**算法训练 Lift and Throw**

时间限制：3.0s   内存限制：256.0MB

问题描述

　　给定一条标有整点(1, 2, 3, ...)的射线. 定义两个点之间的距离为其下标之差的绝对值.  
　　Laharl, Etna, Flonne一开始在这条射线上不同的三个点, 他们希望其中某个人能够到达下标最大的点.  
　　每个角色只能进行下面的3种操作, 且每种操作不能每人不能进行超过一次.  
　　1.移动一定的距离  
　　2.把另一个角色高举过头  
　　3.将举在头上的角色扔出一段距离  
　　每个角色有一个movement range参数, 他们只能移动到没有人的位置, 并且起点和终点的距离不超过movement range.  
　　如果角色A和另一个角色B距离为1, 并且角色B没有被别的角色举起, 那么A就能举起B. 同时, B会移动到A的位置,B原来所占的位置变为没有人的位置. 被举起的角色不能进行任何操作, 举起别人的角色不能移动.同时, 每个角色还有一个throwing range参数, 即他能把举起的角色扔出的最远的距离. 注意, 一个角色只能被扔到没有别的角色占据的位置. 我们认为一个角色举起另一个同样举起一个角色的角色是允许的. 这种情况下会出现3个人在同一个位置的情况. 根据前面的描述, 这种情况下上面的两个角色不能进行任何操作, 而最下面的角色可以同时扔出上面的两个角色. 你的任务是计算这些角色能够到达的位置的最大下标, 即最大的数字x, 使得存在一个角色能够到达x.

输入格式

　　输入共三行, 分别为Laharl, Etna, Floone的信息.  
　　每一行有且仅有3个整数, 描述对应角色的初始位置, movement range, throwing range.  
　　数据保证3个角色的初始位置两两不相同且所有的数字都在1到10之间.</div>

输出格式

　　仅有1个整数, 即Laharl, Etna, Flonne之一能到达的最大距离.

样例输入

9 3 3  
4 3 1  
2 3 3

样例输出

15

样例说明

　　一开始Laharl在位置9, Etna在位置4, Flonne在位置2.  
　　首先, Laharl移动到6.  
　　然后Flonne移动到位置5并且举起Etna.  
　　Laharl举起Flonne将其扔到位置9.  
　　Flonne把Etna扔到位置12.  
　　Etna移动到位置15.

本题的C++参考代码如下：

#include<cstdio>

#include<iostream>

#include<cmath>

#include<cstring>

#include<cstdlib>

#include<algorithm>

#include<map>

using namespace std;

bool bst;

int i,j,k,l,n,m,s,an,tot,len,L,P;

class arr

{

public:

int d,m,r,son,fa;

bool f1,f2,f3;

}a[4];

bool ff[47][47][47][8][8];

void dfs(int n1,int n2)

{

if (ff[a[1].d][a[2].d][a[3].d][n1][n2]) return;

for (int i=1;i<=3;i++)

if (a[i].d<=42) an=max(an,a[i].d);

ff[a[1].d][a[2].d][a[3].d][n1][n2]=true;

int mm=9999;

for (int i=1;i<=3;i++)

mm=min(mm,a[i].d);

int ju[4];

memset(ju,0,sizeof(ju));

for (int i=1;i<=3;i++) if (a[i].d>42) ju[a[i].d-42]=i;

for (int i=1;i<=3;i++)

if (a[i].d<=42)

if (!(n1&(1<<(i-1))))

if (!ju[i])

{

for (int j=max(mm,a[i].d-a[i].m);j<=a[i].d+a[i].m;j++)

{

int pp=a[i].d;

a[i].d=j;

dfs(n1|(1<<(i-1)),n2);

a[i].d=pp;

}

}

for (int i=1;i<=3;i++)

if (a[i].d<=42)

if (!(n2&(1<<(i-1))))

if (!ju[i])

{

for (int j=1;j<=3;j++)

if (abs(a[j].d-a[i].d)==1)

{

int pp=a[j].d;

a[j].d=42+i;

dfs(n1,n2|(1<<(i-1)));

a[j].d=pp;

}

}

for (int i=1;i<=3;i++)

if (a[i].d<=42)

if (ju[i])

{

int y=ju[i];

for (int j=max(mm,a[i].d-a[i].r);j<=a[i].d+a[i].r;j++)

{

a[y].d=j;

dfs(n1,n2);

a[y].d=42+i;

}

}

}

int main()

{

// freopen("throw.out","w",stdout);

//freopen("a.in","r",stdin);

n=3;

while (scanf("%d%d%d",&a[1].d,&a[1].m,&a[1].r)==3)

{

memset(ff,0,sizeof(ff));

// memset(a,0,sizeof(a));

for (int i=2;i<=n;i++)

scanf("%d%d%d",&a[i].d,&a[i].m,&a[i].r);

an=0;

dfs(0,0);

printf("%d\n",an);

}

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

#include <string.h>

#define TRUE 1

#define FALSE 0

#define max(a, b) a > b ? a : b

int x[4];

int l[4];

int t[4];

int ans = 0;

int w[4];

int p[4];

int a[4] = {3, 3, 3, 3};

int near(int s)

