**CS 520: Project 1 - Ghosts in the Maze**

Submitted by:

Dheekshith Dev Manohar Mekala (dm1653)

Sai Surya Siva Prasad Raghavarapu (sr1789)



Computer Science Department - Rutgers University

# Declaration

We, Dheekshith Dev Manohar Mekala, and Sai Surya Siva Prasad Raghavarapu declare that this written submission represents my ideas in my own words where others’ ideas or words have been included; I have adequately cited and referred to the sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated any idea/data/fact/source in my submission. I fully understand that any violation of the above will cause disciplinary action by the institute and can also evoke penal action from the sources which have thus not been appropriately cited or from whom proper permission has not been obtained.

***Gray Ques 1:*** A\* search algorithm is the best algorithm for checking the existence of a path. A\* search algorithm is called the “Best-First Search” algorithm because it uses a heuristic function/value to estimate the overall work needed to reach the goal state. This way, instead of blindly searching/traversing the grid, A\*, with its heuristic value, can always know either how far or how close the search position is to its destination.

***Gray Ques 2:*** We don't always need to replan because when the opponents are chance players, there is a good guarantee that the opponent won't play optimally. If this is the case and also with known bounds for a problem - as in our case (ghosts in a grid) the only bounds would be either the agent gets killed, which will be equal to (-1), or he reaches the goal state (+1). With these bounds, we can implement Alpha-Beta Pruning on any expectiminimax algorithms. This essentially reduces a lot of replanning an agent would do in any other scenario.

***Gray Ques 3:*** If agent 3 predicts there is no successful path in its projected future, it could either re-iterate to see some more projected futures with a path or could simply keep moving away from all the nearby ghosts. It does not explicitly guarantee that success is impossible. Still, due to the probabilistic nature of the agent predicting futures, we can almost certainly say that there might not be a path to the goal state.

**Description of Agents/Maze:**

* After the grid is created, the ghosts won't spawn within the start node and goal node proximities.
* If the created grid doesn't have a path from start node to goal node, my program will scrap that grid and generate a new one until it finds an ideal grid that has a path available.

* Agent 1 - Agent 1 uses A\* search algorithm to go to goal state as fast as possible without avoiding any ghosts.

* Agent 2 - Agent 2 uses proximity and has a 2x2 grid proximity around him. So, if there are any ghosts within his proximity he will try to avoid that ghost by choosing a neighbour node with the highest ghost euclidean distance effectively averting from choosing a neighbour position that is closer to a ghost. Agent 2 if there is a ghost in his proximity will calculate f values of his neighbour node positions but will penalize the neighbour position with very high f value (ghost value along with g,h,f values) if that position is near to a ghost based on euclidean distance to that ghost. Agent 2 has a 2x2 grid proximity around him. So, if there are any ghosts within his proximity he will try to avoid that ghost by a choosing a neighbor node with the highest ghost euclidean distance effectively averting from choosing a neighbour position that is closer to a ghost'''

Agent 2 if there is a ghost in his proximity will calculate f values of his neighbour node positions but will penalize the neighbour position with very high f value (ghost value along with g,h,f values) if that position is near to a ghost based on euclidean distance to that ghost. Agent\_1 will use A\* search algorithm with the heuristic Euclidean Distance from any node to goal node as h value.

* Agent 3 - Agent 3 similar to agent 2 uses a proximity cloud around him and if he spots any ghost in that cloud, he will stop and do beam search (Hill-Climbing - Beam Search) (DFS) by releasing four mini agents in opposite directions and teleports to the nearest best position those mini agents found.

Here, mini agents act like agent 1 without trying to avoid any ghosts in the way. This is a way of penalizing. The pink path represented in the grid shows a mini agents path.

* For agent 4 we don't actually need to accentuate the computational complexity/barriers. We can simply make the agent 4 go in eight directions (diagonally) as opposed to cardinal directions and without the abilities of previous agents, his success rate will be a lot better.

* For agent 1 no matter how big the grid is or how many ghosts there are on the maze, A\* algorithm will always ensure a shortest path in any given n-dimensional maze. Dijsktra’s algorithm can also do the same but it takes more time to find the shortest path.

