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

E01 Maze Problem

18340215 Tianyi Zhang

September 6, 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.

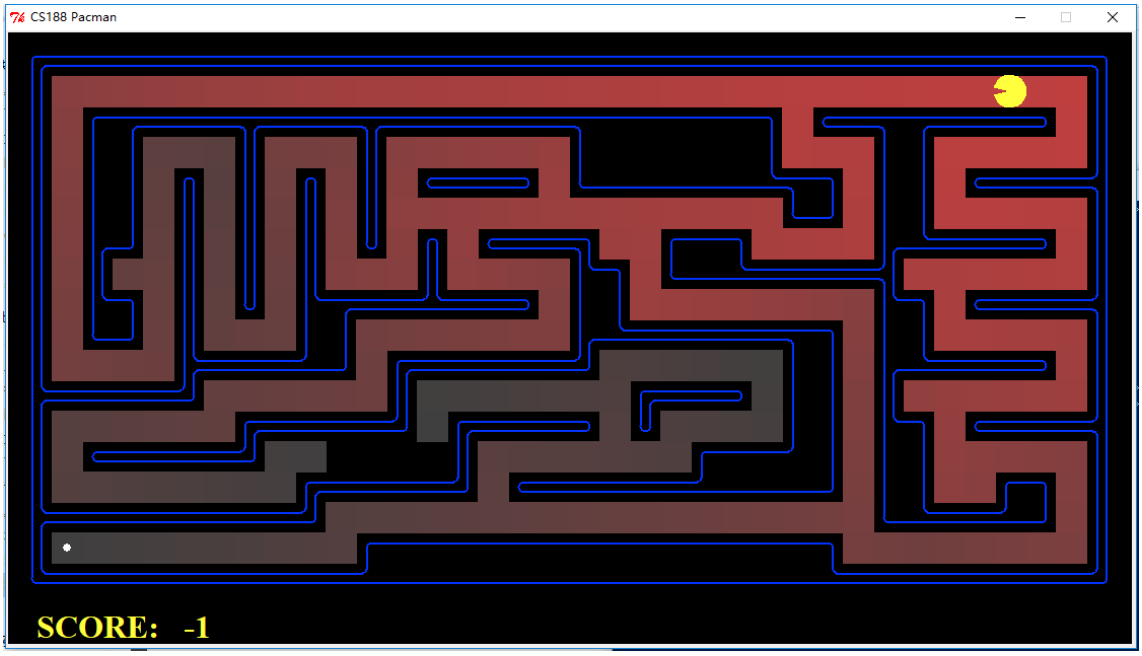


Figure 1: Searching by BFS or DFS

2 Codes

```
//v0 dfs
#include<iostream>
#include<fstream>
#include<string>
#include<vector>
#include<queue>
#include<iomanip>
#define MAX 999
#define PATH '*'
#define VISITED -1
using namespace std;

vector<pair<int, int>> rst;
int min_len = MAX;
```

```
void dfs(vector<vector<char>> map, pair<int, int> curr
, int len, vector<pair<int, int>> path)
{
        path.push_back(curr);
        if (map[curr.first][curr.second] == 'E')
        {
                if (len < min_len)
                {
                        rst = path;
                        min_len = len;
                }
        }
        else if (map[curr.first][curr.second] == '0') {
                map[curr.first][curr.second] = VISITED;
                dfs(map, pair<int, int>(curr.first + 1, curr.second)
                , len + 1, path);
                dfs(map, pair<int, int>(curr.first, curr.second + 1)
                , len + 1, path);
                dfs(map, pair<int, int>(curr.first - 1, curr.second)
                , len + 1, path);
                dfs(map, pair<int, int>(curr.first, curr.second - 1)
                , len + 1, path);
        }
}


int main()
{
        ifstream inFile;
        inFile.open("MazeData.txt", ifstream::in);
        if (!inFile)
        {
                cout << "FILE_OPEN_ERROE!" << endl;
                return 0;
        }
        vector<vector<char>> map;
        pair<int, int> s;
        int j = 0;
        while (1)
        {
                string curr;
                getline(inFile, curr);
                vector<char> vec(curr.begin(), curr.end());
                map.push_back(vec);
                if (curr.empty()) break;
                for (int i = 0; i < vec.size(); i++)
                {
```

