ZZYZ Round #3 Tutourial

1. weini

出题人	验题人
lyfx	shzaiz, kingxbz, lyfx, ws(维尼),star,Micoael_Primo.,TopTom

算是比较基础的构造题吧. 容易观察到第一个数绝对值尽量小，还得是负的. 从而得出结论.

- 考虑$n=1$, 显然没有办法hack.
- 考虑$n=2$. 第一个数绝对值尽量小，还得是负的。此时我们需要构造一种状态满足$2(a+b)=b+k$.
- 因为要求$|a|$最小,并且a是负整数,所以a就是-1了,然后就可以知道$b=k+2$了.

注意1. 我们可以知道k的值可以是$2^{63}-1$的加2就炸$longlong$了

代码实现100分.

```
#include<bits/stdc++.h>
using namespace std;
int main( )
{
    unsigned long long  t,n;
    cin>>t;
    while(t--)
    {
        cin>>n;
        cout<<2<<" -1 "<<n+2<<endl;
    }
    return 0;
}
```

出题者&验题者的吐槽:

> 我曾建议刘巨不要提醒第一个数是负的，可惜他没有采纳 –kingxbz
>
> 很有意思的一道题, 也挺坑的. –shzaiz

2.amethyst

审核难度. [入门], [贪心]

出题人	验题人
ws(维尼)	shzaiz, kingxbz, lyfx, ws(维尼),star,Micoael_Primo.,TopTom

算是一个板子题目.

- Fibonacci-Nim游戏部分:
    - $\bigoplus_{i=1}^{n} sg[i]=0$的时候先手必败.

- sg函数的终态$sg(x)=\operatorname{mex}(x)=0$当且仅当此时为必败态。
- 那么 sg[x]就表示有一个数量为x的堆时的博亦终态, 也就是说表示x通过博将最小的不能达到的状态。
- sg函数可以通过递推来解决。

这个部分的代码实现.

```
void lyfx(int n){
    int i,j;
    memset(sg,0,sizeof(sg));
    for(int i=1;i<=n;i++){
        memset(s,0,sizeof(s));
        for(j=0;f[j]<=i&&j<=n;j++)
        s[sg[i-f[j]]]=1;
        for(j=0;;j++)
            if(!s[j]){
                sg[i]=j;
                break;
            }
    }
}
```

- 再看威佐夫博弈:
  - 可以用SG函数, 也可以用mex实现. 下面给出两种实现方式:
  - 黄金分割的说明:

> 先来考虑两者的最优策略:
>
> 在这道题中，如果当一个人取石子时两堆已经为（$0,0$），那么他处于必败态。
>
> 没错，对于先手，如果他面对的是两堆相等的石子，那么他必胜。
>
> 如果他面对的是两堆不等的石子，他一定尽可能不让局势转化为先手两堆相等的石子，因为这样会使后手必胜
>
> 是的，知道了最优策略，我们就可以知道必胜态与必败态了。
>
> 我们手玩一下:
>
> $(0,0)$
>
> $(1,2)$
>
> $(3,5)$
>
> $(4,7)$
>
> $(6,10)$
>
> $(8,13)$
>
> $(9,15)$
>
> $(11,18)$
>
> 我们会发现，对于(a,b),$(b_i-a_i)=i$
>
> 这种先手必输的局面我们称为奇异局势

> 而这些局势的第一个值即a_i是$mex(i)$,也就是在前面没有出现的最小自然数。
>
> 接下来就可以用SG函数解决问题了;
>
> 我们枚举0~inf中第一个没有被访问过的数$first$，此时的结果是第cnt个必胜态
>
> 那么记录$SG[first][first+cnt]=1$。
>
> 对于两堆，如果$SG[a][b]=1$则表示处在奇异局势中，先手必败，否则，先手必胜!
>
> 然而:
>
> 对于威佐夫博弈，还有一种结论:
>
> 那么说一个结论：每种奇异局势的第一个值(**这里假设第一堆数目小于第二堆的数目**)总是等于当前局势的差值乘上1.618
>
> 这里的0.618即为黄金分割率！！！
>
> 那么我们可以得到结论:
>
> 先假设奇异局势为(a_k, b_k)
>
> $a_k = int(\frac{\sqrt{5}-1}{2} \times k)$
>
> $b_k = a_k + k$
>
> 由此我们可以得知：若两堆物品个数分别为$x, y(x < y)$, 则$k = y - x$,

