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

Maze Problem

17341137 Zhenpeng Song

August 29, 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.

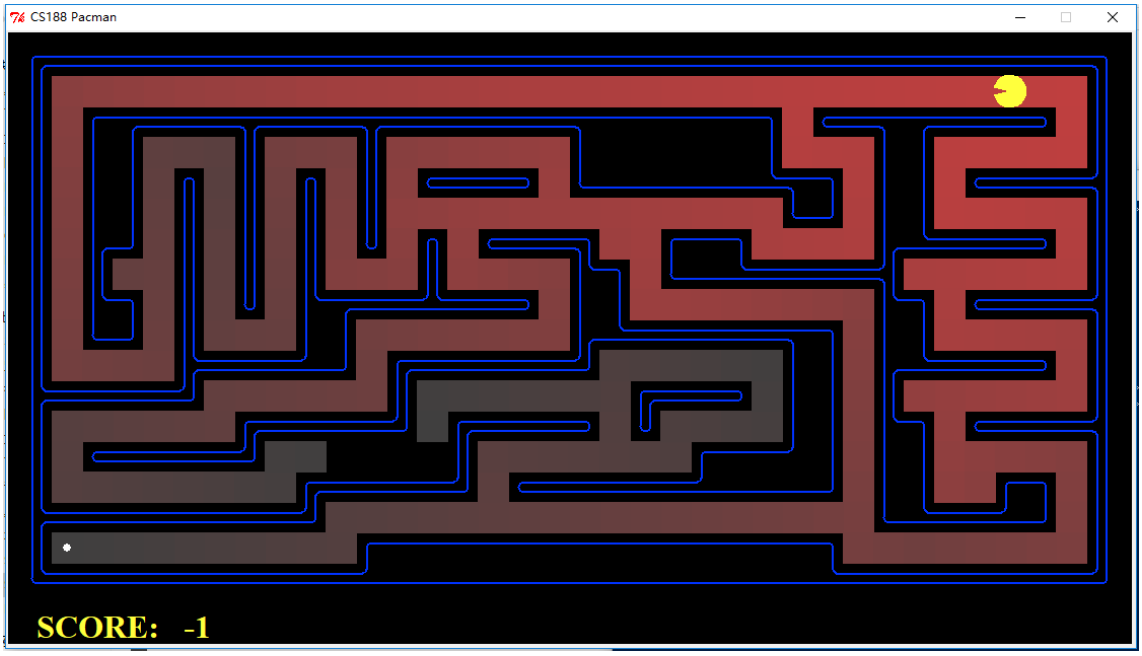

Figure 1: Searching by BFS or DFS

2 Codes

- Python Version

```
# AI - exp 1 - python - Pacman
# ID: 17341137

import sys
sys.setrecursionlimit(10000)


def formatprint(map):
    for i in range(len(map)):
        for j in range(len(map[i])):
            print(map[i][j], end=' ')
        print("")
    print("")
```

```
def printpath(map, path):
    print("Shortest Path with %d steps! (Noted with '#')" % len(path))
    for i in range(len(map)):
        for j in range(len(map[i])):
            if (i, j) in path[1:-1]:
                print('#', end=" ")
            else:
                print(map[i][j], end=" ")
        print("")
    print("")


def dfs(map, x, y, used):
    global res
    if len(res) and len(used) > len(res):
        return
    if x == -1 or y == -1 or x == len(map) or y == len(map[0]) or map[x][y
        return
    elif map[x][y] == 'E':
        # print("goal with %d" % (len(used) + 1))
        if len(used) + 1 < len(res) or len(res) == 0:
            res = used
            res.append((x, y))
        return
    else:
        used.append((x, y))
        dfs(map, x + 1, y, used[:])
        dfs(map, x - 1, y, used[:])
        dfs(map, x, y + 1, used[:])
        dfs(map, x, y - 1, used[:])
        used.remove(used[-1])


if __name__ == "__main__":
    # Generate the Map
    FilePath = "./MazeData.txt"
    Maze_with_nums = []
    res = []
    with open(FilePath, 'r') as Maze:
        for i in Maze.readlines():
            if i[0] == '1' or i[0] == '0':
                Maze_with_nums.append(i[:-1])
    formatprint(Maze_with_nums)

    # Solution
    for i in range(len(Maze_with_nums)):
        for j in range(len(Maze_with_nums[i])):
```

```
                if Maze_with_nums[i][j] == 'S':
                    dfs(Maze_with_nums, i, j, [])
                    break
        # Print the result
        printpath(Maze_with_nums, res)
```

