2019年蓝桥杯软件类大学A组第4题

华东理工大学计算机系教师　罗勇军

原文链接：https://blog.csdn.net/weixin_43914593/category_10721247.html

文章目录

2019省赛A组第4题“**迷宫**”，题目链接：
http://oj.ecustacm.cn/problem.php?id=1455

这一题是初学者训练的好题目。

1、题目描述

下图给出了一个迷宫的平面图，其中标记为1 的为障碍，标记为0 的为可以通行的地方。
010000
000100
001001
110000
迷宫的入口为左上角，出口为右下角，在迷宫中，只能从一个位置走到这个它的上、下、左、右四个方向之一。
对于上面的迷宫，从入口开始，可以按DRRURRDDDR 的顺序通过迷宫，一共10 步。其中D、U、L、R 分别表示向下、向上、向左、向右走。
对于下面这个更复杂的迷宫（30 行50 列），请找出一种通过迷宫的方式，其使用的步数最少，在步数最少的前提下，请找出字典序最小的一个作为答案。
请注意在字典序中D<L<R<U。

2、题解

这题用来**练习搜索**很好！还有**路径打印**，也需要掌握！

图虽然不大，但直接用手画、拿眼睛看是数不出来了，只能编码，这就是出题人的意思吧。

不不不，有人说手画、眼睛数能得到答案：https://www.bbsmax.com/A/kmzLkg9XdG/

出题人还是善良了，这个迷宫其实不复杂，真的能用手数出来。

本题图不大，那么就是鼓励用简单的暴力搜索了，别想太多。

是BFS还是DFS?

（1）要搜所有可能的路径吗？不管用BFS还是DFS，复杂度都是指数的。

（2）题目只要求搜最短路径，这就简单多了，肯定用BFS。BFS也是求最短路径的一种经典算法，不过，它仅用于相邻结点距离为1的情况，这就是本题的情况。

BFS原理，参考这篇博文的“**3 BFS的性质和代码实现**”：
https://blog.csdn.net/weixin_43914593/article/details/104608578

（3）最短路径可能不止一条，不过并不复杂。BFS的特点是：它是逐层扩散的（往BFS的队列中加入邻居结点时，是按距离起点远近的顺序加入的：先加入距离起点为1的邻居结点，加完之后，再加入距离为2的邻居结点，等等），搜完一层，才会继续搜下一层。一条路径是从起点开始，沿着每一层逐步往外走，每多一层，路径长度就增加了1。那么，所有长度相同的最短路径都是从相同的层次扩散出去的。当搜到第一个到达终点的最短路径后，继续搜索，会返回其他可能不是最短的路径。

（4）题目要求返回字典序最小的最短路径，那么只要在每次扩散下一层（往BFS的队列中加入下一层的结点）时，都按字典序“D<L<R<U”的顺序来加下一层的结点，那么第一个搜到的最短路径就是字典序最小的那个。

所以本题就是一个基本的BFS搜索最短路。复杂度只有$O(n)$，n是迷宫内结点的总数，本题有$30 \times 50 = 1500$个点。因为每个点只用搜一次，即进入队列和出队列一次。

本题另一个有用的地方是路径打印，下面给出**两种打印方法。**

3、两种路径打印方法

路径如何打印？一个简单的办法是：每扩展到一个点v，都在v上存储从起点s到v的完整路径$path$。到达终点t时，就得到了从起点s到t的完整路径。这样做的缺点是会占用大量空

间，因为每个点上都存储了完整的路径。

其实不用在结点上存储完整路径，而是在每个点上记录它的前驱结点就够了，这样从终点能一步步回溯到起点，得到一条完整路径（详细解释见《算法竞赛入门到进阶》245页“打印最短路径”）。称这种路径记录方法为“标准方法”。