法1. 黄金分割问题

```
#include<bits/stdc++.h>
#define fint register int
#define h 5001
#define N 364578
using namespace std;
const double gd=(sqrt(5)+1)/2;
int las_x=1;
string ans[]={"lyfx yyds!","xbz yyds!"};
int f[N],vis[N],sg[N];
inline void pre(int n);
inline string dos_a(int a,int b);
inline string dos_b(int a,int b);
int main()
{
    int T;
    cin>>T;
    memset(sg,-1,sizeof(sg));
    f[0]=1,f[1]=1;
    for(fint i=2;i<=100;i++)
    f[i]=f[i-1]+f[i-2];
    pre(5432);
    for(fint i=1;i<=T;i++)
    {
        int a,b;
        cin>>a>>b;
        if(i&1)
        cout<<dos_a(a,b);
```

```
        else
        cout<<dos_b(a,b);
        cout<<endl;
    }
    return 0;
}

inline string dos_a(int a,int b)
{
    return ans[!(sg[a] xor sg[b])];
}

inline string dos_b(int a,int b)
{
    if(a<b)
    swap(a,b);
    return ans[b==(int)((a-b)*(double)gd)];
}

inline void pre(int n)
{
    for(fint i=1;i<=n;i++)
    {
        memset(vis,0,sizeof(vis));
        for(fint j=0;f[j]<=i&&j<=n;j++)
        vis[sg[i-f[j]]]=1;//vis表示当前可选状态组成的集合
        for(fint j=0;;j++)//求mex
        if(!vis[j])
        {
            sg[i]=j;
            break;
        }
    }
    return ;
}
```

法2. SG函数

```
#include<bits/stdc++.h>
#define fint register int
#define h 5001
#define N 364578
using namespace std;
const double gd=(sqrt(5)+1)/2;
string ans[]={"lyfx yyds!","xbz yyds!"};
int f[N],vis[N],sg[N],SG[h][h];
inline void pre(int n);
inline string dos_a(int a,int b);
inline string dos_b(int a,int b);
int main()
{
    int T;
    cin>>T;
```

```
    memset(sg,-1,sizeof(sg));
    f[1]=1,f[2]=1;
    for(fint i=3;i<=100;i++)
    f[i]=f[i-1]+f[i-2];
    pre(1001);
    for(fint i=1;i<=T;i++)
    {
        int a,b;
        cin>>a>>b;
        if(i&1)
        cout<<dos_a(a,b);
        else
        cout<<dos_b(a,b);
        cout<<endl;
    }
    return 0;
}

inline string dos_a(int a,int b)
{
    return ans[!(sg[a] xor sg[b])];
}

inline string dos_b(int a,int b)
{
    if(SG[a][b])
    return ans[1];
    return ans[0];
}

inline void pre(int n)
{
    for(fint i=1;i<=n;i++)
    {
        memset(vis,0,sizeof(vis));
        for(fint j=1;f[j]<=i&&j<=n;j++)
        vis[sg[i-f[j]]]=1;//vis表示当前可选状态组成的集合
        for(fint j=0;;j++)//求mex
        if(!vis[j])
        {
            sg[i]=j;
            break;
        }
    }
    memset(vis,0,sizeof(vis));
    SG[0][0]=1;
    int cnt=1;
    for(fint i=1;i<=n;i++)
    {
        int firsts;
        for(fint j=0;;j++)
        if(!vis[j])
        {
```

```
            firsts=j;
            break;
        }
        SG[firsts][firsts+cnt]=1;
        cnt++;
    }
    return ;
}
```

审核难度. [普及-], [博弈论]

出题者&验题者的吐槽:

> **我曾建议王巨把普通Nim改为K-Nim游戏，不过他没有采纳。** –kingxbz

3. lemon

出题人	验题人
shzaiz	shzaiz, kingxbz, lyfx, ws(维尼),star,Micoael_Primo.,TopTom

- 10% 瞎暴力. 期望得分10pts.
- 50% : 链上DP. 链还是有顺序的. 那么因为在前面不存在NASA序列. 所以每一步可以分开考虑.

