design you algorithm. Next, prove the statement above.

   (b) Does you algorithm work if there are multiple sources or sinks? Which part of the argument will break? Can you construct a counter-example illustrating this?

5. The 2-SAT problem is defined as follows. Consider a propositional formula ϕ in 2-CNF defined as follows:

$$\phi = (l_{1,1} \vee l_{2,2}) \wedge (l_{2,1} \vee l_{2,2}) \wedge \cdots \wedge (l_{m,1} \vee l_{m,2}),$$

   where each $l_{i,j}$ is either a boolean variable x_k or its negation $\overline{x}_k$. You want to check if there is a truth assignment to the variables $x_1, x_2, \ldots, x_n$ such that the propositional formula ϕ is satisfiable. For example, the formula $\phi = (x_1 \vee \overline{x}_2) \wedge (\overline{x}_1 \vee \overline{x}_3) \wedge (x_2 \vee x_3)$ is satisfiable with the truth assignment $x_1 = 1, x_2 = 1$, and $x_3 = 0$.

   Let us design a linear-time algorithm for checking the satisfiability as follows. First define a graph G_ϕ on $2n$ vertices as follows: for each i, there is a vertex corresponding to x_i and one of $\overline{x}_i$. Now, for each clause $(l_i \vee l_j)$, add a directed edge from $\overline{l}_i$ to l_j and one from $\overline{l}_j$ to j_i.

(a) Show that if G_ϕ has a strongly connected component containing x_i and $\overline{x}_i$ for some i, then ϕ is not satisfiable.

Hint: Recall that $(x \vee y)$ is logically equivalent to $\overline{x} \Rightarrow y$, and to $\overline{y} \Rightarrow x$

(b) Now, show that if none of the strongly connected components contains a variable and its negation, then the formula ϕ is satisfiable.

For this, start by considering a sink component in $\mathrm{SCC}(G_\phi)$, and assign all the literals in it to be true, and all their negations to be false. Show that you can extend this idea suitably to get a satisfying assignment for ϕ.

(c) Verify that this idea gives a linear-time algorithm to check the satisfiability (and find a satisfying assignment, if one exists) for a 2-CNF formula ϕ.

6. The *transitive closure* of a directed graph G is a graph G^* such that $(u, v) \in G^*$ iff there is a path from u to v in G. In this question, we will see how to find the transitive closure using boolean matrix multiplication. Given two boolean matrices A and B, the boolean matrix product C is defined as: $C[i, j] = \bigvee_k (A[i, k] \wedge B[k, j])$. It is known that there is an algorithm to perform boolean matrix multiplication in time $O(n^\omega)$ where $2 < \omega \leq 2.37$.

(a) Let A be the adjacency matrix of G. Show that the adjacency matrix of the G^* is A^n where you perform boolean matrix multiplication. (Use induction)

(b) Use the part above to design an $O(n^\omega \log n)$-time algorithm to compute G^*.

(c) Suppose that G is a *directed acyclic graph*. Let $v_1, v_2, \ldots, v_n$ be a topological ordering of the vertices of G. Let G_1 be the graph on the vertices $v_1, \ldots, v_{n/2}$ and G_2 be the graph on the vertices $v_{n/2}, \ldots, v_n$, and let G_3 be the graph $G \setminus \{G_1 \cup G_2\}$. Let A_1, A_2 and A_3 be the respective adjacency matrices of G_1, G_2, and G_3.

Show that the adjacency matrix of G^* is given by $A_1^{n/2} A_3 A_2^{n/2}$.

(d) Use the part above in a divide-and-conquer algorithm to obtain an $O(n^\omega)$-time algorithm to compute the transitive closure of directed acyclic graphs.

(e) Show that if G is strongly connected, then G^* is the graph such that for every $u, v \in V$, (u, v) and (v, u) are edges.

(f) Use parts above to design an $O(n^\omega)$-time algorithm to compute the transitive closure of a directed graph G.