模板

Ether Strike

2022 年 10 月 31 日

目录

1 数学

1.1 线性代数

1.1.1 矩阵乘法

```
using Matrix = array<array<int,80>,80>;
using Vector = array<int,80>;
Vector operator *(Vector a, Matrix b){
        Vector res={0};
        for(int i=0; i<m; i++)
                for(int j=0; j<m; j++)
                        inc(res[j], 1ll*a[i]*b[i][j]%mod);
        return res;
}
Vector operator *(Matrix b, Vector a){
        Vector res={0};
        for(int i=0; i<80; i++)
                for(int j=0; j<80; j++)
                        inc(res[i], 1ll*a[j]*b[i][j]%mod);
        return res;
}
Matrix operator *(Matrix a, Matrix b){
        Matrix res={0};
        for(int i=0; i<80; i++)
                for(int j=0; j<80; j++)
                        for(int k=0; k<80; k++)
                                inc(res[i][j], 1ll*a[i][k]*b[k][j]%mod);
        return res;
}
Matrix power(Matrix a, ll b){
        Matrix res={0};
        for(int i=0; i<80; i++)
                res[i][i]=1;
        for(;b;b>>=1,a=a*a)
                if(b&1)
                        res=res*a;
        return res;
}
inline void solve(){
        int n;
        cin >> n;
        Vector now={0};
        now[0] = 1;
        a = power(a, n);
```

```
        now = a * now;
}
```

1.1.2 幺半群–证明结合律的工具

满足结合律就可以使用矩阵加速 (如 max,+)
定义：零元 (O)，加法 ($+$)，乘法 ($\times$)
矩阵乘法的形式: $c_{i,j} = \sum a_{i,k} \times b_{k,j}$
如果一个运算满足结合律，当且仅当它满足

1. $O \times A = O$
2. $(a+b)+c = a+(b+c)$
3. $(a \times b) \times c = a \times (b \times c)$
4. $(a+b) \times c = a \times c + b \times c$
5. $a \times (b+c) = a \times b + a \times c$

example: $c_{i,j} = min\{max\{a_{i,k}, b_{k,j}\}\}$ 此处 min 为加法，max 为乘法，零元为 ∞

1.1.3 线性基

```
struct Linear{
//cnt != n说明可以构成0
    ll p[64];//根据数字的范围来开
    int cnt, n;
    Linear(){cnt = n = 0; for(int i=0; i<64; i++) p[i] = 0;}
    inline void clear(){cnt = n = 0; for(int i=0; i<64; i++) p[i] = 0;}
    inline void copy(Linear b){
        for(int i=0; i<64; i++)
                p[i] = b.p[i];
    }
    inline bool insert(ll x){
        ++n;
        for(int i=63; i>=0; i--)
            if(x&(1ll<<i)){
                if(!p[i]){p[i]=x; cnt++; return 1;}
                x^=p[i];
            }
        return 0;
    }
    inline ll query(ll x){
        if(cnt != n) --x;
        if(!x) return 0;
        ll res = 0;
        for(int i=0; i<64; i++)
```

```
            if(p[i]){
                if(x&1) res^=p[i];
                x>>=1;
            }
        if(x) return -1;
        return res;
    }
    inline ll mx(){//线性基最大值
        ll res = 0;
        for(int i=63; ~i; i--)
                if(p[i] && (res^p[i]) > res) res^=p[i];
        return res;
    }
};
inline Linear merge(Linear a, Linear b){
    for(int i=0; i<=63; i++)
        a.insert(b.p[i]);
    return a;
}
```

1.1.4 矩阵树定理

$O(N^3)$ 求以每个点为根的外向树

```
#include <bits/stdc++.h>
using namespace std;
#define fo(i,j,k) for(int i=(j),end_i=(k);i<=end_i;i++)
#define ff(i,j,k) for(int i=(j),end_i=(k);i< end_i;i++)
#define fd(i,j,k) for(int i=(j),end_i=(k);i>=end_i;i--)
#define all(x) (x).begin(),(x).end()
#define cle(x) memset(x,0,sizeof(x))
#define lowbit(x) ((x)&-(x))
#define VI vector<int>
#define ll long long
const ll mod=1e9+7;
inline ll Add(ll x,ll y){x+=y; return (x<mod)?x:x-mod;}
inline ll Dec(ll x,ll y){x-=y; return (x<0)?x+mod:x;}
inline ll Mul(ll x,ll y){return x*y%mod;}
inline ll Pow(ll x,ll y)
{
    y%=(mod-1);ll ans=1;for(;y;y>>=1,x=x*x%mod)if(y&1) ans=ans*x%mod;
    return ans;
}
const int N=505;
int n;
```

```
struct matrix{
    ll a[N][N];
    matrix(){fo(i,0,n) fo(j,0,n) a[i][j]=0;}
    void clear(int n)
    {
        fo(i,0,n) fo(j,0,n) a[i][j]=0;
    }
    inline ll* operator [](int x){return a[x];}
};
inline ll det(matrix a,int n)
{
    ll d=1,inv,tmp;
    int flag=1;
    fo(i,1,n)
    {
        int k=i;
        fo(j,i+1,n) if(a[j][i]) {k=j; break;}
        if(k!=i) {fo(j,i,n) swap(a[k][j],a[i][j]); flag=-flag;}
        if(!a[i][i]) return 0;
        inv=Pow(a[i][i],mod-2);
        fo(j,i+1,n)
        {
            tmp=a[j][i]*inv%mod;
            fo(k,i,n) a[j][k]=Dec(a[j][k],a[i][k]*tmp%mod);
        }
        d=d*a[i][i]%mod;
    }
    if(flag==-1) d=(mod-d%mod)%mod;
    return d;
}
int r(matrix a)
{
    int d=0;
    ll inv,tmp;
    fo(i,1,n)
    {
        int k=i;
        fo(j,i+1,n) if(a[j][i]) {k=j; break;}
        if(k!=i) fo(j,i,n) swap(a[i][j],a[k][j]);
        if(!a[i][i]) continue;
        d++;
        inv=Pow(a[i][i],mod-2);
        fo(j,i+1,n)
        {
```

```
            tmp=a[j][i]*inv%mod;
            fo(k,i,n) a[j][k]=Dec(a[j][k],a[i][k]*tmp%mod);
        }
    }
    return d;
}
ll l[N],p[N];
inline void solve(matrix a,ll *p)
{
    static ll c[N][N],d[N][N];
    fo(i,0,n) fo(j,0,n) c[i][j]=d[i][j]=0;
    auto check = [&](const int &id,ll *g){
        fo(i,1,n) p[i]=0;
        p[id]=1;
        ll tmp;
        fd(i,n,1)
            if(l[i])
            {
                if(!c[i][i])
                {
                    fd(j,i,1) c[i][j]=l[j];
                    fo(j,1,n) d[i][j]=p[j];
                    return 0;
                }
                tmp=Pow(c[i][i],mod-2)*l[i]%mod;
                fd(j,i,1) l[j]=Dec(l[j],c[i][j]*tmp%mod);
                fo(j,1,n) p[j]=Dec(p[j],d[i][j]*tmp%mod);
            }
        fo(i,1,n) g[i]=p[i];
        return 1;
    };
    fo(i,1,n)
    {
        fo(j,1,n)
            l[j]=a[j][i];
        if(check(i,p)) return;
    }
}
void solve(matrix A)
{
    int rank=r(A);
    assert(rank!=n);
    {
        matrix B,C;
```

```
        static ll x[N],y[N];
        C.clear(n);
        fo(i,1,n) fo(j,1,n) C[j][i]=A[i][j];
        solve(A,y);
        solve(C,x);
        int c=0,r=0;
        fo(i,1,n) if(y[i]) {c=i; break;}
        fo(i,1,n) if(x[i]) {r=i; break;}
        B.clear(n-1);
        fo(i,1,n)
            if(i!=r)
                fo(j,1,n)
                    if(j!=c)
                        B[i-(i>r)][j-(j>c)]=A[i][j];
        ll tmp=det(B,n-1);
        tmp=((r+c)%2==1)?(mod-tmp)%mod:tmp;
        tmp=Pow(y[c]*x[r]%mod,mod-2)*tmp%mod;
        fo(i,1,n)
            printf("%lld%c",tmp*x[i]%mod*y[i]%mod,(i==n)?'\n':' ');
        return;
    }
}
void solve()
{
    int m,x,y;
    scanf("%d%d",&n,&m);
    if(n==1) {printf("1\n"); return;}
    matrix A;
    fo(i,1,m)
    {
        scanf("%d%d",&x,&y);
        A[y][y]++;
        A[x][y]--;
    }
    fo(i,1,n) fo(j,1,n) if(A[i][j]<0) A[i][j]+=mod;
    solve(A);
}
int main()
{
    int T; scanf("%d",&T);
    while(T--) solve();
    return 0;
}
```

1.2 gcd 相关

```
//luogu P5656 二元一次不定方程
ll exgcd(int a, int b, ll &x, ll &y){
	if(b == 0){
		x = 1; y = 0;
		return a;
	}
	ll tx, ty;
	int d = exgcd(b, a % b, tx, ty);
	x = ty; y = tx - a / b * ty;
	return d;
}

inline ll calc(int a, int b){//excrt
	//t%N = a t % M = b
	//Nx+a=My+b=t
	int N = n<<1, M = m<<1;
	ll x, y;
	int d = exgcd(N, M, x, y);
	int c = b - a;
	if(c % d != 0) return 2e18;
	ll lcm = 1ll * N * M / d;
	x = 1ll * x * (b-a) / d % lcm;
	ll res = 1ll * x % lcm * N % lcm + a;
	res = (res + lcm) % lcm;
	if(res == 0) res = lcm;
	return res;
}

inline void solve(){
	int a, b, c;
	cin >> a >> b >> c;
	ll x, y;
	int d = exgcd(a, b, x, y);
	if(c % d)
		cout << -1 << ln;
	else {
		ll rx = x * c / d, ry = y * c / d;
		// cout << rx << " " << ry << ln;
		if(rx <= 0){

			ll tmp = abs(rx) / (b / d) + 1;
			rx += tmp * b / d;
			ry -= tmp * a / d;
```

```
            if(ry <= 0) {
                cout << rx << " ";

                tmp = abs(ry) / (a / d) + 1;
                ry += tmp * a / d;
                rx -= tmp * b / d;

                cout << ry << ln;
                return;
            }
        } else if(ry <= 0){
            ll tmp = abs(ry) / (a / d) + 1;
            ry += tmp * a / d;
            rx -= tmp * b / d;
            ll tmpy = ry;
            if(rx <= 0){

                tmp = abs(rx) / (b / d) + 1;
                rx += tmp * b / d;
                ry -= tmp * a / d;
                cout << rx << " " << tmpy << ln;

                return;
            }
        }
        // cout << rx << " " << ry << ln;

        ll L = -rx * d / b, R = ry * d / a;
        if(rx + L * b / d == 0) L++;
        if(ry - R * a / d == 0) R--;
        cout << R - L + 1 << " " << rx + L * b / d << " " << ry - R * a / d
        //rx <= ry

    }
}
```

1.3 数相关

1.3.1 生成函数

1. $e^{dx}[\frac{x^n}{n!}] = \frac{d^n}{n!}$
2. $(e^x - 1)^d[\frac{x^n}{n!}] = \left\{ {n \atop d} \right\} d!$
3. $\frac{1}{(x-1)^d}[x^n] = \binom{n+d-1}{d-1}$

