上海电力大学程序设计迎新赛题解 by suepacm

T1 First Meet

题目信息

write by StarryKiller

test by Empty_Dust,loveyileina

难度：入门

题目内容

题目链接（洛谷）
在这个世界里，每一个人都有一个友善值 v，其中 v 是一个正整数。定义两个人之间的**不友善度**为他们的友善值的最小公约数。

形式化地说，设 A 的友善值为 v_1，B 的友善值为 v_2，则 A B 间的不友善度为 $\gcd(v_1,v_2)$，其中 $\gcd(a,b)$ 代表 a,b 的最大公约数，即最大的整数 q，使得 $a \bmod q = b \bmod q = 0$。

给你 Empty_Dust 的友善值 v_1，请找到一个人的友善值 v_2，满足 $v_2 \leq v_1$ 且他们间的不友善度最小。这个人就是 Starrykiller 捏。

形式化地说，给定正整数 v_1，请找到 $1 \leq v_2 \leq v_1$，最小化 $\gcd(v_1,v_2)$。输出你找到的 v_2。

输入格式

本题存在多组测试数据。

第一行为一个整数 T，表示测试数据组数。

下面 T 行，每行一个整数 v_1，表示 Empty_Dust 的友善值。

输出格式

对于每组测试数据，输出一行一个整数 v_2，表示你找到的符合题意的友善值。

若有多解，请输出最大的一个。

题解

由 $v_1 - 1$ 从大到小找，利用辗转相除法寻找最大公约数。

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int n, v1, v2, temp, t1, t2;
    cin >> n;
    for (int i = 0;i < n;++i)
    {
        cin >> v1;
        for (int j = v1 - 1;j > 0;--j)
        {
            v2 = j, t1 = v1;
            while (temp != 0)
            {
                temp = t1 % v2;
                t1 = t1 / v2;
                t2 = v2;
                v2 = temp;
            }
            if (t2 == 1)
            {
                cout << j << endl;
                break;
            }
        }
    }

    return 0;
}
//by loveyileina 得分:5
```

性质1 $\forall v_1 \in \mathbb{Z} \cap [2, +\infty], \gcd(v_1, v_1 - 1) = 1$
也就是说，对于所有的大于 1 的整数 v_1， $v_2 = v_1 - 1$ 时，不友善度一定能取到最小值。

证明:
设 $x \mid v_1, x \mid v_1 - 1$，即 $\exists q_1, q_2 \in \mathbb{Z}, q_1 x = v_1, q_2 x = v_2$ 。

不难得到 $(g1 - g2)x = 1$ 。故 $a = 1$ 是能满足要求的最大的整数。证毕。

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

signed main(){
    ios::sync_with_stdio(false);
    cin.tie(0),cout.tie(0);
    int t,tmp;cin>>t;

    while(t--){
        cin>>tmp;
        cout<<tmp-1<<'\n';
    }
    return 0;
}
//by Empty_Dust 得分：5
```

特别地，我们注意到，当 $v_1 = 1$ 时， 1 应当为满足条件的最大值。

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

signed main(){
    ios::sync_with_stdio(false);
    cin.tie(0),cout.tie(0);
    int t,tmp;cin>>t;

    while(t--){
    cin>>tmp;
    if(tmp==1)
        cout<<1<<'\n';
    else
        cout<<tmp-1<<'\n';
    }
    return 0;
}
//by Empty_Dust 得分：45
```

不开 **long long** 见祖宗

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

signed main(){
    ios::sync_with_stdio(false);
    cin.tie(0),cout.tie(0);
    int t,tmp;cin>>t;

    while(t--){
    cin>>tmp;
    if(tmp==1)
        cout<<1<<'\n';
    else
        cout<<tmp-1<<'\n';
    }
    return 0;
}
//by Empty_Dust 得分：100
```

