CSCI261 Analysis of Algorithms, Homework 5

Due Friday, November 6, 2020, 11:59pm

Problem 1

Given is a undirected graph and two of its vertices s and t. Give an $O(n+m)$ algorithm that computes the number of shortest paths from s to t.

Problem 2

Given are n courses and for each course given are its prerequisites. Let P_i be the set of prerequisite courses for the i-th course and let $m = |P_1| + |P_2| + \ldots + |P_n|$. Give an $O(n+m)$ algorithm that finds the size of the longest prerequisite chain, i.e., the longest sequence of courses for which for every element in the sequence the previous element is its prerequisite. You may assume that the data is consistent, i.e., there are no "prerequisite loops."

Problem 3

Given is an $a \times b$ matrix where every cell corresponds to either an empty space or a wall. This matrix represents a maze and in it are also two things: things One and thing Two. The things start at different empty spaces. And, somehow, they got synchronized: They do exactly the same movements—when thing One goes west, so does thing Two; when thing One goes east, so does thing Two; and the same happens when they go south or north. They move at the same time: For example, if thing One is east of thing Two and they move east, both things move. The things cannot go to a location where there is an obstacle (or the other thing, unless it is moving away). For example, if there is an obstacle east of thing One but not of thing Two, if they go east only thing Two moves. Or, if east of thing One there is an obstacle and west of it there is thing Two, if they try to move east, they will stay in their current locations as thing One is blocked by the obstacle and thing Two by thing One.

Things One and Two want to get out of the house, and they want to do so at the very same time (that is, they want to leave the house with the same move). The things get out of the house if they move north in row 1, south in row a, west in column 1, or east in column b. This is proving to be very tricky. Please help them by designing an $O((ab)^2)$ algorithm that will tell them how to get out in the smallest number of moves (or it will tell them that the task is impossible).