比赛链接

补题情况

题目	蒋贤蒙	王赵安	王智彪
A	2	0	0
C	2	1	0
E	2	0	1
F	0	1	2
G	2	0	2
I	2	1	0
J	2	1	1

题解

C. Cheating and Stealing

题意

给定一个长度为 n 的 01 串 S，对每个 $i=1\sim n$，询问下述流程的结果：

1. 初始化答案为 0
2. 找到最短的前缀满足至少有 i 个 0 或者 i 个 1，且 0 的个数和 1 的个数的差值不小于 2，如果没有满足条件的前缀则输出答案
3. 对这个前缀，如果 1 比 0 多，则答案加一
4. 删除这个前缀，跳回操作 2

题解

首先不难发现对于固定 i，由于每次操作至少取出长度为 i 的前缀，所以上述操作最多执行 $O\left(\frac ni\right)$ 次，所以总操作次数 $O(n\log n)$。

所以如果能 $O(1)$ 找到每次操作满足条件的前缀，即可 $O(n\log n)$ 解决此题。

首先考虑如何找到至少有 i 个 0 或者 i 个 1 的最短前缀，可以提前记录第 k 个 0 和 1 的位置，分被为 $p(0,k)$ 和 $p(1,k)$。

于是可以 $O(1)$ 跳转。接下来在这个位置的基础上寻找满足 0 的个数和 1 的个数的差值不小于 2 的位置。

如果当前位置已经满足条件，则已经找到前缀。否则假如当前位置的得分差为 1，则在移动一位，使得分差为 0 或 2。如果是 2 则也已经找到前缀。

接下来只需要考虑得分差为 0 的情况，提前维护 next 数组表示从得分相同到比赛结束的位置。

不难发现有 $\text{next}(i)=(s[i]==s[i+1])?(i+1):\text{next}(i+2)$，提前预处理后也可以 $O(1)$ 跳转。

```
const int MAXN=1e6+5,MAXM=21,mod=998244353;
char s[MAXN];
int pre[MAXN],nxt[MAXN],det[MAXN],p1[MAXN],p0[MAXN],cnt1,cnt0;
int quick_pow(int n,int k){
    int ans=1;
    while(k){
        if(k&1)ans=1LL*ans*n%mod;
        n=1LL*n*n%mod;
        k>>=1;
    }
    return ans;
}
int solve(int n,int i){
    int lef=1,ans=0;
    while(true){
        int c1=cnt1-pre[lef-1],c0=n-lef+1-c1,rig=n+1;
        if(c1>=i)
        rig=min(rig,p1[pre[lef-1]+i]);
        if(c0>=i)
        rig=min(rig,p0[lef-1-pre[lef-1]+i]);
        if(rig==n+1)
        break;
        if(abs(det[rig]-det[lef-1])<2){
            if(det[rig-1]!=det[lef-1])
            rig++;
            rig=nxt[rig];
        }
        if(rig==0)
        break;
        if(det[rig]-det[lef-1]>=2)
        ans++;
        lef=rig+1;
    }
    return ans;
}
int main(){
    int n=read_int();
    scanf("%s",s+1);
    _rep(i,1,n){
        if(s[i]=='W'){
            p1[++cnt1]=i;
            pre[i]=1;
            det[i]=1;
        }
        else{
            p0[++cnt0]=i;
            pre[i]=0;
            det[i]=-1;
        }
```

```
        pre[i]+=pre[i-1];
        det[i]+=det[i-1];
    }
    for(int i=n-1;i;i--)
    nxt[i]=(s[i]==s[i+1])?i+1:nxt[i+2];
    int ans=0;
    _rep(i,1,n)
    ans=(ans+1LL*solve(n,i)*quick_pow(n+1,i-1))%mod;
    enter(ans);
    return 0;
}
```

E. Eert Esiwtib

题意

给定一棵以 1 为根的点权树，记第 i 个点的原始点权为 a_i□每条边有一种操作符，可能为 $\text{OR},\text{AND},\text{XOR}$□

设路径 $u\to v$ 上的点权和操作符依次为 $p_1,e_1,p_2\cdots e_{k-1},p_k$□则路径的权重 $w(u,v)=p_1 e_1(p_2 e_2(\cdots (p_{k-2}e_{k-2}(p_{k-1}e_{k-1}p_k))\cdots))$□

