# 1. Rescue

# Code：

#include<iostream>

#include<cstring>

#define Max 40100

using namespace std;

struct List{

int x; //横坐标

int y; //纵坐标

int sum; //总步数

};

char a[210][210];

int vis[210][210];

int Next[4][2] = {

{1, 0},

{-1, 0},

{0,1},

{0,-1}};

int main() {

ios::sync\_with\_stdio(false);

int n, m; while(cin>>n>>m) {

//初始化与定义变量

memset(vis, 0, sizeof(vis)); //每次循环vis数组都需初始化

int startx, starty; //起点的x、y坐标

int Min = 999999999; //最后的输出结果

bool flag = false; //终止循环的标记

//输入

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

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

cin>>a[i][j];

if(a[i][j] == 'a') startx=i, starty=j;

}

//队列初始化

int head=0, tail=0; //头结点，尾结点

List l[Max];

l[tail].x = startx;

l[tail].y = starty;

l[tail].sum = 0;

tail++;

vis[startx][starty] = 1;

flag = 0; //标记是否到达目标点

//当目标不为空时循环

while(head < tail) { //如果尾指针小于等于头指针，代表队列中无元素，返回

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

int tx = l[head].x + Next[i][0];

int ty = l[head].y + Next[i][1];

//终止本次循环的三种条件： 越界、该点是障碍物、已经被搜索过。

//1、若越界则返回

if(tx<0 || tx>=n || ty<1 || ty>=m) continue;

//2、若没被搜索过且该点不是障碍物，则：

if(!vis[tx][ty])

if(a[tx][ty] != '#') {

vis[tx][ty]=1;

l[tail].x = tx;

l[tail].y = ty;

//如果是x，则走两步

if(a[tx][ty] == 'x') l[tail].sum = l[head].sum+2;

else l[tail].sum=l[head].sum+1;

tail++; //每次循环后，tail都需自加

}

if(a[tx][ty]=='r') //若等于r，则与Min比较，取较小值。

Min = min(l[tail-1].sum, Min);

}

head++; //每次循环都要丢弃头部元素

}

//输出

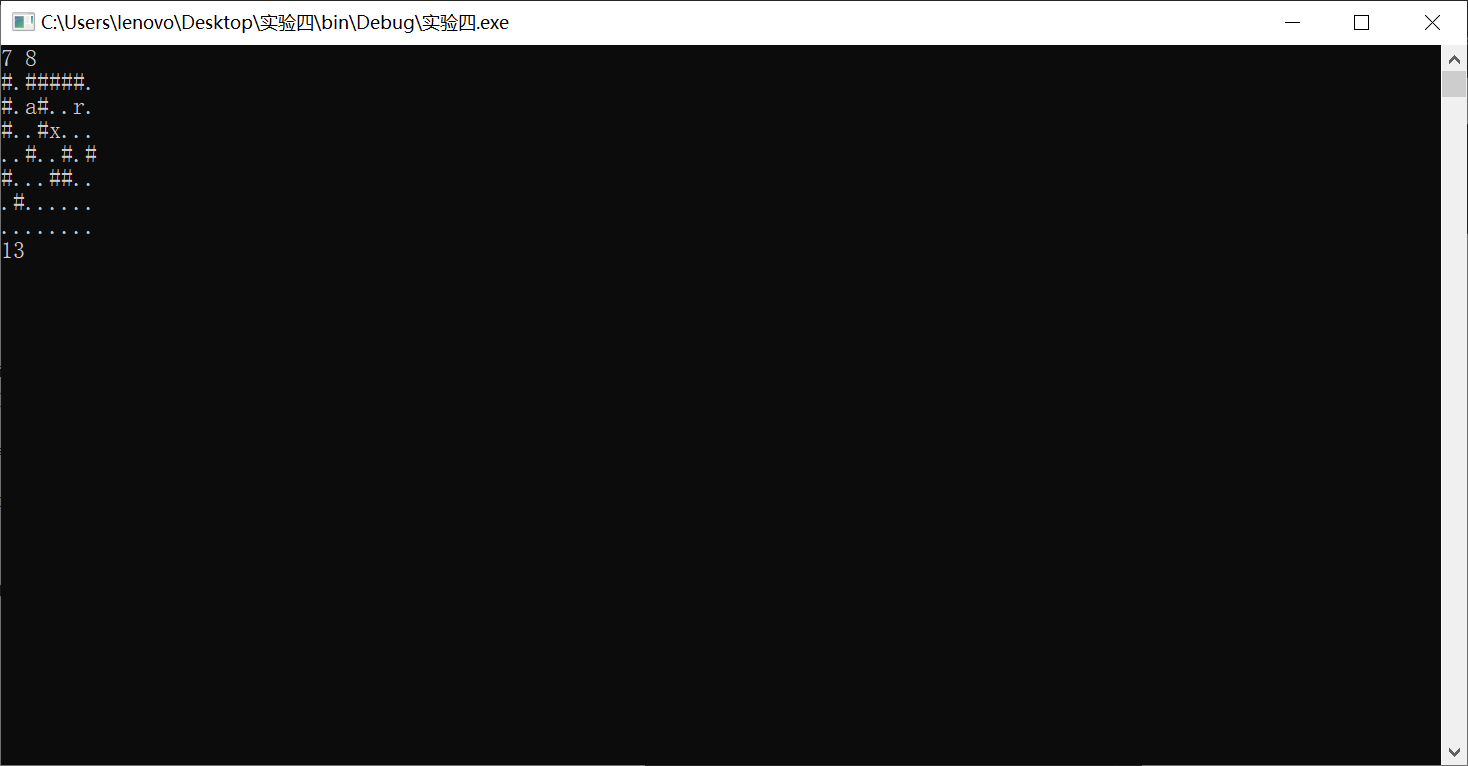
if(Min == 999999999) cout << "Poor ANGEL has to stay in the prison all his life." << endl;

