Advanced Algorithms

Assignment 1

Due: February 17, 2026

Instructions

Problem 1 guides you in developing your own analysis of a natural algorithm for maximum flow. Problems 2-4 are designed to demonstrate how the problems we are learning about in class are more generally applicable than one might expect. A major goal of the course is not only to learn how to solve these fundamental problems, but to recognize when a naturally occurring problem is a thinly disguised version of one of the problems we have covered.

These problems are hard! You are *not expected to know how to solve them immediately*. You are encouraged to work together on them or come to me if you are stuck. However, you should write up your solutions yourself. Please list the people you worked with at the top of your submission. Looking for answers on the internet is not allowed, nor is working with an AI-powered system for any part of this assignment. You must understand everything you submit and I reserve the right to ask you to orally explain your answer to me. You may write your solutions by hand or in latex. Either way, submit them on gradescope by 10:00pm on Tuesday, February 17. Good luck!

Problem 1 - The Widest Path Method (30 points)

This problem explores another natural heuristic for choosing a good augmenting path. Recall that in the Edmonds-Karp algorithm, we choose the shortest augmenting path from s to t in the residual network G_f in each iteration. Suppose that instead, we always select the augmenting path on which we can push the most flow.

For example, in the network in Figure 1, this algorithm would push 3 units of flow on the path $s \to v \to w \to t$ in the first iteration, and 2 units on $s \to w \to v \to t$ in the second iteration.

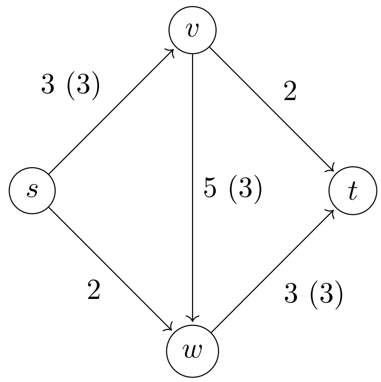

Figure 1: Edges show capacity; flow in parentheses.

(a) Recall how Dijkstra's shortest path algorithm computes the shortest length path between a pair of nodes in a graph. Write down the algorithm at a high level in English or pseudo-code (you can Google this) (this will be helpful for part (b)).

(b) Define the bottleneck capacity of a path to be the minimum weight of edges along that path. Show how to modify Dijkstra's shortest-path algorithm so that it computes an s-t path with maximum possible bottleneck capacity.

This shows how we can implement one iteration of the Widest Path Method. Our next step will be to analyze how many iterations this method can take in the worst case.

(c) Think about why the following fact is true / seems reasonable (you don't have to prove it): suppose G is a flow network, and the maximum-flow value in G is F^*. If f is a flow in G with value F, then the maximum flow value in the residual network G_f is equal to $F^* - F$.

(d) You may assume the fact in part (c) to solve this question! Suppose G is a flow network with max-flow value F^* and f is a flow of value F in G. Show that there is an augmenting path in G_f such that every edge has residual capacity at least $(F^* - F)/m$, where $m = |E|$.

[Hint: if Δ is the maximum amount of flow that can be pushed on any s-t path of G_f, consider the set of vertices reachable from s along edges in G_f with residual capacity more than Δ. Relate the capacity of this (s,t)-cut in G_f to $F^* - F$.]

(e) You may assume the answer to part (d) to solve this question. Prove that the Widest-Path variant of the Ford-Fulkerson algorithm terminates within $O(m \log F^*)$ iterations, where F^* is defined as in the previous problem.

[Hint: you might find the inequality $1 - x \le e^{-x}$ for $x \in [0, 1]$ useful.]