T2 Stars in the sky

题目信息

write by StarryKiller

test by Empty_Dust

难度：普及−

题目内容

题目链接（洛谷）

Starrykiller 喜欢星星。她现在一共有 n 颗星星，从左往右排成一列，依次编号为 $1, 2, \cdots, n$。

有一个数列 a，描述每颗星星的阴阳性。对于第 i 颗星星，若她是阴性的，则 $a_i = 0$；若她是阳性的，则 $a_i = 1$。

Starrykiller 想要知道，当她给定 l, r 时，$[l, r]$ 中的最大的满足阳性星星的个数和阴性星星的个数相等的区间的长度。特别地，如果不存在，则答案为 0。

输入格式

本题存在多组数据。

第一行，一个整数 T，表示有 T 组数据。

对于每组数据，第一行为 n, l, r，含义见题目描述。

第二行有 n 个非 0 即 1 的整数，描述数列 a。

输出格式

一行一个整数，表示所有答案的异或值。

题解

利用前缀和思想，将阴性看作 -1 , 阳性看作 1 。
显然有：
性质1 $\forall i, j \in \mathbb{N}$, 若 $sum_i = sum_j$, 则 $len = i - j$ 合法。

因此我们可以利用哈希表/数组记录第一次到达某个点的下标 i 。
即：从左往右遍历数组，记录一个整型 sum 若为 0 则 sum 自减，若为 1 则 sum 自增， 新到达的坐标若在哈希表/数组中没有记录，则在该位置记录当前遍历到的下标 i ， 若有记录，则根据记录的最早下标和当前下标计算出长度 len 。

解法1：哈希表

```cpp
#include <bits/stdc++.h>
#define int long long
using namespace std;
int nums[500007];

signed main() {
    ios::sync_with_stdio(false);
    cin.tie(0), cout.tie(0);
    int t;cin >> t;
    int ans = 0;
    while (t--) {
        int n, l, r;
        cin >> n >> l >> r;
        l--, r--;
        for (int i = 0;i < n;++i)
            cin >> nums[i];
        map<int, int> mp;
        int len = 0, sum = 0;
        mp[0] = l - 1;
        for (int i = l;i <= r;++i) {
            if (nums[i] == 1)
                sum++;
            else
                sum--;
            if (mp.count(sum) == 0)
                mp[sum] = i;
            else
                len = max(len, i - mp[sum]);
        }
        ans ^= len;
    }
    cout << ans;
    return 0;
}
//by Empty_Dust 得分：100
```

解法2：数组

```cpp
#include <bits/stdc++.h>

using namespace std;
constexpr int MAXN=5e5+10;
int ans=0;

map<int,int> m; int n, l, r;
int a[MAXN];
void solve() {
    cin>>n>>l>>r;
    m.clear();
    int cur=0;
    if(l==1) m[0]=0;
    for (int i=1; i<=n; ++i) {
        cin>>a[i]; if (!a[i]) a[i]--;
        a[i]+=a[i-1];
        if (i>=l-1 && i<=r) {
            if (m.count(a[i]))
                cur=max(cur,i-m[a[i]]);
            else m[a[i]]=i;
        }
    }
    // cout<<cur<<'\n';
    ans^=cur;
}

signed main() {
    int T; cin>>T;
    while (T--) solve();
    cout<<ans;
}
//by StarryKiller 得分: 100
```

T3 NNEZ

题目信息

write by StarryKiller

test by Empty_Dust,loveyileina

难度：普及

题目内容

题目链接（洛谷）

南宁二中的社团会不定期举办随机宅舞活动。（Empty_Dust：好看爱看）

有两支不同的宅舞，宅舞 A 有 n 人会跳，宅舞 B 有 m 人会跳。由于一个人只会跳一支舞，所以一共有 $(n+m)$ 个人。

观众会给这 $(n+m)$ 个人的舞蹈评分。宅舞 A 的评分分别为 $a_1, a_2, \cdots, a_n$，宅舞 B 的评分分别为 $b_1, b_2, \cdots, b_m$。

