# 1. Project team

|  |  |  |  |
| --- | --- | --- | --- |
| Team leader | [20P3844] | [Youssef Emad] | [20P3844@eng.asu.edu.eg] |
| Team member | [18P7231] | [Hamza Abdellah] | [18P7231@eng.asu.edu.eg] |
| Team member | [18P8121] | [Basel Mohamed] | [18P8121@eng.asu.edu.eg] |
| Team member | [18P9203] | [Mostafa Essam] | [18P9203@eng.asu.edu.eg] |
| Team member | [20P8736] | [Heba Hesham] | [20P8736@eng.asu.edu.eg] |

# 2. Introduction and Background

Entertainment: The project has been undertaken as part of an effort to develop a fun and challenging game or puzzle that involves maze-solving. The maze solver application can be used as a tool to generate and solve these mazes, providing players with an interesting and engaging experience.

## 2.1 Describe an opportunity or problem that the project is to address.

* The maze solver application can be used to solve a variety of maze-related problems, such as finding the shortest path through a maze, navigating through a maze, or testing the efficiency of maze-solving algorithms. It can also be used for educational purposes to teach students about graph theory and search algorithms such as BFS,DFS and A\*.
* One specific opportunity for the maze solver application is in the field of robotics. Robots are often used in environments that require navigation through complex mazes, such as in search and rescue operations, exploration of unknown territories, or inspection of hazardous areas. The maze solver application can be used to test and optimize the navigation algorithms of these robots, ensuring that they can efficiently and safely navigate through the maze-like environments they are designed to operate in.
* Another potential use case for the maze solver application is in game development. Maze-solving algorithms can be used to generate procedural mazes in games, which can provide players with a challenging and unpredictable experience. The maze solver application can be used to test and optimize these algorithms, ensuring that the generated mazes are interesting and solvable.

# 3. Objectives

* Developing efficient maze-solving algorithms: One of the primary objectives of a maze solver project is to develop efficient algorithms that can solve mazes quickly and accurately. This may involve exploring different search algorithms such as Breadth-First Search (BFS), Depth-First Search (DFS), or A\* search, and optimizing these algorithms to improve their performance.
* Generating solvable mazes: Another objective of a maze solver project is generating mazes that can be solved using the developed algorithms. This may involve developing algorithms that can generate random mazes of different sizes and complexities, while ensuring that the mazes are solvable.
* Testing and validating maze-solving algorithms: The maze solver project also be aimed at testing and validating the developed algorithms on different types of mazes. This may involve creating a large database of mazes with varying sizes and complexities, and testing the algorithms on these mazes to ensure that they can solve them efficiently.
* Developing a user-friendly interface: Another objective of a maze solver project may be to develop a user-friendly interface that allows users to input mazes and visualize the maze-solving process. This may involve developing a graphical user interface (GUI) that allows users to input mazes using a mouse or keyboard, and displays the maze-solving process in real-time.

# 4. System overview

## 4.1. pyamaze Module

### 4.1.1 Class COLOR(Enum):

*This class is created to use the Tkinter colors easily.  
Each COLOR object has two color values.  
The first two objects (dark and light) are for theme and the two color  
values represent the Canvas color and the Maze Line color respectively.  
The rest of the colors are for Agents.  
The first value is the color of the Agent and the second is the color of  
its footprint*

### 4.1.2 Class agent:

*The agents can be placed on the maze.  
They can represent the virtual object just to indcate the cell selected in Maze.  
Or they can be the physical agents (like robots)  
They can have two shapes (square or arrow)*def \_\_init\_\_(self, parentMaze, x=None, y=None, shape='square', goal=None, filled=False, footprints=False,  
 color: COLOR = COLOR.blue):  
 *parentmaze--> The maze on which agent is placed.  
 x,y--> Position of the agent i.e. cell inside which agent will be placed  
 Default value is the lower right corner of the Maze  
 shape--> square or arrow (as string)  
 goal--> Default value is the goal of the Maze  
 filled--> For square shape, filled=False is a smaller square  
 While filled =True is a biiger square filled in complete Cell  
 This option doesn't matter for arrow shape.  
 footprints--> When the aganet will move to some other cell, its footprints  
 on the previous cell can be placed by making this True  
 color--> Color of the agent.  
  
