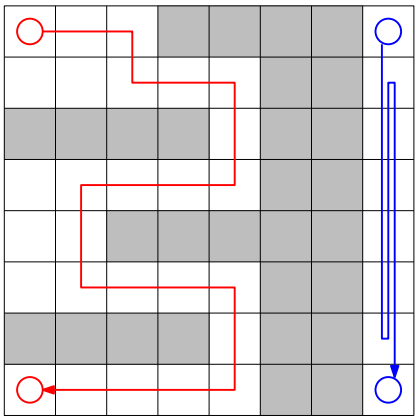
CS/ECE 374 A (Spring 2020)
Homework 8 (due Apr 2 Thursday at 10am)

Instructions: As in previous homeworks.

Problem 8.1: We are given an $n \times n$ grid, each cell of which is either black or white (black cells represent "obstacles" to avoid). Initially, Alice is at a given position (i_0, j_0) and Bob is at a given position (i'_0, j'_0). At each time step, Alice may stay at its cell or move to the cell one unit to the left, or to the right, or above, or below, provided that the new cell is white; similarly for Bob. In the interest of social distancing, we insist that at every time step t, Alice and Bob must have distance at least a given value B (i.e., the distance between Alice's position at time t and Bob's position at the same time t is at least B). Here, the *distance* between (i, j) and (i', j') is defined as $|i - i'| + |j - j'|$ (this is called the L_1 *distance*). At the final time step, Alice wants to be at a specified position (i_f, j_f) and Bob wants to be at a specified position (i'_f, j'_f).



$B = 5$

(a) (8.5 points) Design and analyze an algorithm to find a traveling schedule for Alice and Bob satisfying the above constraints, or decide that a solution does not exist, by constructing a graph and invoking a known graph algorithm. Aim for $O(n^4)$ running time.

(Hint: build a graph where each vertex is a pair of cells. Lab 9a, Problem 2 uses a vaguely similar idea.)

(b) (1.5 points) Suppose that we change the constraint to insist that at every time step, Alice and Bob must have distance *at most* B. Give a better upper bound on the running time when B is smaller than n.

Problem 8.2: We are given a directed graph $G = (V, E)$ with n vertices and m edges, where each vertex v has a value $\tau(v)$. We say that v is a *relative* of u iff there exists a vertex z such that there exists a path from z to u and a path from z to v.

Given a fixed vertex u, describe an efficient algorithm to output all relatives of u. Aim for $O(m+n)$ time.

(Hint: there are different ways to solve this problem, but one way is to define a new graph and just apply a known graph algorithm as a black box.)

Problem 8.3: We are given a directed graph $G = (V, E)$ with n vertices and m edges. We say that a vertex v is *dangerous* if there exists a walk that starts at v and has length (i.e., number of edges) at least n. (Recall that in a walk, repeated vertices are allowed.)

Give an efficient algorithm to compute the list of all dangerous vertices. Remember to prove correctness of your algorithm.

(Hint: apply the strongly connected components algorithm from class and consider the meta-graph. Come up with a characterization of when a vertex is dangerous in terms of the meta-graph. Aim for $O(m+n)$ time, although slower, correct solutions would still receive partial credit.)