Empty_Dust 想要邀请会跳宅舞 A 的 i 和会跳宅舞 B 的 j 来一起跳舞。这场舞蹈的精彩程度是 $a_i + b_j$。她想要知道，前 k 大的精彩程度的和是多少。

由于答案可能很大，输出答案模 114514191 之后的结果。

输入格式

共 3 行。

第 1 行，三个整数 n, m, k。

第 2 行，n 个整数 $a_1, a_2, \cdots, a_n$。

第 3 行，m 个整数 $b_1, b_2, \cdots, b_m$。

输出格式

共一行一个数，代表前 k 大的精彩程度的和模 114514191 后的结果。

题解

下面是一位初学者的答案，我反复查看，但最终还是没有看懂，将它放在这里是为了证明：就算你乱写也能拿15分。

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;

signed main()
{
	int n, m, k;
	cin >> n >> m >> k;
	vector<int> a(n);
	vector<int> b(m);
	for (int i = 0; i < n; ++i)cin >> a[i];
	for (int i = 0; i < m; ++i)cin >> b[i];
	sort(a.begin(), a.end(), [&](int a, int b) {return a > b; });
	sort(b.begin(), b.end(), [&](int a, int b) {return a > b; });
	int l1 = 0, l2 = 0, sum = 0;
	for (int i = 0; i < k; ++i)
	{
		sum = sum + a[l1] + b[l2];
		while (a[l1] == a[l1 + 1])
		{
			sum += (a[l1] + b[l2]);
			l1++;
			i++;
		}
		while (b[l2] == b[l2 + 1])
		{
			sum += (a[l1] + b[l2]);
			l2++;
			i++;
		}
		if (a[l1 + 1] + b[l2] > b[l2 + 1] + a[l1])
			l1++;
		else
			l2++;
		if (l1 >= n)
			l1--;
		if (l2 >= m)
			l2--;
	}
	cout << sum % 114514191;
	return 0;
```

```
}
//by loveyileina 得分：15 *modified by Empty_Dust
```

设 i 为数组 a 的下标，j 为数组 b 的下标，$sum_{i,j}=a_i+b_j$ 。

显而易见的，每一个 sum 都可以与唯一的二元组 (i,j) 组成一个三元组 $(i,j,sum_{i,j})$,因此，我们只需要将它们以 sum 的大小关系排序即可。

堆/优先队列是一个能在 $O(n\log n)$ 时间复杂度内完成这一目标的数据结构。

因此，我们可以：

1. 边计算边取模，防止在计算过程中越界。
2. 使用结构体简化思考。
3. 先对两个数组**由大到小排序**。

朴素地思考到，对每个 $sum_{i,j}$,略小的必然是 $sum_{i+1,j}$ 或 $sum_{i,j+1}$ 。

此时，显然 $sum_{max}=sum_{0,0}$ ，次大的显然是 $max\{sum_{0,1},sum_{1,0}\}$,不妨设 $sum_{0,1}>sum_{1,0}$,则第三大的是 $max\{sum_{1,1},sum_{1,0},sum_{0,2}\}$ 。

我们注意到，如果此时 $sum_{1,0}$ 是最大值，那么简单地将 $sum_{1,1}$ 和 $sum_{2,0}$ 放入堆中，会造成重复的问题。

因此我们需要

1. 将所有 $(a_i,0)$ 数对都先加入堆。(来自 $StarryKiller$ 的思路)
2. 维护一个 j 指针记录当前已经加入堆的 b 数组下表。

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
const int MAXN=1e6+7;
const int mod =114514191;//9810好臭
int a[MAXN];
int b[MAXN];
struct Node
{
    int i;
    int j;
    int value;
    bool operator()(Node x,Node y){
        return x.value<y.value;
    }
};

signed main(){
    ios::sync_with_stdio(false);
    cin.tie(0),cout.tie(0);
    signed n,m,k;
    cin>>n>>m>>k;
    for(int i=0;i<n;++i)cin>>a[i];
    for(int i=0;i<m;++i)cin>>b[i];
    sort(a,a+n);
    sort(b,b+m);
    reverse(a,a+n);
    reverse(b,b+m);
    priority_queue<Node,vector<Node>,Node> pq;
    for (int i = 0; i < n; i++)
    {
        pq.push({i,0,a[i]+b[0]});
    }
    int sum = 0;
    for(signed i=0;i<k;++i){
        Node cut = pq.top();
        pq.pop();
        sum+=cut.value%mod;
        sum%=mod;
        if(cut.j<m-1){
            pq.push({cut.i,cut.j+1,a[cut.i]+b[cut.j+1]});
        }
    }
```

```
    cout<<sum;
    return 0;
}
//by Empty_Dust 得分：100
```

