　在很久很久以前，有一位大师级程序员，实力高强，深不可测，代码能力无人能及。从来没有人听说过他的真名，只知道他在完成一段代码后，总会跟上一行注释“十四出品，必属精品”，于是他在编程江湖上便有了绰号“十四”。

　　随着十四大师声名远播，意图登门拜访，寻求编程秘法的人也渐渐多了起来。然而，正如他无人知晓的真名一般，十四大师的真面目也少有人得见。这并不是因为大师住处隐秘，而是因为大师居所前有着他亲自布下的阵法，它困住了无数虔诚的求知者。

　　作为十四大师崇拜者的小胖经过苦心探寻，终于有一天得到了法阵的地图。

　　地图显示，法阵是方形的，纵横皆为五里，在地图上简示为5\*5的矩阵，且只由0或1组成。其中，0表示可以走的路，1表示阻止通行的屏障。左上角和右下角分别是阵的入口和出口，这两个位置的数字保证为0。

　　既然得到了地图，破解法阵自然不再是难事。现在，小胖不仅想要走出法阵，还想知道怎样才能用最短的路线走出法阵。

**Input**

　　输入是一个5 × 5的二维数组，仅由0、1两数字组成，表示法阵地图。

**Output**

　　输出若干行，表示从左上角到右下角的最短路径依次经过的坐标，格式如样例所示。数据保证有唯一解。

**Sample Input**

0 1 0 0 0  
0 1 0 1 0  
0 0 0 0 0  
0 1 1 1 0  
0 0 0 1 0

**Sample Output**

(0, 0)  
(1, 0)  
(2, 0)  
(2, 1)  
(2, 2)  
(2, 3)  
(2, 4)  
(3, 4)  
(4, 4)

**Hint**

　　坐标(x, y)表示第x行第y列，行、列的编号从0开始，且以左上角为原点。

　　另外注意，输出中分隔坐标的逗号后面应当有一个空格。

#include<iostream>

#include<stdio.h>

using namespace std;

int dx[]={0,0,1,-1};

int dy[]={1,-1,0,0};

int a[10][10];

struct node

{

int x,y,pre;

};

node q[500];

void print(int t)

{

if(q[t].pre!=-1)

{

print(q[t].pre);

cout<<"("<<q[t].x<<", "<<q[t].y<<")\n";

}

}

void bfs(int x1,int y1)

{

int head(0),tail(1),xx,yy;

q[head].x=x1;

q[head].y=y1;

q[head].pre=-1;

while(head<tail)

{

for(int i=0;i<4;i++)

{

xx=q[head].x+dx[i];

yy=q[head].y+dy[i];

if(xx==4&&yy==4)print(head);

if(xx>4||yy>4|xx<0||yy<0||a[xx][yy])continue;

else

{

a[xx][yy]=1;

q[tail].x=xx;

q[tail].y=yy;

q[tail].pre=head;

tail++;

}

}

head++;

}

}

int main()

{

//freopen("input.txt","r",stdin);

for(int i=0;i<=4;i++)

for(int j=0;j<=4;j++)

cin>>a[i][j];

cout<<"(0, 0)\n";

bfs(0,0);

cout<<"(4, 4)";

}

//这个不存在走过了就不能走的问题，所以没有开个visit数组存走过没有

//并且这个也没有墙

运用了结构体和自己写的队列

这就不能用queue，因为回头来找父节点的时候还要之前的点在队列中，而不能出队，只是头指针和为指针动动

最先到的一定是最短路