Links for Videos:

Agent 3:

<https://youtu.be/qXuJBlI-adg>

<https://youtu.be/zN-ewag2tt8>

Agent 2:

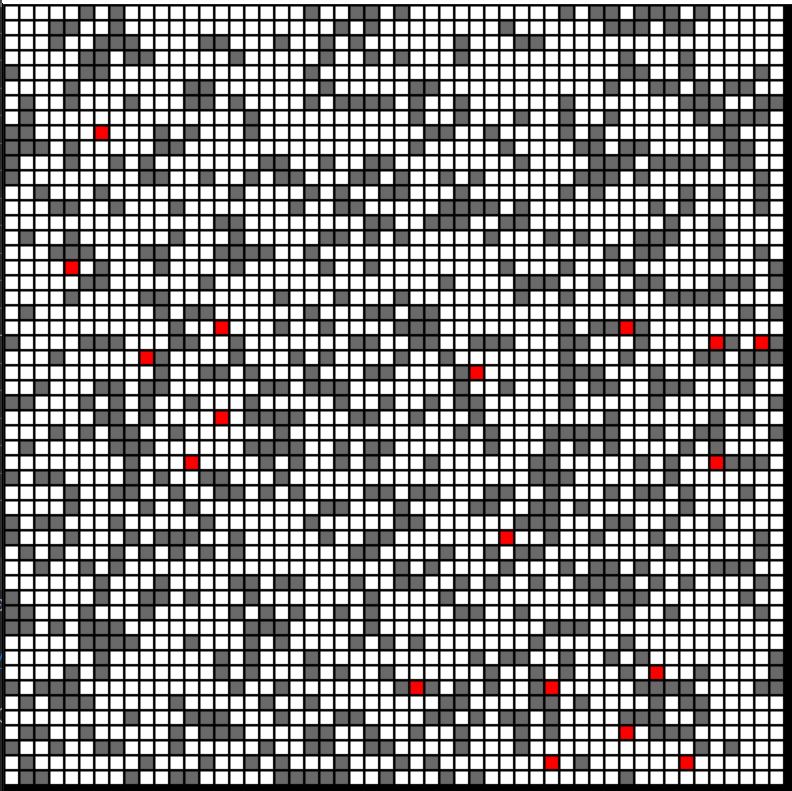
<https://youtu.be/Q7Ku9DfA_LE>

Agent 1:

