**Artificial Intelligence and Neural Network Lab Project**

|  |  |  |
| --- | --- | --- |
| **Project Title** | **AI maze solver** | |
| **Section** | **Registration Number** | **Name** |
| C | 0105-BSCS-19 | Syed Muhammad Usman Ali Bukhari |
| C | 0036-BSCS-19 | Malik Muhammad Mueed Aslam |

**Installation Requirement**

* pygame library (import pygame)
* math (import math)
* random (import random)
* time (import time)
* copy (import copy)

**Why choose these algorithms (DFS, BFS, A\*)?**

In this maze problem, the solution is to find the goal node in the bottom right corner of the maze. Since the node to be searched is the deepest node so Depth-First Search Algorithm is more beneficial than Breadth-First Search. Other uninformed searches like UCS would have no effective benefit because the path cost of each step is the same everywhere. The reason why A\* search is better than all of the mentioned algorithms is that it only explores nodes near to goal node thus providing the shortest path.

**Step by step visualization of the Project:**

|  |  |  |
| --- | --- | --- |
| **Step 1:** Run the project a maze will be generated. | **Step 2:** Choose the algorithm you want to apply as follows:  BFS (Press 1), A\*(Press 2),  DFS (Press 3) | **Step 3:** Press space to simulate the algorithm traversing the graph. |

**Challenges**

Implementing the A\* algorithm was a bit tricky. I spent hours debugging to make my algorithm work properly.

The most difficult part of the project was creating the GUI and visualizing the maze.

**Limitation of project**

The project generates a random maze each time as a result the A\* algorithm is unable to find a path in some of the random mazes