2020牛客暑期多校训练营（第六场）

比赛情况

题号	A	B	C	D	E	F	G	H	I	J	K
状态	-	O	O	-	O	-	Ø	O	-	O	O

O 在比赛中通过 Ø 赛后通过 ! 尝试了但是失败了 - 没有尝试

比赛时间

2020-07-27 12:00-17:00

题解

B - Binary Vector

求 n 个 n 维 01 向量线性无关的概率。

找到分子的通项 $2^{\frac {n(n-1)}2} \prod_{i=1}^n(2^i-1)$线性的 dp 一下求出来。

```
#pragma GCC optimize(2)
#pragma GCC optimize(3,"Ofast","inline")
#include<bits/stdc++.h>
#define ALL(x) (x).begin(),(x).end()
#define ll long long
#define db double
#define ull unsigned long long
#define pii_ pair<int,int>
#define mp_ make_pair
#define pb push_back
#define fi first
#define se second
#define rep(i,a,b) for(int i=(a);i<=(b);i++)
#define per(i,a,b) for(int i=(a);i>=(b);i--)
#define show1(a) cout<<#a<<" = "<<a<<endl
#define show2(a,b) cout<<#a<<" = "<<a<<"; "<<#b<<" = "<<b<<endl
using namespace std;
const ll INF = 1LL<<60;
const int inf = 1<<30;
const int maxn = 2e7+5;
const int M = 1e9+7;
inline void fastio() {ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);}
ll qpow(ll a,ll b) {ll
s=1;while(b){if(b&1)s=(s*a)%M;a=(a*a)%M;b>>=1;}return s;}

int ans[maxn],inv2[maxn],s[maxn],s2[maxn];
```

```
void init()
{
    int n = 2e7;

    inv2[1] = qpow(2,M-2);
    rep(i,2,n){
        inv2[i] = 1LL*inv2[i-1]*inv2[1]%M;
    }
    rep(i,2,n){
        inv2[i] = 2LL*inv2[i-1]%M*inv2[i]%M*inv2[i]%M;
    }


    ll pow2 = 1;
    s[0] = s2[0] = 1;
    rep(i,1,n) {
        s[i] = 1LL * s[i-1] * pow2 % M;
        pow2 = 2LL * pow2%M;

        s2[i] = 1LL * s2[i-1] * (pow2-1)%M;
    }


    rep(i,1,n){
        ans[i] = 1LL * s[i] * s2[i] % M * inv2[i] % M;
    }
    rep(i,2,n) ans[i] ^= ans[i-1];
}

int main()
{
    fastio();init();
    int _;
    for(cin>>_;_;_--){ int n;
        cin>>n;
        cout<<ans[n]<<endl;
    }
    return 0;
}
```

C - Combination of Physics and Maths

定义一个矩阵的压强是矩阵和除以最下面一行的和，现在要从一个矩阵中取一个子矩阵使得这个矩阵的压强最大。

首先我们可以意识到，如果取了$a_{i,j}$作为底中的一个元素，那么取$a_{1...i-1,j}$一定是更最优的，

我们考虑两个单列矩阵，如果一列的压强是a,另一列的压强是b且$a>b$,那么把后一列加入第一列一定会使压强变小，所以最优的应该是某个元素到顶的。预处理前缀和然后枚举底就行了。

```
#pragma GCC optimize(2)
#pragma GCC optimize(3,"Ofast","inline")
#include<bits/stdc++.h>
#define ALL(x) (x).begin(),(x).end()
#define ll long long
#define db double
#define ull unsigned long long
#define pii_ pair<int,int>
#define mp_ make_pair
#define pb push_back
#define fi first
#define se second
#define rep(i,a,b) for(int i=(a);i<=(b);i++)
#define per(i,a,b) for(int i=(a);i>=(b);i--)
#define show1(a) cout<<#a<<" = "<<a<<endl
#define show2(a,b) cout<<#a<<" = "<<a<<"; "<<#b<<" = "<<b<<endl
using namespace std;
const ll INF = 1LL<<60;
const int inf = 1<<30;
const int maxn = 2e7+5;
const int M = 1e9+7;
inline void fastio() {ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);}
ll qpow(ll a,ll b) {ll
s=1;while(b){if(b&1)s=(s*a)%M;a=(a*a)%M;b>>=1;}return s;}

int ans[maxn],inv2[maxn],s[maxn],s2[maxn];

void init()
{
    int n = 2e7;

    inv2[1] = qpow(2,M-2);
    rep(i,2,n){
        inv2[i] = 1LL*inv2[i-1]*inv2[1]%M;
    }
    rep(i,2,n){
        inv2[i] = 2LL*inv2[i-1]%M*inv2[i]%M*inv2[i]%M;
    }


    ll pow2 = 1;
    s[0] = s2[0] = 1;
    rep(i,1,n) {
        s[i] = 1LL * s[i-1] * pow2 % M;
        pow2 = 2LL * pow2%M;

        s2[i] = 1LL * s2[i-1] * (pow2-1)%M;
```

```
    }


    rep(i,1,n){
        ans[i] = 1LL * s[i] * s2[i] % M * inv2[i] % M;
    }
    rep(i,2,n) ans[i] ^= ans[i-1];
}

int main()
{
    fastio();init();
    int _;
    for(cin>>_;_;_--){ int n;
        cin>>n;
        cout<<ans[n]<<endl;
    }
    return 0;
}
```

