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

E01 Maze Problem

18340013 Conghao Chen

September 3, 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.

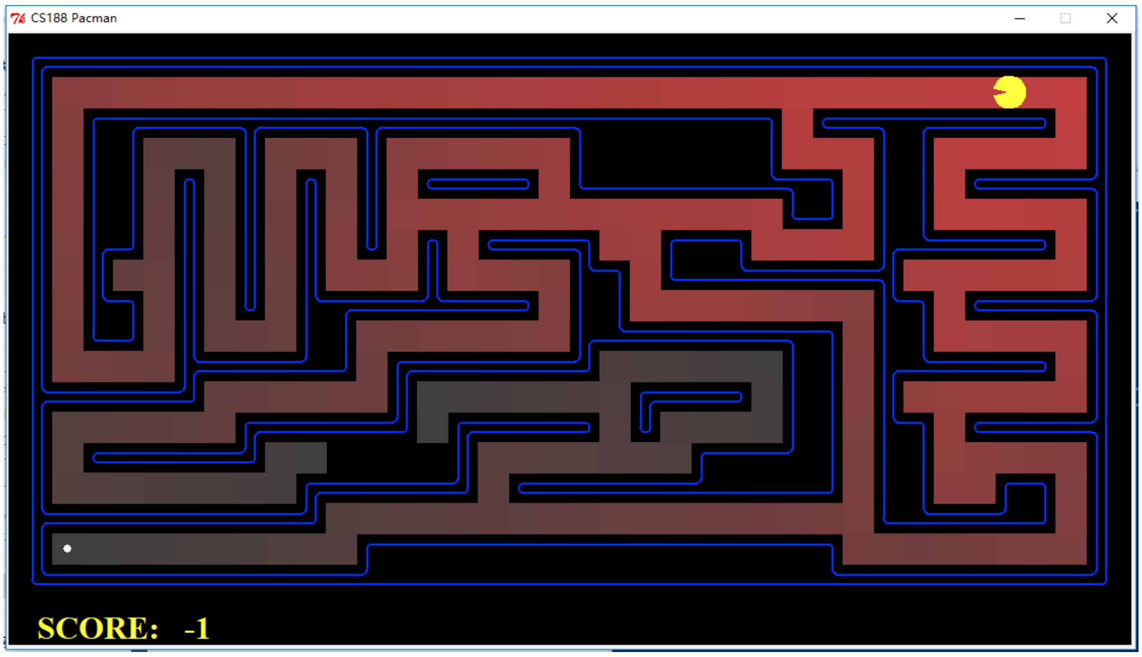


Figure 1: Searching by BFS or DFS

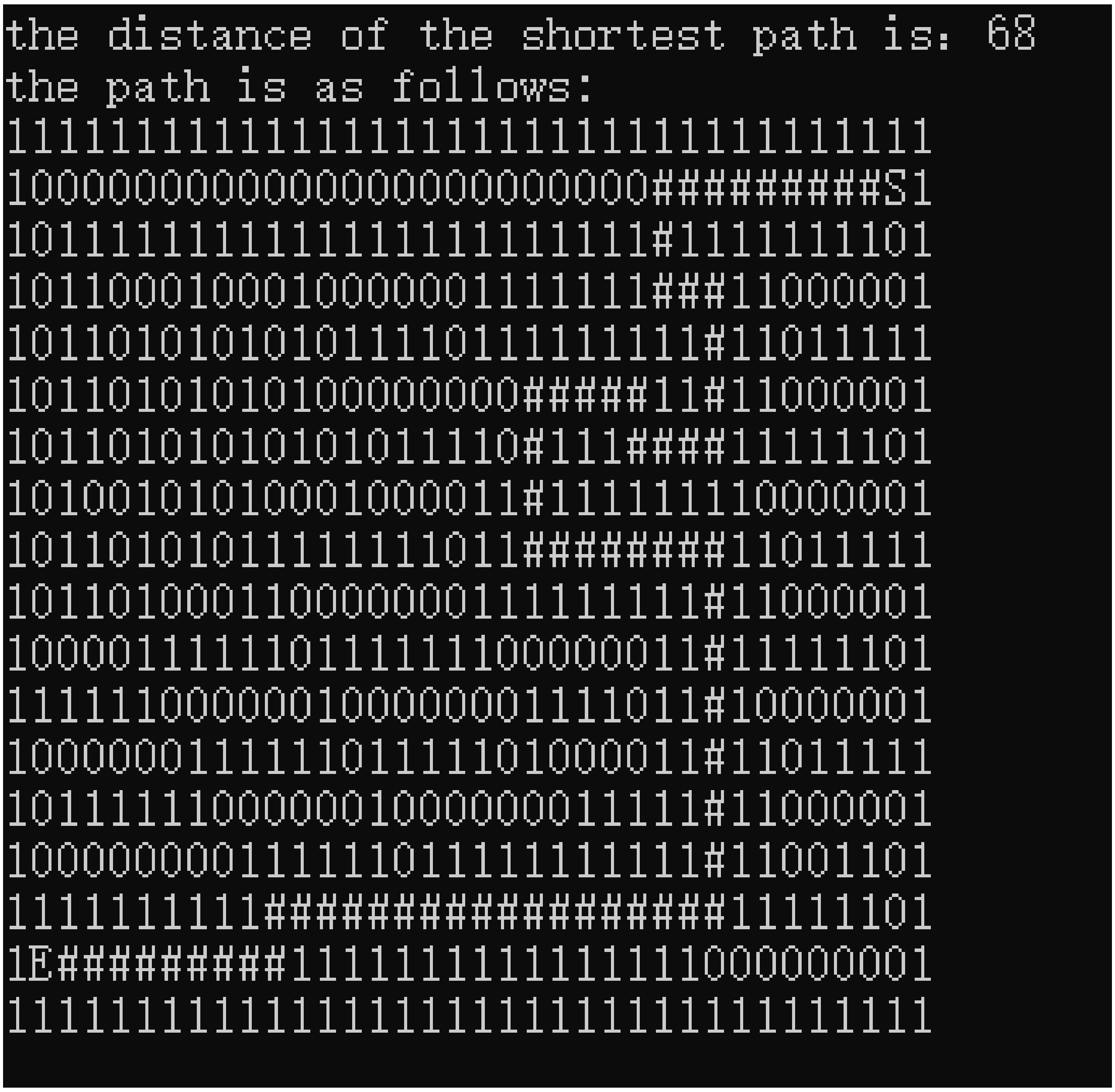
2 Codes

```
#include<iostream>
#include<bits/stdc++.h>
using namespace std;

char Mazedata[100][100];                    //matrix of maze
int sx, sy, ex, ey;                        //coordinate of start point
    and end point
int shortdis=1000000;                     //the distance of the
    shortest path,start with 1000000
bool visited[100][100];                   //have visited or not
vector< pair<int, int> > dir;            //directions:W/A/S/D
vector< pair<int, int> > shortpath;     //record the shortest path

void dfs(int x,int y,int length,vector< pair<int, int> > &path){
```

```
visited[x][y]=1;
path.push_back(make_pair(x, y));
//update the shortest path when get to the end point
if(x==ex&&y==ey){
if(length<shortdis){
shortdis=length;
shortpath=path;
}
return;
}
//search in all directions
for(int i=0; i<dir.size(); i++){
int nx=x+dir[i].first;
int ny=y+dir[i].second;
if(Mazedata[nx][ny]!='1'&&visited[nx][ny]!=1){
length++;
dfs(nx,ny,length,path);
length--;
visited[nx][ny] = 0;
path.pop_back();
}
}
return;
}

int main(){
int m=1, n;
//express the directions:W/A/S/D
dir.push_back(make_pair(-1, 0));
dir.push_back(make_pair(0, -1));
dir.push_back(make_pair(0, 1));
dir.push_back(make_pair(1, 0));
//read the data of maze
ifstream input("MazeData.txt");
char s[100];
input.getline(s,100);
n=strlen(s);
strcpy(Mazedata[0],s);
while(!input.eof()){
visited[m][0]=1;
visited[m][n-1]=1;
for(int j=0;j<n;j++){