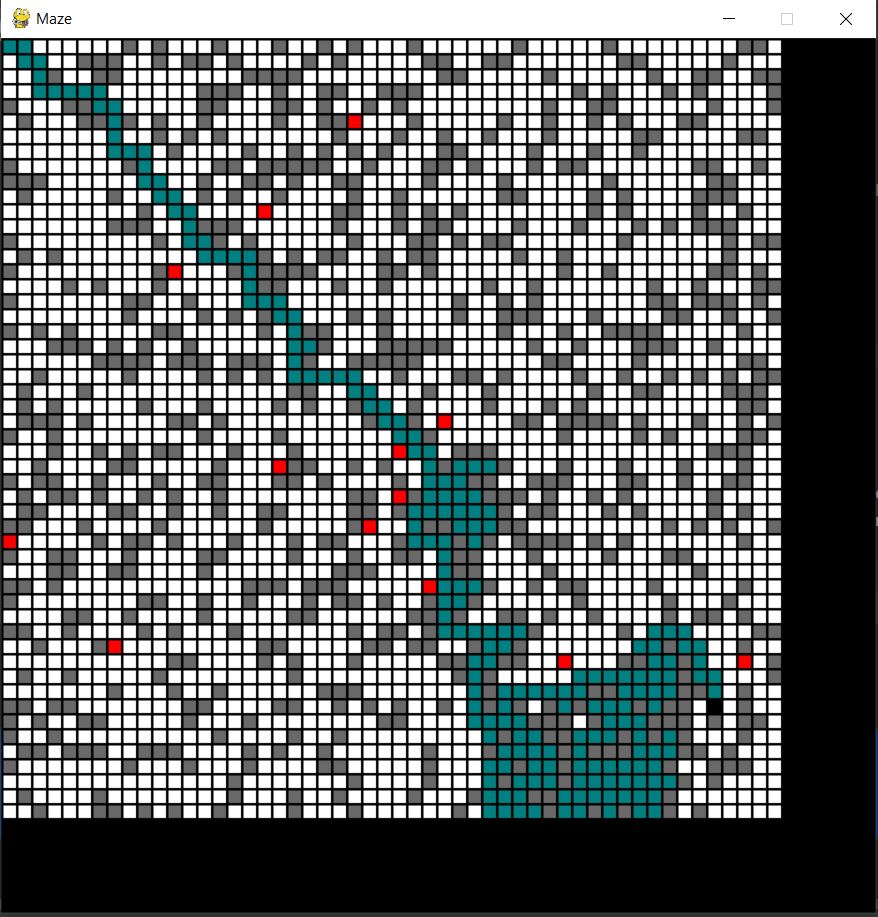
<https://youtu.be/FvDk99z-IsI>

<https://youtu.be/aXPX0OexPWo>

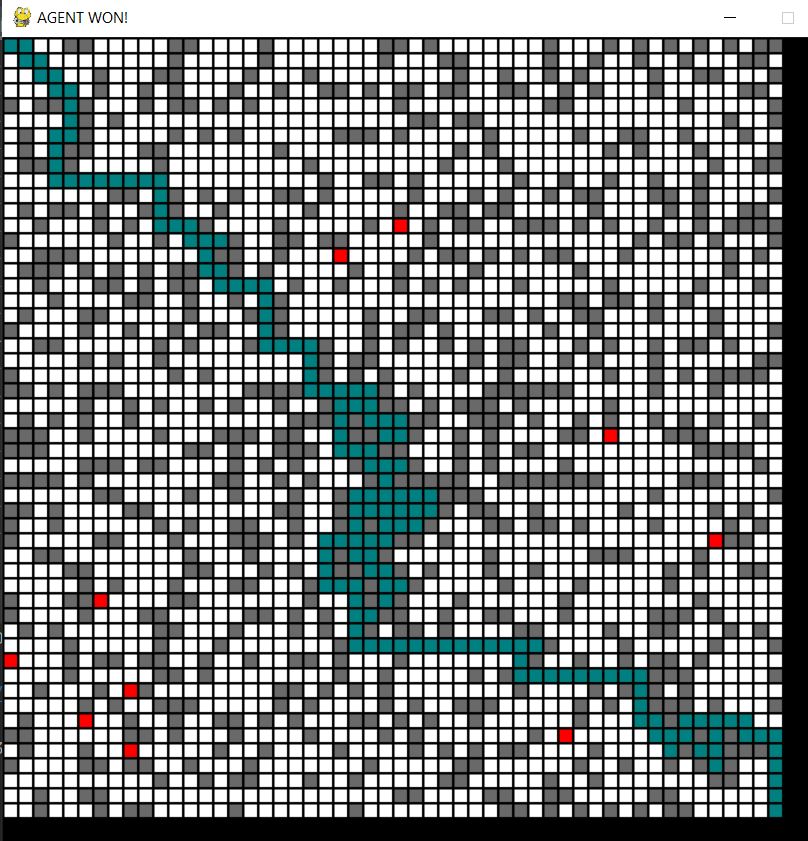
**The Grid:**



**Agent 1 (Death):**

****

**Agent 1:**

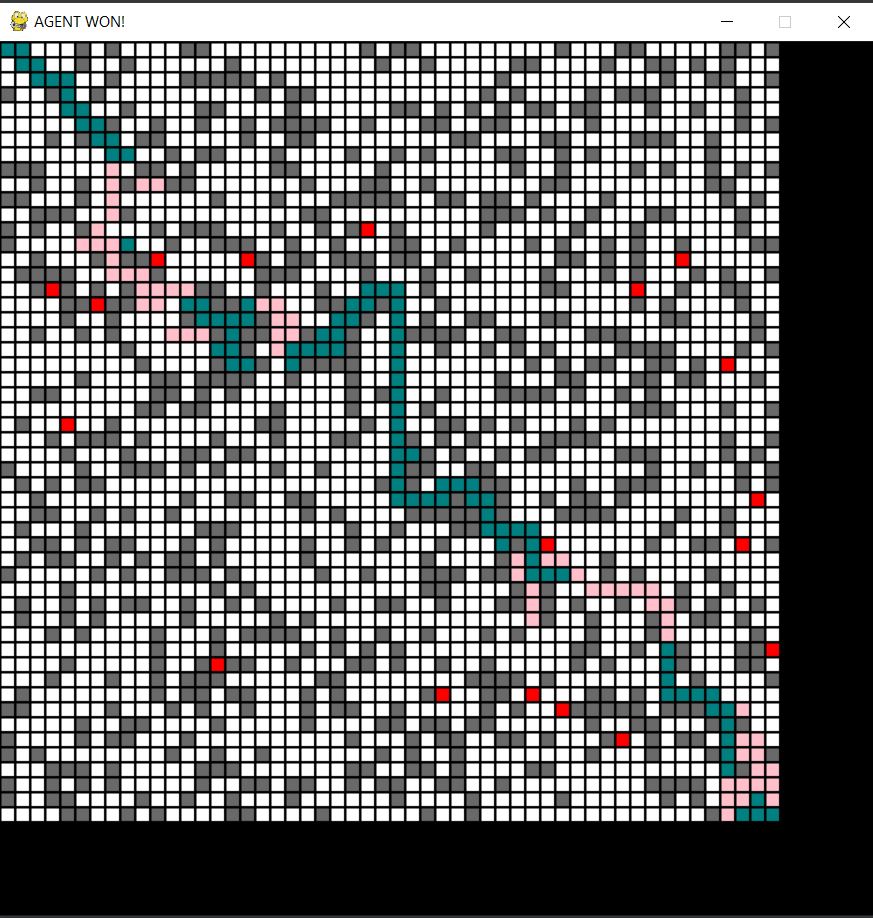
****

**Agent 2:**

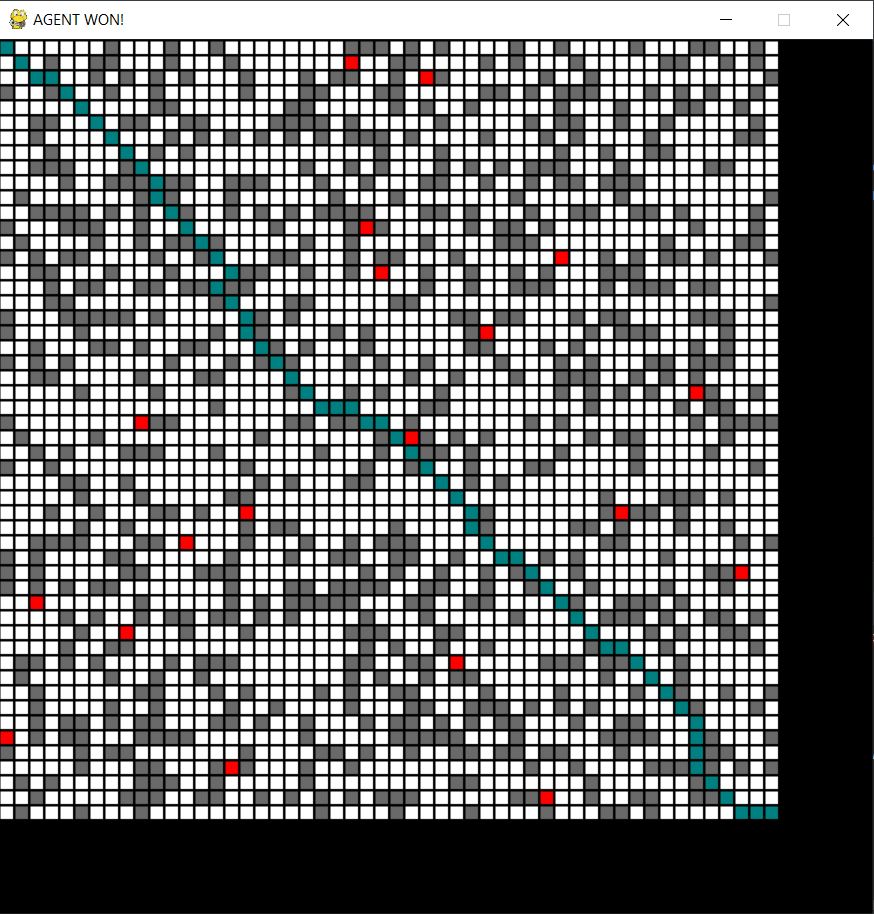
****

**Agent 3:**

(Pink path represents (Hill-climbing Beam Search (DFS)) by mini agents.

****

**Agent 4:**

****

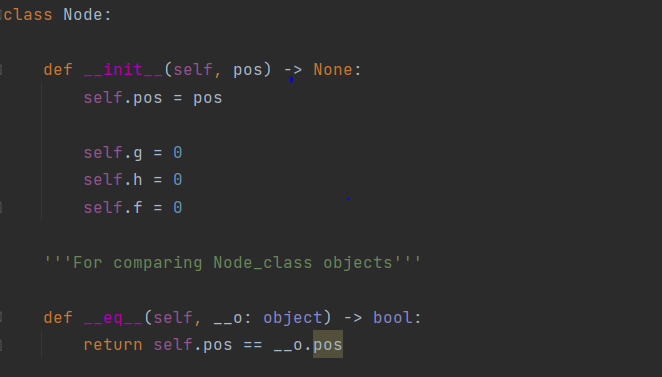
**Creation of the Maze:**

I created a numpy 2-dimensional matrix that randomly creates blocks based on Prof. Charles probability requirement. ‘1’ denotes an unblocked path. ‘0’ denotes a blocked path. ‘9’ denotes ghost positions.



