Algorithms
Graph Modeling Project

A detailed description of the maze problem including a figure is provided separately[1]. You may want to look at that first to get a sense of what this project involves! Your assignment is to (1) model the logic maze problem as an explicit graph (2) use an appropriate graph algorithm that we've encountered in class to solve the problem (3) code the algorithm (4) submit your program to Gradescope for verification and (5) write a **brief** report. **Friendly reminder: This is an individual assignment. The departmental collaboration policy (the document you signed at the start of the semester) will be enforced.**

1 Deliverables

The deliverables are (1) a report, submitted to Gradescope, and (2) a submission of your program to Gradescope, where it will be verified through several test cases. Your report must include the following items:

1.1 Problem Modeling [45 pts]

1. (15 points) Explain how you modeled the logic maze as an explicit graph.

   Vertices: Clearly describe what each vertex in your graph represents in relation to the logic maze.

   Edges: Explain the edges of your graph. Are they directed or undirected? What does an edge in your graph model represent in the context of the logic maze?

   Discuss any additional properties of the graph that are crucial for the unmodified[2] BFS to work correctly. This might include how you handled the start/end vertices (an ideal model would have one start vertex and one end vertex) or any other pertinent special features.

2. (10 points) Draw the entire resulting graph for the Spacewreck instance in Fig. 1. Clearly indicate the start vertex, the finish vertex, and the vertices along (a/the) shortest path in your drawing. All vertices and edges should be included, even if they are unreachable from the start vertex. The names of the vertices must be clearly readable, and the figure clean and easy to see.

   We **highly** recommend using graphviz or some other graph visualization software. Drawing this by hand might make you feel like you're assembling IKEA furniture without instructions (you have been warned).

3. (10 points) Argue for your model's correctness. Given any Spacewreck maze instance, convince us that your model represents it correctly, and when combined with a BFS, is guaranteed to find the shortest path if a path exists. This should be a concise yet complete argument (similar to a proof).

[1]The problem was taken from "MAD MAZES: Intriguing Mind Twisters for Puzzle Buffs, Game Nuts and Other Smart People," by Robert Abbott, Bob Adams, Inc. Publishers, 1990.

[2]Note that in its original form as discussed in class, BFS runs until it visits ALL vertices and edges in the graph. A simple modification allows us to terminate the traversal when a singular desired target vertex is reached. We permit this simple modification of BFS in your project.

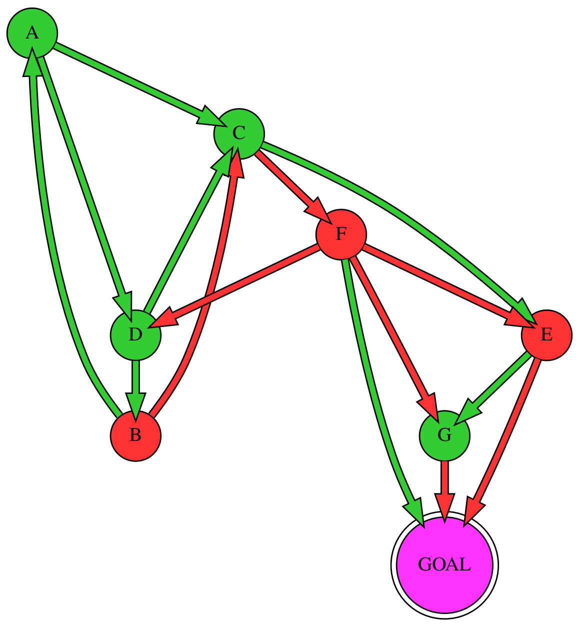

Figure 1: Spacewreck instance. Draw the entire resulting state graph.

4. (10 points) Analyze the space complexity of your graph model by comparing the vertices and edges to the logic maze's dimensions. If the maze comprises **n** rooms and **m** corridors, provide a precise formula for the number of vertices in relation to the size of the original maze, expressed in terms of **n** and/or **m**. Also, specify a tight upper bound for the number of edges, represented using $\mathcal{O}$ notation. This should be based on **n**, **m**, and/or the total vertex count. Provide a clear reasoning/rationale for your conclusions.