Problem 2 - Matrix Rounding (20 points)

In class, we have seen how to solve maximum flow problems where there are upper bounds (capacities) on the amount of flow that can be carried by each edge. What if there is also a minimum amount of flow that should be sent on the edges? It turns out that the maximum flow problem with lower and upper bounds on each edge can be reduced to the standard maximum flow problem (which only has upper bounds). If you want to know how this is done, you can read, for example, Section 25.1 of these notes. This is optional reading, but what you should know for the remainder of this problem is this:

In a flow network $G = (V, E)$ with source s, sink t and integer maximum and minimum capacities ℓ_e and c_e for each edge e, there is an **integer** maximum-flow f from s to t which satisfies $\ell_e \leq f_e \leq c_e$ for all edges $e \in E$, and this can be computed in polynomial time.

(a) Suppose we are given an $m \times n$ matrix A of non-negative real numbers. Our goal is to round A to an integer matrix by replacing each entry x in A with either $\lfloor x \rfloor$ or $\lceil x \rceil$, without changing the sum of entries in any row or column of A.

For example:

$$\begin{bmatrix} 1.2 & 3.4 & 2.4 \\ 3.9 & 4 & 2.1 \\ 7.9 & 1.6 & 0.5 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 4 & 2 \\ 4 & 4 & 2 \\ 8 & 1 & 1 \end{bmatrix}$$

(b) Give a polynomial time algorithm that either rounds the matrix A in this fashion, or reports correctly that no such rounding is possible.

(c) Suppose that all row and column sums of A are integral. Explain why such a rounding is guaranteed to exist.

Problem 3 - Friendship is Magic (25 points)

In sociology, one often studies a graph G in which nodes represent people and edges represent those who are friends with each other. Let's assume for purposes of this question that friendship is symmetric, so we can consider an undirected graph.

Now suppose we want to study this graph G, looking for a "close-knit" group of people. One way to formalize this notion would be as follows. For a subset S of nodes, let $E(S)$ denote the number of edges in S - that is, the edges that have both endpoints in S. We define the cohesiveness of S as $|E(S)|/|S|$. A natural thing to search for would be a set S of people achieving the maximum cohesiveness. As usual, let n be the number of vertices in G and m be the number of edges.

(a) Show that the problem of finding a set S with cohesiveness greater than α is equivalent to finding a set S of vertices satisfying $(m - |E(S)|) + \alpha|S| < m$.

(b) Give a polynomial-time algorithm that takes a rational number α and determines whether there exists a set S with cohesiveness greater than α.

Hint: Check if the minimum s, t-cut in a new graph is less than m.

Hint: The new graph should have a node for each edge in E and vertex in V

(c) Show that the value of $|E(S)|/|S|$ for any set S is at least 0 and at most n.

(d) Show that difference between $|E(A)|/|A|$ and $|E(B)|/|B|$ is either 0 or at least $1/n^2$.

(e) Give a polynomial-time algorithm to find a set S of nodes with maximum cohesiveness. You may assume the answers to parts (b) – (d).

Problem 4 - n degrees of Bacon (25 points)

Some friends of yours have grown tired of the game "Six Degrees of Kevin Bacon" (after all, isn't it just breadth-first search?) and decide to invent the following cooler game, algorithmically speaking. Here's how it works.

You start with a set X of n actresses and a set Y of n actors, and two players P_1 and P_2. Player P_1 names an actress $x_1 \in X$, player P_2 names an actor y_1 who has appeared in a movie with x_1. Player P_0 then names another actress $x_2 \in X$ who has not been named before, player P_1 names an actor y_2 who has appeared in a movie with x_2 and has not been named before, and so on.

Thus, P_0 and P_1 collectively generate a sequence

$$x_1, y_1, x_2, y_2, \ldots$$

such that for each $i \geq 1$, actor y_i has co-starred with actress x_i, and no actor or actress appears more than once in the sequence. A player loses when it is their turn to move, but they cannot name a member of their set who satisfies the rules of the game.

Suppose you are given a specific pair of such sets X and Y, with complete information on who has appeared in a movie with whom. Give a polynomial-time algorithm that decides which of the two players can force a win in a particular instance of this game, and fully justify why your algorithm is correct.