E - Easy Construction

给一个数字 k要构造一个 $1-n$ 的排列，满足对于 $i \in [1,n]$ 都存在一个长度为 i 子串，使得这个子串内数字的和模 n 等于 k

首先考虑 $i=n$的情况，得出结论 n 为偶数时$k=\frac n2$才有解，否则 $k=0$ 才有解。然后再考虑如何构造，当 n 为偶数时，把 n 和 k 放最左边，然后依次从右边加入 $i,n-i$当 n 为奇数时，先在最左边放 n 然后依次放入$i,n-i$

```
#include<bits/stdc++.h>
#define ALL(x) (x).begin(),(x).end()
#define ll long long
#define db double
#define ull unsigned long long
#define pii_ pair<int,int>
#define mp_ make_pair
#define pb push_back
#define fi first
#define se second
#define rep(i,a,b) for(int i=(a);i<=(b);i++)
#define per(i,a,b) for(int i=(a);i>=(b);i--)
#define show1(a) cout<<#a<<" = "<<a<<endl
#define show2(a,b) cout<<#a<<" = "<<a<<"; "<<#b<<" = "<<b<<endl
using namespace std;
const ll INF = 1LL<<60;
const int inf = 1<<30;
```

```
const int maxn = 5e5+5;
inline void fastio() {ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);}


int main()
{
    fastio();
    int n,k;
    cin>>n>>k;
    if(n%2==0){
        if(k!=n/2) cout<<-1<<endl;
        else{
            cout<<n<<" "<<k<<" ";
            rep(i,1,k-1) cout<<i<<" "<<n-i<<" ";
            cout<<endl;
        }
    }else{
        if(k) cout<<-1<<endl;
        else{
            cout<<n;
            rep(i,1,n/2) cout<<" "<<i<<" "<<n-i;
            cout<<endl;
        }
    }
    return 0;
}
```

G - Grid Coloring

$n\times n$ 方格框架涂色， k 种颜色出现次数相等，构造一种方案使得每一行每一列都有至少两种颜色，且同色的边不构成环。

从上到下从左到右按 1 到 k 循环填就好了。

```
#include<bits/stdc++.h>
#define ll long long
#define pii pair<int,int>
#define mp make_pair
using namespace std;
const int N=205;
int r[N][N],c[N][N];
int main()
{
    int t;
    scanf("%d",&t);
```

```
    while(t--)
    {

        int n,k;
        scanf("%d%d",&n,&k);
        if(k==1||n==1||2*(n+1)*n%k){printf("-1\n");continue;}
        int res=0;
        for(int i=1;i<=n;i++)
        {
            for(int j=1;j<=n;j++)r[i][j]=res,res=(res+1)%k;
            for(int j=1;j<=n+1;j++)c[j][i]=res,res=(res+1)%k;
        }
        for(int i=1;i<=n;i++)r[n+1][i]=res,res=(res+1)%k;
        for(int i=1;i<=n+1;i++)
        {
            for(int j=1;j<=n;j++)printf("%d ",c[i][j]+1);
            puts("");
        }
        for(int i=1;i<=n+1;i++)
        {
            for(int j=1;j<=n;j++)printf("%d ",r[i][j]+1);
            puts("");
        }
    }
    return 0;
}
```