针对操作1进行修改。

解法2：

对每个 a_i, b_j 多记录一个值 nxt ， 存放已经遍历到的对方数组下标。

这里有一道类似题目和它使用该方法的题解。

[atcoder abc311] E Meal

```cpp
#include <bits/stdc++.h>
//#define int long long
using namespace std;
const int MAXN = 100010;

int n, m, l;

struct meal
{
    int index;
    int v;
    int nxt = 0;
}a[MAXN], b[MAXN];

struct query {
    int ai;
    int bi;
    int sum;
    bool operator()(query a, query b) {
        return a.sum < b.sum;
    }
};

priority_queue < query, vector<query>, query> pq;
vector<int> ph[MAXN];

void pt(int x, int y) {
    if (x >= n || y >= m || x < b[y].nxt || y < a[x].nxt)
        return;
    query tmp;
    tmp.ai = x;
    tmp.bi = y;
    tmp.sum = a[x].v + b[y].v;
    pq.push(tmp);
    a[x].nxt++;
    b[y].nxt++;
}

signed main() {
    ios::sync_with_stdio(false);
    cin.tie(0), cout.tie(0);
    cin >> n >> m >> l;
    for (int i = 0;i < n;++i) {
```

```
        a[i].index = i;
        cin >> a[i].v;
    }
    for (int i = 0;i < m;++i) {
        b[i].index = i;
        cin >> b[i].v;
    }
    while (l--) {
        int x, y;
        cin >> x >> y;
        x--, y--;
        ph[x].push_back(y);
    }
    sort(a, a + n, [&](meal a, meal b) {return a.v > b.v;});
    sort(b, b + m, [&](meal a, meal b) {return a.v > b.v;});
    for (int i = 0;i < n;++i)
        sort(ph[i].begin(), ph[i].end());
    pt(0, 0);
    while (!pq.empty()) {
        int cai = pq.top().ai;
        int cbi = pq.top().bi;
        int ca = a[cai].index;
        int cb = b[cbi].index;
        auto lb = lower_bound(ph[ca].begin(), ph[ca].end(), cb);
        if (lb == ph[ca].end() || *lb != cb) {
            cout << pq.top().sum;
            return 0;
        }
        pt(cai, cbi + 1);
        pt(cai + 1, cbi);
        pq.pop();
    }
    return 0;
};
//by Empty_Dust AC
```

T4 别来无恙

题目信息

write by StarryKiller

test by Empty_Dust

难度：普及

分类：计算几何

题目内容

题目链接(洛谷)

Starrykiller 非常想念她的朋友。她回忆起了她和朋友在一起度过的快乐的时光。

她和朋友在一起的时候，会玩一个这样的游戏：一个人给出平面内的两条线段（每条线段给定两个端点的坐标），一个人报出这两条线段的交点。

当然，**Starrykiller** 的数学很菜，她不能很快的计算出结果，因此她转而求助于你。请帮帮她求出这两条线段交点的坐标。

输入格式

第一行一个整数 T，表示她们玩了 T 局游戏。

下面 T 行，每行八个实数 $x_1\ y_1\ x_2\ y_2\ x_3\ y_3\ x_4\ y_4$，分别表示点 P_1, P_2, P_3, P_4。其中 P_1P_2 组成一条线段，P_3P_4 组成一条线段。

