Algorithms and Data Structures 1

Student names: Jacopo Ceccuti s215158
Hand-in: Super Mario Run

1 Exercise

1.1

The process to find the maximum number of nodes and edges is really similar so we will focus on one of the two. Let's count the nodes. Given a $k \times k$ matrix we have k nodes in the $(k-1)$'th row; at the end we can imagine a jump that brings Mario to the $(k-3)$'th row, where there are going to be $(k-1)$ nodes. If we go on counting in this way until the top of the matrix, we end up in the following situation where the division by 2 represents that only half of the height of the matrix is used for actual nodes because the rest is used as a "floor" where Mario can move:

$$n = k + ((k-1) + (k-1) + ... + (k-1)) \cdot \frac{k}{2}$$

The above equation can be simplified and written in big "O" notation to get: $O(k^2)$ A really similar process is done for counting the edges. We only need to keep in mind that there is always going to be one less edge than nodes in the final "count" since one egde connects two nodes.

1.2

To check if the graph is *valid* we can use DFS (version for directed graphs) and a list of visited vertices. The DFS algorithm will get as an argument a vertex and it will check if it's the goal field or not. When it finds the goal field it will return *true*, otherwise it just keeps track of the visited vertices and recursively call DFS on the next vertex. The time complexity (worst case) of this algorithm in big "O" notation is $O(n+m)$, where m is the number of edges and n is the number of nodes in the graph.

1.3

In order to find the shortest path from beginning to end, going through the mushroom, we can use a slightly modify version of BFS. The BFS algorithm works as known but it also implements an array of distances (with the source vertex's distance set to 0) and keeps track of the predecessors. During the normal path of BFS we check if the distance to a neighbor can be improved (is less) going through the current vertex, and in that case it's updated and the neighbour is added to the queue to continue to execution. Now we can evaluate the path backtracking on the predecessors creating the actual shortest path. Since we have used BFS the running, in big "O" notation, is $O(n+m)$.