SCHOOL OF DATA AND COMPUTER SCIENCE, SUN YAT-SEN UNIVERSITY

E01 Maze Problem

18364066 Yanzuo Lu

2020 年 9 月 6 日

目录

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.

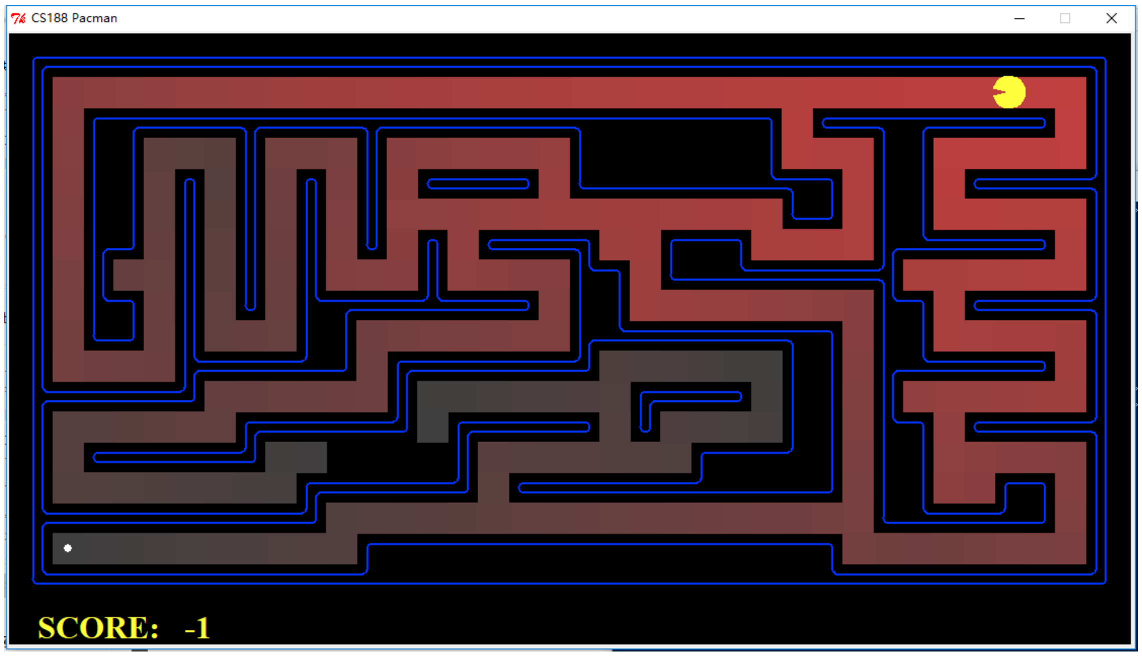

图 1: Searching by BFS or DFS

2 Codes

```
#include <cstdio>
#include <fstream>
#include <queue>
#include <stack>
using namespace std;

constexpr int row = 18;
constexpr int col = 36;
constexpr int dir[4][2] = {{-1, 0}, {1, 0}, {0, -1}, {0, 1}};
```

```
typedef struct Node
{
    int x, y;
}Node;
int maze[row][col];
int vis[row][col];
Node pre[row][col];
Node startn, endn;

void bfs()
{
    queue<Node> q;
    q.push(startn);
    vis[startn.x][startn.y] = 1;
    while (1)
    {
        if (q.empty()) break;
        Node tmpn = q.front();
        q.pop();
        for (int i = 0; i < 4; i++)
        {
            Node next = {tmpn.x + dir[i][0], tmpn.y + dir[i][1]};
            if (!vis[next.x][next.y] && next.x >= 0 && next.x < row &&
                next.y >= 0 && next.y < col && !maze[next.x][next.y])
            {
                pre[next.x][next.y] = tmpn;
                vis[next.x][next.y] = 1;
                q.push(next)
            }
        }
    }
}

void print(Node node)
{
    stack<Node> s;
    int x = node.x;
    int y = node.y;
    while (!(pre[x][y].x == 0 && pre[x][y].y == 0))
    {
        s.push(node);
        node = pre[x][y];
        x = node.x;
        y = node.y;
    }
    int len = s.size();
    while (!s.empty())
    {
        Node tmpn = s.top();
        s.pop();
        printf("(%d, %d)\n", tmpn.x, tmpn.y);
    }
    printf("shortest path length: %d\n", len);
}

int main()
{
    ifstream fin("MazeData.txt");
    for (int i = 0; i < row; i++)
        for (int j = 0; j < col; j++)
        {
            char tmp;
            while (1)
            {
                fin >> tmp;
                if (tmp != '0' && tmp != '1' && tmp != 'S' && tmp != 'E')
                    continue;
                else break;
            }
            if (tmp == 'S')
            {
                startn.x = i;
                startn.y = j;
                maze[i][j] = 1;
            }
            else if (tmp == 'E')
            {
                endn.x = i;
                endn.y = j;
                maze[i][j] = 0;
            }
            else
            {
                maze[i][j] = tmp - '0';
            }
        }
    fin.close();

    bfs();
    // for (int i = 0; i < row; i++)
    //     for (int j = 0; j < col; j++)
    //         printf("(%d, %d)\n", pre[i][j].x, pre[i][j].y);
    print(endn);
    return 0;
}
```