 \_orient--> You don't need to pass this  
 It is used with arrow shape agent to shows it turning  
 position--> You don't need to pass this  
 This is the cell (x,y)  
 \_head--> You don't need to pass this  
 It is actually the agent.  
 \_body--> You don't need to pass this  
 Tracks the body of the agent (the previous positions of it)*

### 4.1.3 Class textLabel:

*This class is to create Text Label to show different results on the window.*

def \_\_init\_\_(self, parentMaze, title, value):

*parentmaze--> The maze on which Label will be displayed.  
 title--> The title of the value to be displayed  
 value--> The value to be displayed*

### 4.1.4 Class maze:

*This is the main class to create maze.*

def \_\_init\_\_(self, rows=10, cols=10):  
 *rows--> No. of rows of the maze  
 cols--> No. of columns of the maze  
 Need to pass just the two arguments. The rest will be assigned automatically*

*maze map--> Will be set to a Dicationary. Keys will be cells and values will be another dictionary with keys=['E','W','N','S'] for East West North South and values will be 0 or 1. 0 means that direction (EWNS) is blocked. 1 means that direction is open.  
 grid--> A list of all cells  
 path--> Shortest path from start(bottom right) to goal(by default top left)  
 It will be a dictionary  
 \_win,\_cell\_width,\_canvas --> \_win and )canvas are for Tkinter window and canvas*

*\_cell\_width is cell width calculated automatically  
 \_agents--> A list of aganets on the maze  
markedCells--> Will be used to mark some particular cell during  
 path trace by the agent.*

#### 4.1.4.1 Class maze: Functions

def \_Open\_‘Direction’ (self, x, y): To remove the Wall of the cell in direction selected

##### def CreateMaze

def CreateMaze(self, x=1, y=1, pattern=None, loopPercent=0, saveMaze=False, loadMaze=None,  
 theme: COLOR = COLOR.dark):  
 *'''  
 One very important function to create a Random Maze  
 pattern--> It can be 'v' for vertical or 'h' for horizontal  
 Just the visual look of the maze will be more vertical/horizontal  
 passages will be there.  
 loopPercent--> 0 means there will be just one path from start to goal (perfect maze)  
 Higher value means there will be multiple paths (loops)  
 Higher the value (max 100) more will be the loops  
 saveMaze--> To save the generated Maze as CSV file for future reference.  
 loadMaze--> Provide the CSV file to generate a desried maze  
 theme--> Dark or Light*

##### def tracePath

def tracePath(self, d, kill=False, delay=300, showMarked=False):  
 *'''  
 A method to trace path by agent  
 You can provide more than one agent/path details  
 '''*

