ACM Templates

XTS

2019 年 10 月 6 日

目录

1 图论

1.1 dijstra

```
struct edge {
    int v, d;
};

struct node {
    int d, u;

    bool operator<(const node &x) const {
        return d > x.d;
    }
};

vector <edge> e[maxn];

void add(int u, int v, int d) {
    e[u].push_back(edge{v, d});
}


int dij(int s, int nd) {
    priority_queue <node> q;
    for (int i = 0; i <= N; i++)d[i] = inf;
    d[s] = 0;
    pre[s] = -1;
    q.push(node{0, s});
    while (!q.empty()) {
        node x = q.top();
        q.pop();
        int u = x.u;
        if (vis[u])continue;
        vis[u] = 1;
        for (auto E:e[u]) {
            if (d[E.v] > d[u] + E.d) {
                d[E.v] = d[u] + E.d;
                pre[E.v] = u;
```

```
                q.push(node{d[E.v], E.v});
            }
        }
    }
}
```

1.2 倍增 LCA

```
const int maxn = 5e5 + 7;
const int maxp = 21; //最大高度 log 1e5->18 1e4->15

int n, m, s;

int fa[maxn], dep[maxn], pa[maxn][maxp];

vector<int> G[maxn];

void dfs(int u, int fa)
{
    pa[u][0] = fa;
    dep[u] = dep[fa] + 1;
    for (int i = 1; i < maxp; i++)
        pa[u][i] = pa[pa[u][i - 1]][i - 1];
    for (int &v : G[u])
    {
        if (v == fa)
            continue;
        dfs(v, u);
    }
}

int lca(int u, int v)
{
    if (dep[u] < dep[v])
        swap(u, v);
    int t = dep[u] - dep[v];
    for (int i = 0; i < maxp; i++)
        if (t & (1 << i))
            u = pa[u][i];
```

```
    for (int i = maxp - 1; i >= 0; i--)
    {
        int uu = pa[u][i], vv = pa[v][i];
        if (uu != vv)
        {
            u = uu;
            v = vv;
        }
    }
    return u == v ? u : pa[u][0];
}
```

1.3 树上路径交

```
//x-y 和 xx-yy 的路径交即为 lca(x,xx),lca(x,yy),lca(y,xx),lca(y,yy) 中最深两点的路径
//注意判断是否有交
int intersection(int x, int y, int xx, int yy) {
    int t[4] = {lca(x, xx), lca(x, yy), lca(y, xx), lca(y, yy)};
    sort(t, t + 4);
    int r = lca(x, y), rr = lca(xx, yy);
    if (dep[t[0]] < min(dep[r], dep[rr]) || dep[t[2]] < max(dep[r], dep[rr]))
        return 0;
    int tt = lca(t[2], t[3]);
    int ret = 1 + dep[t[2]] + dep[t[3]] - dep[tt] * 2;
    return ret;
}
```

1.4 树链剖分

```
//调用 predfs(root,1) dfs(root,root) 多组数据把 clk 赋 0
const int N = 1e5 + 7;
vector<int> G[N];
int n, m, s, mod;

int fa[N], dep[N], idx[N], out[N], ridx[N], sz[N], son[N], top[N], v[N], clk;

void predfs(int u, int d)
{
    dep[u] = d;
```

```
    sz[u] = 1;
    int &maxs = son[u] = -1;
    for (int &v : G[u])
    {
        if (v == fa[u])
            continue;
        fa[v] = u;
        predfs(v, d + 1);
        sz[u] += sz[v];
        if (maxs == -1 || sz[v] > sz[maxs])
            maxs = v;
    }
}
void dfs(int u, int tp)
{
    top[u] = tp;
    idx[u] = ++clk;
    ridx[clk] = u;
    if (son[u] != -1)
        dfs(son[u], tp);
    for (int &v : G[u])
        if (v != fa[u] && v != son[u])
            dfs(v, v);
    out[u] = clk;
}

void go(int u, int v)
{
    int uu = top[u], vv = top[v];
    while (uu != vv)
    {
        if (dep[uu] < dep[vv])
        {
            swap(uu, vv);
            swap(u, v);
        }
        //do sth. in [idx[uu],idx[u]]
        u = fa[uu];
```

```
        uu = top[u];
    }
    if (dep[u] < dep[v])
        swap(u, v);
    //do sth. in [idx[v],idx[u]]
}

int up(int u, int d)
{
    while (d)
    {
        if (dep[u] - dep[top[u]] < d)
        {
            d -= dep[u] - dep[top[u]];
            u = top[u];
        }
        else
            return ridx[idx[u] - d];
        u = fa[u];
        --d;
    }
    return u;
}
int finds(int u, int rt)
{ // 找 u 在 rt 的哪个儿子的子树中
    while (top[u] != top[rt])
    {
        u = top[u];
        if (fa[u] == rt)
            return u;
        u = fa[u];
    }
    return ridx[idx[rt] + 1];
}
```