定义 $\text{Tree}(u)$ 表示 u 的子树，不包括 u 本身。接下来若干询问，每次给定 d,u□将每个点的点权变为 $a_i+d\times i$□求

$$ \text{OR}_{v\in Tree(u)}w(u,v),\text{AND}_{v\in Tree(u)}w(u,v),\text{XOR}_{v\in Tree(u)}w(u,v) $$

注意，每组询问对点权的修改独立，即一个询问对点权的修改不影响另一个询问。同时有 $0\le d\le 100$□

题解

发现 d 很小，直接枚举 d□暴力树形 dp 即可。设 $\text{dp}(u,0/1/2)$ 表示 u 求 $\text{OR},\text{AND},\text{XOR}$ 情况下的答案，大力分类讨论即可。

注意权值直接运算时每个位是独立的，所以在讨论时可以只考虑权值是 $0/1$ 的情况，然后枚举 u 是 $0,1$ 考虑一下即可。

例如，考虑边是 XOR 的情况，计算下式

$$ (a_u\oplus v_1)|(a_u\oplus v_2)|\cdots (a_u\oplus v_k) $$

首先假设 $a_u=0$□得到 $(a_u\oplus v_1)|(a_u\oplus v_2)|\cdots |(a_u\oplus v_k)=v_1|v_2|\cdots |v_k=\text{dp}(v,0)$□

假设 $a_u=1$□得到 $(a_u\oplus v_1)|(a_u\oplus v_2)|\cdots |(a_u\oplus v_k)=(\sim v_1)|(\sim v_2)|\cdots |(\sim v_k)=\sim (v_1\And v_2\And \cdots \And v_k)=\sim \text{dp}(v,1)$□

于是有 $\text{dp}(u,0)|=((\sim a_u)\And \text{dp}(v,0))|(a_u\And (\sim \text{dp}(v,1)))$□注意

$\text{dp}(v)$ 不包含 v 本身的贡献所以还有 $\text{dp}(u,0)|=a_u|a_v$。

总时间复杂度 $O(nd)$。

```
const int MAXN=1e5+5,MAXV=105;
struct Edge{
    int to,w,next;
}edge[MAXN];
int head[MAXN],edge_cnt;
void Insert(int u,int v,int w){
    edge[++edge_cnt]=Edge{v,w,head[u]};
    head[u]=edge_cnt;
}
LL a0[MAXN],a[MAXN],dp[MAXN][3];
int sz[MAXN];
void dfs(int u){
    dp[u][0]=dp[u][2]=0;
    dp[u][1]=-1;
    sz[u]=1;
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        dfs(v);
        sz[u]+=sz[v];
        LL t;
        if(edge[i].w==0)t=a[u]|a[v];
        else if(edge[i].w==1)t=a[u]&a[v];
        else t=a[u]^a[v];
        dp[u][0]|=t;
        dp[u][1]&=t;
        dp[u][2]^=t;
        if(sz[v]==1)continue;
        sz[v]--;
        if(edge[i].w==0){
            dp[u][0]|=a[u]|dp[v][0];
            dp[u][1]&=a[u]|dp[v][1];
            if(sz[v]&1)
            dp[u][2]^=a[u]|((~a[u])&dp[v][2]);
            else
            dp[u][2]^=(~a[u])&dp[v][2];
        }
        else if(edge[i].w==1){
            dp[u][0]|=a[u]&dp[v][0];
            dp[u][1]&=a[u]&dp[v][1];
            dp[u][2]^=a[u]&dp[v][2];
        }
        else{
            dp[u][0]|=(a[u]&(~dp[v][1]))|((~a[u])&dp[v][0]);
            dp[u][1]&=(a[u]&(~dp[v][0]))|((~a[u])&dp[v][1]);
            if(sz[v]&1)
            dp[u][2]^=a[u]^dp[v][2];
```

```
            else
            dp[u][2]^=dp[v][2];
        }
    }
}
vector<pair<int,int> > c[MAXV];
LL ans[MAXN][3];
int main(){
    int n=read_int(),q=read_int();
    _rep(i,1,n)
    a0[i]=read_int();
    _rep(i,2,n){
        int f=read_int(),s=read_int();
        Insert(f,i,s);
    }
    _for(i,0,q){
        int d=read_int(),u=read_int();
        c[d].push_back(make_pair(i,u));
    }
    _for(d,0,MAXV){
        _rep(i,1,n)
        a[i]=a0[i]+i*d;
        dfs(1);
        for(pair<int,int> t:c[d]){
            _for(j,0,3)
            ans[t.first][j]=dp[t.second][j];
        }
    }
    _for(i,0,q){
        space(ans[i][0]);
        space(ans[i][1]);
        enter(ans[i][2]);
    }
    return 0;
}
```

