

Template

fblogy

September 19, 2018

目录

1	DP	1	5	Others	10
1.1	斜率优化	1	5.1	roman_numerals	10
2	DataStructure	1			
2.1	CDQ	1			
2.2	cartesian_tree	2			
2.3	fenwick_tree	2			
2.4	rmq	2			
2.5	trie	2			
3	Graph	3			
3.1	DMST	3			
3.2	StoerWagner_O(n3)	3			
3.3	StoerWagner_O(nmlog(m))	4			
3.4	max_clique_BK	4			
3.5	max_clique_fastest	4			
4	Math	5			
4.1	CRT	5			
4.2	Euler_power	5			
4.3	FFT	6			
4.4	FFTMOD	6			
4.5	NTT	7			
4.6	abel 变换	8			
4.7	bell	8			
4.8	lindstrom_gessel_viennot_lemma	8			
4.9	math_function	8			
4.10	min_25	8			
4.11	prime	9			

1 DP

1.1 斜率优化

```
const int N=400005;
const int mod=1e9+7;

ll a[N],f[N],s[N];
int n,t,l,r,pos;

struct node{
    ll x,y;
    node(ll x=0,ll y=0):x(x),y(y){}
} h[N];

ll cal(int x) {
    return f[x]-s[x]+a[x+1]*x;
}

bool check(const node &a,const node &b,const node &c) {
    return (c.y-b.y)*(b.x-a.x)>(b.y-a.y)*(c.x-b.x);
} // 判断斜率递增

bool check2(const node &a,const node &b,int w) {
    return (b.y-a.y)<(b.x-a.x)*w;
} // 判断后比前优

int main(){
    ios::sync_with_stdio(false);
    while (cin>>n>>t) {
        rep(i,1,n+1) cin>>a[i];
        sort(a+1,a+n+1);
        rep(i,1,n+1) s[i]=s[i-1]+a[i];
        l=1,r=0;
        rep(i,t,n+1) {
            if (i-t<t) f[i]=s[i]-a[i]*i;else {
                while (l<r && check2(h[l],h[l+1],i)) l++;
                f[i]=h[l].y+s[i]-h[l].x*i;
            }
            pos=i+1;if (pos<t) continue;
            node tmp(a[pos+1],cal(pos));
            while (l<r && !check(h[r-1],h[r],tmp)) r--;
            h[++r]=tmp;
        }
        cout<<f[n]<<endl;
    }
    return 0;
}
```

```
const int N = 200005;
const int mod = 1e9+7;

int p1, p2, pos, n, k, nn, ans[N];

struct node{
    int x, y, z, num, ans;
    bool operator == (const node & b) const{
        return x == b.x && y == b.y && z == b.z;
    }
} a[N], tmp[N];

template<class T>
struct Fenwick{
    static const int N = 2e5+7;
    int n;T a1[N],a2[N];
    void ini(int _n){
        fill_n(a1+1,n,_n,0);fill_n(a2+1,n,_n,0);
    }
    void add(T *a,int p,T d) { for(; p<=n; p+=p & -p) a[p]+=d; }
    void add(int l,int r,T d) {
        add(a1, l, d), add(a1, r + 1, -d);
        add(a2, l, d * (l - 1)), add(a2, r + 1, -d * r);
    }
    T sum(T *a,int p) { T r=0; for(; p>=1; p-=p & -p) r+=a[p]; return r; }
    T pre(int p) { return !p ? 0 : sum(a1, p) * p - sum(a2, p); }
    T qry(int l,int r) {return pre(r)-pre(l-1); }
};

Fenwick<int> fen;

bool cmp(node a, node b){
    if (a.x != b.x) return a.x < b.x;
    if (a.y != b.y) return a.y < b.y;
    return a.z < b.z;
}

bool cmp2(node a, node b){
    //if (a.y != b.y) return a.y < b.y;
    //return a.z < b.z;
    return a.y < b.y;
}

void CDQ(int l, int r){
    if (l == r) {
        a[l].ans = a[l].num - 1;
        return;
    }
    int mid = l + r >> 1;
    CDQ(l, mid);
    CDQ(mid+1, r);
    pos = 1;
    rep(i, mid+1, r+1) {
        while (pos <= mid && a[pos].y <= a[i].y) {
            fen.add(fen.a1, a[pos].z, a[pos].num);