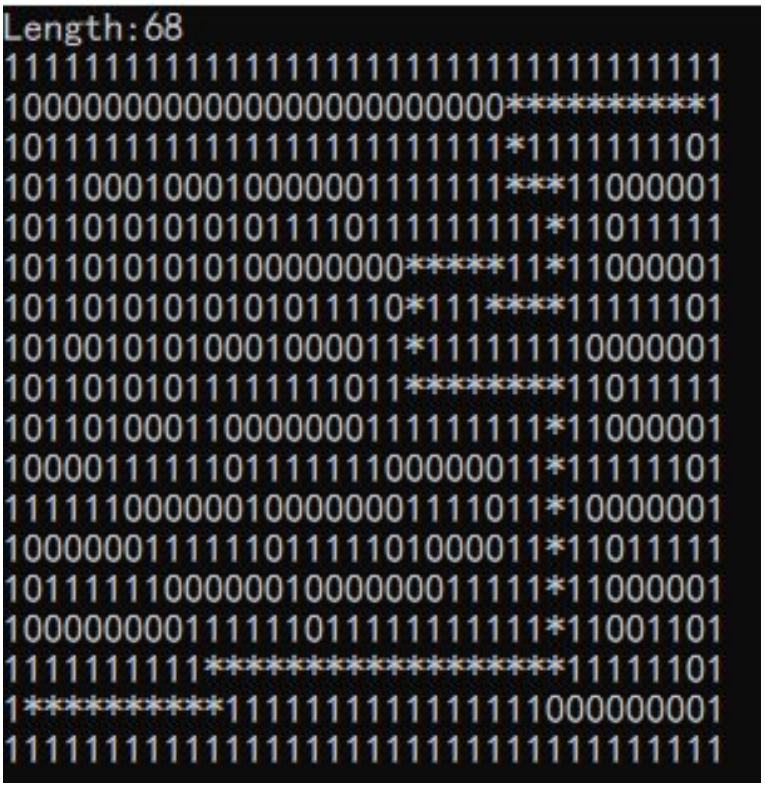
```
                        if (vec[i] == 'S')
                        {
                                s.first = j;
                                s.second = i;
                                map[j][i] = '0';
                        }
                }
                j++;
        }
        inFile.close();
        vector<pair<int, int>> tem;
        dfs(map, s, 0, tem);

        for (auto i : rst)
        {
                map[i.first][i.second] = PATH;
        }
        cout << "Length:" <<min_len;
        puts("");
        for (auto i : map)
        {
                for (auto j : i)
                {
                        if (j == VISITED) cout << '0';
                        else cout << j;
                }
                puts("");
        }
        return 0;
}

//v1 bfs
#include<iostream>
#include<fstream>
#include<string>
#include<vector>
#include<queue>
#include<deque>
#include<iomanip>
#define VISITED -1
#define PATH '*'
using namespace std;

int main()
{
        ifstream inFile;
        inFile.open("MazeData.txt", ifstream::in);
        if (!inFile)
        {
                cout << "FILE_OPEN_ERROE!" << endl;
```

