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

# Maze Problem

17341111 Xuehai Liu

September 1, 2019

## Contents

# 1 Task

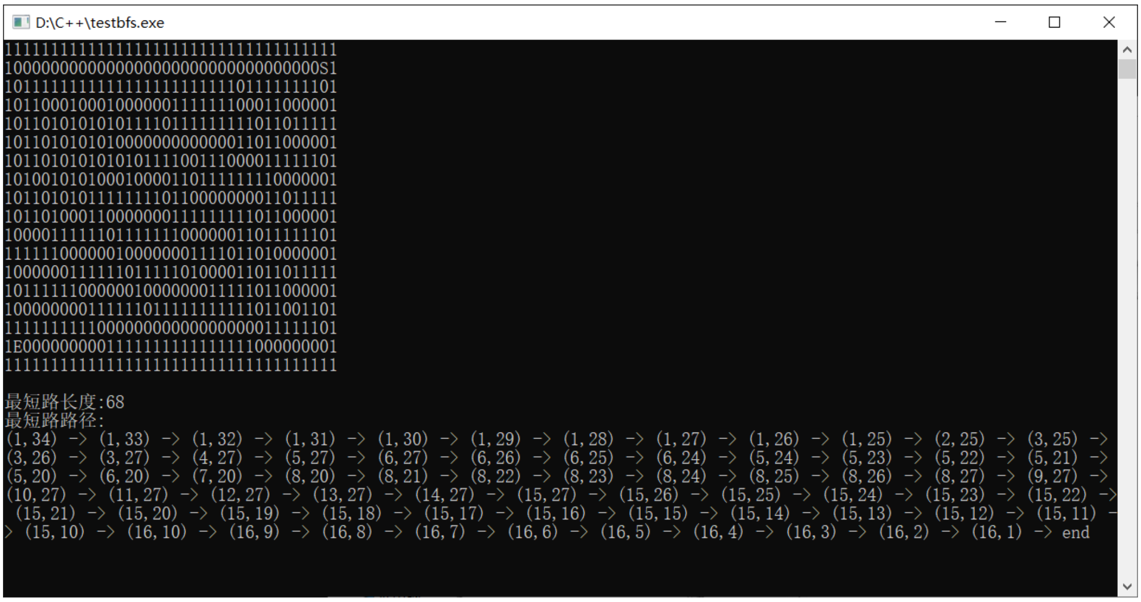
- Please solve the maze problem (i.e., find the shortest path from the start point to the finish point) by using BFS or DFS (Python or C++)
- The maze layout can be modeled as an array, and you can use the data file `MazeData.txt` if necessary.
- Please send `E01_YourNumber.pdf` to `ai_201901@foxmail.com`, you can certainly use `E01_Maze.tex` as the LaTeXtemplate.

Figure 1: Searching by BFS or DFS

# 2 Codes

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

int migong[10000];
int fa[10000];
int visit[10000]={0};
queue <int> bestpath;
int row;
int col;

//cal the length of path
int countlen(int x)
{
    int len = 0;
    while(fa[x]!=-1)
    {
        len++;
        x = fa[x];
    }
    return len;
}

//require the path which ends at x
queue<int> getPath(int x)
{
    queue <int> path;
    path.push(x);
    while (fa[x]!=-1)
    {
        x = fa[x];
        path.push(x);
    }
    return path;
}

//print the result stack
void print(queue<int> q)
{
    int a[10000];
    int j=0;
    int size = q.size();
    for (int i=0;i<size;i++)
    {
        a[size-i-1] = q.front();
        q.pop();
    }
    for (int i=0;i<size;i++)
    {
        int r = a[i]/col;
        int c = a[i]%col;
        cout<<"("<<r<<","<<c<<")"<<"␣->␣";
    }
    cout<<"end";
    cout<<endl;
}

int bfs(int begin )
{
    queue <int>q;
    q.push(begin);
    int len = 0;
    while (!q.empty())
    {
        int temp = q.front();
        visit[temp] = 1;
        q.pop();
        if(migong[temp]==3)//find the end
        {
            len = countlen(temp);
            bestpath = getPath(temp);
            break;
        }
        //left
        if( temp%col != 0 && migong[temp-1] != 1 && !visit[temp-1])
        {
            q.push(temp-1);
            fa[temp-1] = temp;
        }
        //right
        if ( (temp+1)%col !=0 && migong[temp+1] != 1 && !visit[temp+1])
        {
            q.push(temp+1);
            fa[temp+1] = temp;
        }
        //upside
        if( temp-col>=0 && migong[temp-col] != 1 && !visit[temp-col] )
        {
            q.push(temp-col);
            fa[temp-col] = temp;
        }
        //downside
        if( temp+col<row*col && migong[temp+col] != 1 && !visit[temp+col])
        {
            q.push(temp+col);
            fa[temp+col] = temp;
        }
    }
    return len;

}

int main()
{
    char ch;
    int i=0,j=0;
    int begin;
    ifstream infile;
    infile.open("MazeData.txt");
    infile>>noskipws;
    //read char from file including '\n'
    while ( !infile.eof())
    {
        infile>>ch;
        cout<<ch;
        if(ch=='\n')
        {
            row++;
            col = j;
            continue;
        }
        if(row==0)
            j++;
        if(ch=='0'||ch=='1')
            migong[i++] = ch-'0';
        else if (ch=='S')
        {
            begin = i;
            migong[i++] = 2;
        }
        else if (ch=='E')
            migong[i++] = 3;
    }
    infile.close();
    fa[begin] = -1;
    cout<<"shortest length:"<<bfs(begin)<<endl;
    cout<<"shortest path:"<<endl;
    print(bestpath);
    Sleep(1000000);
}
```

## 3 Results



```
111111111111111111111111111111111111
1000000000000000000000000000000000S1
101111111111111111111111101111111101
101100010001000000111111100011000001
101101010101011110111111111011011111
101101010101000000000000011011000001
101101010101010111100111000011111101
101001010100010000110111111110000001
101101010111111110110000000001011111
101101000110000000011111111011000001
100001111110111111100000011011111101
111111000000100000000111011010000001
100000011111101111101000011011011111
101111110000000100000001111011000001
100000000111110111111111111011001101
111111111100000000000000000011111101
1E0000000001111111111111111000000001
111111111111111111111111111111111111

最短路长度:68
最短路路径:
(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, 26) -> (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) -> end
```