H - Harmony Pairs

数位dp□状态记录一下数位和差值□ $A\le B$ 的限制和 $B\le N$ 的限制即可。

```
#pragma GCC optimize(2)
#pragma GCC optimize(3,"Ofast","inline")
#include<bits/stdc++.h>
#define ll long long
using namespace std;
const int N=105;
const ll mod=1e9+7;
char s[N];
ll dp[N][2005][2][2];
ll dfs(int dep,int d,bool lim1,bool lim2)
{
    if(dep>=strlen(s))return (d>0);
    if(dp[dep][d+900][lim1][lim2]!=-1) return dp[dep][d+900][lim1][lim2];
    dp[dep][d+900][lim1][lim2] = 0;
    for(int i=0;i<10;i++)
```

```
    for(int j=0;j<10;j++)
    {
        if(lim2&&j>s[dep]-'0')continue;
        if(lim1&&i>j)continue;
        dp[dep][d+900][lim1][lim2]+=dfs(dep+1,d+i-
j,lim1&&(i==j),lim2&&(j==s[dep]-'0'));
        dp[dep][d+900][lim1][lim2]%=mod;
    }
    return dp[dep][d+900][lim1][lim2];
}
int main()
{
    memset(dp,-1,sizeof(dp));
    scanf("%s",s);
    printf("%lld\n",dfs(0,0,1,1));
    return 0;
}
```

J - Josephus Transform

对排列 $1,2,3,\cdots,n$ 做 m 次操作。每次操作是进行 x 次 k 约瑟夫问题。求最后的排列。

把操作看成置换，如果已知一次 k 约瑟夫问题的置换，那么可以用类似快速幂的方法在 $nmlogx$ 得到答案。考虑如何求一次 k 约瑟夫问题的置换，用线段树暴力模拟就行，复杂度为 $nmlogn$所以总复杂度$nm(logn + logx)$

```
#include<bits/stdc++.h>
#define ALL(x) (x).begin(),(x).end()
#define ll long long
#define db double
#define ull unsigned long long
#define pii_ pair<int,int>
#define mp_ make_pair
#define pb push_back
#define fi first
#define se second
#define rep(i,a,b) for(int i=(a);i<=(b);i++)
#define per(i,a,b) for(int i=(a);i>=(b);i--)
#define show1(a) cout<<#a<<" = "<<a<<endl
#define show2(a,b) cout<<#a<<" = "<<a<<"; "<<#b<<" = "<<b<<endl
using namespace std;
const ll INF = 1LL<<60;
const int inf = 1<<30;
const int maxn = 1e5+5;
inline void fastio() {ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);}

int a[maxn],n,b[maxn],tmp[maxn];
```

```
int tr[maxn<<2];
void build(int id,int l,int r)
{
    tr[id] = 0;
    if(l==r) {tr[id]=1;return ;}
    int mid = (l+r)>>1;
    build(id<<1,l,mid);build(id<<1|1,mid+1,r);
    tr[id] = tr[id<<1] + tr[id<<1|1];
}
void update(int id,int stl,int str,int pos)
{
    if(stl==str){
        tr[id] = 0;
        return ;
    }
    int mid = (stl+str)>>1;
    if(pos<=mid) update(id<<1,stl,mid,pos);
    else update(id<<1|1,mid+1,str,pos);
    tr[id] = tr[id<<1] + tr[id<<1|1];
}
int query(int id,int stl,int str,int l,int r)
{
    if(stl==l && str==r){
        return tr[id];
    }
    int mid = (stl+str)>>1;
    if(r<=mid) return query(id<<1,stl,mid,l,r);
    else if(l>mid) return query(id<<1|1,mid+1,str,l,r);
    else return query(id<<1,stl,mid,l,mid) +
query(id<<1|1,mid+1,str,mid+1,r);
}
int query2(int id,int l,int r,int k)
{
    if(l==r) return l;
    int mid = (l+r)>>1;
    if(tr[id<<1]>=k) return query2(id<<1,l,mid,k);
    else return query2(id<<1|1,mid+1,r,k-tr[id<<1]);
}
void perm(int *a,int *b,int x)
{
    while(x){
        if(x&1){
            rep(i,1,n) tmp[i] = a[b[i]];
            rep(i,1,n) a[i] = tmp[i];
        }
        rep(i,1,n) tmp[i] = b[b[i]];
        rep(i,1,n) b[i] = tmp[i];
        x>>=1;
    }
}
```

```
int main()
{
    fastio(); int m,k,x;
    cin>>n>>m;
    rep(i,1,n) a[i] = i;
    while(m--){
        cin>>k>>x;
        build(1,1,n);
        int last = 0;
        rep(i,1,n){
            int r = last==n?0:query(1,1,n,last+1,n);
            if(r>=k){
                int pre = last?query(1,1,n,1,last):0;
                b[i] = query2(1,1,n,k+pre);
                update(1,1,n,b[i]);
                last = b[i];
            }else{
                int tt = k - r;
                tt %= (n-i+1);
                if(tt == 0) tt = n-i+1;
                b[i] = query2(1,1,n,tt);
                update(1,1,n,b[i]);
                last = b[i];
            }
        }
        perm(a,b,x);
    }
    rep(i,1,n) cout<<a[i]<<" \n"[i==n];
    return 0;
}
```