**Node Class:**

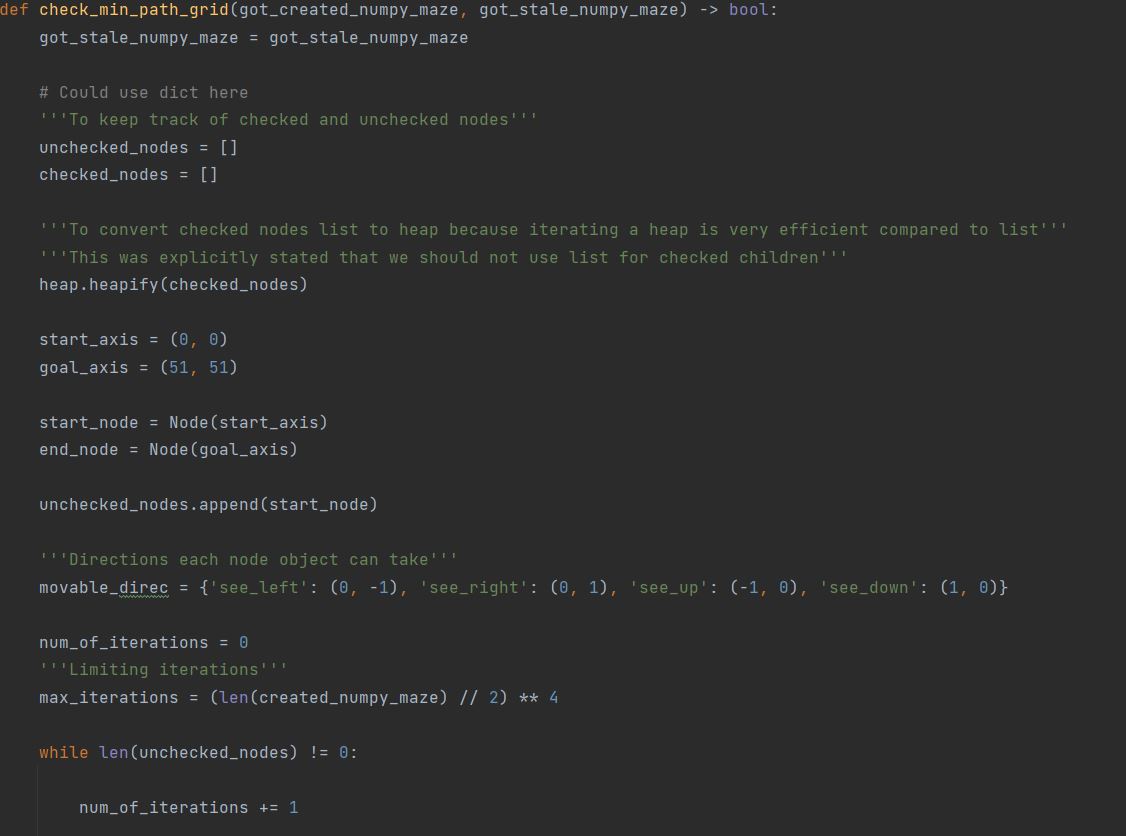
Node class because each position on the grid should have 3 values (g\_value, h\_value, f\_value) other than its position.