```

2 DataStructure

2.1 CDQ

```

    return v[1];
}

```

2.3 fenwick_tree

```

// index : [1, n]
// time : nlogn
// support : segment add, sum
// !!!! : use before init()!
template<class T>
struct Fenwick{
    static const int N = 2e5+7;
    int n; T a1[N], a2[N];
    void ini(int _n){
        fill_n(a1+1, n=_n, 0); fill_n(a2+1, n=_n, 0);
    }
    void add(T *a, int p, T d) { for(; p<=n; p+=p & -p) a[p]+=d; }
    void add(int l, int r, T d) {
        add(a1, l, d), add(a1, r + 1, -d);
        add(a2, l, d * (1 - 1)), add(a2, r + 1, -d * r);
    }
    T sum(T *a, int p) { T r=0; for(; p>=1; p-=p & -p) r+=a[p]; return r; }
    T pre(int p) { return !p ? 0 : sum(a1, p) * p - sum(a2, p); }
    T qry(int l, int r) { return pre(r)-pre(l-1); }
};

```

2.4 rmq

```

// index : [0, n)
// limit: 2^M >= N
// !!!! : bud()
namespace RMQ {
    const int N = ::N, M = 22;
    int lg2[N];
    int st[N][M];
    void bud(int n, int a[]) {
        assert((1 <= M) > n);
        lg2[0] = -1; rep(i, 1, n + 1) lg2[i] = lg2[i >> 1] + 1;
        rep(i, 0, n) st[i][0] = a[i];
        rep(j, 1, M) rep(i, 0, n) {
            if (i + (1 <= j) > n) break;
            st[i][j] = max(st[i][j - 1], st[i + (1 <= j - 1)][j - 1]);
        }
    }
    int qry(int l, int r) {
        if (l > r) swap(l, r);
        int lv = lg2[r - l + 1];
        return max(st[l][lv], st[r - (1 <= lv) + 1][lv]);
    }
}

```

2.5 trie

```

    pos++;
}
a[i].ans += fen.sum(fen.a1, a[i].z);
}
rep(i, 1, pos) fen.add(fen.a1, a[i].z, -a[i].num);
p1 = 1; p2 = mid+1;
rep(i, 1, r+1){
    if (p1 > mid) {tmp[i] = a[p2]; p2++;}
    else if (p2 > r) {tmp[i] = a[p1]; p1++;}
    else if (a[p1].y <= a[p2].y) {tmp[i] = a[p1]; p1++;}
    else {tmp[i] = a[p2]; p2++;}
}
rep(i, 1, r+1) a[i] = tmp[i];
}

int main(){
    freopen("a.in", "r", stdin);
    ios::sync_with_stdio(0);
    cin.tie(0);
    //cout << setiosflags(ios::fixed);
    //cout << setprecision(2);
    cin >> n >> k;
    rep(i, 1, n+1) cin >> a[i].x >> a[i].y >> a[i].z;
    sort(a+1, a+n+1, cmp);
    nn = 0;
    rep(i, 1, n+1) {
        if (i > 1 && a[i] == a[i-1]) {
            a[nn].num++;
        } else {
            a[++nn] = a[i];
            a[nn].num = 1;
        }
    }
    fen.ini(N);
    CDQ(1, nn);
    rep(i, 1, nn+1) ans[a[i].ans] += a[i].num;
    rep(i, 0, n) cout << ans[i] << endl;
    return 0;
}

```

2.2 cartesian_tree

```

// desc : bud a cartesian tree from a[0] .. a[n - 1]
// time : O(N)
// !!!! : return rt, a[n] will be rewrite
int ls[N], rs[N];
int cartesian_tree(int a[], int n) {
    a[n] = INT_MAX;
    vi v(1, n);
    fill_n(ls, n, -1), fill_n(rs, n, -1);
    rep(i, 0, n) {
        while (a[v.back()] < a[i])
            ls[i] = v.back(), v.pop_back();
        v.pb(rs[v.back()] = i);
    }
}

```

```
ans+=in[i];int v=i;
while(vis[v] != i && id[v] == -1 && v!=root){
    vis[v] = i;v = pre[v];
}
if(v != root && id[v] == -1) {
    for(int u=pre[v];u != v;u = pre[u]) id[u] = cnt;
    id[v] = cnt++;
}
}
if(cnt == 0) break;
rep(i,0,n) if(id[i] == -1) id[i] = cnt++;
rep(i,0,m) {
    int v=e[i].v;
    e[i].u = id[e[i].u]; e[i].v = id[e[i].v];
    if(e[i].u != e[i].v) e[i].d -= in[v];
}
n=cnt;root=id[root];
}
return ans;
}
} tree;
```