1.3.2 斯特林数

第一类斯特林数: 将 p 个不同的物品分成 k 个非空循环排列的方法数

$S(p,0)=0,p\geq 1$

$S(p,p)=1,p\geq 0$

$S(p,k)=(p-1)S(p-1,k)+S(p-1,k-1),p-1\geq k\geq 1$

生成函数 (可以用分治 fft 加速): $F(x)=\prod_{i=0}^{n-1}(x+i)$

第二类斯特林数

$S_2(p,0)=0,p\geq 1$

$S_2(p,p)=1,p\geq 0$

$S_2(p,k)=kS_2(p-1,k)+S_2(p-1,k-1),p-1\geq k\geq 1$

恒等式:$n^m=\sum_{k=0}^{m}S(m,k)\cdot\binom{n}{k}k!$

生成函数 (可以从恒等式二项式反演过来, 可以用 fft 加速): $S(n,k)=\sum_{i=0}^{k}(-1)^i\binom{k}{i}(k-i)^n$

下降幂公式: $x^n=\sum_{i=0}^{n}\left\{{n \atop i}\right\}x^{\underline{i}}$

k 个球	m 个盒子	是否允许有空盒子	方案数
各不相同	各不相同	是	m^k
各不相同	各不相同	否	$m!S_2(k,m)$
各不相同	完全相同	是	$\sum_{i=1}^{m}S_2(k,i)$
各不相同	完全相同	否	$S_2(k,m)$
完全相同	各不相同	是	$C\ (m+k-1,k)$
完全相同	各不相同	否	$C(k-1,m-1)$
完全相同	完全相同	是	$\frac{1}{(1-x)(1-x^2)\ldots(1-x^m)}[x^k]$
完全相同	完全相同	否	$\frac{x^m}{(1-x)(1-x^2)\ldots(1-x^m)}[x^k]$

1.3.3 卡特兰数

1. 括号匹配: 你有 n 个左括号，n 个右括号，问有多少个长度为 2n 的括号序列使得所有的括号都是合法的

2. 进出栈问题有一个栈，我们有 2n 次操作，n 次进栈，n 次出栈，问有多少中合法的进出栈序列

3. 312 排列一个长度为 n 的排列 $\{a\}$，只要满足 $i<j<k$ 且 $a_j<a_k<a_i$ 就称这个排列为 312 排列, 求 n 的全排列中不是 312 排列的排列个数

4. 不相交弦问题在一个圆周上分布着 2n 个点，两两配对，并在这两个点之间连一条弦，要求所得的 2n 条弦彼此不相交的配对方案数

5. 从 (1, 1) 走到 (n, n) 且不穿过对角线有多少条路径

```
const int N=2e6+5;
const int mod = 20100403;
int n, m, fac[N], inv[N], cal[N];
inline int power(int a, int b){
    int res=1;
    while(b){
        if(b&1)
            res=1ll*res*a%mod;
        b>>=1;
```

```
        a=1ll*a*a%mod;
    }
    return res;
}
inline int C(int n, int m){
    return 1ll*fac[n]*inv[m]%mod*inv[n-m]%mod;
}
inline void pre(int n){
    fac[0]=1;
    for(int i=1; i<=n; i++)
        fac[i]=1ll*fac[i-1]*i%mod;
    inv[n]=power(fac[n], mod-2);
    for(int i=n-1; i>=0; i--)
        inv[i]=1ll*inv[i+1]*(i+1)%mod;
}
inline void Caterlan(int n){
    return (C(2*n, n)-C(2*n, n-1)+mod)%mod
}
```

1.3.4 组合数

- 链上不相邻地取 k 个数 $C(n-r+1,r)$
- 大小为 n 的环上不相邻的选 r 个点 $C(n-r-1,r-1)+C(n-r,r)=C(n-r,r)\frac{n}{n-r}$

1.4 博弈论

1.4.1 sg 函数

sg(x)=mex{sg(y)} y 为 x 的可达状态点
游戏和的 SG 函数等于各个游戏 SG 函数的异或和
一般是把 sg 函数打印出来找规律

1.4.2 Nash 均衡

对于一个决策，有 p 的概率选择，$(1-p)$ 的概率不选择，设选择的代价为 x，不选择的代价为 y，可以得到 $z=px+(1-p)y$，是一个喜闻乐见的线性规划式子

- nim – 取最后一个的胜利: 先手必胜要求 sg 函数不为 0
- anti-nim – 取最后一个的失败: 先手必胜要求 sg 函数不为 0 且存在一个 $sg(x)>1$ 或所有 $sg(x)=1$ 且有偶数个游戏

1.5 多项式

1.5.1 FFT

精度可能不大够，预处理单位根会好很多

```cpp
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef unsigned long long ull;
typedef pair<int,int> pii;
#define pb push_back
#define ln '\n'

//FFT多项式乘法
//n和m表示两个多项式的次数
//a[i].x和b[i].x表示第i项
//结果保存在a[i].x中， 注意是浮点数， 需要取整输出
const int N = 1<<20;
// #define double long double
const double pi = acos(-1);
typedef vector<ll> poly;
//根据精度判断double还是long double
//long double可能会导致速度变慢
//#define double long double
struct com{//复数
	com(double xx=0, double yy=0){x=xx; y=yy;}
	double x, y;
	com operator +(com const &b) const{
		return com(x+b.x, y+b.y);
	}
	com operator -(com const &b) const{
		return com(x-b.x, y-b.y);
	}
	com operator *(com const &b) const {
		return com(x*b.x-y*b.y, x*b.y+b.x*y);
	}
	com conj(){return com(x, -y);}
}A[N<<1], B[N<<1], rev[N<<1];

const int L = N<<1;
inline void fft_init(){
	for(int i=0; i<N<<1; i++)
		rev[i] = com(cos(2*i*pi/L), sin(2*i*pi/L));
}

void fft(com *f, int n, int flag){
	for(int i=(n>>1), j=1; j<n; j++){
		if(i<j) swap(f[i], f[j]);
		int k = (n>>1);
```

```
                while(i&k){i^=k; k>>=1;}
                i^=k;//蝴蝶变换
        }
        for(int k=2; k<=n; k<<=1){
                // com rt(cos(2*pi/k), sin(2*pi/k)*flag);//单位根
                //dft && idft
                int rt = L/k;
                for(int i=0; i<n; i+=k){
                        int w = 0;
                        for(int j=i; j<(i+(k>>1)); j++){
                                com u = f[j], v = f[j+(k>>1)] * (flag == 1?rev[w]:r
                                f[j] = u+v;
                                f[j+(k>>1)] = u-v;
                                w = w + rt;
                        }
                }
        }
        if(flag == -1){//idft
                for(int i=0; i<n; i++)
                        f[i].x=f[i].x/n;
        }
}

poly operator*(poly a, poly b){
        int n = a.size() - 1, m = b.size() - 1;
        for(m+=n, n=1; n<=m; n<<=1);
        a.resize(n); b.resize(n);
        for(int i=0; i<n; i++)
                A[i] = com(a[i], 0),
                B[i] = com(b[i], 0);
        fft(A, n, 1); fft(B, n, 1);
        for(int i=0; i<n; i++)
                A[i] = A[i] * B[i];
        fft(A, n, -1);
        poly c(m+1);
        for(int i=0; i<=m; i++){
                if(A[i].x > 0)
                        c[i] = (ll)(A[i].x + 0.5);
                else
                        c[i] = (ll)(A[i].x - 0.5);
        }
        return c;
}

```

```
int main(){
        // freopen("0.in","r",stdin);
    ios::sync_with_stdio(false);
    cin.tie(0);
    int n, m;
        cin >> n >> m;
        fft_init();
        poly a(n+1), b(m+1);
        for(int i=0; i<=n; i++)
                cin >> a[i];
        for(int i=0; i<=m; i++)
                cin >> b[i];
        a = a * b;
        for(int i=0; i<=n+m; i++)
                cout << a[i] << " ";
}
```

1.5.2 单位根反演

1. $[n|a] = \frac{1}{n}\sum_{k=0}^{n}\omega_n^{ak}$
2. $[a = b(mod\ n)] = [a - b = 0(mod\ n)] = [n|(a-b)] = \frac{1}{n}\sum_{i=0}^{n} -1\omega_n^{(a-b)k} = \frac{1}{n}\sum_{i=0}^{n-1}\omega_n^{ak}\omega_n^{-bk}$

1.5.3 NTT

跟 FFT 同理，只是用原根 $G^{\frac{(P-1)}{n}}$ 来代替 w_n^1

```
//#pragma GCC optimize(2)
//#pragma comment(linker, "/STACK:102400000,102400000")
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef unsigned long long ull;
typedef pair<int,int> pii;
#define pb push_back
#define ln '\n'

const int g=3;
const int mod = 998244353;

inline void inc(int &a, int b){
    a+=b;
    if(a>=mod) a-=mod;
    if(a<0) a+=mod;
}
inline void dec(int &a, int b){
```

```
    a-=b;
    if(a<0) a+=mod;
    if(a>=mod) a-=mod;
}

inline int power(int a, int b){
    int res = 1;
    for(; b; b>>=1, a=1ll*a*a%mod)
        if(b&1)
            res=1ll*res*a%mod;
    return res;
}

typedef vector<int> poly;
void ntt(int *a, int n, int flag){
    for(int i=(n>>1), j=1; j<n; j++){
        if(i<j) swap(a[i], a[j]);
        int k = (n>>1);
        while(i&k){i^=k; k>>=1;}
        i^=k;
    }
    for(int k=2; k<=n; k<<=1){
        int rt = power(g, (mod-1)/k);
        if(flag == -1)
            rt = power(rt, mod-2);
        for(int i=0; i<n; i+=k){
            int del = 1;
            for(int j=i; j<i+k/2; j++){
                int u = a[j], v = 1ll * del * a[j+k/2] % mod;
                a[j] = (u + v) % mod;
                a[j+k/2] = (u - v + mod) % mod;
                del = 1ll * del * rt % mod;
            }
        }
    }
    if(flag == -1){
        int inv = power(n, mod-2);
        for(int i=0; i<n; i++)
            a[i] = 1ll * a[i] * inv % mod;
    }
}

poly operator*(poly a, poly b){
    int m = b.size(), n = a.size();
```

```cpp
    for(m+=n, n=1; n<=m; n<<=1);
    a.resize(n); b.resize(n);
    ntt(a.data(), n, 1); ntt(b.data(), n, 1);
    for(int i=0; i<n; i++)
        a[i] = 1ll * a[i] * b[i] % mod;
    ntt(a.data(), n, -1);
    a.resize(m-1);
    return a;
}

poly operator*(poly a, int b){
    int m = a.size();
    for(int i=0; i<m; i++)
        a[i] = 1ll * a[i] * b % mod;
    return a;
}

poly operator*(int a, poly b){
    int m = b.size();
    for(int i=0; i<m; i++)
        b[i] = 1ll * b[i] * a % mod;
    return b;
}

poly operator+(poly a, poly b){
    int m = b.size(), n = a.size();
    a.resize(max(n, m));
    for(int i=0; i<b.size(); i++)
        inc(a[i], b[i]);
    return a;
}

poly operator-(poly a, poly b){
    int m = b.size(), n = a.size();
    a.resize(max(n, m));
    for(int i=0; i<b.size(); i++)
        dec(a[i], b[i]);
    return a;
}

void modxk(poly &a, int k){
    if(a.size() > k) a.resize(k);
}