1.5 点分治

```
//求树上长度小于等于 k 的路径数
```

```
int n, m, k, cnt;

long long ans;

struct edge
{
    int to, d;
};

bool vis[maxn];

int h[maxn];

vector<edge> G[maxn];

int q[maxn], fa[maxn], sz[maxn], mx[maxn], dep[maxn];

void get_dep(int u, int fa, int d)
{
    dep[u] = d;
    h[++cnt] = d;
    for (int i = 0; i < G[u].size(); i++)
    {
        edge e = G[u][i];
        int v = e.to;
        if (vis[v] || v == fa)
            continue;
        get_dep(v, u, d + e.d);
    }
}

int get_rt(int u)
{
    int p = 0, cur = -1;
    q[p++] = u;
    fa[u] = -1;
    while (++cur < p)
    {
```

```
        u = q[cur];
        mx[u] = 0;
        sz[u] = 1;
        for (int i = 0; i < G[u].size(); i++)
        {
            edge e = G[u][i];
            if (!vis[e.to] && e.to != fa[u])
                fa[q[p++]] = e.to] = u;
        }
    }
    for (int i = p - 1; i >= 0; i--)
    {
        u = q[i];
        mx[u] = max(mx[u], p - sz[u]);
        if (mx[u] * 2 <= p)
            return u;
        sz[fa[u]] += sz[u];
        mx[fa[u]] = max(mx[fa[u]], sz[u]);
    }
}

long long cal(int u, int x)
{
    long long res = 0;
    cnt = 0;
    get_dep(u, -1, 0);
    sort(h + 1, h + 1 + cnt);
    int r = cnt;
    for (int l = 1; l < r; l++)
    {
        while (h[l] + h[r] > x && r > l)
            r--;
        res += r - l;
    }
    return res;
}

void dfs(int u)
```

```
{
    u = get_rt(u);
    vis[u] = true;
    ans += cal(u, k);
    for (int i = 0; i < G[u].size(); i++)
    {
        edge e = G[u][i];
        int v = e.to;
        if (vis[v])
            continue;
        ans -= cal(v, k - e.d * 2);
        dfs(v);
    }
}
```

1.6 强连通分量

```
const int maxn = 2e4 + 7;

int n, m, top;

vector<int> G[maxn];

int st[maxn], pre[maxn], low[maxn], sccno[maxn], clk, cnt;

void dfs(int u)
{
    pre[u] = low[u] = ++clk;
    st[++top] = u;
    for (int &v : G[u])
    {
        if (!pre[v])
        {
            dfs(v);
            low[u] = min(low[u], low[v]);
        }
        else if (!sccno[v])
            low[u] = min(low[u], pre[v]);
    }
```

```
    if (low[u] == pre[u])
    {
        cnt++;
        for (;;)
        {
            int x = st[top--];
            sccno[x] = cnt;
            if (u == x)
                break;
        }
    }
}

void find_scc(int n)
{
    clk = cnt = top = 0;
    memset(sccno, 0, sizeof(sccno));
    memset(pre, 0, sizeof(pre));
    for (int i = 1; i <= n; i++)
    {
        if (!pre[i])
            dfs(i);
    }
}
```

1.7 双连通分量

```
//
const int maxn = 1e3 + 7;

struct edge {
    int u, v;
};

int n, m, cnt;

stack <edge> st;

int pre[maxn], iscut[maxn], bccno[maxn], clk, bcc_cnt;
```

```
vector<int> G[maxn], bcc[maxn];

void init() {
    for (int i = 1; i <= n; i++)G[i].clear();
    n = 0;
}

int dfs(int u, int fa) {
    int lowu = pre[u] = ++clk;
    int child = 0;
    for (int i = 0; i < G[u].size(); i++) {
        int v = G[u][i];
        edge e = edge{u, v};
        if (!pre[v]) {
            st.push(e);
            child++;
            int lowv = dfs(v, u);
            lowu = min(lowu, lowv);
            if (lowv >= pre[u]) {
                iscut[u] = 1;
                bcc[++bcc_cnt].clear();
                for (;;) {
                    edge x = st.top();
                    st.pop();
                    if (bccno[x.u] != bcc_cnt) {
                        bcc[bcc_cnt].push_back(x.u);
                        bccno[x.u] = bcc_cnt;
                    }
                    if (bccno[x.v] != bcc_cnt) {
                        bcc[bcc_cnt].push_back(x.v);
                        bccno[x.v] = bcc_cnt;
                    }
                    if (x.u == u && x.v == v)break;
                }
            }
        } else if (pre[v] < pre[u] && v != fa) {
            st.push(e);
```

```
                lowu = min(lowu, pre[v]);
            }
        }
    if (fa < 0 && child == 1)iscut[u] = 0;
    return lowu;
}

void find_bcc(int n) {
    memset(pre, 0, sizeof(pre));
    memset(iscut, 0, sizeof(iscut));
    memset(bccno, 0, sizeof(bccno));
    clk = bcc_cnt = 0;
    for (int i = 1; i <= n; i++) {
        if (!pre[i])dfs(i, -1);
    }
}
```

1.8 仙人掌求环

```
void dfs(int u, int fa) {
    vis[u] = 1;
    f[u] = fa;
    dep[u] = dep[fa] + 1;
    for (auto v:e[u]) {
        if (v == fa)continue;
        if (vis[v]) {
            if (dep[u] < dep[v])continue;
            int cnt = 1;
            int x = u;
            while (x != v) {
                cnt++;
                x = f[x];
            }
            // cnt: 环的长度
        } else dfs(v, u);
    }
}
```

2 字符串

2.1 回文树

```
//num[i] 表示以节点 i 表示的最长回文串的最右端点为回文串结尾的回文串个数
#include<bits/stdc++.h>
using namespace std;
const int maxa=2e5+20,cha=30,MAXN=1e6+20;//maxa 字符串最长长度, cha 字符集大小
typedef long long ll;
struct PalindromicTree
{
    int next[maxa][cha],fail[maxa],cnt[maxa],len[maxa],num[maxa];
    int tot,s[maxa],p,last;
    int newnode(int l)
    {
        for(int i=0;i<cha;i++) next[p][i]=0;
        len[p]=l;
        cnt[p]=num[p]=0;
        return p++;
    }
    void init()
    {
        tot=p=last=0;
        s[0]=-1,fail[0]=1;//0 为存储 偶数回文串 树根节点, 1 为存储 奇数回文串 树根节点
        newnode(0);/*p==0, 偶数回文串树根节点编号为 0,len 值为 0*/
        newnode(-1);/*p==1, 奇数回文串树根节点编号为 1,len 值为-1*/
    }
    int getfail(int x)
    {
        while(s[tot-len[x]-1]!=s[tot]) x=fail[x];
        return x;
    }
    void insert(char c)
    {
        c-='a';
        s[++tot]=c;
        int cur=getfail(last);
        if(!next[cur][c])
        {
```

```
            int now=newnode(len[cur]+2);
            fail[now]=next[getfail(fail[cur])][c];
            next[cur][c]=now;
            num[now]=num[fail[now]]+1;
        }
        last=next[cur][c];
        cnt[last]++;
    }
    void makecnt() //统计本质相同的回文串的出现次数
    {
        for(int i=p-1;i>=2;i--) cnt[fail[i]]+=cnt[i];//根节点 cnt 无意义, i>=2 即可
    }
}pt;
char str[maxa];
ll ans=0;
int main()
{
    scanf("%s",str);
    int len=strlen(str);
    pt.init();
    for(int i=0;i<len;i++) pt.insert(str[i]);
    pt.makecnt();
    cout<<(pt.p-1)/2<<endl;
    return 0;
}
```

