Shasha Alvares
CS 590 Project 3
Due Date: May 7th, 2025

1. What type of graph would you use to model the problem input (detailed in the Section 3.1), and how would you construct this graph? (I.e., what do the vertices, edges, etc., correspond to?) Be specific here; we discussed a number of different types of graphs in class.

A directed unweighted graph is best to solve this problem.

Directed because the connections between nodes do not go both ways. For example, if the value at trampoline A is 3 and the location is (i,j), as long as it is within the boundaries of the game board, you can definitely move from trampoline A to trampoline B located at (i, j+3) if you are going East. However, unless the value of trampoline B is also 3 (which is not guaranteed), you cannot move freely back to trampoline A. That is why the graph is directed.

Unweighted because every move Jim makes requires equal effort. The problem does not say that it requires more energy/there is a penalty to make one move versus another, which is why the edges are unweighted.

I would construct a graph where there are row*column vertices, and each vertice represents the coordinates (i,j), where i cannot be greater than r and j cannot be greater than c. The value (representing how many vertices Jim can jump within the bounds of the gameboard) at every vertex is k where matrix[i][j] = k.

Every edge represents a possible trampoline jump of the fixed k value that Jim can jump from the current trampoline (i,j) as long as it is within the bounds of the game board. (i+k, j) means he moved South, (i-k, j) means he moved North, (i, j+k) means he moved East, and (i, j-k) means he moved West.

Shasha Alvares
CS 590 Project 3
Due Date: May 7th, 2025

2. What algorithm will you use to solve the problem? Be sure to describe not just the general algorithm you will use, but how you will identify the sequence of moves Jim must take in order to reach the goal.

The algorithm I will use to solve this problem is BFS. The goal of the problem is to find the shortest number of jumps from the starting point matrix[0][0] to the end point matrix[r-1][c-1]. Since each jump has the same cost which means our graph is unweighted, BFS guarantees being able to find the shortest number of jumps because that is a property of using BFS in an unweighted graph.

To identify the sequence of moves Jim will take to reach the goal, I would use a HashMap to keep track of the last trampoline coordinates visited from the current trampoline coordinates and use a set to keep track of trampolines that have been marked as visited during BFS, so we do not check them again. Once the goal has been reached I would backtrack to reconstruct the path from the goal coordinates to the starting coordinates.

To map the coordinates to the direction (N, S, E, or W) that was moved in, I would use another hashmap that stores the current trampoline coordinates and the direction that was moved in as a string (N, S, E, W) ) to get to this current trampoline. This will enable me to reconstruct the path from the end point to the start point. I would then just have to backtrack to output the path taken from the start to end coordinates in terms of the direction (N, S, E, W,) moved in.