{

int i = 1;

for (; i <= 3; i++)

{

if (s == x[i] + 1 || s == x[i] - 1)

{

return TRUE;

}

}

return FALSE;

}

//dfs深度遍历

void dfs(int d)

{

int i = 1, j = 1, e = 0;

//每次都取最远（大）的位置

for (; i <= 3; i++)

{

ans = max(ans, x[i]);

}

for (i = 1; i <= 3; i++)

{

//是否可以进行操作

if (w[i])

{

continue;

}

//a[i] == 1 || a[i] == 3（未进行移动且不可抛出）

if ((a[i] & 1) && !(a[i] & 4))

{

for (j = 1; j <= l[i]; j++)

{

x[i] += j;

a[i] ^= 1;

if (near(x[i]) || j == l[i])

{

dfs(d + 1);

}

x[i] -= j;

x[i] -= j;

if (near(x[i]) || j == l[i])

{

dfs(d + 1);

}

x[i] += j;

a[i] ^= 1;

}

}

//a[i] == 2 || a[i] == 3 || a[i] == 5（未进行举起）

if (a[i] & 2)

{

for (j = 1; j <= 3; j++)

{

if (i != j && !w[j] && t[i] > 0)

{

if (x[i] == x[j] + 1 || x[j] == x[i] + 1)

{

w[j] = 1;

a[i] ^= 2;

a[i] ^= 4;

p[i] = j;

e = x[j];

x[j] = -j;

dfs(d + 1);

x[j] = e;

w[j] = 0;

a[i] ^= 2;

a[i] ^= 4;

}

}

}

}

if (a[i] & 4)

{

for (j = 1; j <= t[i]; j++)

{

w[p[i]] = 0;

a[i] ^= 4;

e = x[p[i]];

x[p[i]] = x[i] + j;

if (near(x[p[i]]) || j == t[i])

{

dfs(d + 1);

}

x[p[i]] -= j;

x[p[i]] -= j;

if (near(x[p[i]]) || j == t[i])

{

dfs(d + 1);

}

x[p[i]] = e;

a[i] ^= 4;

w[p[i]] = 1;

}

}

}

return ;

}

int main()

{

int i = 1;

for (; i <= 3; i++)

{

scanf("%d %d %d", &x[i], &l[i], &t[i]);

}

dfs(1);

printf("%d\n", ans);

return 0;

}

本题的Java参考代码如下：

import java.util.Scanner;

public class Main {

static int ans;

static int[] mov = new int[4],

thr = new int[4],

init = new int[4],

lift = new int[4];

static boolean[] islifted = new boolean[4],

ismov = new boolean[4],

isthr = new boolean[4];

//static int[][] hh = new int[1000][4];

public static void main(String[] args){

Scanner in = new Scanner(System.in);

for(int i = 1; i < 4; i++){

init[i] = in.nextInt();

mov[i] = in.nextInt();

thr[i] = in.nextInt();

}

new Main().dfs(1);

System.out.println(ans);

}

public boolean near(int s){

for(int i = 1; i < 4; i++){

if(Math.abs(s - init[i]) == 1)

return true;

}

return false;

}

public void dfs(int cur){

/\*for(int i = 1; i < 4; i++)

hh[cur][i] = init[i];\*/

for(int i = 1; i < 4; i++)

if(ans < init[i]){

ans = init[i];

/\*System.out.println("cur = " + cur);

for(int k = 1; k <= cur; k++){

for(int j = 1; j < 4; j++)

System.out.print(hh[k][j] + " ");

System.out.println();

}\*/

}

for(int i = 1; i < 4; i++){

if(islifted[i])continue;

if(!ismov[i] && lift[i] == 0){

for(int j = 1; j <= mov[i]; j++){

init[i] += j;

ismov[i] = true;

if(near(init[i]) || j == mov[i])

dfs(cur + 1);

init[i] -= j \* 2;

if(near(init[i]) || j == mov[i])

dfs(cur + 1);

init[i] += j;

ismov[i] = false;

}

}

if(lift[i] == 0 && !isthr[i]){

for(int j = 1; j < 4; j++){

if(!islifted[j] && Math.abs(init[i] - init[j]) == 1 && thr[i] > 0){

islifted[j] = true;

lift[i] = j;

int e = init[j];

init[j] = -j;

dfs(cur + 1);

islifted[j] = false;

lift[i] = 0;

init[j] = e;

}

}

}

if(lift[i] != 0 && !isthr[i]){

for(int j = 1; j <= thr[i]; j++){

int k = lift[i];

islifted[k] = false;

isthr[i] = true;

lift[i] = 0;

int e = init[k];

init[k] = init[i] + j;

if(near(init[k]) || j == thr[i])

dfs(cur + 1);

init[k] -= j \* 2;

if(near(init[k]) || j == thr[i])

dfs(cur + 1);

islifted[k] = true;

isthr[i] = false;

init[k] = e;

lift[i] = k;

}

}

}

}

}