3.2 StoerWagner_O(n3)

```
struct StoerWagner{
    static const int N = 305;
    static const int INF = 0x3f3f3f3f;;
    int n;
    int g[N][N], val[N];
    bool vis[N], use[N];
    void init(int _n) {
        n = _n;
        fill_n(use + 1, n, 0);
        rep(i, 1, n+1) fill_n(g[i] + 1, n, 0);
    }
    void add_edge(int u, int v, int w) {
        g[u][v] += w;
        g[v][u] += w;
    }
    void merge(int u, int v) {
        rep(i, 1, n+1) {
            g[v][i] += g[u][i];
            g[i][v] += g[i][u];
        }
        use[u] = 1;
    }
    int MinimumCutPhase(int cnt, int &s, int &t) {
        fill_n(val + 1, n, 0);
        fill_n(vis + 1, n, 0);
        t = 1;
        while (--cnt) {
            vis[s = t] = 1;
            rep(i, 1, n+1) if (!vis[i] && !use[i]) val[i] += g[t][i];
            int ma = 0;
            rep(i, 1, n+1) if (!vis[i] && !use[i] && val[i] >= ma) {
```

```
// 'a' .. 'z'
// !!!! : ini()
struct Trie {
    static const int N = ::N;
    int son[N][26], _;
    int ne() { fill_n(son[_], 26, -1); return _++; }
    void ini() { _ = 0; ne(); }
    void insert(const string &s) {
        int u = 0;
        for (auto ch : s) {
            int &v = son[u][ch - 'a'];
            if (!v) v = ne();
            u = v;
        }
    }
    void comb(int a, int b) { // a into b
        //TODO : something Like sz[b] += sz[a];
        rep(i, 0, 26) if (~son[a][i]) {
            int &as = son[a][i];
            int &bs = son[b][i];
            if (!bs) bs = ne();
            comb(as, bs);
        }
    }
} tree;
```

3 Graph

3.1 DMST

```
struct edge{int u,v,d};
struct DMST{
    static const int N = 105 , M = N * N , inf = 2e9;
    edge e[M];int n,m,vis[N],pre[N],id[N];int in[N];
    void ini(int n) {this->n=n,m=0;}
    void addedge(int u,int v,int d) {e[m++]=edge({u,v,d});}
    int run(int root){
        int ans=0;
        while(1){
            rep(i,0,n) in[i]=inf;
            rep(i,0,m){
                int u=e[i].u , v=e[i].v;
                if(e[i].d < in[v] && u != v){
                    in[v] = e[i].d , pre[v] = u;
                }
            }
            rep(i,0,n) {
                if(i == root) continue; if(in[i] == inf) return -1;
            }
            int cnt = 0;in[root] = 0;
            memset(id,-1,sizeof(*id)*n);
            memset(vis,-1,sizeof(*vis)*n);
            rep(i,0,n){
```

```
        if (!vis[v]) q.push(mp(val[v] += data[p], v));
    }
    while (!q.empty() && (vis[q.top().se] || val[q.top().se] != q.top().fi)) {
        q.pop();
    }
    if (q.empty()) return 0;
    t = q.top().se; q.pop();
    return val[t];
}
int solve() {
    int res = INF;
    for (int i = n, s, t; i > 1; --i) {
        res = min(res, MinimumCutPhase(i, s, t));
        if (res == 0) break;
        merge(s, t);
    }
    return res;
}
} SW;
```

3.3 StoerWagner_O(nmlog(m))

```
struct StoerWagner {
    static const int N = 3005, M = 100005 * 2, INF = 0x3f3f3f3f;
    int head[N], val[N], e, n;
    int to[M], ne[M], data[M];
    bool vis[N];
    int fa[N], link[N];
    void init(int _n) {
        n = _n;
        fill_n(head + 1, n, -1);
        fill_n(link + 1, n, -1);
        rep(i, 1, n+1) fa[i] = i;
        e = 0;
    }
    void add_edge(int u, int v, int w) {
        to[e] = v; data[e] = w; ne[e] = head[u]; head[u] = e++;
        to[e] = u; data[e] = w; ne[e] = head[v]; head[v] = e++;
    }
    int findset(int u) {
        return u == fa[u] ? u : fa[u] = findset(fa[u]);
    }
    void merge(int u, int v) {
        int p = u;
        while (~link[p]) p = link[p];
        link[p] = v;
        fa[v] = u;
    }
    int MinimumCutPhase(int cnt, int &s, int &t) {
        fill_n(val + 1, n, 0);
        fill_n(vis + 1, n, 0);
        priority_queue<pii> q;
        t = 1;
        while (--cnt) {
            vis[s = t] = 1;
            for (int u = s; ~u; u = link[u]) {
                for (int p = head[u]; ~p; p = ne[p]) {
                    int v = findset(to[p]);
```