2.2 广义后缀自动机

```
//cnt[i] 表示节点 i 表示的本质不同的串的个数(建树时求出的不是完全的,最后 count() 函数
↪ 跑一遍以后才是正确的)
#include<bits/stdc++.h>
using namespace std;
const int maxa=2e5+20,cha=30,MAXN=1e6+20;//maxa 字符串最长长度, cha 字符集大小
typedef long long ll;
struct SAM{
    int cnt,las,root;
    int ch[MAXN][30],fa[MAXN],len[MAXN];

    inline void init(){root = las = ++cnt;}
```

```
    inline void insert(int x){
        int p = las;
        int np = las = ++cnt;
        len[np] = len[p] + 1;
        for(;p && !ch[p][x];p = fa[p]) ch[p][x] = np;
        if(!p) fa[np] = root;
        else{
            int q = ch[p][x];
            if(len[q] == len[p] + 1) fa[np] = q;
            else {
                int nq = ++cnt;
                fa[nq] = fa[q];fa[q] = fa[np] = nq;
                memcpy(ch[nq],ch[q],sizeof(ch[nq]));
                len[nq] = len[p] + 1;
                for(;p && ch[p][x] == q;p = fa[p]) ch[p][x] = nq;
            }
        }
    }
    inline long long sum(int x){
        return len[x] - len[fa[x]];
    }
}T;
char str[maxa];
ll ans=0;
int main()
{
    scanf("%s",str);
    int len=strlen(str);

    ll ans=0;
    T.init();
    for(int i = 0;i < len;i++) T.insert(str[i] - 'a');
    T.las = T.root;
    for(int i = len-1;i >=0;i--) T.insert(str[i] - 'a');
    for(int i = 0;i <= T.cnt;i++) ans = ans + T.sum(i);

    return 0;
```

```
}
```

2.3 kmp

```
// 数组为 0 - n-1 时 , nxt[1 - n] 是有意义的
// nxt[i] 为满足 x[i-z...i-1]=x[0...z-1] 的最大 z 值
// len = i-nxt[i] 为 s[0 - i-1] 的最小循环节（可能是 aaabbbaaab 这种不完整的循环）如果
↪   i mod len = 0 才一定是 aaabbbaaabbb 这种完整循环串
void kmp_pre(char nxt[], int m, char t[]) {
    int i, j;
    j = nxt[0] = -1;
    i = 0;
    while (i < m) {
        while (-1 != j && t[i] != t[j])j = nxt[j];
        nxt[++i] = ++j;
    }
}

//用 t 匹配 s position 中存所有成功匹配的起点
void kmp(int nxt[], int n, int s[], int m, int t[]) {
    for (int i = 0, j = 0; i < n; i++) {
        if (j < n && s[i] == t[j]) {
            j++;
        } else {
            while (j > 0) {
                j = nxt[j];
                if (s[i] == t[j]) {
                    j++;
                    break;
                }
            }
        }
        if (j == m)position.push_back(i - m + 1);
    }
}
```

3 数据结构

4 数学

4.1 带预处理 BGSG

```
#include <bits/stdc++.h>
#define LL long long
using namespace std;
unordered_map<LL, LL> mp;
int loop, up;
LL n, x0, a, b, p, v;
LL _pow(LL a, LL b, LL Mod)
{
    LL ret = 1;
    LL mul = a % Mod;
    while (b > 0)
    {
        if (b & 1)
            ret = ret * mul % Mod;
        mul = mul * mul % Mod;
        b = b >> 1;
    }
    return ret;
}
LL inv(LL x, LL P)
{
    return _pow(x, P - 2, P);
}
void pre_BSGS(int p, int a)
{ //预处理一部分 a^y
    mp.clear();
    up = ceil(pow(p, 2.0 / 3));

    int t = 1;
    for (int i = 0; i <= up; i++)
    {
        if (i == up)
            loop = t;
        mp[t] = i;
        t = 1LL * t * a % p;
```

```
    }
}
LL BSGS(LL B, LL N, LL P)
{ //B^ans=N(%p)
    int m = ceil(pow(p, 1.0 / 3));
    int obj = inv(N, P);
    for (int i = 1; i <= m; i++)
    {
        obj = 1LL * obj * loop % P;
        if (mp.count(obj))
        {
            return 1LL * i * up - mp[obj];
        }
    }
    return -1;
}

int main()
{
    int T;
    scanf("%d", &T);
    while (T--)
    {
        scanf("%lld%lld%lld%lld%lld", &n, &x0, &a, &b, &p);
        int Q;
        scanf("%d", &Q);
        pre_BSGS(p, a);
        while (Q--)
        {
            scanf("%lld", &v);
            LL bi = b + ((a - 1) * v % p);
            bi = bi % p;
            bi = bi * inv((b + x0 * (a - 1)) % p, p) % p;
            LL ans = BSGS(a, bi, p); //a^ans=bi(%p)
            if (ans >= n)
                ans = -1;
            printf("%lld\n", ans);
        }
```

```
    }
    return 0;
}
```

