School of Data and Computer Science, Sun Yat-sen University

Maze Problem

17341068 Yangfan Jiang

September 1, 2019

Contents

1 Task

- Please solve the maze problem (i.e., find the shortest path from the start point to the finish point) by using BFS or DFS (Python or C++)
- The maze layout can be modeled as an array, and you can use the data file `MazeData.txt` if necessary.
- Please send `E01_YourNumber.pdf` to `ai_201901@foxmail.com`, you can certainly use `E01_Maze.tex` as the LaTeXtemplate.

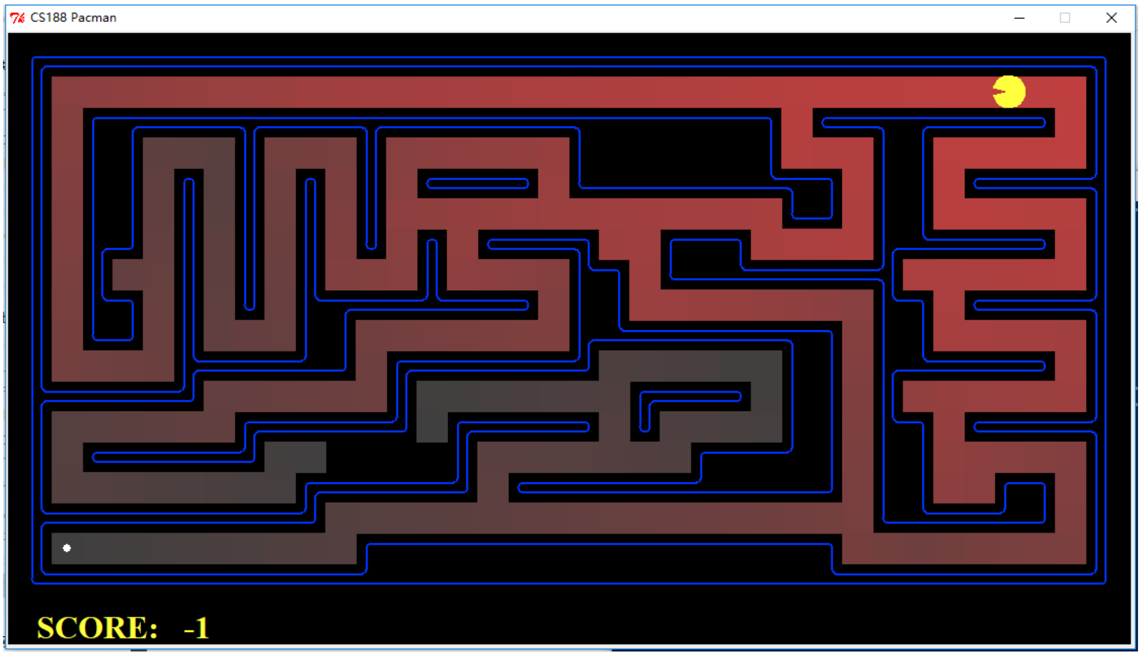


Figure 1: Searching by BFS or DFS

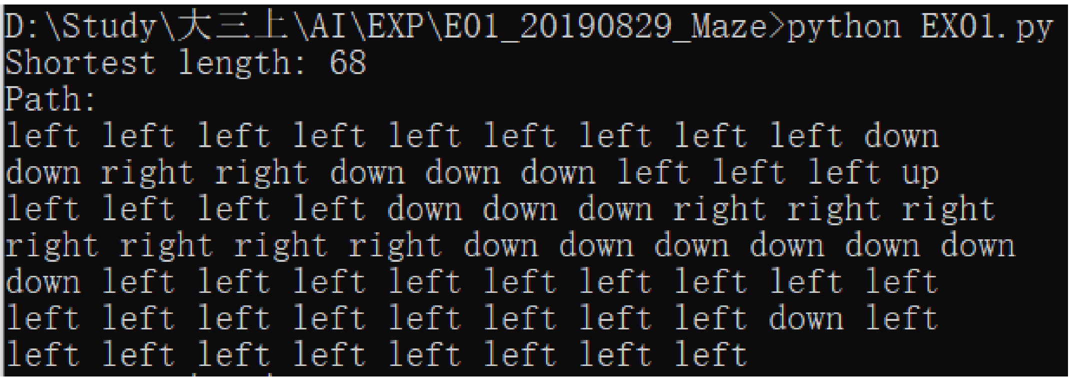
2 Codes

```
# AI Experiment #1: Maze Problem
# 2019/8/29

__author__ = 'Yangfan Jiang (jiangyf29@mail2.sysu.edu.cn)'

'''Solver code of AI experiment #1: Uninformed Search
   Maze Problem (shortest path) by using BFS'''

def load_data(file_name, mode):
    '''
    Load data file

    Input:
    file_name (string): file name
```

```python
    mode (int): 0 or 1, two types of data

    Returns:
    maze (2D list): 2D list represents a maze
    wall (char): character read from data file which represents wall
    '''
    line_cnt = 0
    if mode == 0:
        end = 18
        wall = '%'
        road = ' '

    elif mode == 1:
        end = 39
        wall = '1'
        road = '0'
    else:
        raise Exception('Bad Parameter')

    # 2D list represents a maze
    maze = []

    with open(file_name) as data:
        for line in data:
            if line_cnt == end:
                break
            if mode == 0 or line_cnt >= 21:
                maze.append(list(line))
            line_cnt += 1
    return maze, wall

def print_maze(maze):
    '''
    Maze data visualization, may helpful for debug
    '''
    for i in maze:
        print(''.join(i), end='')

def bfs(maze, start_pos, wall):
    curr_len = 0
    x = start_pos[0]
    y = start_pos[1]
    frontier = [start_pos]
    #print_maze(maze)

    while frontier:
        new_frontier = []
        for next_pos in frontier:
            x = next_pos[0]
            y = next_pos[1]
            path = next_pos[2]

            if maze[x][y] == 'E':
                print('Shortest length:', curr_len)
                print('Path:')
                cnt = 0
                for i in next_pos[2]:
                    print(i,end=' ')
                    cnt += 1
                    if cnt % 10 == 0:
                        print()
                return

            maze[x][y] = wall
            if maze[x-1][y] != wall:
                new_frontier.append([x-1, y, path+['up']])
            if maze[x][y-1] != wall:
                new_frontier.append([x, y-1, path+['left']])
            if maze[x+1][y] != wall:
                new_frontier.append([x+1, y, path+['down']])
            if maze[x][y+1] != wall:
                new_frontier.append([x, y+1, path+['right']])
        frontier = new_frontier
        curr_len += 1

if __name__ == '__main__':
    maze_file = 'MazeData.txt'
    maze ,wall= load_data(maze_file, 1)
    start_pos = [1, 34, []]
    bfs(maze, start_pos, wall)
```

3 Results

The shortest path length is 68,and the trace has been printed .



```
D:\Study\大三上\AI\EXP\E01_20190829_Maze>python EX01.py
Shortest length: 68
Path:
left left left left left left left left left down
down right right down down down left left left up
left left left left down down down right right right
right right right right down down down down down down
down left left left left left left left left left
left left left left left left left left down left
left left left left left left left left
```

Figure 2: Result:length and the Trace