  考虑如下3个问题
  - 情况1 . 0 0 0 0 0 0 ... 0 0 (n 个0), 有多少种情况?
  - 情况2 . m 0 0 0 0 0 ... 0 0 (n个0), 有多少种情况?
  - 情况3 . x 0 0 0 0 0 0 0 y(n个0)
    - 情况3.1 x= y, 有多少种情况?
    - 情况3.2 x != y, 有多少种情况?
  - **[答案]**
  - 填入一个数后, 变为情况2.
  - 显然每个位置上的数只需要和前一个位置不同即可，后 n−1 个位 置每个位置都有 k−1 种替换方案，所以方案总数为$(k-1)^{n-1}$
  - 考虑在两边有限制的情况下的方案数，显然两边有数分为两种情况：数字相同和数字不同。设状态为`f[i][j]`，表示数 列除两边非零的数外，中间 0 的个数为 i。对于 j，0 为两边数字不同，1 为 两边数字相同。那么考虑转移方程：
    - 初始状态`f[0][0] = 1,f[0][1] = 0`。
    - 当两边的数相同时，考虑填 i−1 个 0 时的情况，由于填 i−1 个 0 的末位置与填 i 个 0 的末位置相邻，所以两者必须不同 (a,⋯,a,b,a)， 即 `f[i][1] = (k−1)f[i−1][0]`。
    - 当两边的数不同时，由于填 i−1 个 0 的末位置与填 i 个 0 的末位置 相邻，所以两者必须不同 (a,⋯,a,c,b)，当 c = a 时，有`f[i−1][1]` 种 方案，当 c!= a 且 c != b 时，有 `(k−2)f[i−1][0]`种方案，即 `f[i][0] = f[i−1][1] + (k−2)f[i−1][0]`。
    - 处理一下,复杂度 O(n)。

部分2代码实现:

```
f[0][i] = (int)add(f[1][i - 1], mul((k - 2), f[0][i - 1]));
f[1][i] = (int)mul((k - 1), f[0][i - 1]);
```

- 满分: 注意到不同的0可以分段处理. 描述一个序列需要两个.
    - 100%: 考虑DFS的时候入栈, 出栈的时候对于答案的变化. 变化可能有两种情况
        - 还在0中间
        - 已经又1与之分割.

      两者显然是不同的处理策略, 所以需要存储两个数值ans1, ans2. 如果是0的话, 就对于ans2进行修改, 使之乘上当前数. 那么最后答案就是ans1*ans2.(乘法原理).
    - 如果是1的话, 就相当于对于ans1进行累加. 加上dp的数. 最后答案就是ans1.
    - 关于出栈的时候, 记忆化搜索, 记录下当前变量的状态, 记得回溯就行了.

满分代码实现:

```
void dfs(int u, int fa) {
    ll an2 = ans2, pvv = -1, bkk = -1, com = 0, an1 = ans1;
    s[++tp] = alp[u];
    if (u == 1 && s[tp] == 0) {
        ans2 = mul(ans2, k);
        an2 = ans2;
        if(deb) printf("pos[%d] :%d  ...(1)\n", u, mul(ans2, ans1));
        finalans ^= mul(ans2, ans1);
    } else if (s[tp] == 0) {
        if (!combo) {
            pv = tp - 1;
            pvv = pv;
        }else{
            pvv= pv;
        }
        combo = 1;
        com = 1;
        ans2 = mul(ans2, (k - 1));
        an2 = ans2;
        if(deb)  printf("pos[%d] :%d   ...(2)\n", u, mul(ans2, ans1));
        finalans ^= mul(ans2, ans1);
    } else {
        // if(pv==-1){pv = tp; pvv = pv;}

        bk = tp;
        bkk = bk;
        if (combo) {
            if (pv == -1) {
                // printf("Ouch\n");
                //  printf("%d %d\n",k-1,bk-2);
                ans1 = mul(1, qpow(k - 1, bk - 1));
                an1 = ans1;
            } else if (s[pv] == s[bk]) {
                ans1 = mul(ans1, f[1][bk - pv - 1]);
                an1 = ans1;
```

```
            } else {
                ans1 = mul(ans1, f[0][bk - pv - 1]);
                an1 = ans1;
            }
            if(deb)  printf("pos[%d] :%d   ...(3)\n", u, ans1);
            finalans ^= ans1;
            ans2 = 1;
            an2 = ans2;

        } else {
            if(deb) printf("pos[%d] :%d   ...(4)\n", u, ans1);
            finalans ^= ans1;
        }
        pv = tp, bk = -1;
        pvv = pv, bkk = bk;
        combo = false;
        com = combo;
    }
    for (int i = head[u]; i; i = e[i].nxt) {
        int v = e[i].to;
        if (v == fa)
            continue;
        dfs(v, u);
        pv = pvv;
        ans2 = an2;
        ans1 = an1;
        combo = com;
        bk = bkk;
        tp--;
    }

}
```