K - K-Bag

问给出的序列是否是若干个$1-k$序列组成序列的子序列（好绕）

首先特判序列里有大于k的情况。

其余的上权值线段树，对于前k个位置，只要没有次数大于1就行，后面的dp转移，权值线段树查长度为k的区间是否次数都为1，如果是就转移，然后最后k个位置和前k个位置类似处理。

```
#include <bits/stdc++.h>
#define il inline
using namespace std;
typedef long long ll;
const int N=5e5+5;
const int inf = 1e9+5;
int mx[N<<2],a[N],id[N],cnt;
```

```
bool f[N];
void build(int p,int l,int r) {
    mx[p] = 0;
    if (l==r) return ;
    int mid = (l+r)>>1;
    build(p<<1,l,mid);
    build(p<<1|1,mid+1,r);
}
void Update(int p,int l,int r,int a,int b) {
    if (l==r) {
        mx[p]+=b;
        return ;
    }
    int mid = (l+r)>>1;
    if (a<=mid)Update(p<<1,l,mid,a,b);
    else Update(p<<1|1,mid+1,r,a,b);
    mx[p] = max(mx[p<<1],mx[p<<1|1]);
}
int Getans(int p,int l,int r,int a,int b) {
    if (l>=a&&r<=b)
        return mx[p];
    int mid = (l+r)>>1;
    int ans = 0;
    if (a<=mid)ans = max(ans,Getans(p<<1,l,mid,a,b));
    if (b >mid)ans = max(ans,Getans(p<<1|1,mid+1,r,a,b));
    return ans;
}
int getid(int x) {
    int l = 1,r = cnt+1;
    while (l<r) {
        int mid = (l+r)>>1;
        if (id[mid]<x)l = mid+1;
        else r = mid;
    }
    return l;
}
int main()
{
    int cas;
    scanf("%d",&cas);
    while (cas--) {
        int n,k;
        scanf("%d%d",&n,&k);
        bool flag = false;
        for (int i = 1;i<= n;i++) {
            scanf("%d",&a[i]);
            if (a[i] > k || a[i] <= 0) flag = true;
            id[i] = a[i];
        }
        if (flag) {printf("NO\n");continue;}
```

```
        cnt = 0;
        id[0] = -1;
        sort(id+1,id+n+1);
        for (int i = 1;i<= n;i++)
            if (id[cnt]!=id[i])
                id[++cnt] = id[i];
        build(1,1,cnt);
        for (int i = 1;i<= n;i++)f[i] = false;
        f[0] = true;
        for (int i = 1;i<= n && i <= k;i++) {
            Update(1,1,cnt,getid(a[i]),1);
            if (Getans(1,1,cnt,1,cnt)==1)
                f[i] = true;
        }
        for (int i = k+1;i<= n;i++) {
            int tp = i-k;
            Update(1,1,cnt,getid(a[i]),1);
            Update(1,1,cnt,getid(a[tp]),-1);
            if (Getans(1,1,cnt,1,cnt)==1)
                f[i] |= f[i-k];
        }
        int bg = max(1,n-k+1);
        bool ans = f[n];
        for (int i = bg;i<= n && !ans;i++)
        {
            if (f[i-1] && Getans(1,1,cnt,1,cnt)==1)
                ans = true;
            Update(1,1,cnt,getid(a[i]),-1);
        }
        if (ans) printf("YES\n");
        else printf("NO\n");
    }
    return 0;
}
```