- CPP Version

```
// AI - exp 1 - cpp - Pacman
// ID:17341137

#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <utility>
using namespace std;
vector < pair < int, int > > res;

void formatprint(vector < string > & map) {
    for (int i = 0; i < map.size(); i++) {
        for (int j = 0; j < map[0].size(); j++) {
            cout << map[i][j] << " ";
        }
        cout << endl;
    }
}


void printpath(vector < string > map, vector < pair < int, int > > path) {
    cout << "Shortest Path with " << path.size() << "steps! (Noted with '#
    for (int i = 0; i < path.size(); i++) {
        map[path[i].first][path[i].second] = '#';
    }
    formatprint(map);
}


bool in(vector < pair < int, int > > path, int itemx, int itemy) {
    for (int i = 0; i < path.size(); i++) {
        if (path[i].first == itemx && path[i].second == itemy) {
            return true;
        }
    }
    return false;
}

void dfs(vector <string> & map, int x, int y, vector <pair<int, int> > use
    if (res.size() && res.size() < used.size())
        return;
```

```
    else if (x == -1 || y == -1 || x == map.size() || y == map[0].size() |
        return;
    else if (map[x][y] == 'E') {
        if (used.size() + 1 < res.size() || res.empty()) {
            res = used;
            res.push_back(make_pair(x, y));
        }
        return;
    }
    else {
        used.push_back(make_pair(x, y));
        dfs(map, x + 1, y, used);
        dfs(map, x - 1, y, used);
        dfs(map, x, y + 1, used);
        dfs(map, x, y - 1, used);
        used.pop_back();
    }
}


int main() {
    string FilePath = "./MazeData.txt";
    ifstream MazeFile(FilePath.c_str());
    vector < string > Maze_with_nums;
    vector < pair < int, int > > used;
    if (MazeFile.is_open()) {
        string buf;
        while (getline(MazeFile, buf)) {
            if (buf[0] == '0' || buf[0] == '1') {
                Maze_with_nums.push_back(buf);
            }
        }
        formatprint(Maze_with_nums);
    }

    // Solution
    for (int i = 0; i < Maze_with_nums.size(); i++) {
        for (int j = 0; j < Maze_with_nums[0].size(); j++) {
            if (Maze_with_nums[i][j] == 'S') {
                dfs(Maze_with_nums, i, j, used);
                break;
            }
        }
    }

    // Print the result
    printpath(Maze_with_nums, res);
    return 0;
}
```