4.2 十进制矩阵快速幂

```
#include <bits/stdc++.h>
using namespace std;
long long Mod;
struct Matrix
{
    long long a[2][2];
    Matrix operator*(const Matrix &b) const
    {
        Matrix tmp;
        for (int i = 0; i < 2; i++)
            for (int j = 0; j < 2; j++)
                tmp.a[i][j] = 0;
        for (int i = 0; i < 2; i++)
            for (int j = 0; j < 2; j++)
                for (int k = 0; k < 2; k++)
                {
                    tmp.a[i][j] += a[i][k] * b.a[k][j];
                    tmp.a[i][j] %= Mod;
                }
        return tmp;
    }
};
Matrix Matrixpow(Matrix x, int b)
{
    Matrix ans1;
    ans1.a[0][0] = 1;
    ans1.a[1][1] = 1;
    ans1.a[1][0] = 0;
    ans1.a[0][1] = 0;
    Matrix ret = x;
    while (b)
    {
        if (b & 1)
```

```
            ans1 = ans1 * ret;
        ret = ret * ret;
        b = b / 2;
    }
    return ans1;
}
int main()
{
    ios::sync_with_stdio(false);
    long long x0, x1, a, b;
    cin >> x0 >> x1 >> a >> b;
    string s;
    cin >> s;
    cin >> Mod;
    int len = s.length();
    Matrix ans;
    ans.a[0][0] = 1;
    ans.a[1][1] = 1;
    ans.a[1][0] = 0;
    ans.a[0][1] = 0;
    Matrix mult;
    mult.a[0][0] = a;
    mult.a[0][1] = b;
    mult.a[1][0] = 1;
    mult.a[1][1] = 0;
    for (int i = len - 1; i >= 0; i--)
    {
        ans = ans * Matrixpow(mult, s[i] - '0');
        mult = Matrixpow(mult, 10);
    }
    long long tot = ans.a[1][1] * x0 % Mod + ans.a[1][0] * x1 % Mod;
    tot = tot % Mod;
    cout << tot << endl;
    return 0;
}
```

4.3 exlucas

```
#include <bits/stdc++.h>
#define rep(ii, a, b) for (int ii = a; ii <= b; ii++)
#define ll long long
using namespace std;
ll qpow(ll a, ll b, ll nmod)
{
    ll res = 1;
    while (b)
    {
        if (b & 1)
            res = res * a % nmod;
        a = a * a % nmod;
        b >>= 1;
    }
    return res;
}

void exgcd(ll a, ll b, ll &x, ll &y, ll &d)
{
    b ? (exgcd(b, a % b, y, x, d), y -= x * (a / b)) : (x = 1, y = 0, d = a);
}
ll inv(ll a, ll m)
{
    ll x, y, d;
    exgcd(a, m, x, y, d);
    return x >= 0 ? x % m : x % m + m;
}
ll crt(ll a, ll m, ll m0)
{ //m0 | m; (m0,m/m0)=1
    return m / m0 * inv(m / m0, m0) % m * a % m;
}

ll fact(ll n, ll p, ll pk)
{ //(n!/p^x)%(p^k)
    if (n <= 1)
        return 1;
    ll ans = 1, tmp = n % pk;
```

```
    rep(i, 1, pk)
    {
        if (i % p)
            ans = ans * i % pk;
    }
    ans = qpow(ans, n / pk, pk);
    rep(i, 1, tmp)
    {
        if (i % p)
            ans = ans * i % pk;
    }
    return ans * fact(n / p, p, pk) % pk;
}

ll c(ll n, ll m, ll p, ll pk)
{ //c(n,m)%(p^k)
    ll sum = 0;
    for (ll i = n; i; i /= p)
        sum += i / p;
    for (ll i = m; i; i /= p)
        sum -= i / p;
    for (ll i = n - m; i; i /= p)
        sum -= i / p;
    return qpow(p, sum, pk) * fact(n, p, pk) % pk * inv(fact(m, p, pk), pk) % pk *
     ↪  inv(fact(n - m, p, pk), pk) % pk;
}

ll fac[40][2], pf; //0 p;  1 pk
void getfac(ll n)
{
    ll tmp = sqrt(n);
    rep(i, 2, tmp)
    {
        if (n % i == 0)
        {
            fac[++pf][0] = i, fac[pf][1] = 1;
            while (n % i == 0)
                n /= i, fac[pf][1] *= i;
```

```
        }
    }
    if (n > 1)
        fac[++pf][0] = n, fac[pf][1] = n;
}
ll exlucas(ll n, ll m, ll p)
{
    ll ans = 0;
    getfac(p);
    rep(i, 1, pf)
    {
        ans = (ans + crt(c(n, m, fac[i][0], fac[i][1]), p, fac[i][1])) % p;
    }
    return ans;
}
int main()
{
    ll n, m, p;
    cin >> n >> m >> p;
    cout << exlucas(n, m, p) << endl;
    return 0;
}
```

4.4 BM

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
using VI = vector<int64_t>;
class Linear_Seq
{
public:
    static const int N = 50020; ///多项式系数最大值
    int64_t res[N], c[N], md[N], COEF[N] /**COEF 是多项式系数 */, Mod;
    vector<int> Md;
    inline static int64_t gcdEx(int64_t a, int64_t b, int64_t &x, int64_t &y)
    {
        if (!b)
        {
```