下面用代码分别演示这两种路径打印方法。

3.1 简单方法

确实非常简单，到达终点后，用 `cout<<now.p<<endl;` 一句就打印出了完整路径。

（注意，下面2个代码，提交到oj.ecustacm.cn，都是“答案错误”，因为这是个填空题，没有输入，只有输出。可以自己运行得到答案后，提交到OJ时，直接打印答案就行了。）

```
#include<bits/stdc++.h>
using namespace std;
struct node{
    int x;
    int y;
    string p;  //path,记录从起点(0,0)到这个点(x,y)的完整路径
};
char a[31][51];  //存地图

char k[4]={'D','L','R','U'};
int dir[4][2]={{1,0},{0,-1},{0,1},{-1,0}};
int vis[30][50];  //标记。vis=1: 已经搜过，不用再搜
void bfs(){
    node start; start.x=0;  start.y=0;   start.p="";  //定义起点
    vis[0][0]=1;               //标记起点被搜过

    queue<node>q;   q.push(start);   //把第一个点放进队列，开始BFS
    while(!q.empty()){
        node now = q.front();  //取出队首
        q.pop();
        if(now.x==29 && now.y==49){ //第一次达到终点，这就是字典序最小的最短路径
            cout<<now.p<<endl;        //打印路径：从(0,0)到(29,49)
            return;
        }
        for(int i=0;i<4;i++){  //扩散邻居结点
            node next;
            next.x = now.x+dir[i][0];
            next.y = now.y+dir[i][1];
            if(next.x<0||next.x>=30||next.y<0||next.y>=50)  //越界了
                continue;
            if(vis[next.x][next.y]==1||a[next.x][next.y]=='1')  //vis=1:已经搜过;
```

```
                continue;
            vis[next.x][next.y]=1;  //标记被搜过
            next.p = now.p+k[i];  //记录完整路径：把上一个点的路径，加上这一步后，复制给
            q.push(next);
        }
    }
}
int main(){
    for(int i=0;i<30;i++)  cin>>a[i];  //读题目给的地图数据
    bfs();
}
```

3.2 标准方法

注意看 print_path()，它是递归函数，先递归再打印。从终点开始，回溯到起点后，再按从起点到终点的顺序，正序打印出完整路径。

```
//User: 19031010128
#include<bits/stdc++.h>
using namespace std;
struct node{
    int x;
    int y;
};
char a[31][51];  //存地图

char k[4]={'D','L','R','U'};
int dir[4][2]={{1,0},{0,-1},{0,1},{-1,0}};
int vis[30][50];  //1:已经搜过，不用再搜

char pre[31][51]; //用于查找前驱点。例如pre[x][y] = 'D'，表示上一个点往下走一步到了(x,y
void print_path(int x,int y){      //打印路径：从(0,0)到(29,49)
  if(x==0 && y==0)   //回溯到了起点，递归结束，返回
     return;
  if(pre[x][y]=='D')  print_path(x-1,y);   //回溯，往上 U
  if(pre[x][y]=='L')  print_path(x,  y+1); //回溯，往右 R
  if(pre[x][y]=='R')  print_path(x,  y-1);
  if(pre[x][y]=='U')  print_path(x+1,y);
  printf("%c",pre[x][y]);                  //最后打印的是终点
}

void bfs(){
    node now,next;
```

```
    queue<node>q;
    now.x=0;
    now.y=0;
    vis[0][0]=1;
    q.push(now);

    while(!q.empty()){
        now=q.front();
        q.pop();
        if(now.x==29 && now.y==49){ //第一次达到终点，这就是字典序最小的最短路径
            print_path(29,49);      //打印路径，从终点回溯到起点。但是打印出来是从起点到
            return;
        }
        for(int i=0;i<4;i++){
            next.x = now.x+dir[i][0];
            next.y = now.y+dir[i][1];
             if(next.x<0||next.x>=30||next.y<0||next.y>=50)  //越界
                continue;
            if(vis[next.x][next.y]==1||a[next.x][next.y]=='1')  //vis=1:已经搜过;
                continue;
            vis[next.x][next.y]=1;
            pre[next.x][next.y] = k[i];   //记录点(x,y)的前驱
            q.push(next);
        }
    }
}
int main(){
    for(int i=0;i<30;i++)  cin>>a[i];  //读题目给的地图数据，30行，每行50个
    bfs();
}
```

参考文献：《算法竞赛入门到进阶》 清华大学出版社，网购：京东 当当