

Homework # 1

Instructor: Dr. Antonio Blanca
TA: Guneykan Ozgul

Release Date: 09/02/2022

Notice: Type your answers using LaTeX and make sure to upload the answer file on Gradescope before the deadline. Recall that for any problem or part of a problem, you can use the “I’ll take 20%” option. For more details and the instructions read the syllabus.

Problem 1. Processing the Graph

(a.) Prove or disprove the following statement: if $\{u, v\} \in E$ is an edge in an undirected graph $G = (V, E)$, and during depth-first search the post-visit numbers of $u, v \in V$ satisfy $post(u) < post(v)$, then v is an ancestor of u in the DFS tree.

(b.) You are given a tree $T = (V, E)$ (in adjacency list format), along with a designated root node $r \in V$. Recall that u is said to be an *ancestor* of v in the rooted tree if the path from r to v in T passes through u . You wish to preprocess the tree so that queries of the form “is u an ancestor of v ?” can be answered in constant $O(1)$ time. The preprocessing itself should take linear time. How can this be done?

Problem 2. One-way Streets

State College Police Department has decided to make all the streets one-way. The mayor of State College Borough claims that there is still a way to drive legally from any intersection in the city to any other intersection, but the opposition is not convinced. You have been asked to write a computer program to determine whether the mayor is right. However, the mayor’s term is about to end and he has said that he will not run for another term, and there is just enough time to run a *linear-time* algorithm.

(a.) Formulate this problem graph-theoretically, and explain how it can indeed be solved in linear time (on the number of roads and intersection).

(b.) Suppose it now turns out that the mayor’s original claim is false. He then claims something weaker: if you start driving from Westgate building, navigating one-way streets, then no matter where you reach, there is always a way to drive legally back to Westgate Building. Formulate this weaker property as a graph-theoretic problem, and carefully show how it too can be checked in linear time.

Problem 3. Make It Strongly Connected

Imagine you are given a directed graph $G = (V, E)$ and you have been asked to find the minimum number of (directed) edges that are needed to be added to G such that the resulting graph is strongly connected. Give an algorithm that solves this problem in $\mathcal{O}(|V| + |E|)$ time. What is the space complexity of your algorithm?

Problem 4. 2SAT

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:

$$(x_1 \vee \hat{x}_2) \wedge (\hat{x}_1 \vee \hat{x}_3) \wedge (x_1 \vee x_2) \wedge (\hat{x}_3 \vee x_4) \wedge (\hat{x}_1 \vee x_4).$$

This instance has a satisfying assignment: set x_1, x_2, x_3 , and x_4 to **true**, **false**, **false**, and **true**, respectively.

Given an instance \mathcal{I} of 2SAT with n variables and m clauses, construct a directed graph $G_{\mathcal{I}} = (V, E)$ as follows:

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

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

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

(c.) Conclude that there is a linear-time algorithm for solving 2SAT.

Problem 5. Verify max flow

Suppose someone presents you with a solution to the maximum flow problem a flow network G . Give a linear time algorithm to determine whether the solution does indeed give a maximum flow.

Problem 6. Ford Fulkerson

Assume you are given the following flow network (Figure 1) with source s , sink t and capacities on the edges.

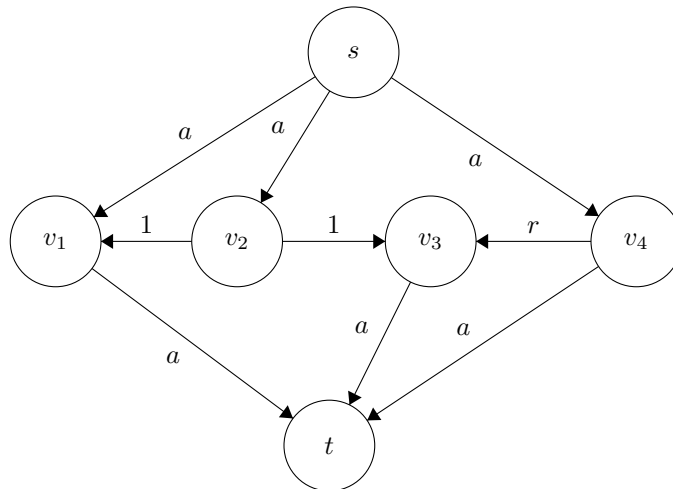


Figure 1: Flow network

where $r = \frac{(\sqrt{5} - 1)}{2}$ and $a \geq 2$ is an integer.

(a.) Show that maximum flow of the network is $2a + 1$.

(b.) Let $p_0 = \{s, v_2, v_3, t\}$, $p_1 = \{s, v_4, v_3, v_2, v_1, t\}$, $p_2 = \{s, v_2, v_3, v_4, t\}$ and $p_3 = \{s, v_1, v_2, v_3, t\}$ be $s - t$ paths in the residual graph. Suppose that the Ford-Fulkerson algorithm chooses to augment along the paths $p_0, p_1, p_2, p_1, p_3, p_1, p_2, p_1, p_3, p_1, p_2, p_1, p_3, \dots$ in this order. Show that after augmenting along p_3 each time the residual capacities of edges (v_2, v_1) , (v_4, v_3) and (v_2, v_3) are of the form r^k , r^{k+1} and 0 respectively for some $k \in \mathbb{N}$.

(c.) Prove that if we use the augmenting paths in the sequence above an infinite number of times, the total

flow converges to $3 + 2r$. Note that since the max flow is $2a + 1$, this show that the Ford-Fulkerson algorithm never terminates and the flow doesn't even converge to the maximum flow.

Problem 7. Removing an edge

Suppose you are given an s - t flow network $G = (V, E)$. You are also given an integer maximum s - t flow in G , defined by a flow value $f(e)$ on each edge e . Now suppose we pick a specific edge $e \in E$ and increase its capacity by one unit. Show how to find a maximum flow in the resulting capacitated graph in time $O(|V| + |E|)$.

Problem 8. Reducing maximum flow

Imagine you are given a flow network with all edge capacities equal to 1. In other words, you have directed graph $G = (V, E)$, a source $s \in V$, and a sink $t \in V$; and $c_e = 1$ for every $e \in E$. You are also given a parameter k . Your goal is to delete k edges so as to reduce the maximum $s - t$ flow in G by as much as possible. In other words, you should find a set of edges $F \subseteq E$ so that $|F| = k$ and the maximum $s - t$ flow in $G' = (V, E \setminus F)$ is as small as possible subject to this. Give an efficient algorithm to solve this problem.

Problem 9. Moving Branches

A company has n branches in one city, and it plans to move some of them to another city. The expenses for operating the i -th branch is a_i per year if it stays in the original city, and is b_i per year if it is moved to the new city. The company also needs to pay c_{ij} per year for traveling if the i -th and j -th branches are not in the same city. Design a polynomial time algorithm to decide which branches should be moved to the new city such that the total expenses (including operating and traveling expenses) per year is minimized.

Problem 10. Max flow with queries

You are given a directed graph $G = (V, E)$, where $|V| = n$ and $|E| = m$. Each edge has a capacity. Exactly k edges (e_1, e_2, \dots, e_k) have capacities equal to zero, initially. Someone asks you q queries and suppose that q is much larger than k . In each query, she gives you k integers w_1, w_2, \dots, w_k , where w_i is the capacity that she wants to be assigned to e_i , for $1 \leq i \leq k$. Give an algorithm to find the maximum flow that goes from a given vertex v to another given vertex u . The running time of your algorithm should be $O(2^k T_{\max_flow}(n, m) + q2^k)$, where $T_{\max_flow}(n, m)$ is the time complexity of computing the max flow.