```
        x = 1;
        y = 0;
        return a;
    }
    int64_t d = gcdEx(b, a % b, y, x);
    y -= (a / b) * x;
    return d;
}
static int64_t Inv(int64_t a, int64_t Mod)
{
    int64_t x, y;
    return gcdEx(a, Mod, x, y) == 1 ? (x % Mod + Mod) % Mod : -1;
};

inline void mul(int64_t *a, int64_t *b, int k)
{ ///下边的线性齐次递推用的.
    fill(c, c + 2 * k, 0);
    for (int i(0); i < k; ++i)
        if (a[i])
            for (int j(0); j < k; ++j)
                c[i + j] = (c[i + j] + a[i] * b[j]) % Mod;
    for (int i(2 * k - 1); i >= k; --i)
        if (c[i])
            for (size_t j(0); j < Md.size(); ++j)
                c[i - k + Md[j]] = (c[i - k + Md[j]] - c[i] * md[Md[j]]) % Mod;
    copy(c, c + k, a);
}
VI BM(VI s)
{ ///BM 算法求模数是质数的递推式子的通项公式，可以单独用
    VI C(1, 1), B(1, 1);
    int L(0), m(1), b(1);
    for (size_t n(0); n < s.size(); ++n)
    {
        int64_t d(0);
        for (int i(0); i <= L; ++i)
            d = (d + (int64_t)C[i] * s[n - i]) % Mod;
        if (!d)
            ++m;
```

```
        else
        {
            VI T(C);
            int64_t c(Mod - d * Inv(b, Mod) % Mod);
            while (C.size() < B.size() + m)
                C.push_back(0);
            for (size_t i(0); i < B.size(); ++i)
                C[i + m] = (C[i + m] + c * B[i]) % Mod;
            if (2 * L <= (int)n)
            {
                L = n + 1 - L;
                B = T;
                b = d;
                m = 1;
            }
            else
                ++m;
        }
    }
    return C;
}
int solve(int64_t n, VI A, VI B)
{ //线性齐次递推:A 系数,B 初值 B[n]=A[0]*B[n-1]+...
    ///这里可以可以单独用, 给出递推系数和前几项代替矩阵快速幂求递推式第 n 项.
    int64_t ans(0), cnt(0);
    int k(A.size());
    for (int i(0); i < k; ++i)
        md[k - i - 1] = -A[i];
    md[k] = 1;
    Md.clear();
    for (int i(0); i < k; ++i)
    {
        res[i] = 0;
        if (md[i])
            Md.push_back(i);
    }
    res[0] = 1;
    while ((1LL << cnt) <= n)
```

```
        ++cnt;
    for (int p(cnt); ~p; --p)
    {
        mul(res, res, k);
        if ((n >> p) & 1)
        {
            copy(res, res + k, res + 1);
            res[0] = 0;
            for (size_t j(0); j < Md.size(); ++j)
                res[Md[j]] = (res[Md[j]] - res[k] * md[Md[j]]) % Mod;
        }
    }
    for (int i(0); i < k; ++i)
        ans = (ans + res[i] * B[i]) % Mod;
    return ans + (ans < 0 ? Mod : 0);
}
inline static void extand(VI &a, size_t d, int64_t value = 0)
{
    if (d <= a.size())
        return;
    a.resize(d, value);
}
static int64_t CRT(const VI &c, const VI &m)
{ ///中国剩余定理合并
    int n(c.size());
    int64_t M(1), ans(0);
    for (int i = 0; i < n; ++i)
        M *= m[i];
    for (int i = 0; i < n; ++i)
    {
        int64_t x, y, tM(M / m[i]);
        gcdEx(tM, m[i], x, y);
        ans = (ans + tM * x * c[i] % M) % M;
    }
    return (ans + M) % M;
}

static VI ReedsSloane(const VI &s, int64_t Mod)
```

```
{ ///求模数不是质数的递推式系数
    auto L = [](const VI &a, const VI &b) {
        int da = (a.size() > 1 || (a.size() == 1 && a[0])) ? a.size() - 1 :
        ↪  -1000;
        int db = (b.size() > 1 || (b.size() == 1 && b[0])) ? b.size() - 1 :
        ↪  -1000;
        return max(da, db + 1);
    };
    auto prime_power = [&](const VI &s, int64_t Mod, int64_t p, int64_t e) {
        vector<VI> a(e), b(e), an(e), bn(e), ao(e), bo(e);
        VI t(e), u(e), r(e), to(e, 1), uo(e), pw(e + 1);
        ;
        pw[0] = 1;
        for (int i(pw[0] = 1); i <= e; ++i)
            pw[i] = pw[i - 1] * p;
        for (int64_t i(0); i < e; ++i)
        {
            a[i] = {pw[i]};
            an[i] = {pw[i]};
            b[i] = {0};
            bn[i] = {s[0] * pw[i] % Mod};
            t[i] = s[0] * pw[i] % Mod;
            if (!t[i])
            {
                t[i] = 1;
                u[i] = e;
            }
            else
                for (u[i] = 0; t[i] % p == 0; t[i] /= p, ++u[i])
                    ;
        }
        for (size_t k(1); k < s.size(); ++k)
        {
            for (int g(0); g < e; ++g)
            {
                if (L(an[g], bn[g]) > L(a[g], b[g]))
                {
                    int id(e - 1 - u[g]);
```

```
            ao[g] = a[id];
            bo[g] = b[id];
            to[g] = t[id];
            uo[g] = u[id];
            r[g] = k - 1;
        }
    }
    a = an;
    b = bn;
    for (int o(0); o < e; ++o)
    {
        int64_t d(0);
        for (size_t i(0); i < a[o].size() && i <= k; ++i)
            d = (d + a[o][i] * s[k - i]) % Mod;
        if (d == 0)
        {
            t[o] = 1;
            u[o] = e;
        }
        else
        {
            for (u[o] = 0, t[o] = d; !(t[o] % p); t[o] /= p, ++u[o])
                ;
            int g(e - 1 - u[o]);
            if (!L(a[g], b[g]))
            {
                extand(bn[o], k + 1);
                bn[o][k] = (bn[o][k] + d) % Mod;
            }
            else
            {
                int64_t coef = t[o] * Inv(to[g], Mod) % Mod * pw[u[o] -
                ↪ uo[g]] % Mod;
                int m(k - r[g]);
                extand(an[o], ao[g].size() + m);
                extand(bn[o], bo[g].size() + m);
                auto fun = [&](vector<VI> &vn, vector<VI> &vo, bool f) {
                    for (size_t i(0); i < vo[g].size(); ++i)
```