完整Code.

```
#include <bits/stdc++.h>
using namespace std;
#define N 500010
#define MOD 998244353
#define ll long long
bool deb = 0;
int f[2][N];
int alp[N];
int s[N] = {}, tp = 0;
ll pv = -1;
ll bk = -1;
ll ans1 = 1;
ll ans2 = 1;
bool combo = false;
int n, k;
ll finalans = 0;
ll add(ll a, ll b) {
    return ((a % MOD) + (b % MOD)) % MOD;
```

```
}
ll mul(ll a, ll b) {
    return ((a % MOD) * (b % MOD)) % MOD;
}
ll qpow(ll u, ll v) {
    ll rep = 1;
    while (v > 0) {
        if (v & 1) {
            rep = mul(rep, u);
        }
        u = mul(u, u);
        v >>= 1;
    }
    return rep;
}
ll sub(ll a, ll b) {
    return ((a % MOD) - (b % MOD) + MOD) % MOD;
}
ll divi(ll a, ll b) {
    return mul(a, qpow(b, MOD - 2));
}
struct Edge {
    int to, nxt;
} e[N];
int head[N], cnt = 0;
void adde(int u, int v) {
    e[++cnt].to = v;
    e[cnt].nxt = head[u];
    head[u] = cnt;
}

void dfs(int u, int fa) {
    ll an2 = ans2, pvv = -1, bkk = -1, com = 0, an1 = ans1;
    s[++tp] = alp[u];
    if (u == 1 && s[tp] == 0) {
        ans2 = mul(ans2, k);
        an2 = ans2;
        if(deb) printf("pos[%d] :%d  ...(1)\n", u, mul(ans2, ans1));
        finalans ^= mul(ans2, ans1);
    } else if (s[tp] == 0) {
        if (!combo) {
            pv = tp - 1;
            pvv = pv;
        }else{
            pvv= pv;
        }
        combo = 1;
        com = 1;
        ans2 = mul(ans2, (k - 1));
        an2 = ans2;
        if(deb)  printf("pos[%d] :%d   ...(2)\n", u, mul(ans2, ans1));
        finalans ^= mul(ans2, ans1);
    } else {
```

```cpp
        // if(pv==-1){pv = tp; pvv = pv;}

        bk = tp;
        bkk = bk;
        if (combo) {
            if (pv == -1) {
                // printf("Ouch\n");
                //  printf("%d %d\n",k-1,bk-2);
                ans1 = mul(1, qpow(k - 1, bk - 1));
                an1 = ans1;
            } else if (s[pv] == s[bk]) {
                ans1 = mul(ans1, f[1][bk - pv - 1]);
                an1 = ans1;
            } else {
                ans1 = mul(ans1, f[0][bk - pv - 1]);
                an1 = ans1;
            }
            if(deb)  printf("pos[%d] :%d   ...(3)\n", u, ans1);
            finalans ^= ans1;
            ans2 = 1;
            an2 = ans2;

        } else {
            if(deb) printf("pos[%d] :%d   ...(4)\n", u, ans1);
            finalans ^= ans1;
        }
        pv = tp, bk = -1;
        pvv = pv, bkk = bk;
        combo = false;
        com = combo;
    }
    for (int i = head[u]; i; i = e[i].nxt) {
        int v = e[i].to;
        if (v == fa)
            continue;
        dfs(v, u);
        pv = pvv;
        ans2 = an2;
        ans1 = an1;
        combo = com;
        bk = bkk;
        tp--;
    }

}

int main() {

    cin >> n >> k;
    for (int i = 1; i <= n; i++)
        cin >> alp[i];
    for (int i = 1; i < n; i++) {
        int x;
```

```
        cin >> x;
        adde(i+1, x);
        adde(x, i+1);
    }
    f[0][0] = 1;
    for (int i = 1; i <= n; i++) {
        // Since the mod value is 998244353 so there will not be
        // overflow things. Casts are ok.
        f[0][i] = (int)add(f[1][i - 1], mul((k - 2), f[0][i - 1]));
        f[1][i] = (int)mul((k - 1), f[0][i - 1]);
    }
    dfs(1, 0);
    printf("%lld\n", finalans);
}
```

审核难度. [提高], [树上DP]

出题者&验题者的吐槽:

> 我曾建议张蔡(shzaiz)把链改成无序的，不过他没有采纳 –**kingxbz**
>
> 看,我还是很良心的 /cy – **shzaiz**
>
> 这题还是挺水的 — **lyfx**
>
> ~~就这么水的题还想考试?~~ —-**维尼**

4. Euler

出题人	验题人
kingxbz	shzaiz, kingxbz, lyfx, ws(维尼),star,Micoael_Primo.,TopTom

- 20%
  - 拿到题目，忽略一堆废话，题意是这样的:

    $aa = \sum_{i=l_1}^{r_1}\sum_{j=l_1}^{r_1}(a_ib_j - a_jb_i)^2 \mod 1e9+7, bb = \sum_{i=l_2}^{r_2}\sum_{j=l_2}^{r_2}(a_ib_j - a_jb_i)^2 \mod 1e9+7$

    求$gcd(q^{aa}-1, q^{bb}-1)\%998244353$

    我们看aa，bb都是在一个区间内的求和操作，那么，我们就直接暴力求和呗!

    我们可以发现，$n<1000, q<1000$是可以**两重for循环**暴力求解的

    对于每次询问，我们可以在$O(n^2)$的时间内解决问题。加上q次询问，我们可以在最坏复杂度$O(qn^2)$,内解决问题。而$n=1000, q=1000$,且数据不可能跑满（随机数据），这样显然可以通过评测。
  - 但是如何求$gcd(q^{aa}-1, q^{bb}-1).\%998244353$? **[答案]** 见小学课本.
  - `fpow(q,gcd(a,b),998244353)-1`就是答案

> 简要证明:
>
> 根据幼儿园数学知识：$q^a-1|q^{ka}-1$。（**PS**:|是整除符号）

> > 说明: 我们将式子改写为$q^{ka}-1^{ka}$
> >
> > 那么对其进行因式分解
> >
> > 可以化为$(q^a-1^a)*(\ldots\ldots)$
> >
> > 那么显然$q^a-1^a|q^{ka}-1^{ka}$
> >
> > 所以$q^a-1|q^{ka}-1$
>
> 接下来看式子:
>
> 设$x=gcd(a,b)$,$x|a$
>
> 那么$q^a=q^{kx}$,所以$q^x-1|q^{kx}-1$.
>
> 同理$x|b,q^b=q^{lx}$可得$q^x-1|q^{lx}-1$
>
> k,l为gcd的系数，我们知道$q^{gcd(a,b)}|q^a-1\&\&q^{gcd(a,b)}|q^b-1$.
>
> 所以$gcd(q^a-1,q^b-1)=q^{gcd(a,b)}-1$

代码实现1. 20pts

```
signed main()
{
    int n,q;
    cin>>n>>q;
    for(fint i=1;i<=n;i++)
    cin>>a[i];
    for(fint i=1;i<=n;i++)
    cin>>b[i];
    while(q--)
    {
        int op;
        cin>>op;
        int i,x,y,p,l,r,ll,rr;
        if(op==1)
        cin>>i>>x>>y,a[i]=x,b[i]=y;
        if(op==2)
        {
            cin>>p>>l>>r>>ll>>rr;
            int aa=0,bb=0;
            for(fint i=l;i<=r;i++)
            for(fint j=l;j<=r;j++)
            aa+=(a[i]*b[j]-a[j]*b[i])*(a[i]*b[j]-a[j]*b[i]),aa%=mods;
            for(fint i=ll;i<=rr;i++)
            for(fint j=ll;j<=rr;j++)
            bb+=(a[i]*b[j]-a[j]*b[i])*(a[i]*b[j]-a[j]*b[i]),bb%=mods;
            cout<<(fpow(p,gcd(aa,bb),mods2)-1+mods2)%mods2)<<endl;
        }
    }
}
```

- 当然由于:
  - $\sum_{i=1}^{n}\sum_{j=1}^{n}(a_ib_j-a_jb_i)^2 \mod 1e9+7,$

- 恒等于：$2\times\sum_{i=1}^{n}\sum_{j=i+1}^{n}(a_ib_j-a_jb_i)^2 \mod 1e9+7$
- 这份暴力代码同样可以通过20pts：