else cout << Min << endl;

}

return 0;

}



# 1. I NEED A OFFER!

#include<stdio.h>

#include<string.h>

double dp[10000];

double minn(double a,double b)

{

if(a>b)return b;

else

return a;

}

int main()

{

int n,m,j;

while(scanf("%d%d",&n,&m)!=EOF,n||m)

{

int i,v[10000];

double f[10000];

for(i=1;i<=m;i++)

{

scanf("%d%lf",&v[i],&f[i]);

f[i]=1-f[i];

}

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

dp[i]=1.0;

for(i=1;i<=m;i++)

{

for(j=n;j>=v[i];j--)

{

dp[j]=minn(dp[j],dp[j-v[i]]\*f[i]);

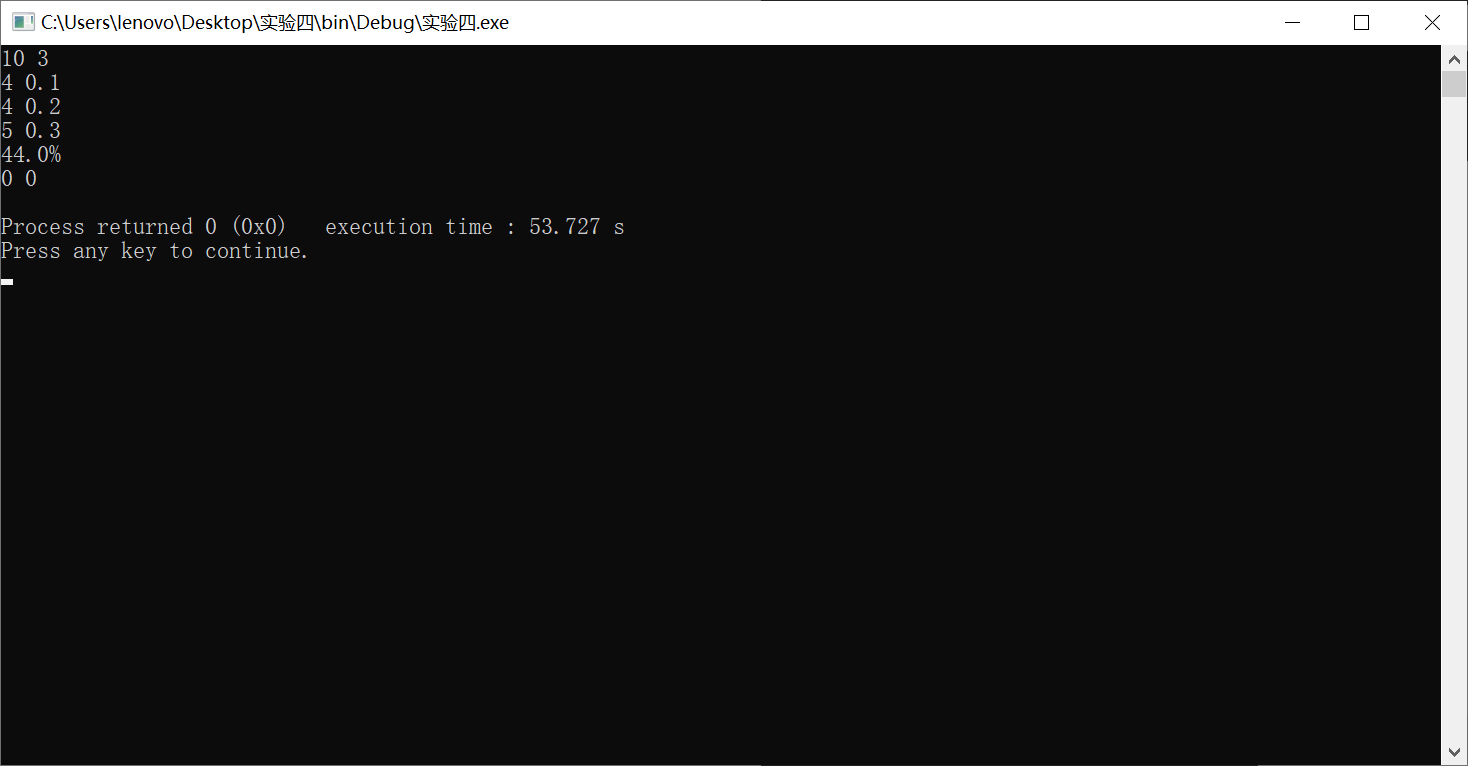
}

}

printf("%.1lf%%\n",(1-dp[n])\*100);

}

}



# 1. 非常可乐

#include <iostream>

#include <string.h>

#include <stdio.h>

#include <queue>

using namespace std;

int s,n,m;

int vis[105][105][105];

struct node

{

int s, n, m, step; //三个杯子的可乐和步数