```

```cpp
poly Inv(poly &f){
    poly R{power(f[0], mod-2)};
    for(int i = 1; i<f.size(); i<<=1)
        R = 2*R-R*R*f, modxk(R, i<<1);
    return R;
}

int main(){
    ios::sync_with_stdio(false);
    cin.tie(0);
    int n;
    cin >> n;
    poly f(n);
    for(int i=0; i<n; i++)
        cin >> f[i];
    f = Inv(f);
    // f = inv(f);
    for(int i=0; i<n; i++)
        cout << f[i] << " ";
    cout << ln;
}
```

```cpp
namespace polybase {
    constexpr int mod(998244353), G(3), L(1 << 20);
    //L(20): 长度， 根据题目需要改
    //g(3): 原根， 根据模数改
    //mod(998244353): ntt经典模数

    inline void inc(int &x, int y)
    {
        x += y;
        if (x >= mod) x -= mod;
        if (x < 0) x += mod;
    }

    inline void dec(int &x, int y)
    {
        x -= y;
        if (x < 0) x += mod;
        if (x >= mod) x -= mod;
    }

    int fpow(int x, int k = mod - 2)
    {
        int r = 1;
```

```
        for (; k; k >>= 1, x = 1LL * x * x % mod)
        {
            if (k & 1) r = 1LL * r * x % mod;
        }
        return r;
    }

    int w[L], _ = []
    {
        w[L / 2] = 1;
        for (int i = L / 2 + 1,
            x = fpow(G, (mod - 1) / L); i < L; i++)
            w[i] = 1LL * w[i - 1] * x % mod;
        for (int i = L / 2 - 1; i >= 0; i--)
            w[i] = w[i << 1];
        return 0;
    }();

    void dft(int *a, int n)
    {
        assert((n & n - 1) == 0);
        for (int k = n >> 1; k; k >>= 1)
        {
            for (int i = 0; i < n; i += k << 1)
            {
                for (int j = 0; j < k; j++)
                {
                    int &x = a[i + j], y = a[i + j + k];
                    a[i + j + k] = 1LL *
                        (x - y + mod) * w[k + j] % mod;
                    inc(x, y);
                }
            }
        }
    }

    void idft(int *a, int n)
    {
        assert((n & n - 1) == 0);
        for (int k = 1; k < n; k <<= 1)
        {
            for (int i = 0; i < n; i += k << 1)
            {
                for (int j = 0; j < k; j++)
```

```
                {
                    int x = a[i + j], y = 1LL *
                        a[i + j + k] * w[k + j] % mod;
                    a[i + j + k] =
                        x - y < 0 ? x - y + mod : x - y;
                    inc(a[i + j], y);
                }
            }
        }
        for (int i = 0, inv =
            mod - (mod - 1) / n; i < n; i++)
            a[i] = 1LL * a[i] * inv % mod;
        std::reverse(a + 1, a + n);
    }

    inline int norm(int n) { return 1 << std::__lg(n * 2 - 1); }

    struct poly : public std::vector<int>
    {
        #define T (*this)
        using std::vector<int>::vector;

        void append(const poly &r)
        {
            insert(end(), r.begin(), r.end());
        }

        int len() const { return size(); }

        poly operator-() const
        {
            poly r(T);
            for (auto &x : r) x = x ? mod - x : 0;
            return r;
        }

        poly &operator+=(const poly &r)
        {
            if (r.len() > len()) resize(r.len());
            for (int i = 0; i < r.len(); i++) inc(T[i], r[i]);
            return T;
        }

        poly &operator-=(const poly &r)
```

```
    {
        if (r.len() > len()) resize(r.len());
        for (int i = 0; i < r.len(); i++) dec(T[i], r[i]);
        return T;
    }

    poly &operator^=(const poly &r)
    {
        if (r.len() < len()) resize(r.len());
        for (int i = 0; i < len(); i++)
            T[i] = 1LL * T[i] * r[i] % mod;
        return T;
    }

    poly &operator*=(int r)
    {
        for (int &x : T) x = 1LL * x * r % mod;
        return T;
    }

    poly operator+(const poly &r) const { return poly(T) += r; }

    poly operator-(const poly &r) const { return poly(T) -= r; }

    poly operator^(const poly &r) const { return poly(T) ^= r; }

    poly operator*(int r) const { return poly(T) *= r; }

    poly &operator<<=(int k) { return insert(begin(), k, 0), T; }

    poly operator<<(int r) const { return poly(T) <<= r; }

    poly operator>>(int r) const {
        return r >= len() ? poly() : poly(begin() + r, end()); }

    poly &operator>>=(int r) { return T = T >> r; }

    poly pre(int k) const {
        return k < len() ? poly(begin(), begin() + k) : T; }

    friend void dft(poly &a) { dft(a.data(), a.len()); }

    friend void idft(poly &a) { idft(a.data(), a.len()); }

```

```
        friend poly conv(const poly &a, const poly &b, int n)
        {
            poly p(a), q;
            p.resize(n), dft(p);
            p ^= &a == &b ? p : (q = b, q.resize(n), dft(q), q);
            idft(p);
            return p;
        }

        friend poly operator*(const poly &a, const poly &b)
        {
            int len = a.len() + b.len() - 1;
            if (a.len() <= 16 || b.len() <= 16)
            {
                poly c(len);
                for (int i = 0; i < a.len(); i++)
                    for (int j = 0; j < b.len(); j++)
                        c[i + j] = (c[i + j]
                            + 1LL * a[i] * b[j] % mod) % mod;
                return c;
            }
            return conv(a, b, norm(len)).pre(len);
        }

        poly deriv() const
        {
            if (empty()) return poly();
            poly r(len() - 1);
            for (int i = 1; i < len(); i++)
                r[i - 1] = 1LL * i * T[i] % mod;
            return r;
        }

        poly integ() const
        {
            if (empty()) return poly();
            poly r(len() + 1);
            for (int i = 0; i < len(); i++)
                r[i + 1] = 1LL * fpow(i + 1) * T[i] % mod;
            return r;
        }

        poly inv(int m) const
        {
```

```cpp
            poly x = {fpow(T[0])};
            for (int k = 1; k < m; k *= 2)
            {
                x.append(-((conv(pre(k * 2),
                    x, k * 2) >> k) * x).pre(k));
            }
            return x.pre(m);
        }

        poly log(int m) const { return (deriv()
            * inv(m)).integ().pre(m); }

        poly exp(int m) const
        {
            poly x = {1};
            for (int k = 1; k < m; k *= 2)
            {
                x.append((x * (pre(k * 2) -
                    x.log(k * 2) >> k)).pre(k));
            }
            return x.pre(m);
        }

        poly sqrt(int m) const
        {
            poly x = {1}, y = {1};
            for (int k = 1; k < m; k *= 2)
            {
                x.append(((pre(k * 2)
                    - x * x >> k) * y).pre(k) * (mod + 1 >> 1));
                if (k * 2 < m)
                {
                    y.append(-((conv(x.pre(k * 2),
                        y, k * 2) >> k) * y).pre(k));
                }
            }
            return x.pre(m);
        }

        poly rev() const { return poly(rbegin(), rend()); }

        poly mulT(poly b) { return T * b.rev() >> b.len() - 1; }

#undef T
```

```
    };
}
```

1.5.4 拆系数 FFT

```
// #define double long double
// #define ll __int128
//精度不够的时候要开int128和long double
//空间要开4倍 多项式长度2n, 扩域最多到4n
const int N = 1<<20;
typedef vector<ll> poly;
typedef complex<double> Complex;
const double pi = acos(-1);
const int L = 15, MASK = (1<<L) - 1;
Complex rev[N+7];
void FFT_init(int n){
        for(int i=0; i<n; i++)
                rev[i] = Complex(cos(2*i*pi/n), sin(2*i*pi/n));
}
void FFT(Complex *p, int n){
        for(int i=1, j=0; i<n-1; i++){
                for(int s=n; j^=s>>=1, ~j&s;);
                if(i<j) swap(p[i], p[j]);
                //依旧是蝴蝶变换, 没有什么不一样
        }
        for(int d=0; (1<<d)<n; d++){
                int m = 1<<d, m2 = m<<1, rm = n >> (d+1);
                for(int i=0; i<n; i+=m2)
                        for(int j=0; j<m; j++){
                                Complex &p1 = p[i+j+m], &p2 = p[i+j];
                                Complex t = rev[rm*j]*p1;
                                p1 = p2 - t; p2 = p2 + t;
                        }
        }
}
Complex A[N+7], B[N+7], C[N+7], D[N+7];
//拆系数fft || 任意模数ntt
poly operator*(poly &a, poly &b){
        int n = a.size()-1, m = b.size()-1;
        for(m+=n, n=1; n<=m; n<<=1);
        FFT_init(n);
        a.resize(n); b.resize(n);
        for(int i=0; i<n; i++){
                A[i] = Complex(a[i]>>L, a[i]&MASK);
```

```
                B[i] = Complex(b[i]>>L, b[i]&MASK);//拆成两部分
        }
        FFT(A, n); FFT(B, n);
        for(int i=0; i<n; i++){
                int j = (n-i)%n;
                //conj表示返回一个complex的共轭
                //压缩值域，  前一半和后一半
                Complex da = (A[i] - conj(A[j])) * Complex(0,-0.5),//实部
                        db = (A[i] + conj(A[j])) * Complex(0.5, 0),//虚部
                        dc = (B[i] - conj(B[j])) * Complex(0,-0.5),
                        dd = (B[i] + conj(B[j])) * Complex(0.5, 0);
                C[j] = da*dd+da*dc*Complex(0, 1);//进行复合
                D[j] = db*dd+db*dc*Complex(0, 1);
        }
        FFT(C, n); FFT(D, n);
        poly res(n);
        for(int i=0; i<n; i++){
                ll da = (ll)(C[i].imag()/n+0.5) % mod,
                   db = (ll)(C[i].real()/n+0.5) % mod,
                   dc = (ll)(D[i].imag()/n+0.5) % mod,
                   dd = (ll)(D[i].real()/n+0.5) % mod;
                res[i] = (((dd<<(L<<1))%mod + ((db+dc)<<L)%mod)%mod + da) % mod;
        }
        res.resize(m+1);
        return res;
}
```

1.5.5 拉格朗日插值

一个 n 次多项式可以由 n+1 个点唯一确定

$$F(x) = \sum_{k=1}^{n+1} a_k x^k$$

$$F(x) \equiv F(a)\ (Mod\ (x-a))$$

$$F(x) \equiv y_i\ (Mod\ (x-x_i))\ i \in [0, n]$$

可以用中国剩余定理求出 F(x), 复杂度为 $O(N^2)$

$$M = \prod (x-x_i),\ m_i = \prod_{i \neq j} (x-x_j)$$

$$m_i^{-1} = \prod_{i \neq j} (x_i - x_j)\ (Mod\ (x-x_i))$$

$$F(x) = \sum_{i=1}^{n+1} y_i \cdot \frac{m_i}{m_i^{-1}} (\ Mod\ M)$$

当 x_i 连续的时候，上式可以变成

$$F(x)=\sum_{i=1}^{n+1} y_i\cdot\frac{\prod_{i\neq j}(x-j)}{\prod_{i\neq j}(i-j)}(\ Mod\ M)$$

