Foundations of Algorithms, Spring 2022: Homework 6

Due: Wednesday, April 20, 11:59pm

Problem 1 (20 points: 14 for implementation / 6 for writeup)

As CEO, you are worried about company morale. Mood is contagious at your company. Every time one employee initiates a conversation with a second employee, it affects the mood of the second employee. The second employee takes on the mood of the first employee, adjusted by an additional effect due to the nature of the particular conversation. Unfortunately, the type of conversation never changes between any two employees. It's either always positive, always negative, or always neutral. Your concern is that there may exist a cycle of conversation that could cause some employees that are either part of this cycle or downstream from this cycle to spiral endlessly into fouler and fouler moods.

Design an $O(mn)$ algorithm that determines the number of employees at risk, given that there are n total employees and m conversational links between employees.

Problem 2 (25 points - 18 for coding, 7 for written response)

Given is a $d_1 \times d_2$ array *Chessboard* containing only 0's (empty squares) and 1's (occupied positions). Moreover, there is an unlimited number of dominos, i.e. 1×2 rectangle pieces. Your task is to decide if it is possible to cover all of the empty squares on the chess board by non-overlapping dominos (the dominos cannot cover any of the occupied positions and they cannot "stick out" of the chess board). Your algorithm should run in time $O((d_1 d_2)^3)$.

HINT: note that a chess board (or a checkers board) utilizes two different colors. What must be true about any 1×2 domino that is placed on the board?

Problem 3 (6 points)

The Edmonds-Karp algorithm refines the idea of Ford and Fulkerson in the following way: in every iteration, the algorithm chooses the augmenting path that uses the smallest number of edges (if there are more such paths, it chooses one arbitrarily). Find a graph for which in some iteration the Edmonds-Karp algorithm has to choose a path that uses a backward edge. Run the algorithm on your graph – more precisely, for every iteration (a) draw the residual graph, (b) show the augmenting path taken by the algorithm, and (c) the flow after adding the augmenting path.

Do not make your graph overly complicated.

Problem 4 (10 points)

This problem is about reducing one problem to another problem. Suppose we have two problems P and Q. And suppose we have an algorithm $\mathcal{A}_Q$ that solves problem Q. Can we then use the algorithm $\mathcal{A}_Q$ to solve problem P, with only little extra work? (In this context by "little" we mean that the number of extra steps is polynomially bounded.) If yes, we say that we reduced problem P to problem Q.

We will work with the following problems:

Problem P – Hamiltonian path: Given is an unweighted undirected graph G and two vertices s and t. Does there exist a path from s to t that goes through every vertex exactly once?

Problem Q_1 – Longest path: Given is an unweighted undirected graph G and two vertices s and t. Find the length of the longest path from s to t. The path can go through every vertex at most once.

Problem Q_2 – Shortest path with negative weights: Given is a weighted undirected graph G with arbitrary weights (positive, negative, or zeros) and two vertices s and t. Find the length of the shortest path from s to t. The path can go through every vertex at most once.

Here are your tasks:

a) Reduce problem P to problem Q_1. In particular, for a graph G and vertices s, t for which you want to find a Hamiltonian path, create an input graph G_1 and its vertices s_1 and t_1 that you'll use as the input for algorithm $\mathcal{A}_{Q_1}$. The algorithm will output ℓ, the length of the longest path from s_1 to t_1. Use ℓ to determine whether G has an s-t Hamiltonian path.

- Given G, s, and t, describe how to construct G_1, s_1, and t_1.
- Given ℓ for your G_1, s_1, and t_1, describe how to determine whether G has an s-t Hamiltonian path.

b) Reduce problem P to problem Q_2.

- Given G, s, and t, describe how to construct G_2, s_2, and t_2, the input for algorithm $\mathcal{A}_{Q_2}$. Make sure to specify the edge weights for G_2.
- Given ℓ, the length of the shortest s_2-t_2 path in your G_2, describe how to determine whether G has an s-t Hamiltonian path.

Note: you cannot modify the algorithms $\mathcal{A}_{Q_1}$ or $\mathcal{A}_{Q_2}$ – you do not know how they work! Imagine them being a part of a library for which you cannot see the source code. We often refer to such functions as *black boxes*.