};

int check(int x,int y,int z) //平分条件

{

if(x == 0 && y == z)

return 1;

if(y == 0 && x == z)

return 1;

if(z == 0 && x == y)

return 1;

return 0;

}

int bfs()

{

//三个杯的情况

queue<node> Q;

//next是下一次三个杯的情况

node a, next;

a.s = s;

a.n = 0;

a.m = 0;

a.step = 0;

//是否出现过这种情况

vis[s][0][0] = 1;

Q.push(a);

while(!Q.empty())

{

//目前杯的情况

a = Q.front();

Q.pop();

//如果成功返回步数

if(check(a.s, a.n, a.m))

return a.step;

if(a.n) //当n杯中还有

{

if(a.n > s - a.s) //将n杯倒入s杯中能将s杯倒满

{

//下一次三个杯的情况

next = a;

next.n = next.n - (s - a.s);

next.s = s;

//没倒成这种情况（如果之前出现过这种情况就没意义了）

if(!vis[next.s][next.n][next.m])

{

next.step = a.step + 1;

Q.push(next);

vis[next.s][next.n][next.m] = 1;

}

}

else //将n杯倒入s杯中不能将s杯倒满

{

next = a;

next.s = next.n + next.s;

next.n = 0;

if(!vis[next.s][next.n][next.m])