其中，分子的部分可以变成 $\frac{\prod(x-j)}{x-i}$，可以预处理

分母的部分可以拆成两段阶乘的乘积 $(i-1)!(n+1-i)!(-1)^{n+1-i}$

可以在 O(n) 的时间内求出 $F(x)$

```
//#pragma GCC optimize(2)
//#pragma comment(linker, "/STACK:102400000,102400000")
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef unsigned long long ull;
typedef pair<int,int> pii;
#define pb push_back
#define ln '\n'

const int mod = 998244353;
typedef vector<int> poly;
inline int power(int a, int b){
        int res = 1;
        for(;b;b>>=1,a=1ll*a*a%mod)
                if(b&1)
                        res=1ll*res*a%mod;
        return res;
}

inline int Inv(int x){return power(x, mod-2);}
//如果mod非质数需要换成exgcd

poly Lagrange(const poly &x, const poly &y){//插系数， x不连续， y连续
        int n = x.size();//n个点确定一个n+1次多项式
        poly a(n, 0), b(n+1, 0), c(n+1, 0), f(n, 0);
        for(int i = 0; i < n; i++){
                int A = 1;
                for(int j = 0; j < n; j++)
                        if(i!=j)
                                A = 1ll * A * (x[i] - x[j] + mod) % mod;
                //crt
                a[i] = 1ll * Inv(A) * y[i] % mod;
        }
        b[0] = 1;
        for(int i = 0; i < n; i++){
                for(int j = i + 1; j; j--)
```

```
                        b[j] = (1ll * b[j] * (mod - x[i]) % mod + b[j-1]) % mod;
                b[0] = 1ll * b[0] * (mod - x[i]) % mod;
        }
        for(int i = 0; i < n; i++){
                int inv = Inv(mod - x[i]);
                if(inv){
                        c[0] = 1ll * b[0] * inv % mod;
                        for(int j = 1; j <= n; j++)
                                c[j] = 1ll * (b[j] + mod - c[j-1]) * inv % mod;
                } else {
                        for(int j = 0; j < n; j++)
                                c[j] = b[j + 1];
                }
                for(int j = 0; j < n; j++)
                        f[j] = (f[j] + 1ll * a[i] * c[j] % mod) % mod;
        }
        return f;
}

inline int calc(int x, const poly &a){
        int ans = 0;
        int n = a.size(); --n;
        for(;~n;n--)
                ans = (1ll * ans * x % mod + a[n]) % mod;
        return ans;
}

int main(){
        ios::sync_with_stdio(false);
        cin.tie(0);

}
```

```
// CF622F The Sum of the k-th Powers
// 给n, k, 求 sigma i^k
// 注意到答案是一个k+1次的多项式, 线性筛出1..k+2的值然后插值即可
#include <bits/stdc++.h>
using namespace std;
const int N = 1e6 + 5, mod = 1e9 + 7;

int n, k, tab[N], p[N], pcnt, f[N], pre[N], suf[N], fac[N], inv[N], ans;

int qpow(int x, int y) {
  int ans = 1;
  for (; y; y >>= 1, x = 1LL * x * x % mod)
```

```
    if (y & 1) ans = 1LL * ans * x % mod;
  return ans;
}

void sieve(int lim) {
  f[1] = 1;
  for (int i = 2; i <= lim; i++) {
    if (!tab[i]) {
      p[++pcnt] = i;
      f[i] = qpow(i, k);
    }
    for (int j = 1; j <= pcnt && 1LL * i * p[j] <= lim; j++) {
      tab[i * p[j]] = 1;
      f[i * p[j]] = 1LL * f[i] * f[p[j]] % mod;
      if (!(i % p[j])) break;
    }
  }
  for (int i = 2; i <= lim; i++) f[i] = (f[i - 1] + f[i]) % mod;
}

int main() {
  scanf("%d%d", &n, &k);
  sieve(k + 2);
  if (n <= k + 2) return printf("%d\n", f[n]) & 0;
  pre[0] = suf[k + 3] = 1;
  for (int i = 1; i <= k + 2; i++) pre[i] = 1LL * pre[i - 1] * (n - i) % mod;
  for (int i = k + 2; i >= 1; i--) suf[i] = 1LL * suf[i + 1] * (n - i) % mod;
  fac[0] = inv[0] = fac[1] = inv[1] = 1;
  for (int i = 2; i <= k + 2; i++) {
    fac[i] = 1LL * fac[i - 1] * i % mod;
    inv[i] = 1LL * (mod - mod / i) * inv[mod % i] % mod;
  }
  for (int i = 2; i <= k + 2; i++) inv[i] = 1LL * inv[i - 1] * inv[i] % mod;
  for (int i = 1; i <= k + 2; i++) {
    int P = 1LL * pre[i - 1] * suf[i + 1] % mod;
    int Q = 1LL * inv[i - 1] * inv[k + 2 - i] % mod;
    int mul = ((k + 2 - i) & 1) ? -1 : 1;
    ans = (ans + 1LL * (Q * mul + mod) % mod * P % mod * f[i] % mod) % mod;
  }
  printf("%d\n", ans);
  return 0;
}
```

1.6 同余

1.6.1 BSGS

1.6.2 原根

1. 概念及定理

   - 阶 (即幂的最小循环节): a 在模 p 意义下的阶是：最小的整数 k，使得 $a^k \equiv 1(mod\ p)$，记作 $\delta_p(a)$

性质 1.：设 $p \geq 1, gcd(a,p) = 1$，若 $a^n \equiv 1(modp), n > 0$，则 $\delta_p(a)|n$

性质 2.：设 $\delta_p(a)|\phi(p)$

推论 1.：若 p, q 均为奇质数，且 $q|(a^p - 1)$ 则有 $q|(a-1)$ 或 $2p|(q-1)$

推论 2. ：原根 (满足 $\delta_p(g) = \phi(p)$ 的 g，也即 $g^{\phi(p)} \equiv 1(mod\ p)$)

1.7 筛法

1.7.1 常用积性函数关系

0. 若没有声明 $*$ 表示狄利克雷卷积

1. $\epsilon = \mu * I$

2. $id = \varphi * I$

3. $\sum_{i=1}^{n} \frac{\mu(i)}{i} = \frac{\phi(n)}{n}$

1.7.2 PN 筛

1. $f = g * h$

2. g 的前缀和 G 好求

3. $f(p) = g(p)$，此时 $h(p) = 0, h(1) = 1$

4. 有用的值即 $h(n) > 1$ 的 n 只有 $O(\sqrt{n})$ 个

$$S(n) = \sum_{i=1}^{n} f(i) = \sum_{i=1}^{n} (h*g)(i) = \sum_{i=1}^{n} \sum_{d|n} h(d) g(\frac{n}{d}) = \sum_{d=1}^{n} h(d) \sum_{i}^{\lfloor \frac{n}{d} \rfloor} g(i) = \sum_{d=1}^{n} h(d) G(\lfloor \frac{n}{d} \rfloor)$$

1.7.3 杜教筛

1. $f * g$

2. 直接暴力是 $O(n^{\frac{3}{4}})$ 的

3. 预处理前 $O(n^{\frac{2}{3}})$ 项复杂度可以到 $O(n^{\frac{2}{3}})$，这个东西被叫做阈值分治

4. 如果需要多次递归可以先递归一次然后处理出每个需要的元素整除分块对应的根号个元素

$$S(n) = \sum_{i=1}^{n} f(i)$$

$$F(n) = \sum_{i=1}^{n} (f*g)(i) = \sum_{i=1}^{n} \sum_{d|i} f(d) g(\frac{i}{d}) = \sum_{d=1}^{n} g(d) \sum_{i}^{\lfloor \frac{n}{d} \rfloor} f(i) = \sum_{d=1}^{n} g(d) S(\lfloor \frac{n}{d} \rfloor)$$

$$\sum_{i=1}^{n}(f*g)(i)=\sum_{d=1}^{n}g(d)S(\lfloor\frac{n}{d}\rfloor)$$

$$g(1)S(n)=\sum_{i=1}^{n}(f*g)(i)-\sum_{d=2}^{n}g(d)S(\lfloor\frac{n}{d}\rfloor)$$

1.7.4 min25 筛 (待填)

2 数据结构

2.1 单调队列

```
//51nod1275 连续子段的差异
//求满足最大值-最小值<=k的区间有多少个
#include<bits/stdc++.h>
using namespace std;
#define ll long long
#define pb push_back

inline ll read() {
    ll x = 0, f = 1; char ch = getchar();
    for(; ch < '0' || ch>'9'; ch = getchar())
        if(ch == '-') f = -f;
    for(; ch >= '0' && ch <= '9'; ch = getchar())
        x = x * 10 + ch - '0';
    return x * f;
}

#define ln endl

const int N = 5e4+5;
int n, k, a[N], q[2][N], l[2], r[2];
ll ans;

int main(){
    #ifndef ONLINE_JUDGE
        freopen("0.txt","w",stdout);
    #endif
    n=read(); k=read();
    l[1] = l[0] = r[1] = r[0] = 1;
    for(int i=1; i<=n; i++)
        a[i] = read();
    int j = 1;
    q[1][l[1]] = q[0][l[0]] = 1;
    for(int i=1; i<=n; i++){//以i为左端点
        while(l[1] <= r[1] && q[1][l[1]] < i)
            l[1]++;
        while(l[0] <= r[0] && q[0][l[0]] < i)
            l[0]++;
        //把i往左的点都排掉
        while(j<=n && a[q[1][l[1]]] - a[q[0][l[0]]] <= k) {
            ++j;
            while(l[1] <= r[1] && a[q[1][r[1]]] <= a[j])
```

```
                --r[1];
            q[1][++r[1]] = j;
            while(l[0] <= r[0] && a[q[0][r[0]]] >= a[j])
                --r[0];
            q[0][++r[0]] = j;
        }
        ans = ans + (j-i);
        // printf("%d %d\n", i, j);
    }
    printf("%lld\n", ans);
}
```

2.2 线段树

2.2.1 普通线段树

```
//这个N得根据需要改成自己需要的值域大小
struct segtree {
        int l, r;
        ll mn, mx, sum;
}T[N<<2];
int tag[N<<2];
segtree operator +(segtree ls, segtree rs){
        segtree rt;
        rt.l = ls.l; rt.r = rs.r;
        rt.mx = max(ls.mx, rs.mx);
        rt.mn = min(ls.mn, rs.mn);
        rt.sum = ls.sum + rs.sum;
        return rt;
}
void pushup(int rt){
        T[rt] = T[rt<<1] + T[rt<<1|1];
}
void pushdown(int rt){
        if(tag[rt]){
                segtree &ls = T[rt<<1], &rs = T[rt<<1|1];
                ls.sum += 1ll*tag[rt]*(ls.r-ls.l+1);
                rs.sum += 1ll*tag[rt]*(rs.r-rs.l+1);
                ls.mx += tag[rt]; rs.mx += tag[rt];
                ls.mn += tag[rt]; rs.mn += tag[rt];
                tag[rt<<1] += tag[rt];
                tag[rt<<1|1] += tag[rt];
                tag[rt] = 0;
        }
}
```

```
void build(int l, int r, int rt){
	tag[rt] = 0;
	if(l == r){
		T[rt] = (segtree){l, l, 0, 0, 0};
		return;
	}
	int mid = l+r >> 1;
	build(l, mid, rt<<1);
	build(mid+1, r, rt<<1|1);
	pushup(rt);
}
void add(int L, int R, int v, int rt){
	int l = T[rt].l, r = T[rt].r;
	if(L <= l && r <= R)
		return T[rt].sum += 1ll*(r-l+1)*v, T[rt].mx += v,
			T[rt].mn += v, tag[rt]+=v, void();
	int mid = l+r>>1;
	pushdown(rt);
	if(L <= mid)
		add(L, R, v, rt<<1);
	if(R > mid)
		add(L, R, v, rt<<1|1);
	pushup(rt);
}
segtree query(int L, int R, int rt){
	int l = T[rt].l, r = T[rt].r;
	if(L <= l && r <= R)
		return T[rt];
	int mid = l+r>>1;
	pushdown(rt);
	if(L <= mid && R > mid)
		return query(L, R, rt<<1) + query(L, R, rt<<1|1);
	else if(L <= mid)
		return query(L, R, rt<<1);
	else
		return query(L, R, rt<<1|1);
}
```

