比赛链接

题解

B. Cannon

题意

给定上下两行格点的数量，规定一个操作是对于其中一行，选择一个点，模仿中国象棋里的炮的攻击规则，中间有且只有一个点，跳到另一个点，然后消失一个点，问所有这种事件的发生数，分有限制和没有限制。有限制指对第一行操作若干步（可以是 0），然后对第二行操作，之后不能退回到第一行。

题解

显然如果一行有 n 个点，操作一次的方案数是 $2×(n-2)$ 之后剩余 $n-1$ 个点，方案数是 $2×(n-3)$ 所以可以知道 n 个炮操作 m 次的方案数是 $2^{m}{\frac {(n-2)!} {(n-2-m)!}}$ 为了方便起见我们把 $x-=2,y-=2$ 于是转变为求$2^{k} \sum {\frac {k!} {i!(k-i)!}} {\frac {n!} {(n-i)!}} {\frac {m!} {(m-(k-i))!}}$ 和 $2^{k} \sum {\frac {n!} {(n-i)!}} {\frac {m!} {(m-(k-i))!}}$ 两个子问题。

对于第一个子问题，我们发现可以把 $k!$ 提出来，之后对于里面的式子，可以变成两个组合数的乘积，再利用组合数的常用结论，可以推出其等于 $2^{k}k!C(n+m,k)$ 这个式子可以通过预处理 2 的方幂，阶乘以及阶乘的逆来 $O(1)$ 求得，于是总共复杂度 $O(n+m)$

对于第二个子问题 $2^{k} \sum {\frac {n!} {(n-i)!}} {\frac {m!} {(m-(k-i))!}} = 2^{k} \sum {\frac {n!m!} {(n+m-k)!}} {\frac {(n+m-k)!} {(n-i)!(m-(k-i))!}} = 2^{k} {\frac {n!m!} {(n+m-k)!}} \sum_{i=n-k}^{n} C(n+m-k,i)$

虽然没有直接的结论，但是这个可以用步移算法来解决。

设 $S(n,m) = \sum_{i=0}^{m} C(n,i)$

则 $S(n,m+1) = S(n,m) + C(n,m+1),S(n,m-1) = S(n,m) - C(n,m),S(n+1,m) = 2S(n,m) - C(n,m)$

于是相当于一个前缀和的问题，然后我们已知的是 0 步的方法是 1，然后通过枚举 k 我们发现， 减数之间存在关系可以用上述关系式求出下一项的减数，被减数也一样，于是复杂度也是 $O(n+m)$

总结：最难的还是推式子，其次是想起来步移算法（可能只有我会想不起来…）。

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

const ll maxn=1e7+10;
const ll mod=1e9+9;
ll x,y;
ll jc[maxn],inv[maxn],ec[maxn];

