比赛链接

题解

E. Escape along Water Pipe

题意

给一张 $n×m$ 的图，起点在 $(1,1)$ 坐标点的上方，终点在 (n,m) 坐标点的下方。每个点有一个水管，其中 $(0,3)$ 是垂直的管，连接两个方向， 4 是水平管， 5 是竖直管，每个管子可以旋转 $90,180,270$ 度。问向起点灌水，能不能到达终点。

如果不能输出 NO 如果能，输出 YES 并输出方案，方案输出方式是 到达 (x,y) 输出 $0,x,y$ 如果需要旋转，请输出 $1,$ 旋转角度 $,$ 点横坐标 $,$ 点纵坐标。要求操作数不能超过 $20mn$ 。 T 组数据。

数据范围 $T≤10000,2≤m,n≤1000$ 并保证总共的 $n×m$ 不超过 10^{6}。

题解

一道恶心人的搜索/大模拟题。

显然需要 BFS 搜一下，并且对于不同的管子，用类似的处理方式。首先是最基础的 dx,dy 数组，表示位移方向。直管、弯管分别连接哪两个方向，要按照题目顺序标识，这样后面好处理。剩下就是标记是否到达，已经到达这个点(包括到达时的方向方向)上一个点的位置。然后就常规的 BFS 搜索一遍，如果存在解，从终点往回找，存起来，最后输出即可，主要是细节很多。复杂度 $O(n×m)$ 可以通过。

```
#include<bits/stdc++.h>
using namespace std;

int dx[5] = {-1,0,1,0}, dy[5] = {0,1,0,-1};
int a[1010][1010];
int wg[4][2] = {{3,0},{0,1},{1,2},{2,3}};
int zg[2][2] = {{1,3},{0,2}};
char dp[1010][1010][4];
int f[1010][1010][4];
int n,m;
deque<int> q;
vector<int> pans;
const int maxn=1000+5;

bool check(int x,int y) {
    return x>0&&x<=n+1&&y>0&&y<=m;
}

void dfs(int x,int y,int fx) {
    int from = f[x][y][fx];
```

```
    if(!from) return;
    int tmp=from;
    int lfx=tmp%4;
    tmp/=4;
    int ly=tmp%maxn;
    tmp/=maxn;
    int lx=tmp;
    dfs(lx,ly,lfx);
    if(a[lx][ly]<4) {
        for(int i=0; i<4; i++) {
            int tmpi=0;
            while(tmpi<2&&wg[i][tmpi]!=lfx) tmpi++;
            if(tmpi==2) continue;
            int llfx=wg[i][1-tmpi];
            int ffx=llfx>1?(llfx-2):(llfx+2);
            if(ffx==fx) {
                if(a[lx][ly]!=i) {
                    pans.push_back((i-a[lx][ly]+8)%4*maxn*maxn+lx*maxn+ly);
                    a[lx][ly]=i;
                }
                pans.push_back(lx*maxn+ly);
                return;
            }
        }
    } else {
        for(int i=0; i<2; i++) {
            int tmpi=0;
            while(tmpi<2&&zg[i][tmpi]!=lfx) tmpi++;
            if(tmpi==2) continue;
            int llfx=zg[i][1-tmpi];
            int ffx=llfx>1?(llfx-2):(llfx+2);
            if(ffx==fx) {
                if(a[lx][ly]-4!=i) {
                    //printf("%d %d %d %d
%d\n",maxn*maxn+lx*maxn+ly,lx,ly,a[lx][ly],i);
                    pans.push_back(maxn*maxn+lx*maxn+ly);
                    a[lx][ly]=i+4;
                }
                pans.push_back(lx*maxn+ly);
                return;
            }
        }
    }
}

int main() {
    int t;
    scanf("%d",&t);
    while(t--) {
        scanf("%d %d",&n,&m);
```

```
        for(int i=1; i<=n; i++) {
            for(int j=1; j<=m; j++) {
                scanf("%d",&a[i][j]);
            }
        }
        for(int i=1; i<=n+1; i++) {
            for(int j=1; j<=m; j++) {
                for(int k=0; k<4; k++)
                    dp[i][j][k]=f[i][j][k]=0;
            }
        }
        dp[1][1][0]=1;
        q.push_back(maxn*4+4);
        while(!q.empty()) {
            int qf=q.front();
            int tmp=qf;
            q.pop_front();
            int fx=tmp%4;
            tmp/=4;
            int y=tmp%maxn;
            tmp/=maxn;
            int x=tmp;
            if(tmp>n) continue;
            if(a[x][y]<4) {
                for(int i=0; i<4; i++) {
                    int tmpi=0;
                    while(tmpi<2&&wg[i][tmpi]!=fx) tmpi++;
                    if(tmpi==2) continue;
                    int lfx=wg[i][1-tmpi];
                    int xx=x+dx[lfx],yy=y+dy[lfx];
                    int ffx=lfx>1?(lfx-2):(lfx+2);
                    if(check(xx,yy)&&!dp[xx][yy][ffx]) {
                        dp[xx][yy][ffx]=1;
                        f[xx][yy][ffx]=qf;
                        q.push_back(xx*maxn*4+yy*4+ffx);
                        //printf("%d\n",xx*maxn*4+yy*4+ffx);
                    }
                }
            } else {
                for(int i=0; i<2; i++) {
                    int tmpi=0;
                    while(tmpi<2&&zg[i][tmpi]!=fx) tmpi++;
                    if(tmpi==2) continue;
                    int lfx=zg[i][1-tmpi];
                    int xx=x+dx[lfx],yy=y+dy[lfx];
                    int ffx=lfx>1?(lfx-2):(lfx+2);
                    if(check(xx,yy)&&!dp[xx][yy][ffx]) {
                        dp[xx][yy][ffx]=1;
                        f[xx][yy][ffx]=qf;
                        q.push_back(xx*maxn*4+yy*4+ffx);
                        //printf("%d\n",xx*maxn*4+yy*4+ffx);
```

```
                    }
                }
            }
        }
        if(!dp[n+1][m][0]) {
            puts("NO");
            continue;
        }
        puts("YES");
        pans.clear();
        dfs(n+1,m,0);
        int sz=pans.size();
        printf("%d\n",sz);
        for(int i=0; i<sz; i++) {
            int tmp=pans[i];
            if(tmp<maxn*maxn) {
                printf("%d %d %d\n",0,tmp%(maxn*maxn)/maxn,tmp%maxn);
            }else {
                printf("%d %d %d
%d\n",1,tmp/(maxn*maxn)*90,tmp%(maxn*maxn)/maxn,tmp%maxn);
            }
        }
    }
    return 0;
}
```

