As we all know, Harry Porter learns magic at Hogwarts School. However, learning magical knowledge alone is insufficient to become a great magician. Sometimes, Harry also has to gain knowledge from other certain subjects, such as language, mathematics, English, and even algorithm.   
  Dumbledore, the headmaster of Hogwarts, is planning to construct a new teaching building in his school. The area he selects can be considered as an n\*m grid, some (but no more than ten) cells of which might contain stones. We should remove the stones there in order to save place for the teaching building. However, the stones might be useful, so we just move them to the top-left cell. Taking it into account that Harry learned how to operate dig machine in Lanxiang School several years ago, Dumbledore decides to let him do this job and wants it done as quickly as possible. Harry needs one unit time to move his dig machine from one cell to the adjacent one. Yet skilled as he is, it takes no time for him to move stones into or out of the dig machine, which is big enough to carry infinite stones. Given Harry and his dig machine at the top-left cell in the beginning, if he wants to optimize his work, what is the minimal time Harry needs to finish it?

**Input**

They are sever test cases, you should process to the end of file.   
For each test case, there are two integers n and m.(1≤n,m≤50)(1≤n,m≤50).   
The next n line, each line contains m integer. The j-th number of ithith line a[i][j] means there are a[i][j] stones on the jthjth cell of the ithith line.( 0≤a[i][j]≤1000≤a[i][j]≤100 , and no more than 10 of a[i][j] will be positive integer).

**Output**

For each test case, just output one line that contains an integer indicate the minimal time that Harry can finish his job.

**Sample Input**

3 3

0 0 0

0 100 0

0 0 0

2 2

1 1

1 1

**Sample Output**

4

4

状压dp入门题目

bc上的一道题，刚开始想用这个方法做的，因为刚刚做了一个[类似的题](http://www.cnblogs.com/bfshm/p/4032042.html" \t "_blank)，但是想到这只是bc的第二题，

以为用bfs水一下就过去了，结果MLE了，因为bfs的队列里的状态太多了，耗内存太厉害。

题意：

从某一点出发，遍历网格上的一些点，每个点至少访问一次需要的最小时间是多少。

官方题解：

由于Harry的dig machine是无限大的，而装载石头和卸载石头是不费时间的，所以问题可以转化成：从某一点出发，遍历网格上的一些点，每个点至少访问一次需要的最小时间是多少。这就是经典的旅行商问题，考虑到我们必须要遍历的点只有不到10个，可以用状态压缩解决。

Dp[i][j]表示i状态的点被访问过了，当前停留在点j 需要的最少时间。枚举另一点不在i状态内的点k，从点j节点走向点k，状态转移

Dp[i|(1≪k)][k]=min(Dp[i|(1≪k)][k],Dp[i][j]+Dis(j,k))

其中Dis(j,k)表示点j与点k的最短距离，这个可以通过坐标O(1)计算得到。若有t个点包含石头，则算法复杂度为O(n∗m+(t2)∗(2t))。

题意：到达所有正值点回到起点花费的最小时间  
做法：直接找出正值点，二进制状态压缩，表示点是否到达  
dp[state][j]表示到达过的点为state最后一个到达的点为j的最小时间  
详细转移见代码，很水的状态压缩，注意边界值的特判即可

这个不能用n次dijkstra是因为点只能过一次而dijkstra不确定一个点过了多少次

时间复杂度 O(n\*m+t2+2t)

求两点之间距离的初始操作是n\*m

#include <iostream>

#include <cstdio>

#include <cmath>

#include <queue>

#include <cstring>

#include <cstdlib>

#include <algorithm>

#define LL long long

#define INF 0x3f3f3f3f

const int maxn = 50+10;

using namespace std;

struct node

{

int x, y;

} p[maxn];

int a[maxn][maxn], c[maxn][maxn], d[(1<<11)+10][15];

int main()

{

int i, j, k, n, m, cnt;

while(~scanf("%d%d", &n, &m))

{

memset(d,INF,sizeof(d));

cnt = 0;

for(i = 0; i < n; i++)

for(j = 0; j < m; j++)

{

scanf("%d", &a[i][j]);

if(i==0 && j==0)//起点

{

p[cnt].x = i;

p[cnt].y = j;

cnt++;

}

else if(a[i][j]>0)//正值点

{

p[cnt].x = i;

p[cnt++].y = j;

}

}

for(i = 0; i < cnt; i++)

{

for(j = i+1; j < cnt; j++)

c[i][j] = c[j][i] = abs(p[i].x-p[j].x)+abs(p[i].y-p[j].y);

c[i][i] = 0;

}

d[0][0]=0;

for(int state = 0;state<(1<<cnt); state++)//i是从0到2^n

{ //这里i就相当于state

for(j = 0;j<cnt;j++)

if(d[state][j]!=INF) //注意保证该状态存在

for(k = 0;k<cnt;k++)//k是枚举每个点，用来找还没有走过的

{

if((state&(1<<k))>0) continue; //数字i的二进制下第k位是1,也就是这个点已经走过了

if(c[j][k]==INF) continue; //两个点之间不可达

d[state|(1<<k)][k] = min(d[state|(1<<k)][k], d[state][j]+c[j][k]);//用j这个点来更新state和k点，有点dijkstra的味道

//下一步走k这个点

}

}

//在装压dp里用了n次dijkstra

int ans = INF;

for(i=0;i<cnt;i++)

ans=min(d[(1<<cnt)-1][i]+c[0][i],ans);//因为是从0开始的，所以是2^cnt-1

cout<<ans<<endl;

}

return 0;

}