3.4 max_clique_BK

```
//g[i][i] should be 0
//index [0..N)
//O(n ^ 3)
typedef unsigned long long T;
struct BK {
    static const int N = 100; T g[N];
    inline int ctz(T s) { return s ? __builtin_ctzll(s) : 64; }
    int n, ans;
    void ini(int _n) { per(i, 0, n = _n) g[i] = (1ull << n) - 1 - (1ull << i); }
    void gao(T cur, T can, T ban) {
        if (!can && !ban) { ans = max(ans, __builtin_popcountll(cur)); return; }
        if (!can) return;
        int piv = ctz(can | ban), ret = 0;
        T z = can & ~g[piv];
        for (int u = ctz(z); u < n; u += ctz(z >> (u + 1)) + 1) {
            gao(cur | (1ull << u), can & g[u], ban & g[u]);
            can ^= 1ull << u, ban |= 1ull << u;
        }
    }
    int run() { gao(ans = 0, (1ull << n) - 1, 0); return ans; }
} bk;
```

3.5 max_clique_fastest

```
const int N = 130;
typedef bool BB[N];
struct MaxClique {
    const BB *e; int pk, lv; db Tlimit;
    struct ve { int i, d; ve(int i): i(i), d(0) {} }; //ve : Vertex
    struct sc { int a, b; sc(      ): a(0), b(0) {} }; //sc : StepCount
```

4 Math

4.1 CRT

```
const int N=1e5+7;
int n;
ll a[N], mod[N], M, R;

void exgcd(ll a, ll b, ll &x, ll &y){
    if(b == 0){
        x = 1; y = 0;
        return;
    }
    exgcd(b, a % b, y, x);
    y -= a / b * x;
}

ll Inv(ll a, ll mod){
    ll x = 0, y = 0;
    exgcd(a, mod, x, y);
    x %= mod;
    if (x < 0) x += mod;
    return x;
}

ll CRT(int n, ll *a, ll *mod){
    M = mod[1], R = a[1];
    rep(i, 2, n+1) {
        ll g=__gcd(M, mod[i]);
        ll inv = Inv(M / g, mod[i] / g);
        if ((a[i] - R) % g) return -1; // 无解
        R += inv * ((a[i] - R) / g) % (mod[i] / g) * M;
        M = M / g * mod[i];
        R = (R % M + M) % M; // 可能为 0 看是否需要是正整数
    }
    return R;
}
```

