CSC 361

Project

Marwan Alobathani

**Introduction**

Mobile robots are increasingly being employed in many automated environments. Potential applications of mobile robots include a wide range such as service robots for elderly persons, automated guided vehicles for transferring goods in a factory, unmanned bomb disposal robots and planet exploration robots, etc. In this project I will implement four strategies. Breadth first search (BFS), Depth first search (DFS), Greedy Best-first search, AStar.

1. **Problem formulation:**

In this project, we see that the problem of Robot path planning for maze navigation is time and cost, and each strategy has a different time and cost, but we the best strategy is A\* that take less time and less cost

1. **Design**

I create three classes that will solve the path robot problem by using BFS, DFS, greedy, A\*.

**byte**[][] *pixels is 2D array matrix*

*pixels*[1][1] *is the initial state*

*pixels*[28][29] is the goal to reach

first class is Maze that will take nodes (i,j) and the node go up if we decrement i and go down if we increment i and go left if we decrement j and go right if we increment j.

second class is MazePanel its got the run, paint, and other methods used for recursion

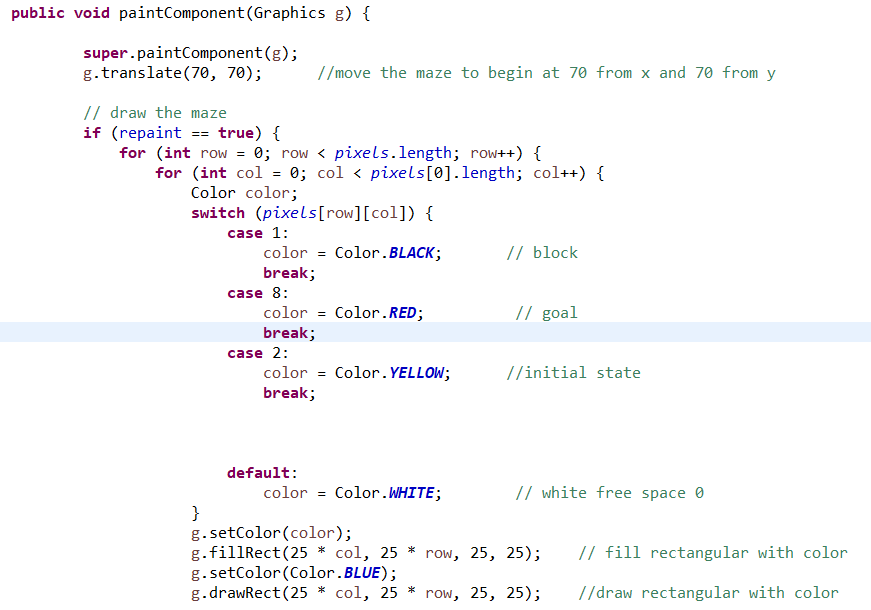
we created a 2D array by reading from picture that will convert it to binary 0 1. 0 is the path and 1 is the wall

third class is the main that will run the project by GUI (Graphical user interface)

1. **Implementation**

**PaintComponent():**

after I convert the image into a 2D binary array matrix, we will draw the maze with different colors. If the pixels row and column are equal to 1 that means it is a wall; then we will draw it black. If pixels is the goal state, then will draw the node red, if it is the initial state, then will draw it yellow, and the others pixels will draw it to white, which is the path.

****

**isInMaze:**

**This method will check if pixels[i][j] in the maze or not, and it will return true if is and false otherwise.**

**public** **boolean** isInMaze(**int** i, **int** j) {

**if** (i >= 0 && i <*pixels*.length && j >= 0 && j < *pixels*.length) {

**return** **true**;

} **else** {

**return** **false**;

}

}

**isClear:**

return true if the node is not a wall and not visited

**public** **boolean** isClear(**int** i, **int** j) {

**assert** (isInMaze(i, j));

**return** (*pixels*[i][j] != ***X*** && *pixels*[i][j] != ***V***);

}

**Mark:**

**It will mark the visited nodes with the color green**

**public** **int** mark(**int** i, **int** j, **byte** value) {

**assert** (isInMaze(i, j)); // it is used for test.if the condition is false it will throw an error named AssertionError.

**int** temp = *pixels*[i][j];

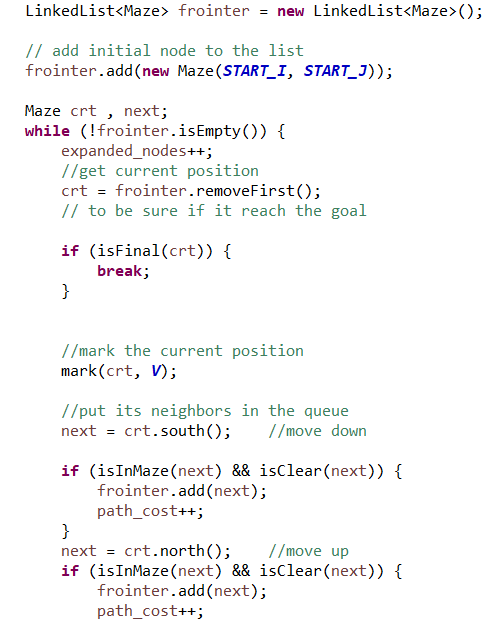
*pixels*[i][j] = value;

**return** temp;

}

**BFS():**

I use in BFS linked list and add the initial node to the list and by using while loop it will check if the list isn’t empty then it will get current position and it will check in every child node until it will reach the goal



**DFS():**

Same as BFS but in DFS, we use a stack to go deeper until it reaches the goal.

**Heuristic():**

It uses to calculate the manhattan distance that it will compute the manhattan distance for the current state and return the sum of the current node and the end node.

1. **Results:**

1-. Breadth first search (BFS)

**صورة تحتوي على عنصر, ساعة حائط, كمبيوتر

تم إنشاء الوصف تلقائياً**

BFS it is visited all node until reaching the goal

2- Depth first search (DFS)

**صورة تحتوي على عنصر, ساعة حائط, كمبيوتر

تم إنشاء الوصف تلقائياً**

We see in DFS it moves deeper until finding the black node (wall), then it will move another way. We use the method isClear to check if the node is not a wall and has not visited before.

3-Greedy Best-first search

صورة تحتوي على عنصر, ساعة حائط, كمبيوتر, كمبيوتر محمول

تم إنشاء الوصف تلقائياً

We see in greedy it is go to node that has a lower cost

1. **Discussion:**

we see in BFS take a lot of time to reach the goal because it is looking everywhere, but in DFS, it takes less time than BFS because it is moved until finding black node then do back trick to move another way but the best way we find it is greedy because it moves to the least cost and takes less time.

1. **Conclusion:**

In conclusion we learned about search strategies Breadth first search (BFS), Depth first search (DFS), Greedy Best-first search we apply this knowledge to Robot path planning for maze navigation. In this programming project we built simple program using java language