2.2.2 动态开点线段树

```
/*
	维护一段递增的序列
	一开始是a[i]=i
	在线段树上二分， 找到最右的一个<T的数 和 最左的一个>T的数
```

```
    找数需要的点最多2e5+2e5个
    所以总共最多需要6e5个点
    区间修改
    单点查询需要的节点最多2e5个
    空间开ki*4即可
    单个点最多需要logN的深度，  也就是
*/
#include<bits/stdc++.h>
using namespace std;

#define ll long long

inline ll read() {
    ll x = 0, f = 1; char ch = getchar();
    for(; ch < '0' || ch>'9'; ch = getchar())
        if(ch == '-') f = -f;
    for(; ch >= '0' && ch <= '9'; ch = getchar())
        x = x * 10 + ch - '0';
    return x * f;
}

inline void chkmin( int &a, int b ) { if(a > b) a = b; }

inline void chkmax( int &a, int b ) { if(a < b) a = b; }

#define _ read()

#define ln endl

const int N=6e5+5;

int cnt;
struct node{
    int l, r, lson, rson, mn, mx, lzy;
}T[N<<5];

void up(int rt){
    T[rt].mn=min(T[T[rt].lson].mn, T[T[rt].rson].mn);
    T[rt].mx=max(T[T[rt].lson].mx, T[T[rt].rson].mx);
}

void down(int rt){
    // if(T[rt].lzy){
    if(!T[rt].lson)
```

```
        T[rt].lson=++cnt,
        T[cnt]=(node){T[rt].l, T[rt].l+T[rt].r>>1, 0, 0, T[rt].l, T[rt].l+T[rt].r>>
    if(!T[rt].rson)
        T[rt].rson=++cnt,
        T[cnt]=(node){(T[rt].l+T[rt].r>>1)+1, T[rt].r, 0, 0, (T[rt].l+T[rt].r>>1)+1
    T[T[rt].lson].mx+=T[rt].lzy;
    T[T[rt].rson].mx+=T[rt].lzy;
    T[T[rt].lson].mn+=T[rt].lzy;
    T[T[rt].rson].mn+=T[rt].lzy;
    T[T[rt].lson].lzy+=T[rt].lzy;
    T[T[rt].rson].lzy+=T[rt].lzy;
    T[rt].lzy=0;
    // }
}

void add(int L, int R, int v, int l, int r, int rt){//区间加法
    if(!rt)
        T[rt=++cnt]=(node){l, r, 0, 0, l, r, 0};
    if(L<=l&&r<=R){
        T[rt].mx+=v, T[rt].mn+=v, T[rt].lzy+=v;
        return;
    }
    int mid=l+r>>1;
    down(rt);
    if(L<=mid)
        add(L, R, v, l, mid, T[rt].lson);
    if(R>mid)
        add(L, R, v, mid+1, r, T[rt].rson);
    up(rt);
}

int queryl(int x, int l, int r, int rt){//二分找到=r的点
    if(l==r)
        return l-(T[rt].mn>=x);
    int mid=l+r>>1;
    down(rt);
    if(T[T[rt].rson].mn<x)//右子树中有<x的点
        return queryl(x, mid+1, r, T[rt].rson);
    else
        return queryl(x, l, mid, T[rt].lson);
}

int queryr(int x, int l, int r, int rt){
    if(l==r)
```

```
        return l+(T[rt].mx<=x);//如果最右边的比x小的点都>=x
    int mid=l+r>>1;
    down(rt);
    if(T[T[rt].lson].mx>x)//左子树中有>x的点
        return queryr(x, l, mid, T[rt].lson);
    else
        return queryr(x, mid+1, r, T[rt].rson);
}

int query(int x, int l, int r, int rt){
    if(l==r)
        return T[rt].mx;
    int mid=l+r>>1;
    down(rt);
    if(x<=mid)
        return query(x, l, mid, T[rt].lson);
    else
        return query(x, mid+1, r, T[rt].rson);

}

int main(){
    #ifndef ONLINE_JUDGE
        freopen("0.out","w", stdout);
    #endif
    int n=read();
    T[cnt=1]=(node){0, (int)1e9, 0, 0, 0, (int)1e9, 0};
    int lastans=0;
    while(n--){
        int Ti=read();
        int L=queryl(Ti, 0, (int)1e9, 1), R=queryr(Ti, 0, (int)1e9, 1);
        // printf("%d %d\n", L, R);
        if(L>=0)
            add(0, L, 1, 0, (int)1e9, 1);
        if(R<=(int)1e9)
            add(R, (int)1e9, -1, 0, (int)1e9, 1);
        int m=read();
        while(m--){
            int x=(read()+lastans)%((int)1e9+1); //x=x+lastans;
            // printf("%d ", x);
            printf("%d\n", lastans=query(x, 0, (int)1e9, 1));
        }
    }
}
```

```
/*
    值域是0..1e9
*/
```

2.2.3 主席树

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

#define ll long long

inline ll read() {
    ll x = 0, f = 1; char ch = getchar();
    for(; ch < '0' || ch>'9'; ch = getchar())
        if(ch == '-') f = -f;
    for(; ch >= '0' && ch <= '9'; ch = getchar())
        x = x * 10 + ch - '0';
    return x * f;
}

inline void chkmin( int &a, int b ) { if(a > b) a = b; }

inline void chkmax( int &a, int b ) { if(a < b) a = b; }

#define _ read()

#define ln endl

const int N = 2e5 + 5;
int n, m, T[N];
int ls[N << 5], rs[N << 5], siz[N << 5], cnt;

inline void insert( int v, int l, int r, int x, int &y)
{
        y = ++cnt; siz[y] = siz[x] + 1;
        if(l == r) return;
        int mid = (l + r) >> 1;
        ls[y] = ls[x]; rs[y] = rs[x];
        if(v <= mid)
                insert(v, l, mid, ls[x], ls[y]);
        else
                insert(v, mid + 1, r, rs[x], rs[y]);
}

```

```
inline int ask( int k, int l, int r, int x, int y)
{
        if(k > siz[y] - siz[x] + 1)
                return 0;
        if(l == r)
                return l;
        int mid = (l + r) >> 1;
        if(siz[ls[y]] - siz[ls[x]] >= k)
                return ask(k, l, mid, ls[x], ls[y]);
        else
                return ask(k - siz[ls[y]] + siz[ls[x]], mid + 1, r, rs[x], rs[y]);
}

int main()
{
        n = _; m = _;
        for( int i = 1, x; i <= n; i++ )
                x = _, insert(x, -1000000000, 1000000000, T[i - 1], T[i]);
        while(m--)
        {
                int l = _, r = _, k = _;
                printf("%d\n", ask(k, -1000000000, 1000000000, T[l - 1], T[r]));
        }
}
```

2.3 并查集

2.3.1 普通并查集

```
struct DSU{
        vector<int>fa,siz;
        int tot;
        DSU(int n){
                fa.resize(n+1);
                siz.resize(n+1);
                for(int i=1; i<=n; i++)
                        fa[i] = i, siz[i] = 1;
                tot = n;
        }
        int find(int x){return fa[x]==x?x:fa[x] = find(fa[x]);}
        bool merge(int x, int y){
                x = find(x); y = find(y);
                if(x == y) return false;
                if(siz[x] > siz[y])
                        swap(x, y);
```

```
                fa[x] = y;
                siz[y] += siz[x];
                tot--;
                return true;
        }
};
```

2.3.2 可删除并查集

```
//动态维护无向图的连通性
int f[N], siz[N];
int top;
pair<int,int> e[10*N];

int find(int x){
    if(f[x] == x) return f[x];
    int par = find(f[x]);
    return par;
}

void add(int x, int y){
    int fx = find(x), fy = find(y);
    if(fx != fy){
        if(siz[fy] > siz[fx])
            siz[fy] += siz[fx], f[fx] = fy, e[++top] = {fy, fx};
        else
            siz[fx] += siz[fy], f[fy] = fx, e[++top] = {fx, fy};
        //最后一条边是谁往谁连的
        res--;
    } else
        e[++top] = {0, 0};//已经联通那么实际上不需要加到图里
}

void del(){
    //删除最后一条边， 回复到上一次的状态
    auto now = e[top--];
    if(now.first) res++;
    siz[now.first] -= siz[now.second];
    f[now.second] = now.second;
}
```

3 字符串

3.1 KMP 算法

```
inline void getfail(char *s){
        //求fail函数
        int n = strlen(s+1);
        for(int i=0; i<=n; i++)
                fail[i] = 0;
        for(int i = 2, j = 0; i <= n; i++){
                while(j && s[j+1] != s[i])
                        j = fail[j];
                if(s[j+1] == s[i])
                        ++j;
                fail[i] = j;
        }
}
```

3.2 后缀数组

后缀排序

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

#define ll long long

inline ll read() {
    ll x = 0, f = 1; char ch = getchar();
    for(; ch < '0' || ch>'9'; ch = getchar())
        if(ch == '-') f = -f;
    for(; ch >= '0' && ch <= '9'; ch = getchar())
        x = x * 10 + ch - '0';
    return x * f;
}

inline void chkmin( int &a, int b ) { if(a > b) a = b; }

inline void chkmax( int &a, int b ) { if(a < b) a = b; }

#define _ read()

#define ln endl

const int N=1e6+5;
char s[N];
```

```
int n, cnt[N], sa[N], sa2[N], rk[N<<1], rk2[N<<1];

inline void getSA(){
	for(int i=1; i<=n; i++)
		rk[i]=s[i], sa[i]=i;
	int m=max(n, 256);
	for(int i=1; i<=m; i++) cnt[i]=0;
	for(int i=1; i<=n; i++) cnt[rk[sa[i]]]++;
	for(int i=1; i<=m; i++)
		cnt[i]+=cnt[i-1];
	for(int i=n; i; i--)
		sa2[cnt[rk[sa[i]]]--]=sa[i];
	for(int i=1; i<=n; i++)
		sa[i]=sa2[i], sa2[i]=0;
	for(int w=1; w<=n; w<<=1){
		for(int i=1; i<=m; i++) cnt[i]=0;
		for(int i=1; i<=n; i++) cnt[rk[sa[i]+w]]++;
		for(int i=1; i<=m; i++)
			cnt[i]+=cnt[i-1];
		for(int i=n; i; i--)
			sa2[cnt[rk[sa[i]+w]]--]=sa[i];
		for(int i=1; i<=n; i++)
			sa[i]=sa2[i], sa2[i]=0;
		for(int i=1; i<=m; i++) cnt[i]=0;
		for(int i=1; i<=n; i++) cnt[rk[sa[i]]]++;
		for(int i=1; i<=m; i++)
			cnt[i]+=cnt[i-1];
		for(int i=n; i; i--)
			sa2[cnt[rk[sa[i]]]--]=sa[i];
		for(int i=1; i<=n; i++)
			sa[i]=sa2[i], sa2[i]=0;
		int tot=0;
		for(int i=1; i<=n; i++){
			if(rk[sa[i]]==rk[sa[i-1]]&&rk[sa[i]+w]==rk[sa[i-1]+w])
				rk2[sa[i]]=tot;
			else
				rk2[sa[i]]=++tot;
		}
		for(int i=1; i<=n; i++)
			rk[i]=rk2[i], rk2[i]=0;
	}
	for(int i=1; i<=n; i++)
		cout<<sa[i]<<" ";
	cout<<ln;
```

```
}

int main(){
        scanf("%s", s+1); n=strlen(s+1);
        getSA();
}
```

4 图论

4.1 最短路 – dij 的最长路是错的

```
struct dijkstra{
        #define N 100005
        #define M 500005
        struct edge{
                int to, nxt;
                double w;
        }e[M];

        int head[N], cnt;
        dijkstra(){clear();}

        inline void clear(){
                memset(head, 0, sizeof(head));
                cnt = 0;
        }

        inline void insert(int u, int v, double w){
                e[++cnt] = (edge){v, head[u], w};
                head[u] = cnt;
        }

        struct node{
                double dis;
                int id;
                bool operator <(const node &b) const{
                        return dis>b.dis;
                }
        };
        priority_queue<node> q;
        double dis[N];
        int vis[N], n;

        inline void dij(int st){
                for(int i=0; i<=n; i++)
                        dis[i] = 2e9, vis[i] = 0;
                dis[st] = 0;
                q.push((node){0, st});
                while(!q.empty()){
                        node now = q.top(); q.pop();
                        if(vis[now.id])
                                continue;
```

```
                        vis[now.id] = 1;
                        int id = now.id;
                        for(int i=head[id]; i; i=e[i].nxt)
                                if(dis[e[i].to] > now.dis + e[i].w)
                                        dis[e[i].to] = now.dis + e[i].w,
                                        q.push((node){dis[e[i].to], e[i].to});
                }
        }
};
```

4.2 二分图

4.2.1 二分图最大匹配

```
const int N=1e4+5;
const int M=1e6+N;
int n;
int vis[M], from[M];
vector<int> v;
vector<std::vector<int>> e(N);
bool dfs(int x){
    for(int y:e[x])
        if(!vis[y]){
            vis[y]=1;
            if(!from[y]||dfs(from[y])){
                from[y]=x;
                return true;
            }
        }
    return false;
}
```