4.2 Euler_power

```
int phi(int n) {
    if (M.count(n)==1) return M[n];
    int r=n, nn=n;
    for(int i=2; i*i<=n; i++) if (n%i==0){
        r=r/i*(i-1);
        while (n%i==0) n/=i;
    }
    if (n>1) r=r/n*(n-1);
    M[nn]=r;
    return r;
}

ll Euler_qpow(ll a, ll b, ll mod) {
    ll res=1; bool ok=(b>0 && a%mod);
}
```

```
typedef vector<ve> ves; ves V; //ves: Vertices
typedef vector<int> cc; cc Q, QMAX; //cc : ColorClass
vector<cc> C;
vector<sc> S;
MaxcLique(BB *conn, int sz, const db tt = 0.025): pk(0), lv(1), Tlimit(tt) {
    rep(i, 0, sz) V.pb(ve(i)); e = conn;
    C.resize(sz + 1);
    S.resize(sz + 1);
}

static bool desc_deg(const ve &a, const ve &b) { return a.d > b.d; }
void ini_col(ves &v) { per(i, 0, sz(v)) v[i].d = min(i, v[0].d) + 1; }
void set_deg(ves &v) { rep(i, 0, sz(v)){v[i].d = 0; rep(j, 0, sz(v)) v[i].d += e[v[i].i][v[j].i]; } }
void deg_sort(ves &R) { set_deg(R); sort(all(R), desc_deg); }
bool cut1(int pi, cc &va) { rep(i, 0, sz(va)) if (e[pi][va[i]]) return true;
return false; }
void cut2(ves &va, ves &vb) { rep(i, 0, sz(va) - 1) if (e[va.back().i][va[i].i]) vb.pb(va[i].i); }
void co_sort(ves &R) {
    int j = 0, maxno = 1, min_k = max(sz(QMAX) - sz(Q) + 1, 1);
    rep(i, 1, 3) C[i].clear();
    rep(i, 0, sz(R)) {
        int pi = R[i].i, k = 1;
        while (cut1(pi, C[k])) k++;
        if (k > maxno) C[maxno = k] + 1].clear(); C[k].pb(pi);
        if (k < min_k) R[j++].i = pi;
    }
    if (j > 0) R[j - 1].d = 0;
    rep(k, min_k, maxno + 1)
        rep(i, 0, sz(C[k]))
            R[j].i = C[k][i], R[j++].d = k;
}

void expand_dyn(ves &R) { // expand_dyn
    S[lv].a += S[lv - 1].a - S[lv].b;
    S[lv].b = S[lv - 1].a;
    for (; sz(R); Q.pop_back(), R.pop_back()) {
        if (sz(Q) + R.back().d <= sz(QMAX)) return;
        Q.pb(R.back().i);
        ves Rp; cut2(R, Rp);
        if (sz(Rp)) {
            if ((db) S[lv].a / ++pk < Tlimit) deg_sort(Rp);
            co_sort(Rp); S[lv++].a++;
            expand_dyn(Rp); --lv;
        } else if (sz(Q) > sz(QMAX)) QMAX = Q;
    }
}

void mcqdyn(int *mxc, int &sz) { // mcqdyn(int maxcLique, int &sz)
    set_deg(V); sort(all(V), desc_deg);
    ini_col(V); rep(i, 0, sz(V) + 1) S[i].a = S[i].b = 0;
    expand_dyn(V); per(i, 0, sz(QMAX)) mxc[i] = QMAX[i];
    sz = sz(QMAX);
}
};
```

```

while (b>0) {
    if (b&1) {
        res=res*a;
        ok!=(res>=mod);
        res%=mod;
    }
    a=a*a;
    ok!=(b>1 && a>=mod);
    a%=mod;
    b>>=1;
}
return res+mod*ok;
}

ll work(int l,int r,int mod) {
    if (mod==1) return 1;
    if (l==r) return a[l];
    return Euler_qpow(a[l],work(l+1,r,phi(mod)),mod);
}

```

4.3 FFT

```

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

```

```

const int MAXN=1<<16;
const double pi=acos(-1);
int N,na,nb,rev[MAXN*2];
struct vir{

```

```

    double re,im;
    vir(double r=0.0,double i=0.0) {re=r,im=i;}
    void print() {printf("%lf %lf\n",re,im);}
} a[MAXN*2],b[MAXN*2],w[2][MAXN*2];

```

```

vir operator +(const vir&A,const vir&B) {return vir(A.re+B.re,A.im+B.im);}
vir operator -(const vir&A,const vir&B) {return vir(A.re-B.re,A.im-B.im);}
vir operator *(const vir&A,const vir&B) {return vir(A.re*B.re-A.im*B.im,A.re*B.im+A.im*B
.re);}

```

```

void FFT(vir*a,int f){
    vir x,y;
    for (int i=0; i<N; i++) if (i<rev[i]) swap(a[i],a[rev[i]]);
    for (int i=1; i<N; i<=1)
        for (int j=0,t=N/(i<1); j<N; j+=i<1)
            for (int k=0,l=0; k<i; k++,l+=t) x=w[f][l]*a[j+k+i],y=x,a[j+k]=y+x,a[j+k+i]
            =y-x;
    if (f) for (int i=0; i<N; i++) a[i].re/=N;
}

```

```

void work(){
    for (int i=0; i<N; i++){
        int x=i,y=0;
        for (int k=1; k<N; x>=1,k<=1) (y<=1)|=x&1;
        rev[i]=y;
    }
}

```

4.4 FFTMOD

```

const int MOD =1e9+7;
const int MAXN=1<<17;
const double PI = acos(-1);

```

```

int N,L,MASK,na,nb;
int a[MAXN],b[MAXN];

```

```

struct vir
{

```

```

    double re,im;
    vir(double r=0.0,double i=0.0) {re=r,im=i;}
    void print() {printf("%lf %lf\n",re,im);}
};

```

```

vir operator +(const vir&A,const vir&B) {return vir(A.re+B.re,A.im+B.im);}
vir operator -(const vir&A,const vir&B) {return vir(A.re-B.re,A.im-B.im);}
vir operator *(const vir&A,const vir&B) {return vir(A.re*B.re-A.im*B.im,A.re*B.im+A.im*B
.re);}

```

```

vir conj(vir a){
    return vir(a.re,-a.im);
}

```

```

vir w[MAXN];
void FFTinit(){
    for (int i = 0; i < N; ++i) {
        w[i] = vir(cos(2 * i * PI / N), sin(2 * i * PI / N));
    }
}