ll C(int n,int m) {
    if(n<m||m<0) return 0;
```

```
    return jc[n]*inv[m]%mod*inv[n-m]%mod;
}
int main() {
    scanf("%lld %lld",&x,&y);
    inv[0]=jc[1]=inv[1]=jc[0]=ec[0]=1;
    ec[1]=2;
    for(int i=2;i<=x+y;i++) {
        ec[i]=ec[i-1]*2%mod;
        jc[i]=jc[i-1]*i%mod;
        inv[i]=(mod-mod/i)*inv[mod%i]%mod;
    }
    for(int i=2;i<=x+y;i++) inv[i]=inv[i-1]*inv[i]%mod;
    ll ans1=0;
    for(int i=0;i<=x+y-4;i++) {
        ans1=ans1^(ec[i]*jc[i]%mod*C(x+y-4,i)%mod);
    }
    printf("%lld ",ans1);
    ll tmp1=1,tmp2=0;
    ll ans2=0;
    x-=2;y-=2;
    ll tt=jc[x]*jc[y]%mod;
    for(int i=0;i<=x+y;i++) {
        ans2=ans2^((ec[x+y-i]*tt%mod*inv[i]%mod*(tmp1-tmp2)%mod+mod)%mod);
        tmp1=(tmp1*2%mod-C(i,x)+mod)%mod;
        //tmp2=(tmp2*2%mod-C(i,x-i-1)*2%mod-C(i,x-i-2)+mod*2)%mod;
        tmp2=(tmp2*2%mod+C(i,i-y))%mod;
    }
    printf("%lld\n",ans2);
    return 0;
}
```

G. League of Legends

题意

给定 n 条线段，要求将线段分为 k 组，使得每组的线段交非空，最大化每组组的线段交之和。

题解

首先假定所有线没有互相包含的关系，将所有线段 $[l_i,r_i)$ 按 r_i 从大到小排列，设 $\text{dp}(i,j)$ 表示将 j 条线段分到 前 i 组得到的答案。

将第 $j+1$ 到 k 条线段分到同一组，如果 $l_k\gt r_{j+1}$则线段交非空且 $\text{dp}(i-1,j)$ 合法，不难得到如下状态转移

$$ \text{dp}(i,k)\gets \text{dp}(i-1,j)+l_k-r_{j+1} $$

由于所有线没有互相包含的关系且 r_i 递减，不难发现 l_i 也递减。

因此对 $i\gt j$□如果 $l_k\le r_{i+1}$□则一定有 $l_k\le r_{j+1}$□同时如果 $l_j\le r_{k+1}$□则 $l_i\le r_{k+1}$□所以决策具有单调性。

于是每个 i□$O(n)$ 单调队列维护所有合法 $\text{dp}(i-1,j)-r_{j+1}$ 即可。总时间复杂度 $O(nk)$□

接下来考虑某些可以包含其他线段的线段，首先将任何一条线段放入一个已经存在的组一定使得答案不增。

而如果要将一条包含其他线段的线段放入一个已经存在的组，则将它放入一个被它包含的线段所在的组一定使得答案不减，是最佳选择。

因此对满足上述条件的线段只有两种最优选择，一种是放入已经存在的组，这样对答案无贡献。一种是创建一个组，这样贡献为该线段长度。

于是可以处理完所有不包含其他线段的线段，然后枚举他们分成的组数 i□然后取前 $k-i$ 条包含其他线段的线段独立建组构成答案。

至于如果判定一条线段是否包含其他线段，可以将所有线段按 r_i 第一关键字从大到小排序 l_i 第二关键字从小到大排序，然后维护最小右端点。

```
const int MAXN=5e3+5,inf=1e9;
struct Seg{
    int l,r;
    bool operator < (const Seg &b)const{
        return l>b.l||(l==b.l&&r<b.r);
    }
}seg[MAXN],a[MAXN];
int len[MAXN],dp[MAXN][MAXN];
pair<int,int> que[MAXN];
int main()
{
    int n=read_int(),k=read_int();
    _for(i,0,n){
        seg[i].l=read_int();
        seg[i].r=read_int();
    }
    sort(seg,seg+n);
    int minr=inf,n1=0,n2=0;
    _for(i,0,n){
        if(seg[i].r<minr){
            minr=seg[i].r;
            a[++n1]=seg[i];
        }
        else
        len[++n2]=seg[i].r-seg[i].l;
    }
    mem(dp,-1);
    dp[0][0]=0;
    _rep(i,1,k){
        int head=0,tail=-1;
        _rep(j,1,n1){
```