4.2.2 二分图最大权匹配

```
const int N = 505;
const int inf = 2e9+233;

int n, nl, nr, m, x, y, z;
int g[N][N];

//nl: 左部图点数
//nr: 右部图点数
//g: 二分图边权
namespace KM{
```

```
	int left[N], right[N];
	int visl[N], visr[N];
	int lx[N], ly[N], slack[N];

	bool augment(int x){
		visl[x] = 1;
		for(int y = 1; y <= n; y++)
			if(visr[y]) continue;
			else {
				int slk = lx[x] + ly[y] - g[x][y];
				if(!slk){
					visr[y] = 1;
					if(!right[y] || augment(right[y])){
						right[y] = x; left[x] = y;
						return 1;
					}
				} else
					slack[y] = min(slack[y], slk);
			}
		return 0;
	}

	void solve(){
		for(int i=1; i<=n; i++)
			for(int j=1; j<=n; j++)
				ly[j] = max(ly[j], g[i][j]);
		for(int i=1; i<=n; i++){
			for(int j=1; j<=n; j++)
				visl[j] = visr[j] = 0, slack[j] = inf;
			if(augment(i)) continue;
			while(1){
				int d = inf, x;
				for(int j = 1; j <= n; j++)
					if(!visr[j]) d = min(d, slack[j]);

				for(int j = 1; j <= n; j++){
					if(!visr[j]){
						ly[j] -= d;
						slack[j] -= d;
						if(!slack[j]) x = j;
					}
					if(!visl[j])
						lx[j] += d;
				}
```

```

                                if(!right[x]) break;
                                visr[x] = 1; visl[right[x]] = 1;
                                x = right[x];
                                for(int y = 1; y <= n; y++)
                                        slack[y] = min(slack[y],
                                                lx[x] + ly[y] - g[x][y]);
                        }

                        for(int j=1; j<=n; j++)
                                visl[j] = visr[j] = 0;
                        augment(i);
                }
        }

        void answer(){
                ll ans = 0;
                for(int i=1; i<=n; i++)
                        ans += lx[i] + ly[i];
                cout << -ans << ln;
        }
}
```

4.3 网络流

最大流效率 $O(N^2M)$, 在二分图上复杂度可以达到 $O(M\sqrt{N})$

```
//要先算好边需要多少，  点需要多少
//dinic复杂度为n^2m
template<typename T>
struct Dinic{
        int s, t, tot;
        T F;
        const T INF = 2e9;
        vector<int> head, d, lst;
        struct Edge{int to, nxt; T w;};
        vector<Edge> e;
        Dinic(int n, int m, int st, int ed){
                head.assign(n+1, -1);
                e.resize(m<<1);
                d.resize(n+1);
                s = st; t = ed; tot = 0; F = 0;
        }
        inline int add(int u, int v, T w){
                e[tot]=(Edge){v, head[u], w};
```

```
			head[u] = tot++;
			if(u == s) F+=w;
			return tot-1;
		}
		inline int insert(int u, int v, T w){
			int tmp = add(u, v, w);
			add(v, u, 0);
			return tmp;
		}
		inline bool bfs(){
			d.assign(d.size(), 0);
			lst = head;
			queue<int> q;
			q.push(s); d[s] = 1;
			while(!q.empty()){
				int u = q.front(); q.pop();
				for(int i=head[u]; ~i; i=e[i].nxt){
					int v = e[i].to;
					if(!d[v] && e[i].w > 0){
						d[v] = d[u] + 1;
						q.push(v);
					}
				}
			}
			return d[t] > 0;
		}
		int dfs(int x, int mn){
			if(x == t) return mn;
			for(int &i = lst[x]; ~i; i=e[i].nxt){
				if(d[e[i].to] == d[x]+1 && e[i].w){
					int val = dfs(e[i].to, min(e[i].w, mn));
					if(val > 0){
						e[i].w -= val;
						e[i^1].w += val;
						return val;
					}
				}
			}
			return 0;
		}
		inline T work(){
			T ans = 0;
			while(bfs())
				while(int d = dfs(s, INF))
```

```
                                ans+=d;
                return ans;
        }
        inline bool check(){//判满流
                return work() == F;
        }
};
```

费用流 (EK) 效率 $O(NM^2F)$

```
template<typename T>
struct EK{
    const T INF = 2e18;
    T cost;
    struct Edge{int to, nxt, w, f;};
    vector<Edge>e;
    vector<bool> vis;
    vector<int> head, lst;
    vector<T> dis;
    int tot, s, t;
    EK(int n, int m, int st, int ed){
        head.assign(n+1, -1);
        e.resize(m<<1);
        dis.resize(n+1);  vis.resize(n+1);
        s = st; t = ed; tot = 0; cost = 0;
    }
    inline int add(int u, int v, int w, int f){
        e[tot]=(Edge){v, head[u], w, f};
        head[u] = tot++;
        return tot-1;
    }
    inline int insert(int u, int v, int w, int f){
        int tmp = add(u, v, w, f);
        add(v, u, -w, 0);
        return tmp;
    }
    inline bool spfa(){
        queue<int> q;
        dis.assign(dis.size(), INF);
        vis.assign(vis.size(), false);
        dis[s] = 0; vis[s] = 1;
        lst = head;
        q.push(s);
        while(!q.empty()){
            int u = q.front(); q.pop();
            for(int i = head[u]; ~i; i = e[i].nxt){
```

```
                int v = e[i].to;
                if(e[i].f > 0 && dis[v] > dis[u] + e[i].w){
                    dis[v] = dis[u] + e[i].w;
                    if(!vis[v]){
                        vis[v] = 1;
                        q.push(v);
                    }
                }
            }
            vis[u] = 0;
        }
        return dis[t] != INF;
    }
    int dfs(int u, int f){
        if(u == t || !f) return f;
        vis[u] = 1;
        int ans = 0;
        for(int &i=lst[u]; ~i; i=e[i].nxt){
            int v = e[i].to;
            if(e[i].f && dis[v] == dis[u] + e[i].w && !vis[v]){
                int val = dfs(v, min(e[i].f, f - ans));
                if(val){
                    e[i].f -= val;
                    e[i^1].f += val;
                    cost += val * e[i].w;
                    ans += val;
                    if(ans == f) break;
                }
            }
        }
        vis[u] = 0;
        return ans;
    }
    inline array<T,2> work(){
        T ans = 0;
        while(spfa()){
            T d = dfs(s, INF);
            ans += d;
        }
        return {ans, cost};
    }
};
```

4.4 强连通分量

```cpp
struct SCC{
//SCC(n) n:点数
    vector<vector<pii>>e;
    stack<int>sta;
    vector<int>id,dfn,low,ins,siz;
    vector<ll>val;
    int tot, scc, n;
    SCC(int nn){
        n = nn;
        e.resize(n+1); siz.resize(n+1);
        id.resize(n+1); dfn.resize(n+1);
        low.resize(n+1); ins.resize(n+1); val.resize(n+1);
        scc = tot = 0;
    }
    void tarjan(int u){
        dfn[u] = low[u] = ++tot;
        sta.push(u); ins[u] = 1;
        for(pii E: e[u]){
            int v = E.first;
            if(!dfn[v]){
                tarjan(v);
                low[u] = min(low[u], low[v]);
            } else if(ins[v]){
                low[u] = min(low[u], dfn[v]);
            }
        }
        if(dfn[u] == low[u]){
            ++scc;
            while(sta.top()!=u){
                int v = sta.top(); sta.pop();
                id[v] = scc;
                siz[scc]++;
                ins[v] = 0;
            }
            ins[u] = 0;
            id[u] = scc;
            siz[scc]++;
            sta.pop();
        }
    }
    inline void remake(){
        for(int i=1; i<=n; i++)
            if(!dfn[i]) tarjan(i);
```

```
        vector<vector<pii>> g(n+1);
        for(int u=1; u<=n; u++)
            for(pii E: e[u]){
                int v = E.first, w = E.second;
                if(id[u] != id[v])
                    g[id[u]].pb({id[v], w});//边权
                else
                    val[id[u]] += w;//点权
            }
        e = g;
    }
};
```

4.5 点双联通分量

边双只需要 low[v] > dfn[u] 即可，然后用强连通分量的弹栈方式

```
struct DCC{

        vector<vector<int>>e;
        vector<vector<int>>g;
        vector<int> dfn, low;
        stack<int> sta;
        int vcc, tot, n;
        //vcc: n+1~2n 方点 可以建Block Forest
        DCC(int nn){
                n = nn;
                e.resize(n+1); g.resize(n<<1|1);
                dfn.resize(n+1); low.resize(n+1);
                tot = 0;
                vcc = n;
        }
        void tarjan(int u, int fa){
                low[u] = dfn[u] = ++tot; sta.push(u);
                for(int v: e[u]){
                        if(v == fa) continue;
                        if(!dfn[v]){
                                tarjan(v, u);
                                low[u] = min(low[u], low[v]);
                                if(low[v] >= dfn[u]){//u是割点
                                        vcc++;//vertice double connected conponents
                                        while(sta.top() != v){
                                                int x = sta.top(); sta.pop();
                                                g[vcc].pb(x); g[x].pb(vcc);
                                        }
```

```
                    sta.pop();
                    g[vcc].pb(v); g[v].pb(vcc);
                    g[vcc].pb(u); g[u].pb(vcc);
                }
            } else //双联通不需要考虑横叉边
                low[u] = min(low[u], dfn[v]);
        }
    }
    inline int work(){
        int tot = 0;
        for(int i=1; i<=n; i++){
            if(!dfn[i]){
                if(!tot)
                    tarjan(i, 0);
                ++tot;
            }
        }
        return tot;
    }
};
```

5 杂项

5.1 快读快输板子

```
inline ll read() {
    ll x = 0, f = 1; char ch = getchar();
    for(; ch < '0' || ch>'9'; ch = getchar())
        if(ch == '-') f = -f;
    for(; ch >= '0' && ch <= '9'; ch = getchar())
        x = x * 10 + ch - '0';
    return x * f;
}
inline void print(ll x){
    if(!x)
        return;
    print(x/10);
    putchar(x%10+'0');
}
inline void write(ll x){
    if(x<0) putchar('-'), x=-x;
    if(!x) putchar('0');
    print(x);
}
inline void writeln(ll x){
    write(x); putchar('\n');
}

//用的时候记得改<<里的内容
struct FastIO{//不可与scanf,printf,getchar,putchar,gets,puts,cin,cout混用
private:
    static const int BUFSIZE=1e5;
    char buf[BUFSIZE];int pos,len;//读入buffer(缓冲器)以及读入指针
    int wpos;char wbuf[BUFSIZE];//输出指针以及输出buffer
    char a[21];//储存输出的数字
#define gc() (pos == len && (len=(pos=0)
    +fread(buf,1,BUFSIZE,stdin),!len)?EOF:buf[pos++])
#define pc(c) (wpos == BUFSIZE ?
    fwrite(wbuf,1,BUFSIZE,stdout),wpos=0,
    wbuf[wpos++]=c:wbuf[wpos++]=c)
public:
    FastIO():wpos(0),pos(0),len(0){}
    ~FastIO(){if(wpos)fwrite(wbuf,1,wpos,stdout),wpos=0;}
    inline char getc(){return gc();}//读取char
    inline void putc(char c){pc(c);}//输出字符
    inline long long rd(){//读取long long
```

```
        long long x=0;char c=gc();bool f=0;
        for(;c<'0'||c>'9';c=gc())f|=c=='-';
        for(;c>='0'&&c<='9';c=gc())x=(x<<3)+(x<<1)+(c^48);
        return f?~x+1:x;
    }
    template<typename T>inline bool read(T &x){//多测读整数while(io.read(n))work();
        x=0;char c=gc();bool f=0;
        for(;c<'0'||c>'9';c=gc()){if(c==EOF)return false;f|=c=='-';}
        for(;c>='0'&&c<='9';c=gc())x=(x<<3)+(x<<1)+(c^48);
        if(f)x=~x+1;return true;
    }
    template<typename T>inline void wt(T x){//输出整数
        if(x<0)pc('-'),x=-x;short h=0;
        for(a[++h]='0'+x%10,x/=10;x;x/=10)a[++h]='0'+x%10;
        while(h)pc(a[h--]);
    }
    template<typename T>inline void wtl(T x){wt(x);pc('\n');}//write line输出整数并
    template<typename T>inline void wtb(T x){wt(x);pc(' ');}//write blank输出整数并
    inline int gets(char *s){
        int l=0;char c=gc();for(;c<=' ';c=gc());
        for(;c>' ';c=gc())s[l++]=c;s[l]=0;
        return l;
    }
    inline void puts(const char *s){
        const char *p=s;while(*p)pc(*p++);
    }//输出字符串 (不带换行)
    template<typename T>inline FastIO & operator >> (T &a){
        return read(a),*this;
    }//io>>a>>b;只能输入整数
    template<typename T>inline FastIO & operator << (T a){
        return wtl(a),*this;
    }//io<<a<<b;输出整数并带有回车
}io;//本地测试出入结束后请输入一次ctrl Z
```