```

```

void FFT(vir p[], int n) {
    for (int i = 1, j = 0; i < n - 1; ++i) {

```

```

init();
mul();
rep(1,0,N) printf("%d\n",a[i]);
}

```

4.5 NTT

```

const int MAXN=1<<17;
const int G=3;
const int P=1004535809; //P=C*2^k+1
int N,na,nb,a[MAXN*2],b[MAXN*2],w[2][MAXN*2],rev[MAXN*2];

ll Pow(ll a,int b)
{
    ll c=1;
    for (;b; b>>=1,a=a*a%p) if (b&1) c=c*a%p;
    return c;
}

void FFT(int*a,int f)
{
    rep(i,0,N) if (i<rev[i]) swap(a[i],a[rev[i]]);
    for (int i=1; i<N; i<=1)
        for (int j=0,t=N/(i<=1); j<N; j+=i<=1)
            for (int k=0,l=0,x,y; k<i; k++,l+=t)
                x=(ll)w[f][l]*a[j+k+1]%P,y=a[j+k],a[j+k]=(y+x)%P,a[j+k+1]=(y-x+p)%P;
    if (f) for (int i=0,x=Pow(N,P-2); i<N; i++) a[i]=(ll)a[i]*x%p;
}

void work()
{
    rep(i,0,N)
    {
        int x=i,y=0;
        for (int k=1; k<N; x>>=1,k<=1) (y<=1)?x&1;
        rev[i]=y;
    }
    w[0][0]=w[1][0]=1;
    for (int i=1,x=Pow(G,(P-1)/N),y=Pow(x,P-2); i<N; i++)
        w[0][i]=(ll)x*w[0][i-1]%P,w[1][i]=(ll)y*w[1][i-1]%P;
}

void init()
{
    scanf("%d",&na); for (int i=0; i<na; i++) scanf("%d",&a[i]);
    scanf("%d",&nb); for (int i=0; i<nb; i++) scanf("%d",&b[i]);
    for (N=1; N<=na|N<=nb; N<=1); N<=1;
}

void doit()
{
    work(),FFT(a,0),FFT(b,0);
    rep(1,0,N) a[i]=(ll)a[i]*b[i]%P;
    FFT(a,1);
    rep(1,0,N) printf("%d\n",a[i]);
}

```

```

for (int s = n; j ^ s >= 1, ~j & s);
if (i < j) {
    swap(p[i], p[j]);
}
}
for (int d = 0; (1 <= d) < n; ++d) {
    int m = 1 <= d, m2 = m * 2, rm = n >> (d + 1);
    for (int i = 0; i < n; i += m2) {
        for (int j = 0; j < m; ++j) {
            vir &p1 = p[i + j + m], &p2 = p[i + j];
            vir t = w[rm * j] * p1;
            p1 = p2 - t;
            p2 = p2 + t;
        }
    }
}
vir A[MAXN], B[MAXN], C[MAXN], D[MAXN];

void init(){
    L=0;
    scanf("%d",&na);rep(i,0,na) scanf("%d",&a[i]);
    scanf("%d",&nb);rep(i,0,nb) scanf("%d",&b[i]);
    for (N=1; N<=na || N<=nb; N<=1) L++; N<=1;
    MASK=(1<=L)-1;
    FFTInit();
    for (int i = 0; i < N; ++i) {
        A[i] = vir(a[i] >> L, a[i] & MASK);
        B[i] = vir(b[i] >> L, b[i] & MASK);
    }
}

void mul() {
    FFT(A, N), FFT(B, N);
    for (int i = 0; i < N; ++i) {
        int j = (N - i) % N;
        vir da = (A[i] - conj(A[j])) * vir(0, -0.5),
            db = (A[i] + conj(A[j])) * vir(0.5, 0),
            dc = (B[i] - conj(B[j])) * vir(0, -0.5),
            dd = (B[i] + conj(B[j])) * vir(0.5, 0);
        C[j] = da * dd + da * dc * vir(0, 1);
        D[j] = db * dd + db * dc * vir(0, 1);
    }
    FFT(C, N), FFT(D, N);
    for (int i = 0; i < N; ++i) {
        ll da = (ll)(C[i].im / N + 0.5) % MOD,
            db = (ll)(C[i].re / N + 0.5) % MOD,
            dc = (ll)(D[i].im / N + 0.5) % MOD,
            dd = (ll)(D[i].re / N + 0.5) % MOD;
        a[i] = ((dd <= (L * 2)) + ((db + dc) <= L) + da) % MOD;
    }
}

int main(){