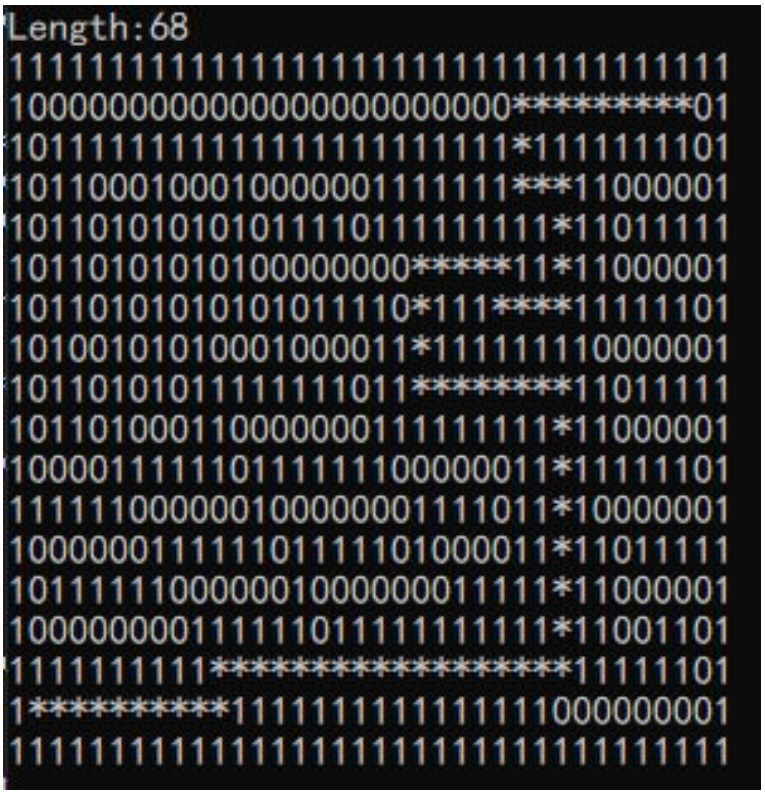
```
        return 0;
}
vector<vector<char>> map;
pair<int, int> s;
pair<int, int> e;
int j = 0;
while (1)
{
        string curr;
        getline(inFile, curr);
        vector<char> vec(curr.begin(), curr.end());
        map.push_back(vec);
        if (curr.empty()) break;
        for (int i = 0; i < vec.size(); i++)
        {
                if (vec[i] == 'S')
                {
                        s.first = j;
                        s.second = i;
                }
                if (vec[i] == 'E')
                {
                        e.first = j;
                        e.second = i;
                        map[j][i] = '0';
                }
        }
        j++;
}
inFile.close();

vector<pair<int, int>> row(map[0].size(), pair<int, int>(0, 0));
vector<vector<pair<int, int>>> fa(map.size(), row);
queue<pair<int, int>> que;
queue<pair<int, int>> emp;
queue<pair<int, int>> hold;
que.push(s);

bool sign = 0;
while (!que.empty())
{
        while (!que.empty())
        {
                pair<int, int> curr = que.front();
                que.pop();
                if (curr == e)
                {
                        sign = true;
                        break;
```

```
                }
                map[curr.first][curr.second] = VISITED;
                if (map[curr.first + 1][curr.second] == '0')
                {
                        hold.push(pair<int, int>
                        (curr.first + 1, curr.second));
                        fa[curr.first + 1][curr.second] =
                        pair<int, int>(curr.first, curr.second);
                }
                if (map[curr.first - 1][curr.second] == '0')
                {
                        hold.push(pair<int, int>
                        (curr.first - 1, curr.second));
                        fa[curr.first - 1][curr.second] =
                        pair<int, int>(curr.first, curr.second);
                }
                if (map[curr.first][curr.second + 1] == '0')
                {
                        hold.push(pair<int, int>
                        (curr.first, curr.second + 1));
                        fa[curr.first][curr.second + 1] =
                        pair<int, int>(curr.first, curr.second);
                }
                if (map[curr.first][curr.second - 1] == '0')
                {
                        hold.push(pair<int, int>
                        (curr.first, curr.second - 1));
                        fa[curr.first][curr.second - 1] =
                        pair<int, int>(curr.first, curr.second);
                }
        }
        if (sign) break;
        que = hold;
        hold = emp;
}

pair<int, int> curr = e;
int len = 0;
while (curr != s)
{
        map[curr.first][curr.second] = PATH;
        len++;
        curr = fa[curr.first][curr.second];
}
cout << "Length:" << len << endl;

for (auto i : map)
{
        for (auto j : i)
```

```
                {
                        if (j == VISITED) cout << '0';
                        else cout << j;
                }
                puts("");
        }

        return 0;
}
```

3 Results


Length:68
1111111111111111111111111111111111111111
1000000000000000000000000**********1
1011111111111111111111111*1111111101
1011000100010000001111111***11000001
1011010101010111101111111*11011111
1011010101010000000*****11*11000001
1011010101010101011110*111****11111101
1010010101000100001 1*111111110000001
1011010101111111101 1********11011111
1011010001100000001111111*11000001
1000011111101111111000000 11*11111101
1111110000001000000011110 11*10000001
1000000111111011111010000 11*11011111
1011111100000010000000111 1*11000001
1000000001111110111111111 1*11001101
1111111111**********1111111 11111101
1**********11111111111111110000000 01
1111111111111111111111111111111111111


Length:68
1111111111111111111111111111111111111111
1000000000000000000000000**********01
1011111111111111111111111*1111111101
1011000100010000001111111***11000001
1011010101010111101111111*11011111
1011010101010000000*****11*11000001
1011010101010101011110*111****11111101
1010010101000100001 1*111111110000001
1011010101111111101 1********11011111
1011010001100000001111111*11000001
1000011111101111111000000 11*11111101
1111110000001000000011110 11*10000001
1000000111111011111010000 11*11011111
1011111100000010000000111 1*11000001
1000000001111110111111111 1*11001101
1111111111**********1111111 11111101
1**********11111111111111110000000 01
1111111111111111111111111111111111111