代码实现2. 20pts

```
signed main()
{
    int n,q;
    cin>>n>>q;
    for(fint i=1;i<=n;i++)
    cin>>a[i];
    for(fint i=1;i<=n;i++)
    cin>>b[i];
    while(q--)
    {
        int op;
        cin>>op;
        int i,x,y,p,l,r,ll,rr;
        if(op==1)
        cin>>i>>x>>y,a[i]=x,b[i]=y;
        if(op==2)
        {
            cin>>p>>l>>r>>ll>>rr;
            int aa=0,bb=0;
            for(fint i=l;i<=r;i++)
            for(fint j=i+1;j<=r;j++)
            aa+=(a[i]*b[j]-a[j]*b[i])*(a[i]*b[j]-a[j]*b[i]),aa%=mods;
            for(fint i=ll;i<=rr;i++)
            for(fint j=i+1;j<=rr;j++)
            bb+=(a[i]*b[j]-a[j]*b[i])*(a[i]*b[j]-a[j]*b[i]),bb%=mods;
            aa*=2,aa%=mods,bb*=2,bb%=mods;
            cout<<fpow(p,gcd(aa,bb),mods2)-1<<endl;
        }
    }
}
```

- 100%
  - 我们观察式子$\sum_{i=1}^{n}\sum_{j=1}^{n}(a_ib_j-a_jb_i)^2 \mod 1e9+7$,

    这个式子恒等于：$2\times\sum_{i=1}^{n}\sum_{j=i+1}^{n}(a_ib_j-a_jb_i)^2 \mod 1e9+7$

    非常显然，不证明了。

    那么我们随便乱拆就可以把式子化简为：

    $\sum_{i=1}^{n}a_i^2b_i^2+\sum_{1\le i<j\le n}a_i^2b_j^2+\sum_{1\le i<j\le n}a_j^2b_i^2$

    这很显然啊！

    我们可以倒推来证明嘛

    $A=\sum_{i=1}^{n}a_i^2b_i^2+\sum_{1\le i<j\le n}a_i^2b_j^2+\sum_{1\le i<j\le n}a_j^2b_i^2$
    $B=\sum_{i=1}^{n}a_i^2b_i^2+2\sum_{1\le i<j\le n}a_ib_ia_jb_j+\sum_{1\le i<j\le n}a_i^2b_j^2-2\sum_{1\le i<j\le n}a_ib_ja_jb_i+\sum_{1\le i<j\le n}a_j^2b_i^2$
    $=\sum_{i=1}^{n}a_i^2b_i^2+\sum_{1\le j\le n}a_i^2b_j^2+\sum_{1\le i<j\le n}a_j^2b_i^2$
    $\therefore A=B$

所以呢，由此可知

$\left(\sum_{i=1}^{n} a_i^2\right)\left(\sum_{i=1}^{n} b_i^2\right)=\left(\sum_{i=1}^{n} a_i b_i\right)^2+\sum_{1\le i<j\le n}\left(a_i b_j-a_j b_i\right)^2$

这就是著名的**拉格朗日恒等式**的内容.

那么对于式子:

$\sum_{i=1}^{n}\sum_{j=1}^{n}(a_i b_j-a_j b_i)^2$

~~(小学的加法交换律知识嘛~~

其实就等于$\left(\sum_{i=1}^{n} a_i^2\right)\left(\sum_{i=1}^{n} b_i^2\right)-\left(\sum_{i=1}^{n} a_i b_i\right)^2$

式子被分为三个部分，我们可以使用

此时，面对1e6的数据，我们显然可以想到用数据结构来维护了。

因为我们需要一个$O(nlogn)$的复杂度.

由于数据是满的，分块，莫队这种$n\sqrt{n}$级别的数据结构咋弄也得T。

所以只能线段树/树状数组。

分别维护$\left(\sum_{i=1}^{n} a_i^2\right)\left(\sum_{i=1}^{n} b_i^2\right)-\left(\sum_{i=1}^{n} a_i b_i\right)^2$

即可。

本题略微卡常，加快读的三份树状数组在**windows**下勉强通过，(在luogu机子上很轻松通过) 不建议使用线段树这种常数大的做法来写(极大概率被卡常)。

在树状数组中，我们单点修改只需通过加法操作即可实现，具体如下:

```
adds(0,i,(1LL*x*x-1LL*a[i]*a[i])%mods),
adds(1,i,(1LL*y*y-1LL*b[i]*b[i])%mods),
adds(2,i,(1LL*x*y-1LL*a[i]*b[i])%mods);
```

最后,

由于我们要求的是

$\sum_{i=1}^{n}\sum_{j=1}^{n}(a_i b_j-a_j b_i)^2$

而不是

$\sum_{i=1}^{n}\sum_{j=i+1}^{n}(a_i b_j-a_j b_i)^2$

所以，记得最后的结果$\times 2$

然后用快速幂维护$gcd(q^{aa}-1,q^{bb}-1)=q^{gcd(a,b)}-1$

别忘了两次操作模数不一样哦!