input>>Mazedata[m][j];
if(Mazedata[m][j]=='S'){
sx=m;
sy=j;
}
if(Mazedata[m][j]=='E'){
ex=m;
ey=j;
}
}
m++;
}
m--;
input.close();
vector< pair<int, int> > path;
dfs(sx,sy,0,path);                    //DFS
if(shortdis==1000000) cout<<"There is no solution to this question.\n";
else{
cout<<"the distance of the shortest path is:"<<shortdis<<endl;
cout<<"the path is as follows:\n";
for(int i=1;i<shortpath.size()-1;i++) Mazedata[shortpath[i].first][
    shortpath[i].second]='#';
for(int i=0;i<m;i++){
for(int j=0;j<n;j++){
cout<<Mazedata[i][j];
}
cout<<endl;
}
}
return 0;
}
```

3 Results

I used C++ programming language and DFS method to solve the problem. DFS itself is not difficult to implement. As for how to record the final path, I opened a container to record the path and update it, also opened a variable to record the length of the path additionally. Finally, we can get the shortest path. The result is as follows:(The shortest path is marked through #.)



```
the distance of the shortest path is： 68
the path is as follows:
111111111111111111111111111111111111
1000000000000000000000000#########S1
10111111111111111111111111#1111111101
1011000100010000001111111###11000001
101101010101011110111111111#11011111
10110101010100000000#####11#11000001
10110101010101011110#111####11111101
10100101010001000011#111111110000001
10110101011111111011########11011111
101101000110000000111111111#11000001
100001111110111111100000011#11111101
111111000000100000001111011#10000001
100000011111101111101000011#11011111
101111110000001000000011111#11000001
100000000111111011111111111#11001101
1111111111##################11111101
1E#########1111111111111111000000001
111111111111111111111111111111111111
```