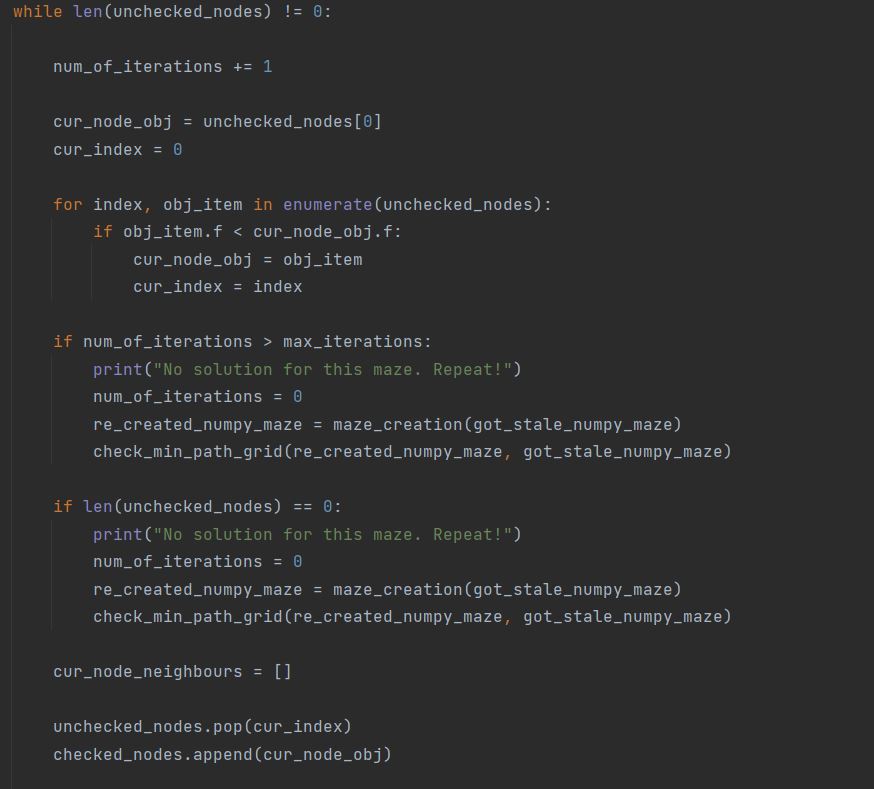
**Check Path Exists:**

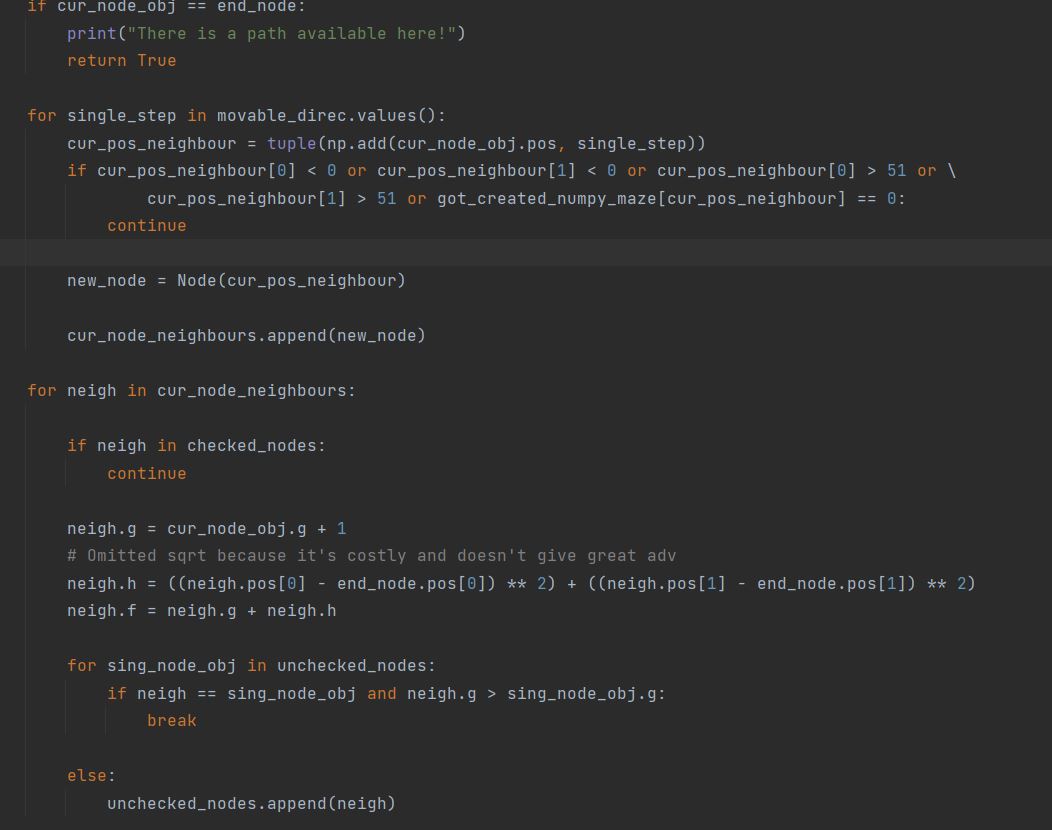
This function will check if a path exists or not from start position to goal position.

Function returns True if there is a path.



'This function will check the path using A\* search algorithm.





**The actual survival rate of agents in a 51x51 grid**.

**Survival of all agents as grid size increases**