输出格式

共 T 行。

对于每一行，若有交点，输出两个实数 $x_C\ y_C$，描述交点的坐标。

否则输出 `qvq` 。

题解

[CSDN]判断两线段相交

[CSDN]计算几何 之 判断两直线是否相交并求两直线交点

```cpp
#include <bits/stdc++.h>
using namespace std;
#define vector point

struct point
{
    double x;
    double y;
    point operator -(point B){
        return {x-B.x,y-B.y};
    }
};

double cross(point a,point b)
{
    return a.x * b.y - b.x * a.y;
}
//判断相交
bool intersection(point a,point b,point c,point d)
{
    if(max(c.x,d.x)<min(a.x,b.x)||max(a.x,b.x)<min(c.x,d.x)||max(c.y,d.y)<min(a.y,b.y)||max(a.y
        return false;
    }
    if(cross(a-d,c-d)*cross(b-d,c-d)>0||cross(d-b,a-b)*cross(c-b,a-b)>0){
        return false;
    }
    return true;
}
//计算位置
inline void pos(point a,point b,point c,point d){
    if(intersection(a,b,c,d)){
        vector w = c-d;
        vector v = a-b;
        vector u = a-c;
        double t = cross(w,u) / cross(v,w);
        point ans={a.x+t*v.x,a.y+t*v.y};
        cout<<ans.x<<' '<<ans.y<<'\n';
    }
    else{
        cout<<"qvq"<<'\n';
    }
}
```

```
signed main(){
    ios::sync_with_stdio(false);
    cin.tie(0),cout.tie(0);
    int n;cin>>n;
    while(n--){
        point a,b,c,d;
        cin>>a.x>>a.y>>b.x>>b.y>>c.x>>c.y>>d.x>>d.y;
        pos(a,b,c,d);
    }
    return 0;
}
//by Empty_Dust
```

T5 Life, the Universe, and Everything

题目信息

write by StarryKiller

test by Empty_Dust

难度：提高

题目内容

题目链接(洛谷)

Empty_Dust 喜欢旅行。她每到一个地方，就会把这个地方的色彩记录下来。她用大写拉丁字母 `A-Z` 来表示不同的颜色，将她经过的地方的色彩按顺序写下来就像是这样的：