3 Results

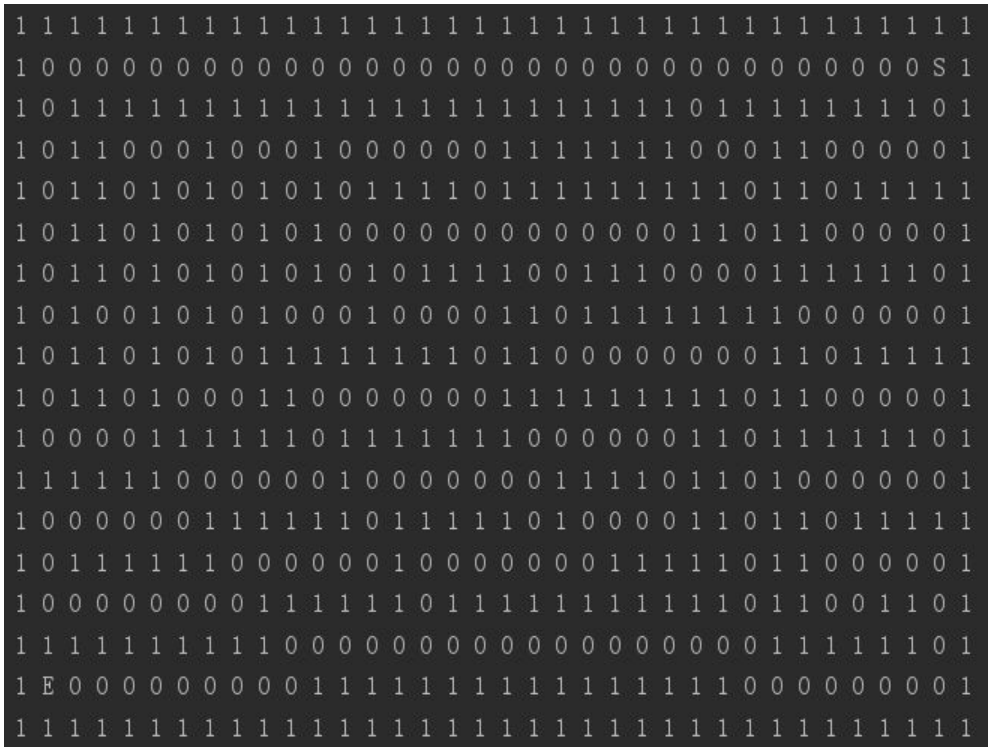

Figure 2: Maze(1s: Walls; 0s: Paths)

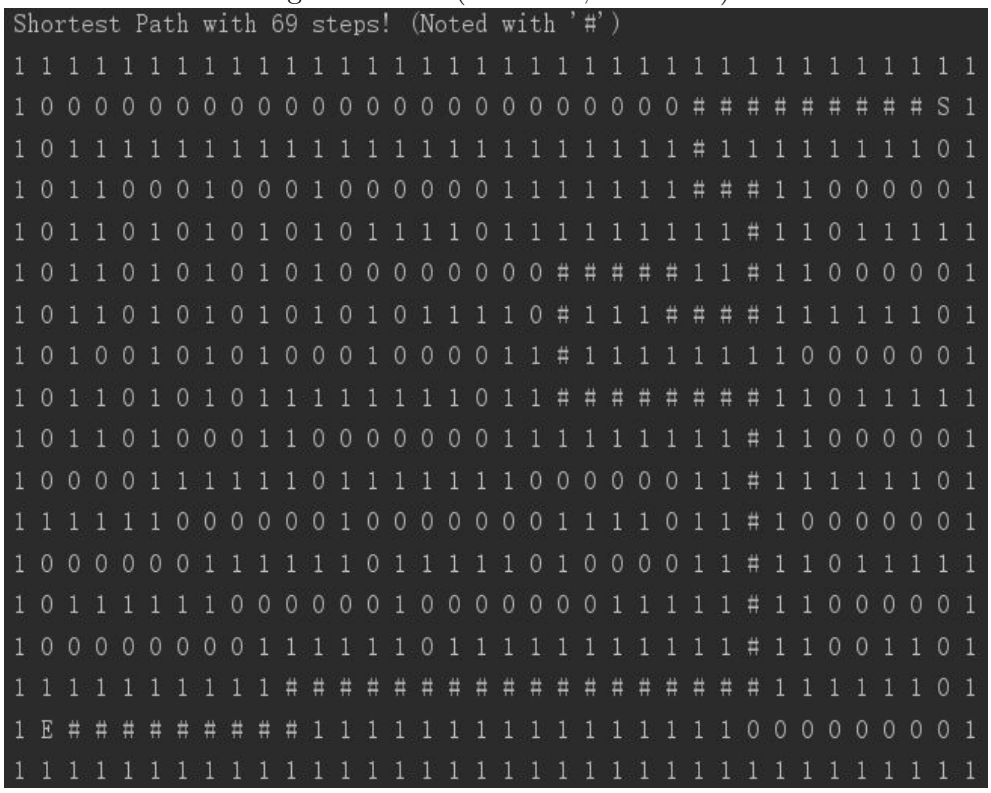

Figure 3: Maze(1s: Walls; 0s: Paths; s: Target Path)