F. Finding Points

题意

给定一个凸包，点按照逆时针给出，然后求凸包内一点，想要这个点与这个凸多边形相邻点组成的 n 个角的最小值最大，求这个最大值 $(4≤n≤100)$

题解

赛场上两分钟出思路，然后看通过率…感觉是不是有坑就没敢写…赛后听说改数据了…血亏！

显然要二分（废话）。

然后对于每一组相邻的点，这个点和这两个点组成的角大于某个角，则这个点一定在这两个点组成的大弓形内，根据圆周角求 n 个圆看面积交即可，然后比赛的时候猜到会有点跑到凸包外面的情况，但是我不会写圆的交再交凸包，这也是我怂了的原因之一，谁知道改数据嘛！

所以就是个板子题…

```
int N;
Point ppx[110];
int main() {
    scanf("%d",&C.n);
    N=C.n;
    for(int i=0;i<C.n;i++) {
        ppx[i].input();
    }
    ppx[C.n]=ppx[0];
    double l=eps,r=2.0*acos(-1)/N;
    while(r-l>1e-15) {
        double mid=(l+r)/2;
        for (int i=0;i<C.n;i++) {
            C.c[i].r=(ppx[i].distance(ppx[i+1])/2/sin(mid));
            double dtmp=C.c[i].r*cos(mid);
            Point ptmp=(ppx[i]+ppx[i+1])/2;
            Point pptmp=(ppx[i]-ptmp);
            pptmp=pptmp.rotright();
            pptmp=pptmp.trunc(dtmp);
            ptmp=ptmp+pptmp;
            C.c[i].p=ptmp;
        }
        C.getarea();
        if(C.ans[C.n]>1e-20)l=mid;
        else r=mid;
    }
    printf("%.20lf\n",l*180/pi);
    return 0;
}
```

G. Greater Integer, Better LCM

题意

给一个大数的质因数分解形式，设为 c 并给出 a 和 b 求最小的 $x+y$ 使得 $lcm(a+x,b+y)=c$ $(1≤n≤18)$ 代表质因子个数， 保证因子幂次和相加不超过 18 $a,b,c≤10^{32}$

题解

类似于数位 dp 我们对每一个质因子进行枚举幂次，并且状压，位为 1 代表这一个因子的幂次取满，

不取满为 0 □ $dp[con]$ 代表这个压缩状态下的相对于 b 的最小代价，也就是 b 最少要补多少才能到这个状态。

于是我们跑出初始每个状态下的最小代价，但是还需要处理 a □我们把每一个末状态放进 vec 里，然后看哪些比 a 大，代价是存的 v 值减 a □剩下至少需要满足 $(1<<n)-1-con$ 的状态，于是我们需要找一个无后效性的求状态数组的方法，就是让 dp 数组变成至少满足这个状态的最小代价。再处理一下就好了

```
#include <bits/stdc++.h>
using namespace std;
#define ll __int128

inline void scan(ll &X) {
    X = 0;
    int w=0;
    char ch=0;
    while(!isdigit(ch)) {
        w|=ch=='-';
        ch=getchar();
    }
    while(isdigit(ch)) X=(X<<3)+(X<<1)+(ch^48),ch=getchar();
    if (w) X = -X;
}
void print(ll x) {
    if (!x) return ;
    if (x < 0) putchar('-'),x = -x;
    print(x / 10);
    putchar(x % 10 + '0');
}

ll n,p[110],q[110],a,b;
ll dp[270000];
vector<pair<ll,int> > d;

void dfs(ll pos,ll value,int con) {
    if(pos==n) {
        d.push_back(make_pair(value,con));
        if(value>=b) dp[con]=value-b;
        return;
    }
    for(int i=0;i<=q[pos];i++) {
        dfs(pos+1,value,(i==q[pos])?(con|(1<<pos)):con);
        value=value*p[pos];
    }
}

int main() {
    memset(dp,0x3f3f3f3f,sizeof(dp));
    scan(n);
    for(int i=0;i<n;i++) scan(p[i]),scan(q[i]);
    scan(a);
```

```
    scan(b);
    dfs(0,1,0);
    for(int i=0;i<n;i++) {
        for(int j=0;j<(1<<n);j++) {
            if(!(j&(1<<i))) dp[j]=min(dp[j],dp[j+(1<<i)]);
        }
    }
    ll ans=1e36;
    for (int i=0;i<d.size();i++) {
        pair<ll,int> x=d[i];
        if (x.first>=a) ans=min(ans,x.first-a+dp[(1<<n)-1-x.second]);
    }
    if(ans) print(ans);
    else puts("0");
    return 0;
}
```