$$\texttt{ABCDAHC}$$

她会经常做这两件事情：

- 数一下第 i 个地方与第 j 个地方之间一共有几个颜色段。
- 将第 i 个地方到第 j 个地方变成一样的色彩。

也许下面的剧情你已经猜到了。由于她去的地方越来越多了，每做一次都要花很长的时间，所以她求助你帮她写一个程序完成这件事。

颜色段的定义：极长的连续相同色彩。

例如 `AABBBDB` 中有 4 个颜色段。

输入格式

共 $(m+2)$ 行。

第一行两个整数 n, m，表示 **Empty_Dust** 去过的地方的数量和操作的数量。

第二行一个长度为 n 的字符串 S，表示初始时每个地方的颜色。

接下来 m 行，对于每一行：

- `1 x y` 表示查询 $[x, y]$ 间的颜色段数量。
- `2 x y C` 表示将 $[x, y]$ 间所有地方的颜色变成 C。

输出格式

对于每一个 1 操作，输出一行一个整数，表示对应区间颜色段的数量。

题解

暴力解法

```cpp
#include <bits/stdc++.h>

using namespace std;

constexpr int MAXN=5e5+10;

string s; int a[MAXN];
int n, m;

signed main() {

    ios::sync_with_stdio(false);
    cin.tie(0),cout.tie(0);
    cin>>n>>m;
    cin>>s;
    for (int i=0; i<n; ++i) {
        a[i+1]=s[i]-'A'+1;
    }
    int op, x, y;  char c;
    while (m--) {
        cin>>op>>x>>y;
        if (op==1) {
            int ans=0;
            // cout<<sg.query(x,y).cnt<<'\n';
            for (int i=x; i<=y; ++i) {
                if (i==x || a[i]!=a[i-1])
                    ans++;
            }
            cout<<ans<<'\n';
        }
        else {
            cin>>c;
            for (int i=x; i<=y; ++i) {
                a[i]=c-'A'+1;
            }
        }
    }
}
//by StarryKiller 得分: 25
```

线段树

```cpp
#include <bits/stdc++.h>
#define int long long
using namespace std;

const int MAXN = 500000;
int n,m;
string str;

struct node
{
    char l;//左侧颜色
    char r;//右侧颜色
    int num;//颜色段数量
    char lazy='x';
    node operator +(node x){
        if(num==0)return x;
        if(x.num==0)return *this;
        int cut=num+x.num;
        if(r==x.l)
            cut--;
        return {l,x.r,cut};
    }
};
node tree[MAXN*4];

node set_up(int l=1,int r=n,int i=1){
    if(l==r){
        node* cut = &tree[i];
        cut->l=str[l-1];
        cut->r=str[l-1];
        cut->num=1;
        return tree[i];
    }
    int mid = l+r>>1;
    return tree[i] = set_up(l,mid,i*2)+set_up(mid+1,r,i*2+1);
}

inline void paint(int i,char x){
    tree[i].l=x;
    tree[i].r=x;
    tree[i].num=1;
    tree[i].lazy=x;
}
```

```
inline void push_down(int l,int r,int i){
    char ly = tree[i].lazy;
    if(ly=='x')
        return;
    paint(i*2,ly);
    paint(i*2+1,ly);
    tree[i].lazy='x';
}

node update(int ul,int ur,char x,int l=1,int r=n,int i=1){
    if(r<ul||ur<l)
        return tree[i];
    if(ul<=l&&r<=ur){
        paint(i,x);
        return tree[i];
    }
    int mid = l+r>>1;
    push_down(l,r,i);
    return tree[i] = update(ul,ur,x,l,mid,i*2)+update(ul,ur,x,mid+1,r,i*2+1);
}

node query(int ql,int qr,int l=1,int r=n,int i=1){
    if(r<ql||qr<l){
        node ret;
        ret.num=0;
        return ret;
    }
    int mid = l+r>>1;
    if(tree[i].lazy!='x'||ql<=l&&r<=qr)
        return tree[i];
    return query(ql,qr,l,mid,i*2)+query(ql,qr,mid+1,r,i*2+1);
}

signed main(){
    ios::sync_with_stdio(false);
    cin.tie(0),cout.tie(0);
    cin>>n>>m;
    cin>>str;
    set_up();
    while(m--){
        int op,x,y;
        cin>>op>>x>>y;
```

```
            if(op==1)
                cout<<query(x,y).num<<'\n';
            else{
                char ch;
                cin>>ch;
                update(x,y,ch);
            }
        }
        return 0;
    }
    //by Empty_Dust 得分：90
```