{

next.step = a.step+1;

Q.push(next);

vis[next.s][next.n][next.m] = 1;

}

}

if(a.n > m - a.m)//将n杯倒入m杯中能将m杯倒满

{

next = a;

next.n = next.n - (m - a.m);

next.m = m;

if(!vis[next.s][next.n][next.m])

{

next.step = a.step+1;

Q.push(next);

vis[next.s][next.n][next.m] = 1;

}

}

else//将n杯倒入m杯中不能将m杯倒满

{

next = a;

next.m = next.n+next.m;

next.n = 0;

if(!vis[next.s][next.n][next.m])

{

next.step = a.step+1;

Q.push(next);

vis[next.s][next.n][next.m] = 1;

}

}

}

if(a.m) //同上

{

if(a.m>s-a.s)

{

next = a;

next.m = next.m-(s-a.s);

next.s = s;

if(!vis[next.s][next.n][next.m])

{

next.step = a.step+1;

Q.push(next);

vis[next.s][next.n][next.m] = 1;

}

}

else

{

next = a;

next.s = next.m+next.s;

next.m = 0;

if(!vis[next.s][next.n][next.m])

{

next.step = a.step+1;

Q.push(next);

vis[next.s][next.n][next.m] = 1;

}

}

if(a.m>n-a.n)

{

next = a;

next.m = next.m-(n-a.n);

next.n = n;

if(!vis[next.s][next.n][next.m])

{

next.step = a.step+1;

Q.push(next);

vis[next.s][next.n][next.m] = 1;

}

}

else

{

next = a;

next.n = next.m+next.n;

next.m = 0;

if(!vis[next.s][next.n][next.m])

{

next.step = a.step+1;

Q.push(next);

vis[next.s][next.n][next.m] = 1;

}

}

}

if(a.s) //同上

{

if(a.s > n - a.n)

{

next = a;

next.s = next.s-(n-a.n);

next.n = n;

if(!vis[next.s][next.n][next.m])

{

next.step = a.step+1;

Q.push(next);

vis[next.s][next.n][next.m] = 1;

}

}

else

{

next = a;

next.n = next.s+next.n;

next.s = 0;

if(!vis[next.s][next.n][next.m])

{

next.step = a.step+1;

Q.push(next);

vis[next.s][next.n][next.m] = 1;

}

}

if(a.s>m-a.m)

{

next = a;

next.s = next.s-(m-a.m);

next.m = m;

if(!vis[next.s][next.n][next.m])

{

next.step = a.step+1;

Q.push(next);

vis[next.s][next.n][next.m] = 1;

}

}

else

{

next = a;

next.m = next.m+next.s;

next.s = 0;

if(!vis[next.s][next.n][next.m])

{

next.step = a.step+1;

Q.push(next);

vis[next.s][next.n][next.m] = 1;

}

}

}

}

return 0;

}

int main()

{

int ans;

while(cin >> s >> n >> m)

{

if(s == 0)

break;

if(s % 2) //奇数肯定不能平分，因为杯子是整数体积大小

{

cout << "NO" << endl;

continue;

}

memset(vis, 0, sizeof(vis));

ans = bfs();

if(ans)

cout << ans << endl;

else

cout << "NO" << endl;

}

return 0;

}

# 

**7. A计划**

#include<stdio.h>

#include<queue>

#include<string.h>

#include<algorithm>

using namespace std;

int book[2][15][15];

char a[2][15][15];

struct node

{

int x,y,z,step;

};

int main()

{

int i,j,k,f,n,m,t,T;

int ta,tb,tc,tx,ty,tz;

int next [4][2]= {0,1,1,0,0,-1,-1,0};

scanf("%d",&T);

while(T--)

{

scanf("%d%d%d",&n,&m,&t);

getchar();

for(i=0; i<2; i++)

{

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

{

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

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

getchar();

}

if(i!=1) getchar();//这个吸收中间那个空白行，注意只有一个空白行。

}

for(i=0; i<2; i++)