```


4.9 math_function

```
const int N = 1e6 + 7;
int n, m, f[N], g[N], h[N], phi[N], u[N], p[N]; //f[n] 为 n 的最小质因子, g[n]=f[n]^k,
phi[n] 为欧拉函数, u[n] 为莫比乌斯函数, h[n] 为一般积性函数

void prime(int n) {
    u[1]=phi[1]=1,h[1]=(0); //1 的时候特判
    rep(i, 2, n+1) {
        if (!f[i]) {
            p[++m]=i;
            f[i] = g[i] = i;
            phi[i] = i - 1;
            u[i] = -1;
            h[i] = (0);
        } // 质数的时候特判
        for (int j = 1, k; j <= m && p[j] <= f[i] && i * p[j] <= n; j++){
            if (p[j] < f[i]) {
                g[k] = p[j];
                phi[k] = phi[i] * phi[p[j]];
                u[k] = u[i] * u[p[j]];
                h[k] = h[i] * h[p[j]];
            }
            else {
                g[k] = g[i] * p[j];
                phi[k] = phi[i] * p[j];
                u[k] = 0;
                h[k] = h[i] / g[i] * (0);
            } // 质数次幂特判
        }
        //phi[i*p[j]]=phi[i]*phi[j]*(p[j]<f[i]?phi[p[j]]:p[j]);
        //u[i*p[j]]=u[i]*u[j]*(p[j]<f[i]?u[p[j]]:0);
    }
    /*
    *phi[i*j]=phi[i]*phi[j] (gcd(i,j)=1)
    phi[i]^j (j|i)
    u[i*j]=u[i]*u[j] (gcd(i,j)=1)
    0 (j|i)
    */
}
```

4.10 min_25

```
struct Min_25{
    // F(i) 要拆成多个完全积性函数的和
    static const int N = 1e6 + 7;
    int Sqr, m, p[N], id1[N], id2[N], tot, cntp;
    ll g[N], sp[N], h[N], n, w[N];
    bool isp[N];
    // f(p) = p ^ k
    ll f(int p) { return 1;}

    // 要求的积性函数 F(p ^ e)
    ll F(int p, int e) { return e == 1 ? -1 : 0;}
```

```
}
int main(){
    init();
    doit();
}
```

4.6 abel 变换

```
## 变换abel
$sum_{i=1}^n a_i * b_i = sum_{i=1}^{n-1} a_i * (b_{i+1} - b_i) + a_n * b_n$
$A_i = sum_{j=1}^i a_j$
```

4.7 bell

```
// desc : 0^k + 1^k + 2^k + .. + (n-1)^k
// time-init : O(n^2)
// time-cal : k + log
namespace Bell {
    const int N = 1000;
    int C[N][N], B[N];
    void ini() {
        rep(i, 0, N) C[i][0] = 1;
        rep(i, 0, N) rep(j, 1, i + 1) C[i][j] = add(C[i - 1][j - 1], C[i - 1][j]);
        B[0] = 1;
        rep(i, 1, N) {
            B[i] = 0;
            rep(j, 0, i) B[i] = add(B[i], MOD - mul(C[i + 1][j], B[j]));
            B[i] = mul(B[i], qpow(C[i + 1][i], MOD - 2)) % MOD;
        }
    }
    int cal(int n, int k) {
        int sum = 0;
        rep(i, 0, k + 1) sum = add(sum, mul(C[k + 1][i], mul(B[i], qpow(n, k + 1 - i))));
        return mul(sum, qpow(k + 1, MOD - 2));
    }
};
```