极致优化

```cpp
#include <bits/stdc++.h>

using namespace std;

constexpr int MAXN=5e5+10;

struct valnode {
    int lc, rc, cnt;
};

struct SGT {
    #define ll(p) tr[p].l
    #define rr(p) tr[p].r
    #define ls(p) (p<<1)
    #define rs(p) (p<<1|1)
    #define cnt(p) tr[p].cnt
    #define lc(p) tr[p].lc
    #define rc(p) tr[p].rc
    #define tag(p) tr[p].tag
    struct node {
        int l, r, cnt;
        int tag, lc, rc;
    } tr[MAXN<<2];

    void pushup(int p) {
        lc(p)=lc(ls(p)), rc(p)=rc(rs(p));
        cnt(p)=cnt(ls(p))+cnt(rs(p))-(rc(ls(p))==lc(rs(p)));
    }

    void pushdown(int p) {
        if (!tag(p)) return;
        tag(ls(p))=tag(rs(p))=
            lc(ls(p))=lc(rs(p))=
            rc(ls(p))=rc(rs(p))=tag(p);
        cnt(ls(p))=cnt(rs(p))=1;
        tag(p)=0;
    }

    void build(int l, int r, int *a, int p=1) {
        ll(p)=l, rr(p)=r;
        if (l==r) {
            lc(p)=rc(p)=a[l];
            cnt(p)=1;
```

```
            return;
        }
        int mid=(l+r)>>1;
        build(l,mid,a,ls(p)); build(mid+1,r,a,rs(p));
        pushup(p);
    }

    void change(int l, int r, int v, int p=1) {
        int cl=ll(p), cr=rr(p);
        if (l<=cl && cr<=r) {
            lc(p)=rc(p)=tag(p)=v, cnt(p)=1;
            return;
        }
        pushdown(p); int mid=(cl+cr)>>1;
        if (l<=mid) change(l,r,v,ls(p));
        if (r>mid) change(l,r,v,rs(p));
        pushup(p);
    }

    valnode query(int l, int r, int p=1) {
        int cl=ll(p), cr=rr(p);
        if (l<=cl && cr<=r)
            return {lc(p),rc(p),cnt(p)};
        pushdown(p); int mid=(cl+cr)>>1;
        valnode
            ansl={0,0,0}, ansr={0,0,0};
        if (l<=mid)
            ansl=query(l,r,ls(p));
        if (r>mid) ansr=query(l,r,rs(p));
        if (!ansl.cnt)
            return ansr;
        if (!ansr.cnt)
            return ansl;
        return {ansl.lc,ansr.rc,ansl.cnt+ansr.cnt-(ansl.rc==ansr.lc)};
    }
} sg;

string s; int a[MAXN];
int n, m;

signed main() {

    ios::sync_with_stdio(false);
```

```
    cin.tie(0),cout.tie(0);
    cin>>n>>m;
    cin>>s;
    for (int i=0; i<n; ++i) {
        a[i+1]=s[i]-'A'+1;
    }
    sg.build(1,n,a);
    int op, x, y;  char c;
    while (m--) {
        cin>>op>>x>>y;
        if (op==1) {
            cout<<sg.query(x,y).cnt<<'\n';
        }
        else {
            cin>>c;
            sg.change(x,y,c-'A'+1);
        }
    }
}
//by StarryKiller 得分：100
```

T6 游戏

题目信息

write by StarryKiller

难度：提高

分类：模拟

题目内容

题目链接(洛谷)