5.2 分治

5.2.1 cdq 分治

最长上升子序列的数量

```
//最长上升子序列计数
//转化为偏序问题
#include<bits/stdc++.h>
using namespace std;
#define ll long long
#define pii pair<int,int>
```

```cpp
#define pb push_back
#define ln '\n'

const int N = 1e5+5;
const int mod = 1e9+7;
int n, a[N], id[N];
pii f[N];

pii calc (pii a, pii b){
    if(a.first < b.first) return b;
    if(a.first > b.first) return a;
    a.second += b.second;
    if(a.second >= mod)
        a.second -= mod;
    return a;
};

void work(int l, int r){
    if(l == r)
        return;
    int mid = l+r>>1;
    work(l, mid);
    for(int i=l; i<=r; i++)
        id[i] = i;
    sort(id+l, id+r+1, [](int x, int y){
        return a[x] == a[y] ? x > y : a[x] < a[y];
    });
    pair<int,int> now = {0,0};

    for(int i=l; i<=r; i++){
        if(id[i] <= mid)
            now = calc(now, f[id[i]]);
        else{
            auto tmp = now;
            tmp.first++;
            f[id[i]] = calc(f[id[i]], tmp);
        }
    }

    work(mid+1, r);
}

int main(){

```

```
    ios::sync_with_stdio(false);
    cin.tie(0);

    cin >> n;
    for(int i=1; i<=n; i++)
        cin >> a[i];
       for(int i=1; i<=n; i++)
           f[i] = {1, 1};//最长， 数量

       work(1, n);
      pair<int,int> ans = {0, 0};
      for(int i=1; i<=n; i++)
          ans = calc(ans, f[i]);

      cout << ans.second << ln;
}
```

5.2.2 树上点分治

*luogu*3806 给定一棵树，询问树上距离为 k 的点对是否存在

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef pair<int,int> pii;
#define pb push_back
#define ln '\n'

const int N = 1e4+5;
const int M = 1e7+5;

vector<pii> e[N];

int n, m;
int a[N];
int val[N], vis[N], sz[N], sum, rt;
//作为重心的价值， 以及该点是否删了， 子树的大小
int res[M];//该长度的链是否出现
int ans[N];//各个询问的答案是否出现

void getroot(int u, int fa){
        sz[u] = 1; val[u] = 0;
        for(auto [v, w]: e[u]){
                if(vis[v] || v == fa)
                        continue;
```

```cpp
                getroot(v, u);
                sz[u] = sz[u] + sz[v];
                val[u] = max(val[u], sz[v]);
        }
        val[u] = max(val[u], sum - sz[u]);
        if(!rt || val[u] < val[rt])
                rt = u;
}

vector<int> vec;

void dfs(int u, int fa, int dist){
        if(dist > 10000000) return;
        vec.pb(dist);
        for(auto [v, w]: e[u])
                if(!vis[v] && v!=fa)
                        dfs(v, u, dist+w);
}

void calc(int u, int fa){
        vector<int> p;
        for(auto [v, w]: e[u]){
                if(vis[v] || v == fa)
                        continue;
                vec.clear();
                dfs(v, u, w);
                for(auto len: vec){
                        for(int i=1; i<=m; i++)
                                if(a[i]-len>=0&&res[a[i] - len])
                                        ans[i] = 1;
                }
                for(auto len: vec){
                        res[len]=1;
                        p.pb(len);
                }
        }
        for(auto len: p)
                res[len] = 0;
}

void work(int u, int fa){
        vis[u] = 1;
        calc(u, fa);
        for(auto [v, w]: e[u]){
```

```
                if(vis[v] || v == fa)
                        continue;
                sum = sz[v]; rt = 0;
                getroot(v, u);//找子树内的重心
                work(rt, u);
        }
}

int main(){
    ios::sync_with_stdio(false);
    cin.tie(0);

    cin >> n >> m;
    for(int i=1; i<n; i++){
        int u, v, w;
        cin >> u >> v >> w;
        e[u].pb({v, w});
        e[v].pb({u, w});
    }

    for(int i=1; i<=m; i++)
        cin >> a[i];

    res[0] = 1;
    rt = 0; sum = n;
    getroot(1, -1);
    // cout << "root:= " << rt << ln;
    work(rt, -1);

    for(int i=1; i<=m; i++)
        cout << (ans[i]?"AYE":"NAY") << ln;

}
```

5.3 莫队

5.3.1 普通莫队

小 z 的袜子：n 双袜子，查询在 [l,r] 中抽到两只同色袜子的概率

```
#include<cstdio>
#include<cmath>
#include<algorithm>
#define ll long long
struct pos{int l,r,id;}q[50005];
int n,m,a[50005],vis[50005],be[50005];
```

```
ll ans[50005],Ans[50005];
inline ll gcd(ll a,ll b){ return b==0?a:gcd(b,a%b); }
bool cmp(pos a,pos b){return be[a.l]==be[b.l]?a.r<b.r:be[a.l]<be[b.l];}
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*10+ch-'0',ch=getchar();
        return x*f;
}
int main() {
        n=read(); m=read(); int siz=sqrt(n-1)+1;
        for(int i=1;i<=n;i++) a[i]=read(),be[i]=(i-1)/siz+1;//scanf("%d",&a[i]);
        for(int i=1;i<=m;i++) q[i].l=read(),q[i].r=read(),q[i].id=i;
        std::sort(q+1,q+m+1,cmp);
        int l=1,r=0;ll res=0;
        for(int i=1;i<=m;i++) {
                while(l<q[i].l) res=res-(1ll*(vis[a[l]]*(vis[a[l]]-1)/2-1ll*(vis[a[
                while(r>q[i].r) res=res-(1ll*(vis[a[r]]*(vis[a[r]]-1)/2-1ll*(vis[a[
                while(l>q[i].l) vis[a[--l]]++,res=res+(1ll*(vis[a[l]]*(vis[a[l]]-1)
                while(r<q[i].r) vis[a[++r]]++,res=res+(1ll*(vis[a[r]]*(vis[a[r]]-1)
                ans[q[i].id]=res; Ans[q[i].id]=1ll*(q[i].r-q[i].l+1)*(q[i].r-q[i].l
                if(l==r) ans[q[i].id]=0;
        }
        for(int i=1;i<=m;i++) {
                if(ans[i]==0) {puts("0/1");continue;}
                ll G=gcd(ans[i],Ans[i]);
                printf("%lld/%lld\n",ans[i]/G,Ans[i]/G);
        }
}
```

5.3.2 带修莫队

数颜色：查询 l..r 中有多少不同颜色的画笔，把第 i 支画笔换成颜色 p

```
#include<cstdio>
#include<cmath>
#include<algorithm>
using namespace std;
using std::sort;
const int N=5e4+5,M=1e6+5;
struct node{int l,r,id,cur;}q[N];
struct Node{int x,c,lst;}b[N];
int n,m,a[N],bl[N],siz,cnt,tot;
int l,r,ans,vis[M],Ans[N],lst[N];
inline void add(int x) {
```

```
	if(++vis[a[x]]==1)++ans;
}
inline void del(int x) {
	if(--vis[a[x]]==0)--ans;
}
inline void upd(int x,int v) {
	if(l<=x&&x<=r) del(x),a[x]=v,add(x);
	else	a[x]=v;
}
bool cmp(node a,node b) {
	if(bl[a.l]!=bl[b.l])
		return bl[a.l]<bl[b.l];
	if(bl[a.r]!=bl[b.r])
		return bl[a.r]<bl[b.r];
	return a.id<b.id;
}
int main() {
	scanf("%d%d",&n,&m);
	siz=pow(n,0.6666);
	for(int i=1;i<=n;i++)scanf("%d",&a[i]),lst[i]=a[i],bl[i]=(i-1)/siz+1;
	for(int i=1;i<=m;i++) {
		char s[10];
		scanf("%s",s);
		if(s[0]=='Q') {
			int l,r;
			scanf("%d%d",&l,&r);
			++cnt;
			q[cnt]=(node){l,r,cnt,tot};//q[i].cur表示位于第几次修改之后
		}else {
			int x,c;
			scanf("%d%d",&x,&c);
			b[++tot]=(Node){x,c,lst[x]}; lst[x]=c;
		}
	}
	l=1; r=0; ans=0;
	int now=0;
	sort(q+1,q+cnt+1,cmp);
	for(int i=1;i<=cnt;i++) {
		while(q[i].cur<now) {//多加了
			upd(b[now].x,b[now].lst);
			now--;
		}
		while(q[i].cur>now) {//少加了
			now++;
```

```
                        upd(b[now].x,b[now].c);
                }
                while(l<q[i].l) del(l++);
                while(l>q[i].l) add(--l);
                while(r<q[i].r) add(++r);
                while(r>q[i].r) del(r--);
                Ans[q[i].id]=ans;
        }
        for(int i=1;i<=cnt;i++)
                printf("%d\n",Ans[i]);
}
```

5.3.3 回滚莫队

只有单侧加没有减的莫队

```
//只有加法的莫队
//https://codeforces.com/contest/1514/problem/D
#include<bits/stdc++.h>
using namespace std;

#define ll long long

inline ll read() {
    ll x = 0, f = 1; char ch = getchar();
    for(; ch < '0' || ch>'9'; ch = getchar())
        if(ch == '-') f = -f;
    for(; ch >= '0' && ch <= '9'; ch = getchar())
        x = x * 10 + ch - '0';
    return x * f;
}

inline void chkmin( int &a, int b ) { if(a > b) a = b; }

inline void chkmax( int &a, int b ) { if(a < b) a = b; }

#define _ read()

#define ln endl

const int N=3e5+5;
int n, m, siz, v[N], bl[N], res[N];
int l, r, R[1005], ans[N], cnt[N], tot[N];
vector<int> mo;
struct Mo{int l, r, id;}a[N];
```

```
inline bool cmp(Mo a, Mo b){return bl[a.l]==bl[b.l]?a.r<b.r:bl[a.l]<bl[b.l];}

inline void solve(int i){
    while(l<R[i]+1) cnt[v[l]]--, l++;
    while(r>R[i]) cnt[v[r]]--, r--;
    while(r<R[i]) ++r, cnt[v[r]]++;
    int tmp=0, res=0;
    for(int j=0; j<mo.size(); j++){
        int x=mo[j];
        while(r<a[x].r){
            r++; cnt[v[r]]++;
            res=max(res, cnt[v[r]]);
        }
        tmp=res;
        while(l>a[x].l){
            --l; cnt[v[l]]++;
            tmp=max(tmp, cnt[v[l]]);
        }
        ans[a[x].id]=max(2*tmp-(a[x].r-a[x].l+1), 1);
        while(l<R[i]+1) cnt[v[l]]--, ++l;
    }
    mo.clear();
}

int main(){
    n=read(); m=read(); siz=sqrt(n);
    for(int i=1; i<=n; i++) {
        v[i]=read(), bl[i]=(i-1)/siz+1;
        R[bl[i]]=max(R[bl[i]], i);
    }
    for(int i=1; i<=m; i++)
        a[i].l=read(), a[i].r=read(), a[i].id=i;
    sort(a+1, a+m+1, cmp);
    l=1; r=0;
    int lst=0;
    for(int i=1; i<=m; i++){
        if(bl[a[i].l]==bl[a[i].r]){
            int tmp=0;
            for(int j=a[i].l; j<=a[i].r; j++){
                tot[v[j]]++;
                if(tot[v[j]]>tmp)
                    tmp=tot[v[j]];
            }
            ans[a[i].id]=max(2*tmp-(a[i].r-a[i].l+1), 1);
```