```
            if(~dp[i-1][j-1]){
                pair<int,int> t=make_pair(dp[i-1][j-1]-a[j].l,-a[j].l);
                while(head<=tail&&que[tail]<t)tail--;
                que[++tail]=t;
            }
            while(head<=tail&&a[j].r+que[head].second<=0)head++;
            if(head<=tail)
            dp[i][j]=que[head].first+a[j].r;
        }
    }
    sort(len+1,len+n2+1,greater<int>());
    _rep(i,1,n2)len[i]+=len[i-1];
    int ans=0;
    _rep(i,0,min(n2,k)){
        if(~dp[k-i][n1])
        ans=max(ans,dp[k-i][n1]+len[i]);
    }
    enter(ans);
    return 0;
}
```

J. Product of GCDs

题意

给 S 个数字，让你求出所有子集大小为 k 的集合中的数的最大公因数之积。

题解

我们可以分成质数来考虑这个问题，分别统计不同数字在整个序列中有多少个数能除开它，这个操作是可以 $mlogm$ 完成的，其中 m 是数字大小范围。

当我们统计出不同质数的方幂在序列中被除开的个数时，我们可以用组合数求出来有多少个集合的最大公约数能够除开这个数，并且统计这个集合的次数恰好等于这个质数的方幂数，我们将所有次数加在一起，是一个质数在最后答案出现的次数，先求出 P 的欧拉函数，用次数取模，最后用快速幂算一下，将所有这些答案乘起来就是最后的答案了，又因为我们算的是乘法并且每次的质数不一样，所以保证了正确性。

其中我们需要预处理出 P 的所有素因子（为了求欧拉函数值），处理出序列中有多少个数字能除开范围内的任意一个数字。处理出组合数对欧拉函数值取模，最后统计答案就是找所有的素数，看他们的方幂出现的次数，相加，对欧拉函数值取模，最后再快速幂乘起来，就可以愉悦 AC 本题了。

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

inline int read() {
    int X=0,w=1;
```

```
    char c=getchar();
    while (c<'0'||c>'9') {
        if (c=='-') w=-1;
        c=getchar();
    }
    while (c>='0'&&c<='9') X=(X<<3)+(X<<1)+c-'0',c=getchar();
    return X*w;
}
inline void write(int x){
    char c[21],len=0;
    if(!x)return putchar('0'),void();
    if(x<0)x=-x,putchar('-');
    while(x)c[++len]=x%10,x/=10;
    while(len)putchar(c[len--]+48);
}

inline ll quick_mul(ll a,ll b,ll p){
    return (ll)((__int128)a*b%p);
}
inline ll quick_power(ll a,ll b,ll p){
    ll ret=1;
    while(b){
        if(b&1) ret=quick_mul(ret,a,p);
        a=quick_mul(a,a,p);
        b>>=1;
    }
    while(ret<0) ret+=p;
    return ret%p;
}

const int maxn=80000+5,maxs=40000+5,maxk=30+2,D=1e7+10,PD=1e6+3;
int T,S,k;
ll P;
int sum[maxn],prime[PD],ps;
long long C[maxs][maxk];
bool isPrime[D];
inline void initial() {
    memset(isPrime, 1, sizeof(isPrime));
    for (int i = 2; i < D; ++i) {
        if (isPrime[i]) {
            prime[ps++]=i;
            for (int j = i + i; j < D; j += i) {
                isPrime[j] = false;
            }
        }
    }
}

int main() {
    initial();
    T=read();
```

```
    while(T--) {
        S=read(); k=read(); scanf("%lld",&P);
        for(int i=0;i<maxn;i++) sum[i]=0;
        for(int i=0,tmp; i<S; i++) {
            tmp=read();
            sum[tmp]++;
        }
        for(int i=2; i<maxn; i++) {
            for(int j=i+i; j<maxn; j+=i) {
                sum[i]+=sum[j];
            }
        }
        long long tmp=P,phi_ans=1;
        for(int i=0; i<ps; i++) {
            if(tmp%prime[i]==0) {
                phi_ans=phi_ans*(prime[i]-1);
                tmp/=prime[i];
                while(!(tmp%prime[i])) {
                    tmp/=prime[i];
                    phi_ans*=prime[i];
                }
            }
        }
        if(tmp>1) phi_ans=phi_ans*(tmp-1);
        for(int i=0; i<=S; i++) {
            C[i][0]=1;
            for(int j=1; j<=i&&j<=k; j++) {
                C[i][j] = C[i-1][j] + C[i-1][j-1];
                while(C[i][j]>=phi_ans+phi_ans) C[i][j]-=phi_ans;
            }
        }
        long long num;
        ll ans=1;
        for(int i=2; i<maxn; i++) {
            if(isPrime[i]) {
                num=0;
                for(ll j=i; j<maxn; j*=i) {
                    num+=C[sum[j]][k];
                    while(num>=phi_ans+phi_ans) num-=phi_ans;
                }
                ans=quick_mul(ans,quick_power(i,num,P),P);
                //printf("%d %d %lld\n",i,num,ans);
            }
        }
        printf("%lld\n",ans);
    }
    return 0;
}
```

赛后总结

jxm过了签到后就一直debug先给自己de然后帮队友de~~后来de不下去直接重写了一份~~。再后来就开始罚坐了，甚至没把所有题看完，或许下次应该在罚坐的时候把所有题先看一遍？

wzb罚时场被干碎了，下次不能这么莽的…