你已经解决了五道题目了。接下来的问题关于一个游戏。

这个游戏在一个 n 行 m 列的方格上进行。方格有三种类型，分别为硬地，软地和禁地。

有一个长和宽都为 1、高为 2 的箱子，起初**可能立在或躺在**起点上。

我们定义箱子的状态：

- 立着：箱子以 1×1 的接触面接触地面。
- 躺着：箱子以 1×2（或者 2×1）的接触面接触地面。

箱子不能立在软地上；箱子的任何部分不能接触禁地，也不能超出格子边界。

每次操作，箱子可以向上、下、左、右沿着对应的棱滚动 90 度，目标是用最少的操作次数，让箱子**立在**终点上。

请你帮助 **Empty_Dust** 求出这个最少的操作次数，让她赢下和 **Starrykiller** 的对局。

输入格式

本题有不定组数据，当出现 $n = m = 0$ 时代表输入结束。这组数据无需被处理。

第一行为 n, m，含义见题面。

接下来 n 行，第 i 行第 j 个字符描述格子 (i, j)：

- `.` 代表硬地。
- `E` 代表软地。
- `#` 代表禁地。
- `X` 代表起点。
- `O` 代表终点。

地图上可能出现一个 `X` 或者两个相邻的 `X`，分别表示箱子立着或者躺着。

起点和终点均可视为硬地。

输出格式

对于每组数据，如果有解，输出一行一个整数，表示最小步数。

否则输出 `Impossible` 。

题解

显然广度搜索，难点在于如何记录已经走过的位置。

```cpp
#include <bits/stdc++.h>
using namespace std;
const int MAXN=5e2+10;
char a[MAXN][MAXN]; int n, m;

// cur=0 箱子立在(x,y)上
// cur=1 箱子躺在(x,y)上，另一格在(x,y+1)
// cur=2 箱子躺在(x,y)上，另一格在(x+1,y)

int atx[]={0,0,1};
int aty[]={0,1,0}; // 另一个格子在哪

// 四个方向分别是：上、右、下、左（按照数学中笛卡尔坐标系的坐标轴方向）

int dx[][4]={
    {0,1,0,-2},
    {0,1,0,-1}, // cur=1
    {0,2,0,-1}  // cur=2
};
int dy[][4]={
    {1,0,-2,0},
    {2,0,-1,0},
    {1,0,-1,0}
};
int dcur[][4]={
    {1,2,1,2},
    {0,1,0,1},
    {2,0,2,0}
};

struct node {
    int x, y;
    int cur, step;
    node(int a, int b, int c, int d=0): x(a), y(b), cur(c), step(d) {}
};

int tx, ty;

bool operator ==(node a, node b) {
    return a.x==b.x && a.y==b.y && a.cur==b.cur;
}
```

```
int vis[MAXN][MAXN][5];

int bfs(node s) {
    queue<node> q; q.push(s); vis[s.x][s.y][s.cur]=1;
    while (q.size()) {
        auto u=q.front(); q.pop();
        for (int i=0; i<4; ++i) {
            int c=u.cur, x=u.x, y=u.y;
            int nx=x+dx[c][i], ny=y+dy[c][i], nc=dcur[c][i];
            int nx2=nx+atx[nc], ny2=ny+aty[nc];
            if (a[nx][ny]=='#' || a[nx2][ny2]=='#') continue;
            if (nx<1 || nx>n || nx2<1 || nx2>n) continue;
            if (ny<1 || ny>m || ny2<1 || ny2>m) continue;
            if (nc==0&&a[nx][ny]=='E') continue;
            if (vis[nx][ny][nc]) continue;
                vis[nx][ny][nc]=1;
            node v={nx,ny,nc,int(u.step+1)};
            q.push(v);
            if (v==node(tx,ty,0)) {
                return v.step;
            }
        }
    }
    return -1;
}

signed main() {
        ios::sync_with_stdio(false);
        cin.tie(nullptr); cout.tie(nullptr);
    int x[3], y[3], tot;
    while (cin>>n>>m && n && m) {
        memset(vis,0,sizeof vis);
        tot=0;
        for (int i=1; i<=n; ++i)
            for (int j=1; j<=m; ++j) {
                cin>>a[i][j];
                if (a[i][j]=='X') {
                    x[++tot]=i, y[tot]=j;
                    a[i][j]='.';
                }
                else if (a[i][j]=='O') {
                    tx=i, ty=j;
                    a[i][j]='.';
```

```
                }
            }
        node u={0,0,0};
        if (tot==1) u={x[1],y[1],0};
        else {
            if (x[1]==x[2]) u={x[1],y[1],1};
            else u={x[1],y[1],2};
        }
        int res=bfs(u);
        if (res!=-1) cout<<res<<'\n';
        else cout<<"Impossible\n";
    }
}
//by StarryKiller 得分：100
```