EC330 Applied Algorithms and Data Structures for Engineers
Spring 2021

Homework 7

Out: April 21, 2021
Due: April 29, 2021

This homework has a written part and a programming part. Both are due at 11:59 pm on April 29. You should submit both parts on Gradescope.

This is an individual assignment. See course syllabus for policy on collaboration.

1. **Single-Source Shortest Path [20 pt]**
   Explain how to modify Bellman-Ford's algorithm to efficiently count the *number* of shortest paths (of equal length) from a source vertex to every other vertex in the graph. You can write the modified algorithm in pseudo code similar to the ones I gave in recent lectures, or write actual C++ code.

2. **All Paths from Source to Destination [20 pt]**
   Given a directed, acyclic graph *G* with *n* nodes, a source node *s* and a destination node *t* in *G*, describe an algorithm in pseudo code that can find all possible paths from *s* to *t*. What is the running time of your algorithm? Give your answer in $\boldsymbol{O}$ (hint: how many paths can there be in the worst case from *s* to *t*?).

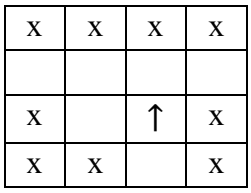
3. **Social Distancing [20 pt]**
   Consider a $n \times n$ grid-world where the value of cell (*i, j*) is 1 if there is a person standing in that cell and 0 otherwise. Suppose we use *Manhattan distance* to measure the distance between any two cells (*i, j*) and (*k, l*), i.e. the distance between (*i, j*) and (*k, l*) is $|i - k| + |j - l|$. Give a social distancing requirement *sd* (a positive integer), describe an algorithm that outputs all persons (in terms of cell coordinates) that violate the social distancing requirement, i.e. the shortest distance from another person in the grid is smaller than *sd*. You should start by describing how you construct a graph model for this problem, and then describe the graph algorithm for solving this problem. What is the running time of your algorithm? Give your answer in $\boldsymbol{O}$. Justify your answer (and state any assumptions you make).

4. **Roomba Escape! [40 pt]**

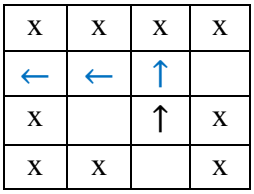
   Our Roomba is damaged and stuck in the middle of a maze and we need to find the quickest way out!
   The maze *M* is a $m \times n$ grid, represented by `vector<vector<bool>>`, where *M*[*i*][*j*] = 0 iff cell *M*[*i*][*j*] is free (otherwise it is blocked). The reason that our Roomba is stuck is because one of its wheels is broken: *the Roomba can only go straight or go*

left. *It cannot turn in place*. You are given the initial coordinate of the Roomba as well as the direction that it is facing.

Write an efficient algorithm to find the shortest escape route from the maze. You can assume that our Roomba can escape the maze if it reaches a free cell on the periphery of the maze. Below is an example.



The Roomba is initially in coordinate (2, 2) (the top left cell has a coordinate of (0, 0)) and facing north, as indicated by the arrow. The squares marked with 'x' are blocked, i.e. the Roomba cannot enter those squares. Observe that the Roomba can escape the maze by taking the following sequence of actions: *go_straight* → *go_left* → *go_straigh*t, as shown by blue arrows below. However, it cannot escape the maze with the sequence *go_straight* → *go_right* since it can only go straight or go left.



Implement the `escape_route` method in `roomba.cpp` which takes inputs the maze *M*, the initial coordinate of the Roomba, the initial direction that it is facing, and returns the *shortest* sequence of actions (stored in a vector with the first action at index 0) that will allow the Roomba to escape the maze. The method should return an empty vector if it is impossible to escape the maze. You can assume that the Roomba does not start on the periphery of the maze. Submit your `roomba.cpp` on Gradescope.

Hint: This is obviously a graph problem so you will want to think about how to construct the graph first, that is, what the vertices and edges are.