G. Game of Swapping Numbers

题意

给定两个长度为 n 的序列 A,B 和 k 次操作，每次操作可以交换 a_i,a_j。

要求在 k 次操作后最小化 $\sum_{i=1}^n|a_i-b_i|$。

题解

首先考虑不存在 k 约束的情况。

可以将本题转化为从 a_i,b_i 中选定 n 个数在前面加上 $+$ 号，其余 n 个数在前面加上 $-$ 号。最后需要最大化所有带符号数的和。

显然贪心选取 a_i,b_i 这 $2n$ 个数中前 n 大加上 $+$ 号即可。然后为了保证合法性，需要使得 a_i,b_i 正好一正一负。

于是考虑任选一组 $(+a_i,+b_i),(-a_j,-b_j)$ 交换 a_i,a_j 直到不存在这种情况为止即可。

接下来考虑 k 有限的情况，显然贪心每次选取收益最大的 $(+a_i,+b_i),(-a_j,-b_j)$ 交换即可。

此时收益为 $2(\min(a_i,b_i)-\max(a_j,b_j))$□于是考虑将 $\min(a_i,b_i)$ 序列从大到小排序□$\max(a_i,b_i)$ 从小到大排序。

然后贪心取前 k 个收益即可。注意 $(+a_i,+b_i)$ 在 $\min(a_i,b_i)$ 序列中一定排在 $(+a_i,-b_i),(-a_i,+b_i),(-a_i,-b_i)$ 的前面。

注意 $(-a_i,-b_i)$ 在 $\max(a_i,b_i)$ 序列中一定排在 $(+a_i,-b_i),(-a_i,+b_i),(+a_i,+b_i)$ 的前面。

于是一定会先让所有 $(+a_i,+b_i)$ 和 $(-a_j,-b_j)$ 配对，至于其他的配对方案计算出来的 $2(\min(a_i,b_i)-\max(a_j,b_j))$ 都是非正数，可以舍去。

注意到 k 可能大于需要交换的次数，但此时如果 $n\gt 2$ 一定可以找到两个同号的 a_i,a_j 做无意义的交换消耗 k□使得 k 正好等于需要交换的次数。

最后 $n=2$ 的情况没有选择只能强制交换，所以不一定可以得到最优解，需要特判。总时间复杂度 $O(n\log n)$□

```
const int MAXN=5e5+5;
int a[MAXN],b[MAXN];
int main()
{
    int n=read_int(),k=read_int();
    _for(i,0,n)a[i]=read_int();
    _for(i,0,n)b[i]=read_int();
    if(n==2){
        if(k&1)swap(a[0],a[1]);
        enter(abs(a[0]-b[0])+abs(a[1]-b[1]));
        return 0;
    }
    LL ans=0;
    _for(i,0,n){
        if(a[i]>b[i])
        swap(a[i],b[i]);
        ans+=b[i]-a[i];
    }
    sort(a,a+n,greater<int>());
    sort(b,b+n);
    _for(i,0,min(n,k))
    ans+=max(0,a[i]-b[i])*2;
    enter(ans);
    return 0;
}
```