5. Code submission: please include your code in an appendix. It should solve the problem using the model and algorithm that you have mentioned in the previous section; and should be similar to your last submission on Gradescope. **If these three do not match, we may deduct (some or all) verification points.**

   As mentioned in class, you **must** modify the graph and not the BFS. Modifying BFS against instructions will result in a **minimum deduction of 25 points** from your overall score on this project. Using a graph traversal algorithm to create the graph model falls into the same category as modifying the algorithm. The model should not be created via a traversal, or the 25-point deduction will apply. The entire graph must be created explicitly (rather than implicitly) before the execution of the BFS. We realize this may not be the most efficient solution method (discuss in your complexity analysis if desired).

1.2 Results [55 pts]

There will be two assignments on Gradescope for verification.

- The Maze Verification Testing assignment is not worth any points on Canvas, but allows infinite submissions. Submit to this assignment first to make sure that you fix any submission-related bugs or typos (a separate document that details how to submit to Gradescope will be released when the autograder assignments are released). It also has some smaller test cases with timing results that you can use to get an idea of your program's run time on Gradescope compared to your computer. While this may have some edge cases, it will certainly not have all edge cases, nor will the inputs be of maximum size (hint: test your code with some edge cases even if you pass all these tests).

- The second assignment, Maze Verification, is worth 55 points. Expect extensive testing - edge cases, large maze sizes (up to 625 rooms), etc. There are three submission attempts. Passing 100% of test cases on the first submission will be awarded with 5 extra credit points. Passing 100% of test cases on the second submission will be awarded with 2.5 extra credit points. To be clear, this extra credit is only awarded if all tests are passed; failing (or exceeding the time limit) on even only one test = no extra credit. The third submission does not have any associated extra credit. Beyond the third submission, a 10% deduction will be applied for each additional submission attempt. **We will not be increasing the number of penalty-free submissions to more than three.**

2 Input Format

Your code will first read the input from a file, which is specified as follows:

- Line 1 contains two integers, $3 \leq n \leq 625$ and $0 \leq m \leq 10n$.
- Line 2 contains $n-1$ space-delimited items where the ith item represents the color of the ith vertex. The vertex with index n is the goal and has no color. There can be up to $n-1$ unique colors.
- Line 3 contains two integers s_1 and s_2, representing the index of the starting rooms of Captain Rocket and Lieutenant Lucky, respectively. You may assume they will not start at the goal.
- Each of the next m lines contains two integers a and b and item c, in that order, representing a corridor with color c from a to b. Note that $a,\ b \in [1, n]$.

The input below corresponds to the example in Fig. 1 where 1 is A, 2 is B, …, 7 is G and 8 is the Goal vertex. The two colors are R (red) and G (green).

```
8 15
G R G G R R G
1 2
1 3 G
1 4 G
2 1 G
2 3 R
3 5 G
3 6 R
4 2 G
4 3 G
5 7 G
5 8 R
6 4 R
6 5 R
6 7 R
```

```
6 8 G
7 8 R
```

3 Output Format

The output will describe a path with each line describing one step. Each line contains a character (either R or L), followed by a space, then an integer x, indicating that Rocket or Lucky respectively moved to room x.

One possible path may start off as follows

```
L 1   // Lucky goes to A
R 4   // Rocket goes to D
L 3   // Lucky goes to C
etc
```

Please output this path concatenated into a single line with all whitespace removed (e.g. L1R4L3).

The output should consist of the **shortest** path to solve the logic maze. If there are multiple shortest paths, output the path that occurs first lexicographically. For example, if we had two shortest paths `L1R4L5` and `L1L4R5`, L comes before R in the alphabet, so we would output the path `L1L4R5`. If we had two shortest paths `L1L4L5` and `L1L300L5`, we would output `L1L300L5`. If no valid sequence of moves exists from the start to the goal, then output "NO PATH".