##### def run(self):

def run(self):  
 *'''  
 Finally to run the Tkinter Main Loop  
 '''*

## 4.2 Algorithms development:

### 4.2.1 DFS: Stack used to store cells

A screenshot of a computer

Description automatically generated with medium confidence

Figure 1: DSF--> pseudocode

### 4.2.2 BFS: Queue used to store cells

A screenshot of a computer

Description automatically generated with medium confidence

Figure 2: BFS--> pseudocode

### 4.2.3 A\*: Priority Queue (Module Queue) to store cells

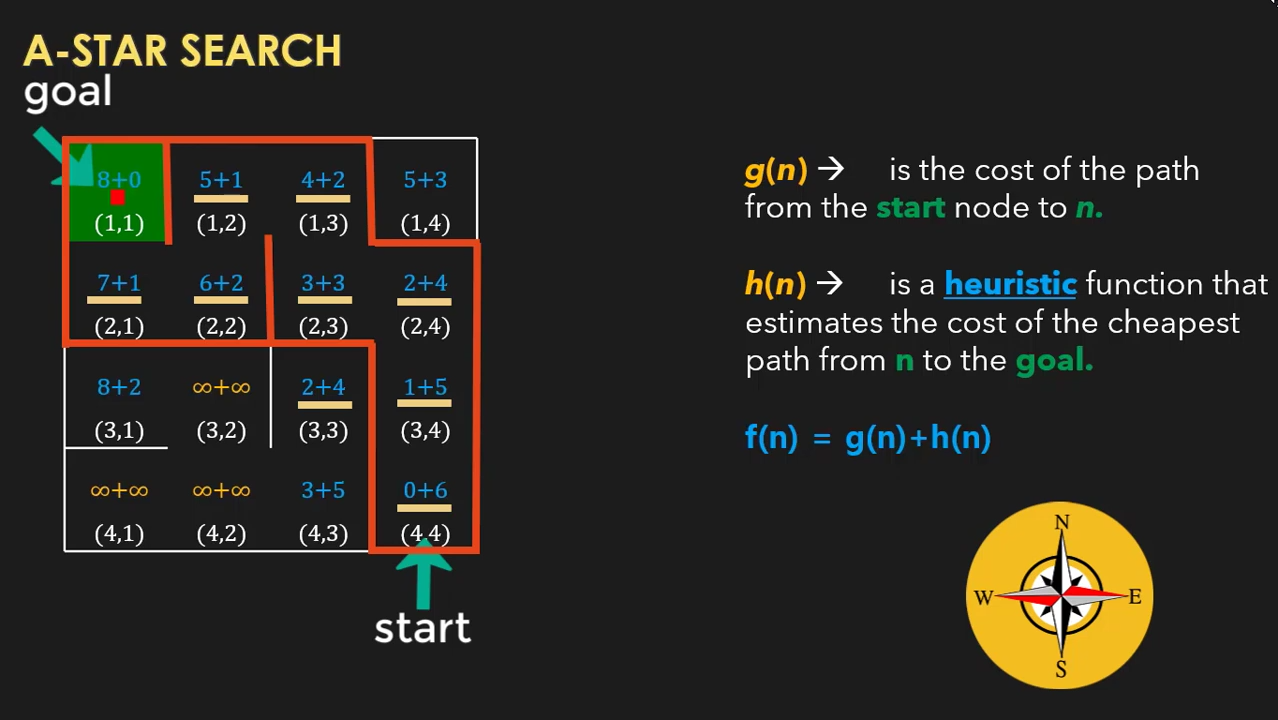


Figure 3: A\*--> Function f(n)