J. Journey among Railway Stations

题意

给定 n 个车站，车站 i 开放时间为 $[u_i,v_i]$□从车站 i 到车站 $i+1$ 需要花费 c_i 时间。接下

来 q 次操作：

1. 询问是否可以从 x 车站出发达到 y 车站，初始时为 0 时刻
2. 修改某个 c_i
3. 修改某个 $[u_i,v_i]$

注意只能在车站开放时间内进入车站，且车站 i 只能直接到达车站 $i+1$。

题解

定义时间函数 $f(l,r,t)$ 表示起始时间为 t 时从车站 l 到车站 $r+1$ 需要花费的时间。不难发现

$$ f(i,i,t)=\begin{cases} c_i, & 0\le t\le u_i\\ c_i+(t-u_i), & u_i\lt t\le v_i \end{cases} $$

同时对任意 $i\le j\lt k$，有 $f(i,k,t)=f(j+1,k,f(i,j,t))$，利用数学归纳法可以证明 $f(i,j,t)$ 总是形如以下形式：

$$ f(i,j,t)=\begin{cases} y, & 0\le t\le x_1\\ y+(t-x_1), & x_1\lt t\le x_2 \end{cases} $$

于是考虑线段树维护区间 $[l,r]$ 的时间函数 $f(l,r,t)$ 顺便维护区间合法性即可，时间复杂度 $O(n\log n)$。

```
const int MAXN=1e6+5;
int u[MAXN],v[MAXN],c[MAXN];
struct Node{
    bool legal;
    LL x1,x2,y;
}s[MAXN<<2];
Node add(const Node &a,const Node &b){
    Node c;
    c.legal=a.legal&&b.legal;
    if(c.legal){
        if(a.y>b.x2)
        c.legal=false;
        else if(a.y>=b.x1){
            c.x1=a.x1;
            c.x2=min(a.x2,c.x1+b.x2-a.y);
            c.y=b.y+a.y-b.x1;
        }
        else{
            c.x1=min(a.x2,a.x1+b.x1-a.y);
            c.x2=min(a.x2,c.x1+b.x2-b.x1);
            c.y=b.y;
        }
    }
    return c;
}
int lef[MAXN<<2],rig[MAXN<<2];
void push_up(int k){
    s[k]=add(s[k<<1],s[k<<1|1]);
```

```
}
void build(int k,int pos){
    s[k].legal=true;
    s[k].x1=u[pos];
    s[k].x2=v[pos];
    s[k].y=u[pos]+c[pos];
}
void build(int k,int L,int R){
    lef[k]=L,rig[k]=R;
    int M=L+R>>1;
    if(L==R){
        build(k,M);
        return;
    }
    build(k<<1,L,M);
    build(k<<1|1,M+1,R);
    push_up(k);
}
void update(int k,int pos){
    if(lef[k]==rig[k]){
        build(k,pos);
        return;
    }
    int mid=lef[k]+rig[k]>>1;
    if(mid>=pos)
    update(k<<1,pos);
    else
    update(k<<1|1,pos);
    push_up(k);
}
Node query(int k,int L,int R){
    if(L<=lef[k]&&rig[k]<=R)
    return s[k];
    int mid=lef[k]+rig[k]>>1;
    if(mid>=R)
    return query(k<<1,L,R);
    else if(mid<L)
    return query(k<<1|1,L,R);
    else
    return add(query(k<<1,L,R),query(k<<1|1,L,R));
}
int main()
{
    int T=read_int();
    while(T--){
        int n=read_int();
        _rep(i,1,n)u[i]=read_int();
        _rep(i,1,n)v[i]=read_int();
        _for(i,1,n)c[i]=read_int();
        build(1,1,n);
        int q=read_int();
```

```
        while(q--){
            int opt=read_int();
            if(opt==0){
                int l=read_int(),r=read_int();
                Node ans=query(1,l,r);
                if(ans.legal)
                puts("Yes");
                else
                puts("No");
            }
            else if(opt==1){
                int pos=read_int();
                c[pos]=read_int();
                update(1,pos);
            }
            else{
                int pos=read_int();
                u[pos]=read_int();
                v[pos]=read_int();
                update(1,pos);
            }
        }
    }
    return 0;
}
```

赛后总结

jxm□玄学场，打表、随机化yyds

王智彪：不会写大模拟的wzb是真的屑，细节处理不到位。