```
            for(int j=a[i].l; j<=a[i].r; j++)
                tot[v[j]]--;
        }else{
            if(bl[a[i].l]==lst);
            else if(mo.size()) solve(lst);
            mo.push_back(i); lst=bl[a[i].l];
        }
    }
    if(mo.size())
        solve(lst);
    for(int i=1; i<=m; i++)
        printf("%d\n", ans[i]);
}
```

5.4 悬线法

```
//51nod最大全1子矩阵
//悬线法， 实际上就是单调栈优化dp
int ans = 0;
for(int i=1; i<=n; i++){
    int top = 0;
    for(int j=1; j<=m; j++){
        a[i][j] = read();
        if(!a[i][j])
            U[i][j] = 0;
        else
            U[i][j] = U[i-1][j] + 1;
        while(top&&U[i][j]<=U[i][s[top]])
            top--;
        L[i][j] = s[top] + 1;
        s[++top] = j;
    }
    s[++top] = m+1;
    for(int j=m; j; j--){
        while(top&&U[i][j]<=U[i][s[top]])
            top--;
        R[i][j] = s[top] - 1;
        s[++top] = j;
    }
    for(int j=1; j<=m; j++)
        ans = max(ans, U[i][j]*(R[i][j]-L[i][j]+1));
}
printf("%d\n", ans);
```

5.5 n 数码问题

对于一个 $n \times m$ 的矩阵，有一个空格，问能否将 $n \times m - 1$ 个元素排序

记一个矩阵的权值为 $F(A)$ =(状态 A 忽略空格时求出的逆序对数量 + 状态 A 将空格移动到状态 B 所需的行数 *(A 的列数 +1))

如果 $F(A) = F(B)$ 那么两个矩阵可以相互转化

5.6 高精度板子

```
const int MAXN = 200;//字符大小， 没压位
struct BigInt
{
    int len, s[MAXN];
    BigInt ()  //初始化
    {
        memset(s, 0, sizeof(s));
        len = 1;
    }
    BigInt (int num) { *this = num; }
    BigInt (const char *num) { *this = num; }  //让this指针指向当前字符串
    BigInt operator = (const int num)
    {
        char s[MAXN];
        sprintf(s, "%d", num);  //sprintf函数将整型映到字符串中
        *this = s;
        return *this;  //再将字符串转到下面字符串转化的函数中
    }
    BigInt operator = (const char *num)
    {
        for(int i = 0; num[i] == '0'; num++) ;  //去前导0
        len = strlen(num);
        for(int i = 0; i < len; i++) s[i] = num[len-i-1] - '0'; //反着存
        return *this;
    }
    BigInt operator + (const BigInt &b) const //对应位相加， 最为简单
    {
        BigInt c;
        c.len = 0;
        for(int i = 0, g = 0; g || i < max(len, b.len); i++)
        {
            int x = g;
            if(i < len) x += s[i];
            if(i < b.len) x += b.s[i];
            c.s[c.len++] = x % 10;  //关于加法进位
            g = x / 10;
```

```
        }
        return c;
    }
    BigInt operator += (const BigInt &b)  //如上文所说，此类运算符皆如此重载
    {
        *this = *this + b;
        return *this;
    }
    void clean()  //由于接下来的运算不能确定结果的长度，先大而估之然后再查
    {
        while(len > 1 && !s[len-1]) len--;  //首位部分‘0’故删除该部分长度
    }
    BigInt operator * (const BigInt &b) //乘法重载在于列竖式，再将竖式中的数转为抽象
    {
        BigInt c;
        c.len = len + b.len;
        for(int i = 0; i < len; i++)
        {
            for(int j = 0; j < b.len; j++)
            {
                c.s[i+j] += s[i] * b.s[j];//不妨列个竖式看一看
            }
        }
        for(int i = 0; i < c.len; i++) //关于进位，与加法意同
        {
            c.s[i+1] += c.s[i]/10;
            c.s[i] %= 10;
        }
        c.clean();  //我们估的位数是a+b的长度和，但可能比它小（1*1 = 1）
        return c;
    }
    BigInt operator *= (const BigInt &b)
    {
        *this = *this * b;
        return *this;
    }
    BigInt operator - (const BigInt &b)  //对应位相减，加法的进位改为借1
    {  //不考虑负数
        BigInt c;
        c.len = 0;
        for(int i = 0, g = 0; i < len; i++)
        {
            int x = s[i] - g;
            if(i < b.len) x -= b.s[i];  //可能长度不等
```

```
                if(x >= 0) g = 0;  //是否向上移位借1
                else
                {
                    g = 1;
                    x += 10;
                }
                c.s[c.len++] = x;
            }
            c.clean();
            return c;
        }
        BigInt operator -= (const BigInt &b)
        {
            *this = *this - b;
            return *this;
        }
        BigInt operator / (const BigInt &b)  //运用除是减的本质，不停地减，直到小于被减
        {
            BigInt c, f = 0; //可能会在使用减法时出现高精度运算
            for(int i = len-1; i >= 0; i--)  //正常顺序，从最高位开始
            {
                f = f*10;  //上面位的剩余到下一位*10
                f.s[0] = s[i];  //加上当前位
                while(f >= b)
                {
                    f -= b;
                    c.s[i]++;
                }
            }
            c.len = len;  //估最长位
            c.clean();
            return c;
        }
        BigInt operator /= (const BigInt &b)
        {
            *this  = *this / b;
            return *this;
        }
        BigInt operator % (const BigInt &b)  //取模就是除完剩下的
        {
            BigInt r = *this / b;
            r = *this - r*b;
            r.clean();
            return r;
```

```
    }
    BigInt operator %= (const BigInt &b)
    {
        *this = *this % b;
        return *this;
    }
    bool operator < (const BigInt &b) //字符串比较原理
    {
        if(len != b.len) return len < b.len;
        for(int i = len-1; i != -1; i--)
        {
            if(s[i] != b.s[i]) return s[i] < b.s[i];
        }
        return false;
    }
    bool operator > (const BigInt &b)  //同理
    {
        if(len != b.len) return len > b.len;
        for(int i = len-1; i != -1; i--)
        {
            if(s[i] != b.s[i]) return s[i] > b.s[i];
        }
        return false;
    }
    bool operator == (const BigInt &b)
    {
        return !(*this > b) && !(*this < b);
    }
    bool operator != (const BigInt &b)
    {
        return !(*this == b);
    }
    bool operator <= (const BigInt &b)
    {
        return *this < b || *this == b;
    }
    bool operator >= (const BigInt &b)
    {
        return *this > b || *this == b;
    }
    string str() const  //将结果转化为字符串（用于输出）
    {
        string res = "";
        for(int i = 0; i < len; i++) res = char(s[i]+'0')+res;
```

```
            return res;
        }
};

istream& operator >> (istream &in, BigInt &x) //重载输入流
{
    string s;
    in >> s;
    x = s.c_str();   //string转化为char[]
    return in;
}

ostream& operator << (ostream &out, const BigInt &x)  //重载输出流
{
    out << x.str();
    return out;
}
```

5.7 整体二分

k 大数查询，1. 将 c 加入编号 [l,r] 的集合种，2. 查询 [l, r] 集合的并集中第 k 大的数是多少

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long
#define pb push_back
#define ln '\n'

inline ll read() {
    ll x = 0, f = 1; char ch = getchar();
    for(; ch < '0' || ch>'9'; ch = getchar())
        if(ch == '-') f = -f;
    for(; ch >= '0' && ch <= '9'; ch = getchar())
        x = x * 10 + ch - '0';
    return x * f;
}

const int N = 5e4+5;
struct node{int l, r; ll sum, lzy;}tree[N<<2];

void pushup(int rt){
        tree[rt].sum = tree[rt<<1].sum + tree[rt<<1|1].sum;
}

void pushdown(int rt){
```

```
        int l = tree[rt].l, r = tree[rt].r;
        if(tree[rt].lzy){
                int mid = l + r >> 1;
                tree[rt<<1].lzy += tree[rt].lzy;
                tree[rt<<1|1].lzy += tree[rt].lzy;
                tree[rt<<1].sum += 1ll * tree[rt].lzy * (mid - l + 1);
                tree[rt<<1|1].sum += 1ll * tree[rt].lzy * (r - mid);
                tree[rt].lzy = 0;
        }
}

void build(int l, int r, int rt){
        tree[rt].l = l; tree[rt].r = r;
        if(l == r)
                return;
        int mid=l+r>>1;
        build(l, mid, rt<<1);
        build(mid+1, r, rt<<1|1);
}

void modify(int L, int R, int v, int l, int r, int rt){
        if(L <= l && r <= R){
                tree[rt].sum += 1ll * (r - l + 1) * v;
                tree[rt].lzy += v;
                return;
        }
        int mid = l + r >> 1;
        pushdown(rt);
        if(L <= mid)
                modify(L, R, v, l, mid, rt<<1);
        if(R > mid)
                modify(L, R, v, mid+1, r, rt<<1|1);
        pushup(rt);
}

ll query(int L, int R, int l, int r, int rt){
        if(L <= l && r <= R)
                return tree[rt].sum;
        int mid = l + r >> 1;
        ll ans = 0;
        pushdown(rt);
        if(L<=mid)
                ans += query(L, R, l, mid, rt<<1);
        if(R>mid)
```

```
                ans += query(L, R, mid+1, r, rt<<1|1);
        return ans;
}

int n, m;
int opt[N], lef[N], rig[N], ans[N];
ll c[N];

void work(int l, int r, vector<int> now){
        if(l>r) return;
        if(l==r){
                for(auto i: now)
                        if(opt[i] == 2)
                                ans[i] = l;
                return;
        }
        int mid = l+r>>1;
        vector<int> L, R;
        for(auto i: now)
                if(opt[i] == 1){
                        if(c[i] > mid)
                                R.pb(i), modify(lef[i], rig[i], 1, 1, n, 1);
                        else
                                L.pb(i);
                } else {
                        ll tmp = query(lef[i], rig[i], 1, n, 1);
                        if(tmp >= c[i])//>mid的有>=k个， 说明答案在mid+1..r
                                R.pb(i);
                        else
                                c[i] -= tmp,
                                L.pb(i);
                }
//      now.clear();
        for(auto i: R)
                if(opt[i] == 1)
                        modify(lef[i], rig[i], -1, 1, n, 1);
        if(L.size())
                work(l, mid, L);
        if(R.size())
                work(mid+1, r, R);
        //对>=mid的修改+1, <的不用管-->没贡献
        //对于修改， 如果c>=mid
}

```

```
int main(){
    #ifndef ONLINE_JUDGE
        freopen("0.txt","w", stdout);
    #endif
    n=read(); m=read();
    build(1, n, 1);
    std::vector<int> now;
    for(int i=1; i<=m; i++){
        opt[i] = read();
        lef[i]=read();
        rig[i]=read();
        c[i]=read();
                if(lef[i] > rig[i]) swap(lef[i], rig[i]);
                now.pb(i);
        }
    work(0, n, now);
    for(int i=1; i<=m; i++)
        if(opt[i] == 2) printf("%d\n", ans[i]);
}
```

5.8 背包回退 (待写)