4.8 lindstrom_gessel_viennot_lemma

```
/*
 * 对于一张无边权的 DAG 图，给定 n 个起点和对应的 n 个终点，这 n 条不相交路径的方案数为矩阵
 * e(a1,b1),e(a1,b2),...e(a1,bn)
 * e(a2,b1),e(a2,b2),...e(a2,bn)
 * .....
 * e(an,b1),e(an,b2),...e(an,bn)
 * 的行列式。
 * 即 M[i][j]=e(ai,bj)
 * e(a,b) 为 a 到 b 的路径方案数
 */
```

```
// 假设都是质数的完全积性函数前缀和去掉 f(1)
ll calc(ll n) { return n - 1; }

void prime(int n){
    cntp = 0; isp[1] = 1;
    rep(i, 2, n+1) {
        if(!isp[i]) p[++cntp] = i;
        for(int j = 1; j <= cntp && i * p[j] <= n; j++){
            isp[i * p[j]] = 1;
            if(i % p[j] == 0) break;
        }
    }
    rep(i, 1, cntp+1) sp[i] = sp[i-1] + f(p[i]);
    p[++cntp] = INT_MAX;
}

ll S(ll x, int y){
    if(x <= 1 || p[y] > x) return 0;
    int k = (x <= Sqr ? id1[x] : id2[n/x]);
    ll ret = -(g[k] - sp[y-1]); // 质数的答案
    for(int i = y; i <= tot && 1ll * p[i] * p[i] <= x; i++){
        ll t1 = p[i], t2 = 1ll * p[i] * p[i];
        for(int e = 1; t2 <= x; e++, t1 = t2, t2 *= p[i]) {
            if (F(p[i], e)) ret += S(x / t1, i + 1) * F(p[i], e);
            ret += F(p[i], e + 1); // 合数的答案
        }
    }
    return ret;
}

ll solve(ll _n) {
    n = _n; if (n == 0) return 0;
    m = 0; Sqr = sqrt(n);
    tot = upper_bound(p + 1, p + cntp + 1, Sqr) - (p + 1);
    for(ll i = 1, j; i <= n; i = j + 1){
        j = n / (n / i);
        w[++m] = n / i;
        g[m] = calc(w[m]);
        w[m] <= Sqr ? id1[w[m]] = m : id2[j] = m;
    }
    rep(j, 1, tot + 1)
        for(int i = 1; i <= m && 1ll * p[j] * p[j] <= w[i]; i++){
            ll t = w[i] / p[j];
            int k = t <= Sqr ? id1[t] : id2[n / t];
            g[i] -= f(p[j]) * (g[k] - sp[j - 1]);
        }
    return S(n, 1) + 1;
}
} _u;
}
```

4.11 prime

```
// time : O(n)
// low[] : optional
const int N = 1e6 + 6;
```

```
int low[N], cntp, p[N];
bool isp[N];

void getprime() {
    fill_n(isp + 2, N - 2, 1);
    rep(i, 2, N) {
        if (isp[i]) p[++cntp] = i;
        for (int j = 0; j < cntp && p[j] * i < N; j++) {
            // low[p[j] * i] = p[j];
            isp[p[j] * i] = 0;
            if (i % p[j] == 0) break;
        }
    }
}

// 优化版欧拉筛法 bitset 需要 O2
const int N = 3e7 + 6;
const int M = 2e6 + 6;
// int low[N],
bitset<N> isp;
int cntp, p[M];

void getprime() {
    cntp = 2; p[0] = 2; p[1] = 3;
    for (int i = 5, k = 1; i <= N; (k & 1) == 1 ? i += 2 : i += 4, k++) {
        if (!isp[k]) p[++cntp] = i;
        for (int j = 2; j < cntp && p[j] * i < N; j++) {
            // low[p[j] * i] = p[j];
            isp[p[j] * i / 3] = 1;
            if (i % p[j] == 0) break;
        }
    }
}

// 优化埃氏筛法空间最小可以不存质数
const int N = 3e8 + 6;
const int M = 2e7 + 6;
int cntp, p[M];
bitset<N / 3 + 1> bit;

void getprime(int n){
    int i, j;
    cntp = 2; p[0] = 2; p[1] = 3;
    for(i = 5, j = 1; i * i <= n; (j & 1) == 1 ? i += 2 : i += 4, j++){
        if(bit[j] == 0) {
            p[++cntp] = i;
            for(int j = i * i; j <= n; j += i)
                if(j % 2 != 0 && j % 3 != 0) bit[j / 3] = 1;
        }
    }
    for( ; i <= n; (j & 1) == 1 ? i += 2 : i += 4, j++) if(bit[j] == 0) p[++cntp] = i;
}
```

5 Others

5.1 roman_numerals

```
const int rom[30] = {
    3000, 2000, 1000, 900, 800, 700, 600, 500, 400, 300, 200, 100,
    90, 80, 70, 60, 50, 40, 30, 20, 10,
    9, 8, 7, 6, 5, 4, 3, 2, 1
};
string smb[30]={
    "MMM", "MM", "M",
    "CM", "DCCC", "DCC", "DC", "D", "CD", "CCC", "CC", "C",
    "XC", "LXXX", "LXX", "LX", "L", "XL", "XXX", "XX", "X",
    "IX", "VIII", "VII", "VI", "V", "IV", "III", "II", "I"
};
string toRoman(ll d) {
    string r;
    rep(i, 0, 30) if (d >= rom[i]) d -= rom[i], r += smb[i];
    return r;
}
```