HW 8

Due on **Wednesday, March 27, 2019 at 10am**

CS/ECE 374: Algorithms & Models of Computation, Spring 2019 Version: **1.33**

Submission instructions as in previous homeworks.

22 (100 PTS.) Graph Morphing

In the following, let G be a graph with n vertices and m edges.

22.A. (20 PTS.) Given an *undirected* graph G, and two vertices $s, t \in \mathsf{V}(\mathsf{G})$, describe a linear time algorithm that directs each edge of G so that the resulting directed graph is a DAG, s is a source of this DAG, and t is a sink.

22.B. (10 PTS.) For a connected undirected graph G, a ***bridge*** is an edge that its removal disconnects the graph. Using **DFS** describe a linear time algorithm that decides if a bridge exists, and if so it outputs it.

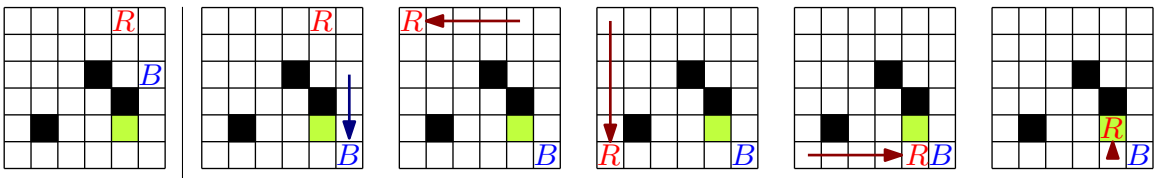
22.C. (20 PTS.) Prove that for an *undirected* connected graph G with n vertices and m edges, the edges can be directed so that the resulting graph is strongly connected $\implies$ there are no bridges in G. (The claim holds with $\iff$. This is the easier direction.)

22.D. (50 PTS.) Given an undirected connected graph G that has no bridges, describe how to modify the **DFS** algorithm so that it outputs an orientation of the edges of G so that the resulting graph is strongly connected. Prove the correctness of your algorithm.

23 (100 PTS.) Robot motion planning.

Consider the following puzzle. You are given a red and blue robots and an $n \times n$ grid – the robots are initially located in two prespecified grid cells. Some grid cells are walls (and marked as such). One cell is marked as the *target*.

In each turn, one of the robots has to move from its current position as far as possible upward, downward, right, or left, stopping just before either (i) hitting the outer wall of the grid, (ii) the other robot, or (iii) an obstacle. The task is get one of the robots to the target cell. Here is an example of an input, and a valid solution (which is not unique in this case):



23.A. (30 PTS.) You are given as input n, a list of obstacle locations, the target, and the initial locations of the robots. Describe and analyze an efficient algorithm to determine the minimum number of steps required to reach the target if possible.

In the monotone variant of the problem, one can choose the initial positions of the robots. In each step, one of the robots is being moved. One of the robots can move only right or down, and the other robot can move only up or left. As before a robot must stop just before hitting something.

23.B. (30 PTS.) Monotone variant longest path.

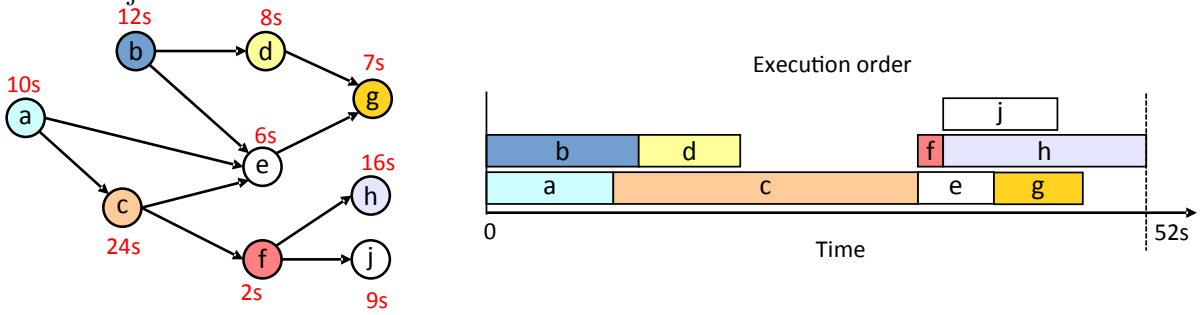
You are given as input n, a list of obstacle locations, and a target cell. Describe and analyze an algorithm to determine the initial locations of the robots that would maximize the number of turns required to reach the target (for the *monotone* variant). You can assume there is a solution for one of the possible initial configurations. The running time of your algorithm should be polynomial in n.

23.C. (40 PTS.) Monotone variant longest stuck.

You are given as input n, a list of obstacle locations, and a target cell. Describe and analyze an algorithm to determine the initial locations of the robots that would maximize the number of turns before either a robot reaches the target, or both robots get stuck (the running time of your algorithm should be polynomial in n).

24 (100 PTS.) Run DAG run!

Suppose we are given a DAG G with n vertices and m edges. The vertices represent jobs and whose edges represent precedence constraints. That is, each edge (u, v) indicates that job u must be completed before job v begins. Each node v also has a weight $T(v)$ indicating the time required to execute job v.



For all the following questions, you can assume that you have as many processors as you want (that work independently in parallel).

24.A. (30 PTS.) Describe an algorithm to determine the shortest interval of time in which all jobs in G can be executed.

24.B. (40 PTS.) Suppose the first job starts at time 0. Describe an algorithm to determine, for each vertex v, the earliest time when job v can begin.

24.C. (30 PTS.) Now describe an algorithm to determine, for each vertex v, the latest time when job v can begin without violating the precedence constraints or increasing the overall completion time (computed in part (A)), assuming that every job that has precedence on v starts at its earliest start time (i.e., computed in part (B)).