```
                    {
                        vn[o][i + m] -= coef * vo[g][i] % Mod;
                        if (vn[o][i + m] < 0)
                            vn[o][i + m] += Mod * (f ? 1 : -1);
                    }
                    while (vn[o].size() && !vn[o].back())
                        vn[o].pop_back();
                };
                fun(an, ao, 1);
                fun(bn, bo, -1);
            }
        }
    }
}
return make_pair(an[0], bn[0]);
};
vector<tuple<int64_t, int64_t, int>> fac;
for (int64_t i(2); i * i <= Mod; ++i)
    if (!(Mod % i))
    {
        int64_t cnt(0), pw(1);
        while (!(Mod % i))
        {
            Mod /= i;
            ++cnt;
            pw *= i;
        }
        fac.emplace_back(pw, i, cnt);
    }
if (Mod > 1)
    fac.emplace_back(Mod, Mod, 1);
vector<VI> as;
size_t n = 0;
for (auto &&x : fac)
{
    int64_t Mod, p, e;
    VI a, b;
    std::tie(Mod, p, e) = x;
```

```
            auto ss = s;
            for (auto &&x : ss)
                x %= Mod;
            std::tie(a, b) = prime_power(ss, Mod, p, e);
            as.emplace_back(a);
            n = max(n, a.size());
        }
        VI a(n), c(as.size()), m(as.size());
        for (size_t i(0); i < n; ++i)
        {
            for (size_t j(0); j < as.size(); ++j)
            {
                m[j] = std::get<0>(fac[j]);
                c[j] = i < as[j].size() ? as[j][i] : 0;
            }
            a[i] = CRT(c, m);
        }
        return a;
    }
    int64_t solve(VI a, int64_t n, int64_t Mod, bool prime = true)
    {
        VI c;
        this->Mod = Mod;
        if (prime)
            c = BM(a); ///如果已经知道系数了，直接输入到 c 就行了，不用调用 BM().
        else
            c = ReedsSloane(a, Mod);
        c.erase(c.begin());
        for (size_t i(0); i < c.size(); ++i)
            c[i] = (Mod - c[i]) % Mod;
        return solve(n, c, VI(a.begin(), a.begin() + c.size()));
    }
};
ll _pow(ll a, ll b, ll Mod)
{
    ll ret = 1LL % Mod;
    ll mul = a % Mod;
    while (b)
```

```
    {
        if (b & 1)
            ret = ret * mul % Mod;
        mul = mul * mul % Mod;
        b >>= 1;
    }
    return ret;
};
ll f[2020], sum[2020];

int main()
{
    VI vec;
    vec.clear();
    for (int i = 0; i <= 2000; i++)
        vec.push_back(sum[i]);
    Linear_Seq bmi;
    printf("%lld\n", bmi.solve(vec, n, P, false));
}
```

4.5 杜教筛

```
#include <bits/stdc++.h>
#define LL long long
const int P = 1e9 + 7;
using namespace std;
LL _pow(LL a, LL b)
{
    LL ret = 1;
    LL mul = a % P;
    while (b)
    {
        if (b & 1)
            ret = ret * mul % P;
        mul = mul * mul % P;
        b = b >> 1;
    }
    return ret;
}
```

```cpp
namespace dujiao
{
unordered_map<LL, LL> mapi;
const int M = 1e6 + 7;
LL sum[M] = {0};
int cnt = 0;
LL prim[M], phi[M];
bool vis[M];
void init(int maxn)
{
    mapi.clear();
    memset(vis, 0, sizeof(vis));
    phi[1] = 1;
    for (int i = 2; i <= maxn; i++)
    {
        if (!vis[i])
        {
            prim[++cnt] = i;
            phi[i] = i - 1;
        }
        for (int j = 1; j <= cnt && prim[j] * i <= maxn; j++)
        {
            vis[i * prim[j]] = 1;
            if (i % prim[j] == 0)
            {
                phi[i * prim[j]] = phi[i] * prim[j];
                break;
            }
            else
                phi[i * prim[j]] = phi[i] * (prim[j] - 1);
        }
    }
    for (int i = 1; i <= maxn; i++)
        sum[i] = (sum[i - 1] + (i * phi[i]) % P) % P;
}
inline LL s_fg(LL n)
{
    LL ans = n * (n + 1) % P;
```

```
    ans = ans * (2 * n + 1) % P;
    ans = ans * _pow(6, P - 2) % P;
    return ans;
}
LL go(LL n)
{
    if (n < M)
        return sum[n];
    if (mapi[n])
        return mapi[n];
    LL ans = s_fg(n);
    for (int i = 2; i <= n;)
    {
        int j = n / (n / i);
        LL t1 = 1LL * j * (j + 1) / 2 - 1LL * i * (i - 1) / 2;
        t1 = t1 % P;
        ans = (ans - t1 * go(n / i) % P + P) % P;
        i = j + 1;
    }
    return mapi[n] = ans;
}
LL solve(LL n)
{
    return go(n);
}
} // namespace dujiao
int main()
{
    int T;
    scanf("%d", &T);
    int M = 1e6 + 7;
    dujiao::init(M);
    while (T--)
    {
        LL n;
        int a, b;
        scanf("%lld%d%d", &n, &a, &b);
        LL ans = dujiao::go(n);
```

```
        ans = (ans - 1 + P) % P;
        ans = ans * _pow(2, P - 2) % P;
        printf("%lld\n", ans);
    }
    return 0;
}
```

4.6 线性基