被吊打的标算

考虑枚举 $S=\{p_1,p_2\cdots p_n\}$ 的所有子集。

对每个子集 $T=\{a_1,a_2\cdots a_i\}$，强制令 x 取 $\{p_1,p_2\cdots p_n\}-T$ 的每个素因子的最高次幂，强制令 y 取 T 的每个素因子的最高次幂。

这样，就消除了 $\text{lcm}(x,y)=c$ 的限制，接下来分别考虑 $x\ge a,y\ge b$ 的限制。

不难发现 x 在 $a_1,a_2\cdots a_i$ 的幂次都是自由的，设 $x=t\cfrac{c}{a_1^{q_1}a_2^{q_2}\cdots a_i^{q_i}}$，因此需要找到最小的 $t\mid a_1^{q_1}a_2^{q_2}\cdots a_i^{q_i}$，且 $x\ge a$。

一种暴力解法为直接枚举 $a_1^{q_1}a_2^{q_2}\cdots a_i^{q_i}$ 的所有因子，最坏情况下 c 有 n 个因子，每个因子幂次均为 1。

此时时间复杂度等价于子集枚举套子集枚举的时间复杂度，可以认为是

$$ \sum_{i=0}^n {n\choose i}2^i=(1+2)^n $$

考虑一个优化，将 $a_1^{q_1}a_2^{q_2}\cdots a_i^{q_i}$ 平均分成两个序列，每个序列枚举因子，对一个序列的因子进行排序，然后另一个序列进行二分查找。

这样里层子集枚举的复杂度优化为 $O\left(n{\sqrt 2}^n\right)$，总时间复杂度为

$$ \sum_{i=0}^n {n\choose i}i{\sqrt 2}^i=(1+\sqrt 2)^{n+1} $$

```
const int MAXN=18;
int n,p[MAXN],q[MAXN];
LL A[1<<MAXN],B[1<<MAXN];
LL vec1[1<<MAXN],vec2[1<<MAXN];
vector<pair<int,int> > d;
int cal(int l,int r,LL *vec){
```

```
    int n=0;
    vec[n++]=1;
    _for(i,l,r){
        int last=n;
        LL x=1;
        _for(j,0,d[i].second){
            x*=d[i].first;
            _for(k,0,last)
            vec[n++]=vec[k]*x;
        }
    }
    return n;
}
void solve(LL v,LL *ans){
    _for(i,0,1<<n){
        LL base=1;
        d.clear();
        _for(j,0,n){
            if(i&(1<<j))
            d.push_back(make_pair(p[j],q[j]));
            else{
                _for(k,0,q[j])
                base*=p[j];
            }
        }
        int n1=cal(0,d.size()/2,vec1);
        int n2=cal(d.size()/2,d.size(),vec2);
        sort(vec1,vec1+n1);
        ans[i]=1e32;
        _for(j,0,n2){
            vec2[j]*=base;
            int pos=lower_bound(vec1,vec1+n1,(v+vec2[j]-1)/vec2[j])-vec1;
            if(pos!=n1)
            ans[i]=min(ans[i],vec1[pos]*vec2[j]);
        }
    }
}
int main(){
    n=read_int();
    _for(i,0,n){
        p[i]=read_int();
        q[i]=read_int();
    }
    LL a=read_LL(),b=read_LL();
    solve(a,A);
    solve(b,B);
    LL ans=1e32;
    int S=(1<<n)-1;
    _for(i,0,1<<n)
    ans=min(ans,A[i]+B[S^i]-a-b);
    enter(ans);
```

```
    return 0;
}
```

