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

E01 Maze Problem

19214808 Yikun Liang

September 1, 2020

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_2020@foxmail.com`, you can certainly use `E01_Maze.tex` as the LaTeXtemplate.

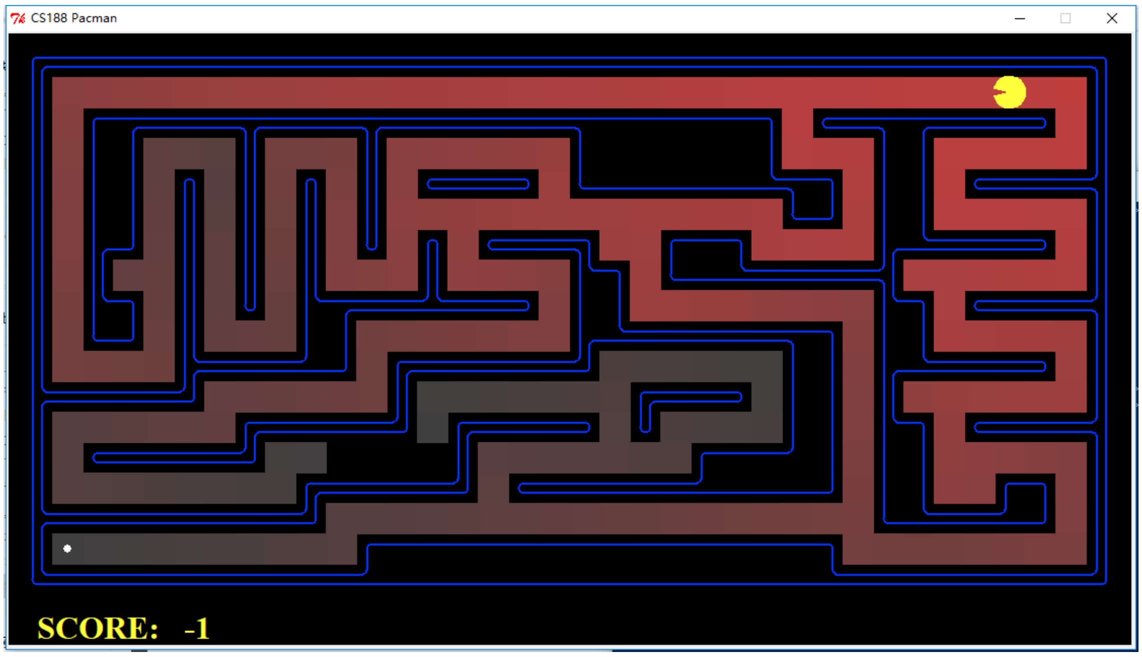
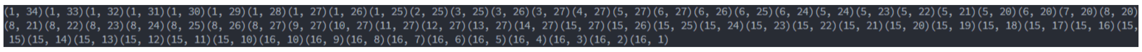


Figure 1: Searching by BFS or DFS

2 Codes

```
import queue
directions = ((-1, 0), (1, 0), (0, -1), (0, 1))
def bfs(q, maze):
    while True:
        cur = q.get()
        for direction in directions:
            next = (cur[0] + direction[0], cur[1] + direction[1])
            # This is used to avoid crossing the boundary.
            try:
                next_val = maze[next[0]][next[1]]
            except:
                continue
            ''' This is used to log last point's position,
            in order to prevent the process from visiting a point twice,
            and to jot down paths.'''
            if next_val == '0':
                maze[next[0]][next[1]] = cur
                q.put(next)
            elif next_val == 'E':
                maze[next[0]][next[1]] = cur
                return next
def read_maze(maze):
    with open("MazeData.txt") as f:
        while True:
            line = f.readline()
            if not line:
                break
            maze.append(list(line))
def init_queue(q):
    for i in range(len(maze)):
        for j in range(len(maze[i])):
            if maze[i][j] == 'S':
                q.put((i, j))
def print_path(maze):
    path = list()
    path.append(end)
    while True:
        cur = maze[path[-1][0]][path[-1][1]]
        if cur == 'S':
            break
        path.append(cur)
    while path:
        cur = path.pop()
        print(cur, end = '')
maze = list()
read_maze(maze)
q = queue.Queue()
init_queue(q)
end = bfs(q, maze)
print_path(maze)
```

3 Results



```
(1, 34)(1, 33)(1, 32)(1, 31)(1, 30)(1, 29)(1, 28)(1, 27)(1, 26)(1, 25)(2, 25)(3, 25)(3, 26)(3, 27)(4, 27)(5, 27)(6, 27)(6, 26)(6, 25)(6, 24)(5, 24)(5, 23)(5, 22)(5, 21)(5, 20)(6, 20)(7, 20)(8, 20)
(8, 21)(8, 22)(8, 23)(8, 24)(8, 25)(8, 26)(8, 27)(9, 27)(10, 27)(11, 27)(12, 27)(13, 27)(14, 27)(15, 27)(15, 26)(15, 25)(15, 24)(15, 23)(15, 22)(15, 21)(15, 20)(15, 19)(15, 18)(15, 17)(15, 16)(15,
 15)(15, 14)(15, 13)(15, 12)(15, 11)(15, 10)(16, 10)(16, 9)(16, 8)(16, 7)(16, 6)(16, 5)(16, 4)(16, 3)(16, 2)(16, 1)
```

Figure 2: Results