代码实现: 100分

```
#include<bits/stdc++.h>
#define fint register int
#define h 5001
#define N 1000001
#define int long long
using namespace std;
const int mods=1e9+7;
const int mods2=998244353;
int n,q,a[N],b[N],t[N][4];
inline int read();
```

```
inline int Pow(int x);
inline void adds(int id,int x, int y);
inline int qry(int id,int x);
inline int lowbit(int x);
inline int gcd(int a,int b);
inline int query(int id, int l, int r);
inline int fpow(int b,int pp,int k);
signed main()
{
    ios::sync_with_stdio(false);
    n=read(),q=read();
    for(fint i=1;i<=n;i++)
    a[i]=read();
    for(fint i=1;i<=n;i++)
    b[i]=read();
    for(fint i=1;i<=n;i++)

adds(0,i,1ll*a[i]*a[i]%mods),adds(1,i,1LL*b[i]*b[i]%mods),adds(2,i,1LL*a[i]*b[i]
%mods);
    int tot=0;
    while(q--)
    {
        int op;
        op=read();
        if(op==1)
        {
            int i,x,y;
            i=read(),x=read(),y=read();
            adds(0,i,(1LL*x*x-1LL*a[i]*a[i])%mods),
            adds(1,i,(1LL*y*y-1LL*b[i]*b[i])%mods),
            adds(2,i,(1LL*x*y-1LL*a[i]*b[i])%mods);
            a[i]=x,b[i]=y;
        }
        else
        {
            int p,l_a,r_a,l_b,r_b;
            p=read(),l_a=read(),r_a=read(),l_b=read(),r_b=read();
            int aa=((1LL*query(0,l_a,r_a)*query(1,l_a,r_a)-
Pow(query(2,l_a,r_a)))%mods+mods)%mods;
            int bb=((1LL*query(0,l_b,r_b)*query(1,l_b,r_b)-
Pow(query(2,l_b,r_b)))%mods+mods)%mods;
            aa*=2,aa%=mods,bb*=2,bb%=mods;
            cout<<(fpow(p,gcd(aa,bb),mods2)-1+mods2)%mods2<<endl;
        }
    }
    return 0;
}

inline int read()
{
    int x=0,f=1;
    char ch=getchar();
    while(ch<'0'||ch>'9')
```

```
    {
        if(ch=='-')
        f=-1;
        ch=getchar();
    }
    while(ch>='0'&&ch<='9')
    {
        x=(x<<1)+(x<<3)+(ch^48);
        ch=getchar();
    }
    return x*f;
}

inline int Pow(int x)
{
    return 1LL*x*x%mods;
}

inline int lowbit(int x)
{
    return x&(-x);
}

inline void adds(int id,int x, int y)
{
    y=(y+mods)%mods;
    for(fint i=x,j=0;i<=n;i+=lowbit(i))
    t[i][id]+=y,t[i][id]>mods?t[i][id]-=mods:j++;
    return ;
}

inline int query(int id, int l, int r)
{
    return qry(id,r)-qry(id,l-1);
}

inline int qry(int id,int x)
{
    int tot=0;
    for(fint i=x,j=0;i;i-=lowbit(i))
    tot+=t[i][id],tot>mods?tot-=mods:j++;
    return tot;
}

inline int fpow(int b,int pp,int k)
{
    int ans=1;
    while(pp)
    {
        if(pp&1LL)
        ans=ans*b%k;
        b=b*b%k;
        pp>>=1LL;
```

```
    }
    return ans;
}

inline int gcd(int a,int b)
{
    if(b==0)
    return a;
    return gcd(b,a%b);
}
/*
3 3
1 2 3
3 2 1
2 3 1 3 1 2
1 2 6 1
2 2 1 3 2 3
*/
```

审核难度. [提高+], [数据结构, 数学]

出题者&验题者的吐槽:

> 这题出简单了，本来还有个区间求前驱的操作，结果删了。原本是在树上搞的，最后也给放序列里了。所以现在你们做到的是**超级弱化版**的。 –kingxbz