```
#include <bits/stdc++.h>
#define ll long long
const int Maxn = 5e4 + 20;
using namespace std;
ll a[Maxn][50];
int cnt[Maxn];
struct L_B
{
    long long d[35]; //若为 LL, 开 63++;
    L_B()
    {
        memset(d, 0, sizeof(d));
    }
    void init()
    {
        memset(d, 0, sizeof(d));
    }
    void insert(ll x)
    {
        for (int i = 32; i >= 0; i--)
        { //or 64
            if (x & (1ll << i))
            {
                if (!d[i])
                {
                    d[i] = x;
                    break;
                }
                x ^= d[i];
            }
```

```
    }
}
bool check(ll x)
{
    for (int i = 32; i >= 0; i--)
    {
        if (x & (1ll << i))
            x ^= d[i];
    }
    return x == 0ll;
}
L_B merge(L_B k)
{
    L_B res, tmp = k;
    res.init();
    for (int i = 0; i <= 32; i++)
    {
        ll x = d[i], y = 0;
        bool flag = false;
        for (int j = 32; j >= 0; j--)
        {
            if ((x >> j) & 1)
            {
                if (k.d[j])
                    x ^= k.d[j], y ^= tmp.d[j];
                else
                {
                    k.d[j] = x;
                    tmp.d[j] = y;
                    flag = true;
                    break;
                }
            }
        }
        if (!flag)
            res.d[i] = y;
    }
    return res;
```

```
    }
} tree[(Maxn) << 2];

void build(int o, int l, int r)
{
    tree[o].init();
    if (l == r)
    {
        for (int j = 1; j <= cnt[l]; j++)
            tree[o].insert(a[l][j]);
    }
    else
    {
        int mid = (l + r) / 2;
        build(o << 1, l, mid);
        build(o << 1 | 1, mid + 1, r);
        tree[o] = tree[o << 1].merge(tree[o << 1 | 1]);
    }
}
bool query(int o, int l, int r, int ql, int qr, ll x)
{
    if (ql <= l && qr >= r)
        return tree[o].check(x);
    int mid = (l + r) / 2;
    bool ans = true;
    if (ql <= mid)
        ans = ans & query(o << 1, l, mid, ql, qr, x);
    if (qr >= mid + 1)
        ans = ans & query(o << 1 | 1, mid + 1, r, ql, qr, x);
    return ans;
}
int main()
{
    int n, m;
    cin >> n >> m;
    for (int i = 1; i <= n; i++)
    {
        cin >> cnt[i];
```

```
        for (int j = 1; j <= cnt[i]; j++)
            cin >> a[i][j];
    }
    build(1, 1, n);
    while (m--)
    {
        int l, r;
        ll x;
        cin >> l >> r >> x;
        if (query(1, 1, n, l, r, x))
        {
            puts("YES");
        }
        else
            puts("NO");
    }
    return 0;
}
```

4.7 原根

原根

模m有原根的充要条件：

$$m=2,4,p^a,2p^a(\text{其中}p\text{是奇素数})$$

一个数m如果有原根，则其原根个数为phi(phi(m))；特别地，对素数有phi(p)=p-1。

假设g是奇素数p的一个原根，则

$$g^1,g^2,\ldots,g^{p-1}$$

在模p意义下两两不同，且结果恰好为1~p-1，由此可以定义“离散对数”，与连续数学中的对数有异曲同工之妙。

离散对数又叫做“指标”，有指标法则：

$$I(ab)\equiv I(a)+I(b)(\%(p-1))$$

$$I(a^k)\equiv k*I(a)(\%(p-1))$$

由此可以把乘法转化为加法。

2017 revenge

题意：给你 n（2e6）个数，和一个数r，问你有多少种方案，使得你取出某个子集，能够让它们的乘积 mod 2017等于r。

题解：2017有5这个原根，可以使用离散对数（指标）的思想把乘法转化成加法。

$$\text{考虑}dp[i][j]\text{表示到第}i\text{个点后取模结果为}j\text{的数的个数}$$

$$f(i,j)=f(i-1,j)+f(i-1,(j-I(a(i))\ );I(i)\text{规定为}i\text{的指标}$$

```
#include <bits/stdc++.h>
using namespace std;
int I[2020];
int main(){
    bitset<2016>f;
    int xx=1;
    for (int i=0;i<2016;i++){
        I[xx]=i;
        xx=(xx*5)%2017;
    }
    int n,r;
    while(~scanf("%d%d",&n,&r)){
        f.reset();
        f.set(I[1]);
        for (int i=1;i<=n;i++){
            int x;
            cin>>x;
            f^=((f<<I[x]) ^ (f>>(2016-I[x])));
        }
        cout<<f[I[r]]<<endl;
    }
    return 0;
```

4.8 欧拉降幂

```
#include <bits/stdc++.h>

const int Maxn = 1000005;
typedef long long ll;
using namespace std;
ll a, b, m;
ll vis[Maxn];
ll prime[Maxn], num = 0;
ll phi[Maxn];

void getphi(ll n = 1000000) {
    phi[1] = 1;
    for (ll i = 2; i <= n; i++) {
        if (!vis[i]) prime[++num] = i, phi[i] = i - 1;
        for (ll j = 1; j <= num; j++) {
            ll k = i * prime[j];
            if (k > n)break;
            vis[k] = 1;
            if (i % prime[j] == 0) {
                phi[k] = phi[i] * prime[j];
                break;
            }
            phi[k] = phi[i] * (prime[j] - 1);
        }
    }
    return;
}

ll qpow(ll a, ll b, ll mod) {
    ll ret = 1;
    while (b) {
        if (b & 1) ret = ret * a % mod;
        a = a * a % mod;
        b >>= 1;
    }
    return ret;
}
```