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

{

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

if(a[i][j][k]=='P')//记录公主的位置，顺便把'P'改为'.',(纯属个人习惯)。

{

ta=i;

tb=j;

tc=k;

a[i][j][k]='.';

}

else if(a[i][j][k]=='#'&&a[i^1][j][k]=='#')//如果这一层是#，另一层还是#，直接标记为墙就行。

{

a[0][j][k]='\*';

a[1][j][k]='\*';

}

else if(a[i][j][k]=='#'&&a[i^1][j][k]=='\*')//这一层是#，另一层是墙，这样会撞死（骑士也不傻）。

{

a[0][j][k]='\*';

a[1][j][k]='\*';

}

}

memset(book,0,sizeof(book));

book[0][0][0]=1;

queue<node>Q;

node p,q;

f=0;

p.x=0,p.y=0,p.z=0,p.step=0;

Q.push(p);

while(!Q.empty())

{

p=Q.front();

Q.pop();

if(p.x==ta&&p.y==tb&&p.z==tc&&p.step<=t&&f==0)

{

f=1;

// printf("%d\n",p.step);

printf("YES\n");

break;

}

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

{

tx=p.x;

ty=p.y+next[i][0];

tz=p.z+next[i][1];

if(a[tx][ty][tz]=='\*'||book[tx][ty][tz]==1||tx<0||tx>1||ty<0||ty>n-1||tz<0||tz>m-1)

continue;

if(a[tx][ty][tz]=='.')

{

book[tx][ty][tz]=1;

q.x=tx,q.y=ty,q.z=tz;

q.step=p.step+1;

Q.push(q);

}

else if(a[tx][ty][tz]=='#')

{

book[tx][ty][tz]=1;

if(tx==0)//传送门可以换层（(⊙o⊙)哦，传送门是什么东东，怎么听着这么熟悉）。

tx+=1;

else tx-=1;

if(a[tx][ty][tz]=='\*'||book[tx][ty][tz]==1||tx<0||tx>1||ty<0||ty>n-1||tz<0||tz>m-1)

continue;

book[tx][ty][tz]=1;

q.x=tx,q.y=ty,q.z=tz;

q.step=p.step+1;

Q.push(q);

}

}

}

while(!Q.empty())

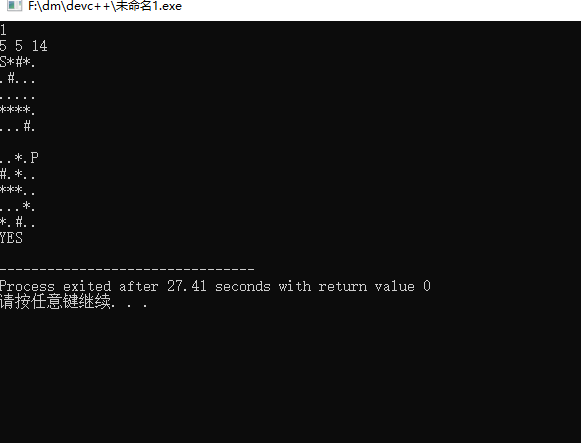
Q.pop();

if(!f) printf("NO\n");

}

return 0;

}



# 哈密顿绕行世界问题

#include<iostream>

#include<cstdio>

#include<cstring>

#include<algorithm>

#include<cmath>

using namespace std;

int map[21][4],m,x=1;

int a[30],b[30];

int dfs(int q,int len,int p){

a[len]=q;

b[q]=1;

for(int i=0;i<3;i++){

int t=map[q][i];

if(t==p&&len==19){

printf("%d: ",x++);

for(int j=0;j<20;j++){

cout<<a[j]<<" ";

}

cout<<p<<endl;

}

if(b[t]==0)dfs(t,len+1,p);

}

b[q]=0;

}

int main(){

for(int i=1;i<=20;i++)

scanf("%d%d%d",&map[i][0],&map[i][1],&map[i][2]);

while(scanf("%d",&m)&&m!=0){

memset(b,0,sizeof(b));

dfs(m,0,m);

}

return 0;

}