3 Results

本次实验我采用的算法是BFS广搜算法，BFS需要设置一个队列用于存储当前步骤所处的位置，每次循环从队列头部取出一个节点后，首先判断是否已经到达目的地，若已到达则退出循环并进入结果输出的函数print中，若未到达则检查当前节点合法的后继节点，一是不能被访问过，二是在迷宫中是可达的，若检查通过则将后继节点放入到队列尾部，以此类推。

代码实现方面，一是为了能够输出路径的结果，需要设置pre数组用于存储经过该节点的路径的上一个节点位置，该数组维度大小与迷宫维度大小相同，二是为了判定后继节点是否合法，需要分别设置maze数组和vis数组分别用于存储该节点在迷宫是否可达以及在循环中是否已被访问过。

由于输出的坐标点对序列较长，我在迷宫原图中绘制了示意图如图2所示，该最短路径长度68。需要额外注意的一点是，在判定后继节点是否合法时，需要先筛选出已经跳出迷宫范围的坐标点对，避免造成数组访问出错。

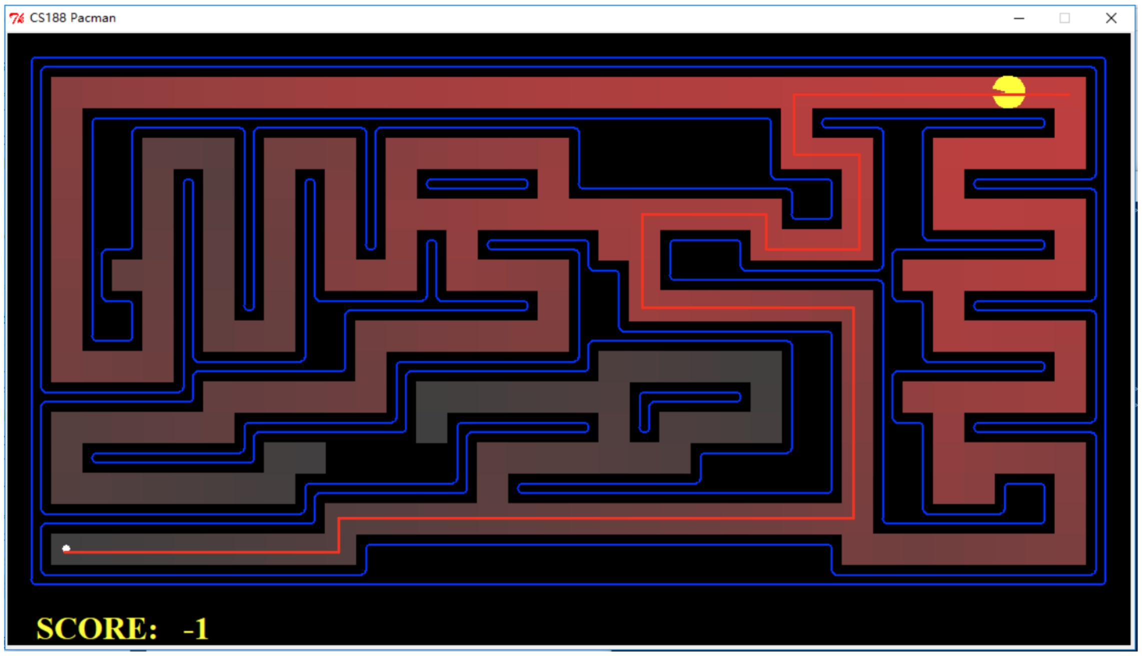

图 2: Shortest Path