I. Interval Queries

题意

给定一个长度为 n 的序列 A接下来有 q 个询问。每次询问给定 l,r,k求

$$ \sum_{i=0}^{k-1}f(l-i,r+i)(n+1)^i $$

其中 f 定义如下

$$ f(l,r)=\max(\{k|\exists x\forall i(0\le i\le k-1\to \exists j(l\le j\le r,a_j=x+i))\}) $$

题解

考虑回滚莫队处理询问，难点在于怎么维护最大的 k

实际上，这等价于维护数轴上的最长连续线段。可以令 $\text{vl}(x)$ 表示以数 x 为左端点的线段在数轴上的右端点。

$\text{vr}(x)$ 表示以数 x 为右端点的线段在数轴上的左端点。于是只需要保证每条线段的两个端点的 vl,vr 正确性即可，不难 $O(1)$ 维护。

时间复杂度 $O\left(n\sqrt q+\sum k\right)$

```
const int MAXN=1e5+5,mod=998244353;
int blk_sz,a[MAXN];
struct query{
    int l,r,k,idx;
    bool operator < (const query &b)const{
        if(l/blk_sz!=b.l/blk_sz)return l<b.l;
        return r<b.r;
    }
}opt[MAXN];
struct node{
    int p1,v1,p2,v2,ans;
}st[MAXN];
int curlen,vis[MAXN],vl[MAXN],vr[MAXN],tp;
void add(int v){
    if(vis[v]++)return;
    int l=vis[v-1]?vl[v-1]:v;
    int r=vis[v+1]?vr[v+1]:v;
    st[++tp]=node{l,vr[l],r,vl[r],curlen};
    vr[l]=r;
    vl[r]=l;
```

```
    curlen=max(curlen,r-l+1);
}
void del(int v){
    if(--vis[v])return;
    vr[st[tp].p1]=st[tp].v1;
    vl[st[tp].p2]=st[tp].v2;
    curlen=st[tp].ans;
    tp--;
}
int solve(int l,int r,int k,int n){
    int ans=curlen,base=1;
    _for(i,1,k){
        add(a[l-i]);
        add(a[r+i]);
        base=1LL*base*(n+1)%mod;
        ans=(ans+1LL*curlen*base)%mod;
    }
    for(int i=k-1;i;i--){
        del(a[r+i]);
        del(a[l-i]);
    }
    return ans;
}
int ans[MAXN];
int main()
{
    int n=read_int(),q=read_int();
    _rep(i,1,n)
    a[i]=read_int();
    _for(i,0,q){
        int l=read_int(),r=read_int(),k=read_int();
        opt[i]=query{l,r,k,i};
    }
    blk_sz=n/sqrt(q)+1;
    sort(opt,opt+q);
    _for(i,0,q){
        if(opt[i].l/blk_sz==opt[i].r/blk_sz){
            _rep(j,opt[i].l,opt[i].r)
            add(a[j]);
            ans[opt[i].idx]=solve(opt[i].l,opt[i].r,opt[i].k,n);
            for(int j=opt[i].r;j>=opt[i].l;j--)
            del(a[j]);
        }
    }
    int lef=1,rig=0,lst=-1;
    _for(i,0,q){
        if(opt[i].l/blk_sz!=opt[i].r/blk_sz){
            if(opt[i].l/blk_sz!=lst){
                while(lef<=rig)del(a[rig--]);
                lst=opt[i].l/blk_sz;
                rig=min(lst*blk_sz+blk_sz-1,n);
```

```
                lef=rig+1;
            }
            while(rig<opt[i].r)add(a[++rig]);
            int tlef=lef;
            while(tlef>opt[i].l)add(a[--tlef]);
            ans[opt[i].idx]=solve(opt[i].l,opt[i].r,opt[i].k,n);
            while(tlef<lef)del(a[tlef++]);
        }
    }
    _for(i,0,q)enter(ans[i]);
    return 0;
}
```