CSCI B505 Spring 20: Written Assignment 3

Due date: Sunday, April 19, 11:59pm

Submit your solution on Canvas. Handwriting is allowed, but you should write clearly and neatly: ambiguities will be treated not in your favor.

Problem 1 (30) You have an $n \times m$ grid. Initially you stay in cell $(1, 1)$, and you want to find a way to cell (n, m). When you stay in cell (i, j), you can move to a next cell, which can be any of $(i-1, j)$, $(i+1, j)$, $(i, j-1)$, or $(i, j+1)$, unless at least one of the following holds:

- The next cell is outside the grid (i.e. you are only allowed to move inside the grid).
- There is a wall between (i, j) and the next cell. Positions of all walls are known in advance.

You have to find any path between cells $(1, 1)$ and (n, m): a list of cells such that the first cell is $(1, 1)$, the last cell is (n, m), and it's possible to move between consecutive cells.

Please write a pseudo-code for an efficient algorithm and analyze its time complexity. Hint: the grid can be represented as an undirected graph, where each edge corresponds to a possible move.

Problem 2 (20+20) For each of the following statements either prove that it's true or show (and explain) a counterexample:

1. For any connected undirected graph G, for any starting vertex, there exists an order in which neighbors are processed, such that BFS and DFS visit vertices in the same order. E.g. for graph with edges $(0, 1)$ and $(0, 2)$, if both DFS and BFS start with vertex 0, and then process 0's neighbors in order $1, 2$, DFS and BFS visit vertices in the same order (first 0, then 1, then 2).
2. Let G be a directed graph, and let u be a vertex such that it has both incoming and outgoing edges. Assume that we run DFS on G. Then there always exists a tree edge such that u is one of its endpoint. Recall that there exists a tree edge between u and v when v was first visited (became GRAY) during the scan of neighbors of u.

Problem 3 (30) You are given a directed graph G. Let $T = \{v_1, \ldots, v_k\}$ be a fixed subset of vertices of the graph, and let s be some fixed vertex. You have to write an algorithm which answers the following question: is any of vertices from T reachable from s. In other words, you have to check if there exists $i \in [1 : k]$ such that there exists a path from s to v_i.

Please write a pseudo-code and analyze its time complexity.