**Question 1.1: Describe how you would frame the maze solver as a search problem.**

A maze solver can be described as a search problem by representing the maze a connection of nodes; these nodes represent each possible position of a player. Each node will have neighbourhood of nodes that it is connected to. We can specify what the start and end nodes are and begin searching from the start node. At each node we will check if it is the goal, if yes then goal found, if no we move the one of the nodes in the current node’s neighbourhood. Our path will be each node that we have visited.

**Question 1.2: Solve the maze using depth-first search.**

1. **Briefly outline the depth-first algorithm.**

The depth first algorithm consists of traversing a given path a far as it can possibly go, we use a stack data structure to store the nodes that will be traversed, the stack is defined to already contain the root node. Coupled with a visited list containing all nodes that have been visited; this is to avoid infinite loops in the search. We begin the search at our chosen root node by popping it off from the stack and marking it as our current node. We then identify the nodes that are connect to the current node and push them onto the stack. Once all neighbouring nodes have been identified we mark the current node as visited and pop the next node off the stack and mark it as our current node. With each new current node, we must check if it is our goal, if yes we will stop the search, if it is not the goal then we will continue searching until the stack is empty.

1. Calendar

   Description automatically generated

The path that depth first search has produced form the start point to end point using maze-Easy.txt as the maze.

1. Shape, arrow

   Description automatically generated

The path of depth first search coupled with statistics on maze-Easy.txt.