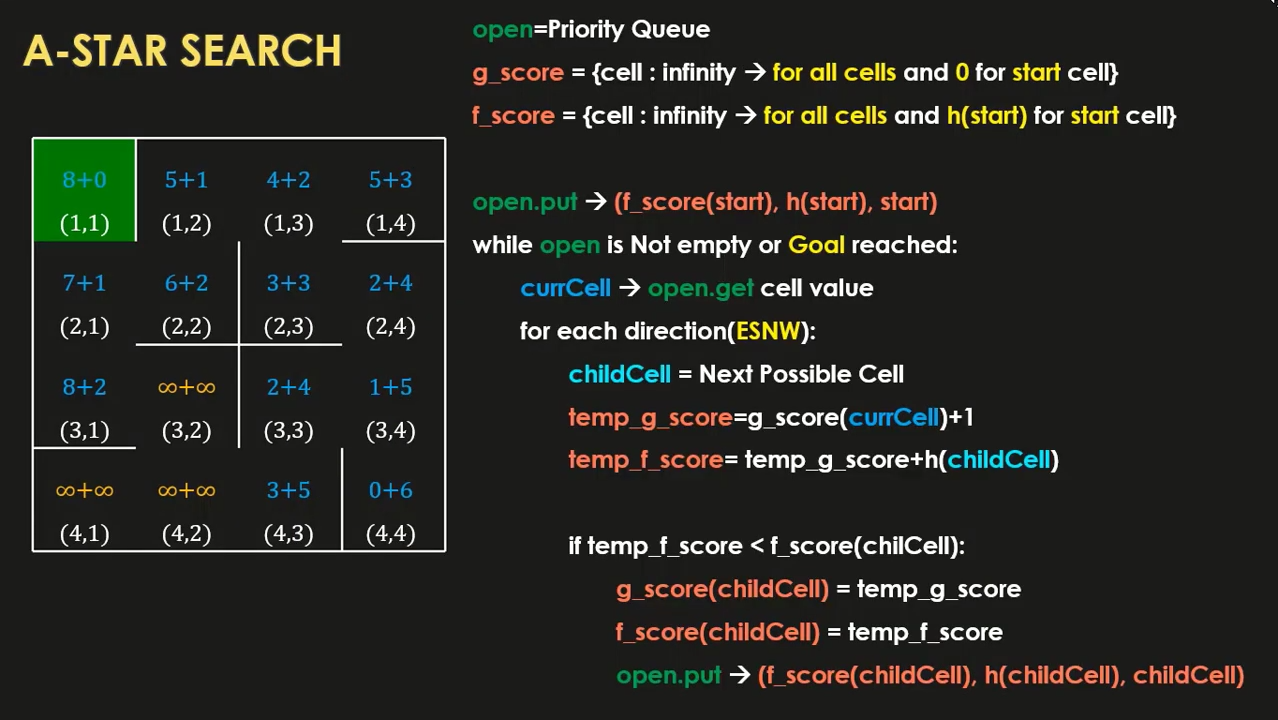


Figure 4: A\*--> pseudocode

## 4.3 Testing and Validation

To ensure that our program is functioning correctly, we employed the "load maze" function from the "create maze" class to load pre-existing maze datasets in CSV format. These datasets contain known paths from the starting point to the goal, which we use to validate the output generated by our program's algorithm. However, we understand that testing with a single dataset is not sufficient to ensure that our program works correctly under all scenarios. Therefore, we have also tested our program with a variety of different maze configurations and path lengths to ensure its reliability and robustness across a range of scenarios. By doing so, we can identify any limitations or weaknesses in our algorithm and make the necessary improvements to ensure optimal performance.

# 5. Methods

1. Searching Algorithms: searching algorithms used to solve the maze, by finding the shortest path from the starting point to the end point.

* Depth-First Search (DFS)
* Breadth-First Search (BFS)
* A\* Search Algorithm

1. Recursive Algorithms: used to generate a maze

* Recursive Backtracking Algorithm

# 6. Results and Evaluation

## 6.1 indicators

1. Accuracy
2. Speed
3. Efficiency

## 6.2 Evaluation criteria

1. Performance
2. Extensibility

## 6.3 Samples of the output

1. The number of mazes solved correctly by each algorithm (BFS, DFS, and A\*).

All of mazes Tested Solved Correctly

1. The Path Length of each algorithm to reach the end point.

* DFS (cyan path): 55
* BFS (yellow path): 49
* A-Star (blue arrow path): 49

1. The Search Length of each algorithm to reach the end point.

* DFS (cyan path): 56
* BFS (yellow path): 600
* A-Star (blue arrow path): 56

1. The time taken by each algorithm to solve different mazes.

* DFS (cyan path): 0.131789 Sec
* BFS (yellow path): 6.28055 Sec
* A-Star (blue arrow path): 1.0809 Sec

A screen shot of a computer screen

Description automatically generated with low confidence

Figure 5: Comparison between all Algorithems

**Grading Criteria**

You should follow the minimum requirements of the project and the grading is based on the discussion as a group as well as each team member in the group. The grading distribution can be follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **Point** | **Out of** | **Grade** | **Notes** |
| Does the report follow the required outline? | **3** |  |  |
| Application/System is running & all required modules are implemented | **3** |  |  |
| Is the dataset presented as a benchmark or collected? | **3** |  |  |
| Does the model/system clearly specify the loss/metric used to evaluate the quality of results? | **3** |  |  |
| Is the final model used to solve the problem obtained by selecting one of at least 2 candidate models? | **3** |  |  |
| Quality of discussion/conclusions. | **5** |  |  |

**Approval Signatures**

|  |  |  |
| --- | --- | --- |
| Supervisor |  | Teaching Assistance |
| Associate Professor / Walaa H. Elashmawi |  | Eng. Ziad Elgayar |

**Grade:**

20