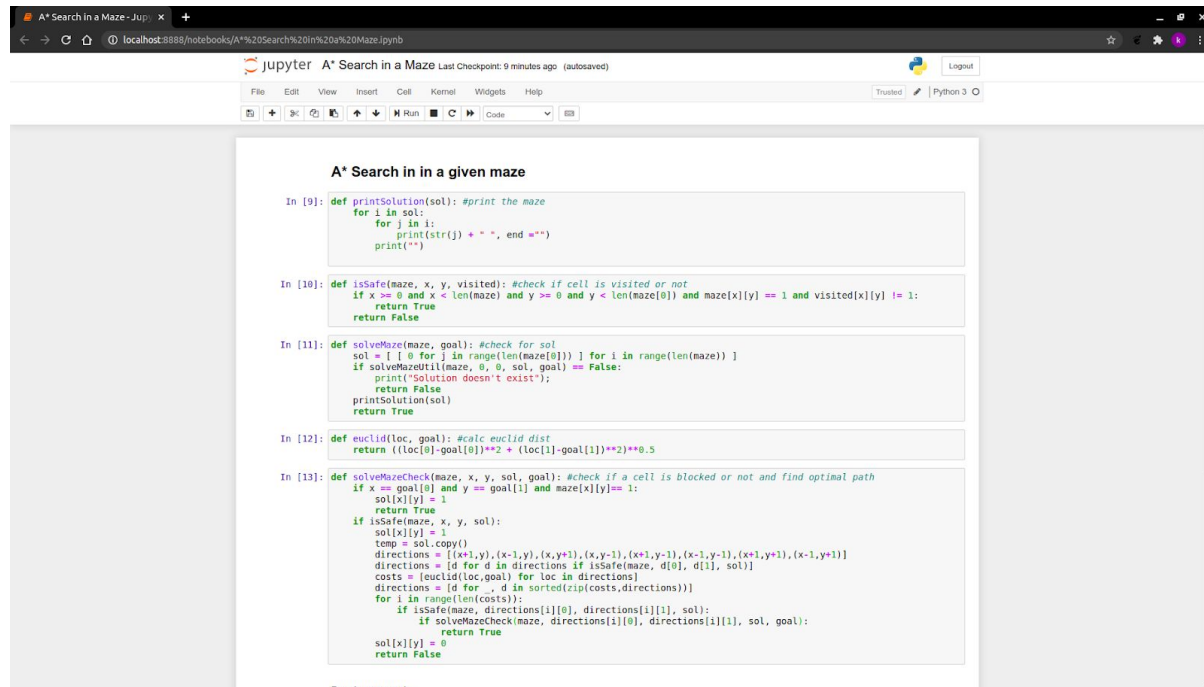


Question: Implement A* algorithm by considering Heuristic function as Euclidean distance to solve the following Maze Structure for the Source and destination provided.

Code and Output:

A screenshot of a Jupyter Notebook titled "A* Search in a Maze". The notebook is running on a local host. The code defines several functions: `printSolution` to print the solution path, `isSafe` to check if a cell is within bounds and not visited, `solveMaze` to find a path from source to goal, and `euclid` to calculate the Euclidean distance. The code is as follows:

```
A* Search in a given maze

In [9]: def printSolution(sol): #print the maze
        for i in sol:
            for j in i:
                print(str(j) + " ", end=" ")
            print("")

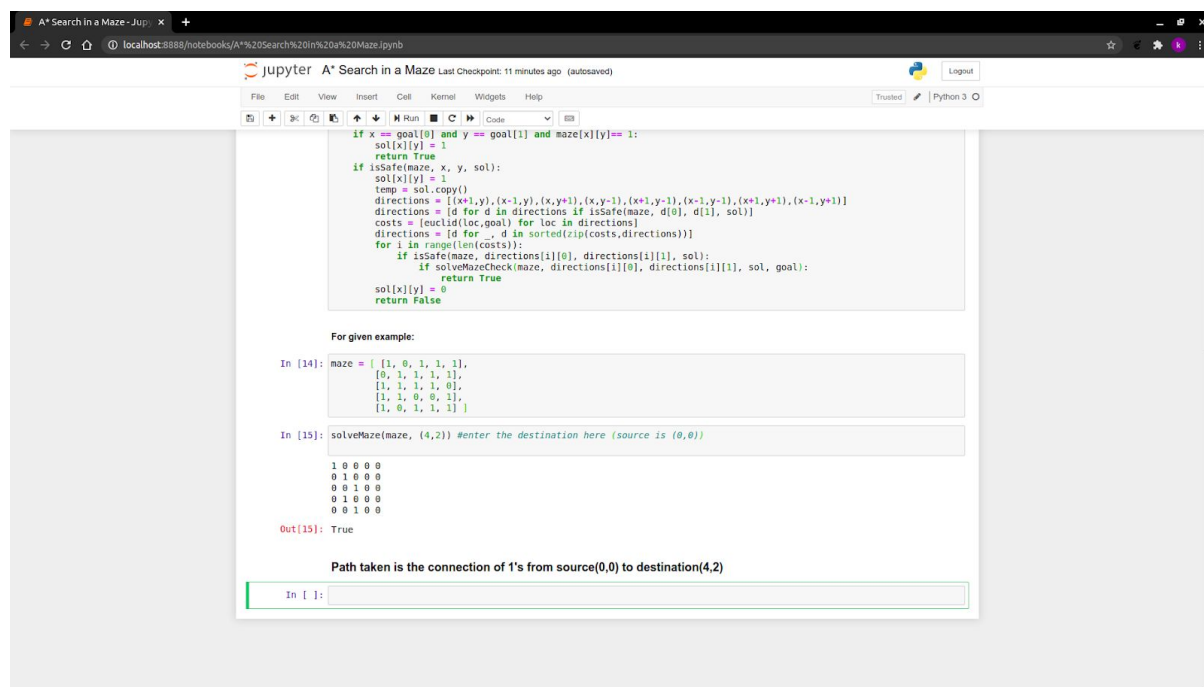
In [10]: def isSafe(maze, x, y, visited): #check if cell is visited or not
         if x >= 0 and x < len(maze) and y >= 0 and y < len(maze[0]) and maze[x][y] == 1 and visited[x][y] != 1:
             return True
         return False

In [11]: def solveMaze(maze, goal): #check for sol
         sol = [ [ 0 for j in range(len(maze[0])) ] for i in range(len(maze)) ]
         if solveMazeUtil(maze, 0, 0, sol, goal) == False:
             print("Solution doesn't exist");
             return False
         printSolution(sol)
         return True

In [12]: def euclid(loc, goal): #calc euclid dist
         return ((loc[0]-goal[0])**2 + (loc[1]-goal[1])**2)**0.5

In [13]: def solveMazeCheck(maze, x, y, sol, goal): #check if a cell is blocked or not and find optimal path
         if x == goal[0] and y == goal[1] and maze[x][y] == 1:
             sol[x][y] = 1
             return True
         if isSafe(maze, x, y, sol):
             sol[x][y] = 1
             temp = sol.copy()
             directions = [(x+1,y),(x-1,y),(x,y+1),(x,y-1),(x+1,y-1),(x-1,y+1),(x+1,y+1),(x-1,y-1)]
             directions = [d for d in directions if isSafe(maze, d[0], d[1], sol)]
             costs = [euclid(loc,goal) for loc in directions]
             directions = [d for _, d in sorted(zip(costs,directions))]
             for i in range(len(costs)):
                 if solveMazeCheck(maze, directions[i][0], directions[i][1], sol, goal):
                     return True
             sol[x][y] = 0
             return False

For given example:
```

A screenshot of the same Jupyter Notebook showing the execution of the A* search algorithm. The code defines a maze and a goal, then calls the `solveMaze` function. The output shows the solution path as a list of coordinates. The code is as follows:

```
For given example:

In [14]: maze = [ [ 1, 0, 1, 1, 1 ],
                  [ 0, 1, 1, 1, 1 ],
                  [ 1, 1, 1, 1, 0 ],
                  [ 1, 1, 0, 0, 1 ],
                  [ 1, 0, 1, 1, 1 ] ]

In [15]: solveMaze(maze, (4,2)) #enter the destination here (source is (0,0))

1 0 0 0 0
0 1 0 0 0
0 0 1 0 0
0 1 0 0 0
0 0 1 0 0

Out[15]: True

Path taken is the connection of 1's from source(0,0) to destination(4,2)

In [ ]:
```

Link to code: <https://github.com/kp10x/AI-Lab/tree/master/AI%20LAB%20TEST%201>

Krishnaprasad V (1BM18CS047)