```
ll check(ll d, ll cmp) {
    ll res = a;
    for (ll i = 1; i <= d; i++) {
        ll tmp = 1;
        for (ll j = 1; j <= res; j++) {
            tmp = tmp * a;
            if (tmp >= cmp) return 1;
        }
        res = tmp;
    }
    return -1;
}

ll fun(ll d) {
    ll res = a;
    for (ll i = 1; i <= d; i++) {
        ll tmp = 1;
        for (ll j = 1; j <= res; j++) tmp = tmp * a;
        res = tmp;
    }
    return res;
}

ll cal(ll d, ll p) {
    if (p == 1) return 0;

    if (d == 0) return a % p;

    ll tmp;
    if (__gcd(a, p) == 1) tmp = cal(d - 1, phi[p]);//gcd(a,p)==1
    else {
        if (check(d - 1, phi[p]) < 0) tmp = fun(d - 1);//gcd(a,p)!=1&&b<phi(p)
        else tmp = cal(d - 1, phi[p]) + phi[p];//gcd(a,p)!=1&&b>=phi(p)
    }
    ll ret = qpow(a, tmp, p);
    return ret;
}
```

```
int main() {
    getphi();
    ll t;
    scanf("%lld", &t);
    while (t--) {
        scanf("%lld%lld%lld", &a, &b, &m);
        if (a == 1 || b == 0) {
            printf("%lld\n", 1 % m);
            continue;
        }
        ll ans = cal(b - 1, m);
        printf("%lld\n", ans % m);
    }
    return 0;
}
```

5 杂项

5.1 约瑟夫问题

约瑟夫问题

简介
所有人围成一圈，顺时针报数，每次报到q的人将被杀掉，被杀掉的人将从房间内被移走。然后从被杀掉的下一个人重新报数，继续报q，再清除，直到剩余一人

q = 2

对于q为2的情况，有个简单的方法。假设有2^n 个人，从1开始数，那么最后活下来的那个一定是1.

那么可以借用这个规律，假设有9个人，从1开始数，数到2，杀死，接下来到3的时候，相当于8个人从3开始数，那么根据上面的规律，活下来的就是3了（8==2^3)

规律为：当q==2时，2^n+t个人活下来的是第2t+1个人。

e.g. q==2，n<=1e100，可以用这个思路做。用大数，找到第一个小于等于2^x 的数，直接得ans==2*(n-2^x)+1

q != 2

设：n+1个数，每次杀第q个的最终结果为A(n+1)； n个每次杀第q个为A(n)
显然A(n+1)=(A(n)+q)%(n+1)，A(1)==0

证明：

n+1个人的游戏中，第一个杀掉的是k−1，那么第二个人是k−1+k 。但是我们可以把杀掉一个人后的情况，看成n个人的游戏的开始，也就是说k−1+k看成n个人的游戏中的k−1。所以有A(n+1)=(A(n)+q)%(n+1)

```
int f(int n,int q){
    if(n==1){
        return 0;
    }
    return (f(n-1,q)+q)%n;
}//若第一个编号为1，ans++即可
```

icpc2018沈阳 K

题意：

给出n个人的环，约瑟夫环背景，求第m个死的位置。

解析：
n+1个人第m个死的位置相当于n 个人第m−1个死的位置转k个下标。即A(n,m)=(A(n−1,m−1)+k)%n

当m较小的时候直接推就行了。

对于m=1e18,k=1e6的数据，显然A(n,m)=(A(n−1,m−1)+k)%n 的这个%n在很多时候没有用处。

A+x*k*<*n*+x

x(k−1)<n−A

x<(*k*−1)*n*−*A*

处理一下就可以了。

板子总结

1、题目：n个人，1至m报数，问最后留下来的人的编号。

公式：f(n,m)=(f(n−1,m)+m)%n，f(0,m)=0;

代码：复杂度O(n)

```
typedef long long ll;
ll calc(int n, ll m) {
    ll p = 0;
    for (int i = 2; i <= n; i++) {
        p = (p + m) % i;
    }
    return p + 1;
}
```

2、题目：n个人，1至m报数，问第k个出局的人的编号。(k<1e6)

公式： f(n,k)=(f(n−1,k−1)+m−1)%n+1

f(n−k+1,1)=m%(n−k+1)

if (f==0) f=n−k+1

代码：复杂度O(k)

```
ll cal1(ll n, ll m, ll k) { // (k == n) equal(calc)
    ll p = m % (n - k + 1);
    if (p == 0) p = n - k + 1;
    for (ll i = 2; i <= k; i++) {
        p = (p + m - 1) % (n - k + i) + 1;
    }
    return p;
}
```

3、题目：n个人，1至m报数，问第k个出局的人的编号。 (m<1e6)

代码：复杂度O(m∗log(m))

```
ll cal2(ll n, ll m, ll k) {
    if (m == 1) return k;
    else {
        ll a = n - k + 1, b = 1;
        ll c = m % a, x = 0;
        if (c == 0) c = a;
        while (b + x <= k) {
            a += x, b += x, c += m * x;
            c %= a;
            if (c == 0) c = a;
            x = (a - c) / (m - 1) + 1;
        }
        c += (k - b) * m;
        c %= n;
        if (c == 0) c = n;
```

```
        return c;
    }
}
```

4、题目：n个人，1至m报数，问编号为k的人是第几个出局的。

代码：复杂度O(n)

```
ll n,k;//可做n<=4e7,询问个数<=100,下标范围[0,n-1]
ll dieInXturn(int n,int k,int x){//n个人，报数k，下标为X的人第几个死亡
    ll tmp=0;
    while(n){
        x=(x+n)%n;
        if(k>n)x+=(k-x-1+n-1)/n*n;
        if((x+1)%k==0){
            tmp+=(x+1)/k;break;
        }else{
            if(k>n){
                tmp+=x/k;
                ll ttmp=x;
                x=x-(x/n+1)*(x/k)+(x+n)/n*n-k;
                n-=ttmp/k;

            }else{
                tmp+=n/k;
                x=x-x/k;
                x+=n-n/k*k;
                n-=n/k;
            }
        }
    }
    return tmp;
}
```

6 计算几何