E01 Maze Problem

19335155 麦子丰
August 31, 2021

1. Task

- Please solve the maze problem (i.e., find the path from the start point to the finish point) by using BFS or DFS.
- 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_course2021@163.com, you can certainly use E01_Maze.tex as the LATEX template.

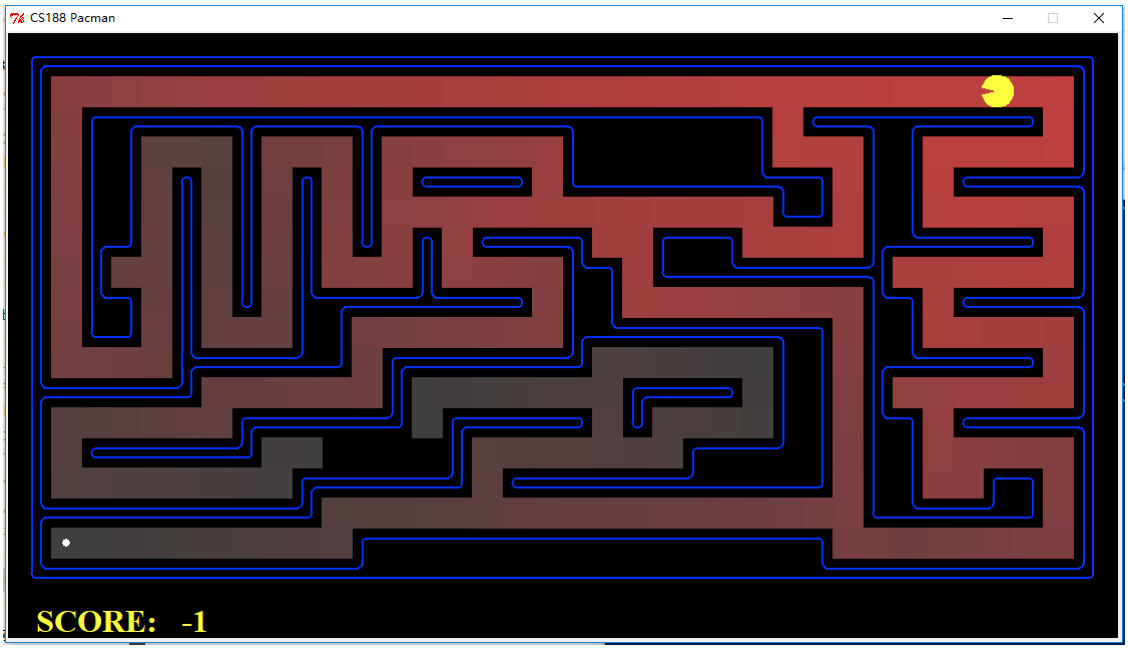

Figure 1: Searching by BFS or DFS

2. Method

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

struct Node{
    int x, y;
    Node(int a=-1, int b=-1) : x(a), y(b){}
    Node(const Node& a) : x(a.x), y(a.y) {}
};

vector<vector<int> > maze;
Node start_node, end_node;
vector<vector<Node> >pre;
```

```cpp
vector<vector<bool> > visited;
const int row = 18, col = 36;
const int dir[4][2] = {{0, -1}, {0, 1}, {-1, 0}, {1, 0}};
bool is_find = false;

void Init(){
    maze.resize(row, vector<int>(col, 0));
    pre.resize(row, vector<Node>(col, Node()));
    visited.resize(row, vector<bool>(col, false));
}

void ReadFile(const char* filename){
    ifstream fin(filename);
    if(!fin.is_open()){
        fprintf(stderr, "Error opening file %s\n", filename);
        return;
    }

    for(int i = 0; i < row; ++i){
        for(int j = 0; j < col; ++j){
            char tmp;
            while(1){
                fin >> tmp;
                if(tmp == '1' || tmp == '0' || tmp == 'S' || tmp == 'E')
break;
            }
            if(tmp == '1' || tmp == '0'){
                maze[i][j] = tmp-'0';
            }
            else if(tmp == 'S'){
                start_node = Node(i, j);
                maze[i][j] = 1;
            }
            else if(tmp == 'E'){
                end_node = Node(i, j);
                maze[i][j] = 0;
            }
        }
    }
    fin.close();
}

void PrintPath(){
    Node node = end_node;
    stack<Node> s;
    while(!(pre[node.x][node.y].x == -1 && pre[node.x][node.y].y == -1)){
        s.push(node);
        node = pre[node.x][node.y];
    }
    s.push(start_node);
    cout << "Path Length: " << s.size() << endl;
    while(!s.empty()){
        node = s.top();
        cout << '(' << node.x << ',' << node.y << ") ";
        s.pop();
    }
}

```

```
74  void BFS(){
75      queue<Node> q;
76      q.push(start_node);
77      visited[start_node.x][start_node.y] = true;
78      while(1){
79          if(q.empty()) break;
80          Node node = q.front();
81          q.pop();
82          for(int i = 0; i < 4; ++i) {
83              Node next(node.x+dir[i][0], node.y+dir[i][1]);
84
85              if(next.x < 0 || next.y < 0 || next.x >= row || next.y >= col)
    continue;
86              if(maze[next.x][next.y] == 1) continue;
87              if(visited[next.x][next.y]) continue;
88
89              pre[next.x][next.y] = node;
90              visited[next.x][next.y] = true;
91              q.push(next);
92          }
93      }
94  }
95
96  int main(){
97      Init();
98      ReadFile("MazeData.txt");
99      BFS();
100     PrintPath();
101 }
```

3. Results

```
Path Length: 69
(1,34) (1,33) (1,32) (1,31) (1,30) (1,29) (1,28) (1,27) (1,26) (1,25) (2,25) (3,25) (3,2
6) (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)
```

本次实验采用了BFS的方式对图进行无信息搜索。由于在本题中，点与点之间的路径长度都是1，因此BFS得出的路径就是最短路径。

BFS采用一个队列来表示当前搜索到的位置。每次从队头取出一个结点后，首先需要判断是否已经达到终点，如果达到终点则停止搜索并输出结果。其次，在每个点都可以做4个方向的运动，但不是每个方向都是合法的，因此需要进行合法性检验。具体来说，一个运动是合法的当且仅当其终点不是墙壁，且在之前的搜索过程中没有被访问过。具体到本题实现中，合法性检验体现在第85~87行的语句中：

```
1  if(next.x < 0 || next.y < 0 || next.x >= row || next.y >= col) continue;
2  if(maze[next.x][next.y] == 1) continue;
3  if(visited[next.x][next.y]) continue;
```

由于需要输出路径，因此在BFS的过程中需要同步维护每个结点在BFS搜索树中的前缀结点，pre数组完成这个工作。当搜索到终点后，从终点出发沿着pre数组指示的前缀结点不断回溯，直至起点，就可以得到BFS搜索路径。