Templates 洛谷模板题

admax

注释:单组时间/总时间/c++14 O2最快总时间(ms)

- 基本结构
  - 头文件&文件IO&关同步
  - fastio
  - 对拍
- 普及-
  - 并查集(n=1e4,op=1e5)(11/54/42)
  - 基数排序(快排略)(1e5)(11/46/31)
  - 素数筛(bitset比欧拉筛快)(1e8)(450/1986/819)
  - 最小生成树(n=5e3,m=2e5)
    - Kruskal(21/93/72)
    - n2 prim(84/284/72)
    - mlogn prim(22/101/72)
  - 快速幂(3/24/24)
- 普及/提高-
  - 负环(从s可达)(bellman-ford)(n=2e3,m=6e3)(86/275/57)
    - 附:全图负环
  - ST表(n=1e5,q=2e6)(178/612/489)
  - 字典树(n=1e5,q=1e5,sumlen=3e6)(347/1028/488)
  - 逆元(n=3e6)(167/288/163)
  - gcd&exgcd(3/34/33)
  - 线段树(区间加,区间和)(n=1e5,op=1e5)(49/188/63)
  - 三分法(3/33/30)
  - 矩阵快速幂(矩阵类)(n=100,k=1e12)(131/522/108)
  - KMP(n=m=1e6)(24/15/11)
  - LCA(n=5e5,q=5e5)
    - 倍增(798/2870/597)
    - tarjan(离线)(536/2450/597)
    - RMQ(392/1930/597)
  - dijkstra(n=1e5,m=2e5)(50/227/83)
- 普及+/提高
  - SCC(tarjan)(n=1e4,m=1e5)(8/62/52)
  - 线段树(区间乘,区间加,区间和)(n=1e5,op=1e5)(84/294/259)
  - 欧拉路径(n=1e5,m=2e5)(56/290/157)
  - 割点(tarjan)(21/217/76)
    - 附:割边(不含重边)
    - 附:割边(含重边)(链式前向星)
  - 全源最短路(n=3e3,m=6e3)(617/2350/778)
  - 边双连通分量(n=5e5,m=2e6)(536/1830/1340)
  - 点双连通分量(n=5e5,m=2e6)(811/2990/1350)
  - 最大流(dinic)(二分图最大匹配)(n=500,e=5e4)(8/3/3)

基本结构

头文件&文件IO&关同步

```
#include <bits/stdc++.h>
#define ll long long
#define endl '\n'
#define ls rt << 1
#define rs rt << 1 | 1
#define maxn 200010
using namespace std;

int main()
{
    std::ios::sync_with_stdio(false);
    std::cin.tie(0);
    std::cout.tie(0);
#ifndef ONLINE_JUDGE
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);
    ll _begin_time = clock();
#endif
    ll _;
    cin >> _;
    while (_--)
        solve();
#ifndef ONLINE_JUDGE
    ll _end_time = clock();
    printf("time = %ld ms\n", _end_time - _begin_time);
#endif
    return 0;
}
```

fastio

```
namespace IO
{
```

```
    const ll L = 1 << 20;
    char inbuf[L], outbuf[L], *S = inbuf, *T = inbuf, *O = outbuf;
    inline char getc()
    {
        if (S == T)
        {
            T = (S = inbuf) + fread(inbuf, 1, L, stdin);
            if (S == T)
                return EOF;
        }
        return *S++;
    }
    inline ll geti()
    {
        char c;
        ll re = 0, w = 1;
        for (c = getc(); c < '0' || c > '9'; c = getc())
            if (c == '-')
                w = -1;
        while (c >= '0' && c <= '9')
            re = re * 10 + (c - '0'), c = getc();
        return re * w;
    }
    inline void flush()
    {
        fwrite(outbuf, 1, O - outbuf, stdout);
        O = outbuf;
    }
    inline void putc(char x)
    {
        *O++ = x;
        if (O - outbuf == L)
            flush();
    }
    void puti(ll x)
    {
        if (x < 0)
            putc('-'), x = -x;
        if (x > 9)
            puti(x / 10);
        putc(x % 10 + '0');
    }
    struct end
    {
        ~end() { flush(); }
    } dummy;
}
```

对拍

```
```

```
::@echo off
:loop
duipai.exe
standard.exe
my.exe
fc standardout.txt myout.txt #diff linux
if not errorlevel 1 goto loop
pause
:end
```

普及-

并查集(n=1e4,op=1e5)(11/54/42)

```
ll fa[maxn], weight[maxn];
inline ll findfa(ll x){return fa[x] == x ? x : fa[x] = findfa(fa[x]);}
void unite(ll x, ll y)
{
    x = findfa(x);
    y = findfa(y);
    if (x == y)
        return;
    if (weight[x] > weight[y])
        swap(x, y);
    fa[x] = y;
    weight[y] += weight[x];
}
```

基数排序(快排略)(1e5)(11/46/31)

```
void radix_sort(ll a[], ll n)
{
    ll *b = new ll[n];
    ll *cnt = new ll[1 << 8];
    ll mask = (1 << 8) - 1;
    ll *x = a, *y = b;
    for (int i = 0; i < 32; i += 8)
    {
        for (int j = 0; j != (1 << 8); j++)
            cnt[j] = 0;
        for (int j = 0; j != n; j++)
            ++cnt[x[j] >> i & mask];
        for (int sum = 0, j = 0; j != (1 << 8); j++)
        {
            sum += cnt[j];
            cnt[j] = sum - cnt[j];
```

```
        }
        for (int j = 0; j != n; j++)
            y[cnt[x[j] >> i & mask]++] = x[j];
        swap(x, y);
    }
    delete[] cnt;
    delete[] b;
}
```

素数筛(bitset比欧拉筛快)(1e8)(450/1986/819)

```
bitset<maxn> isp;
int p[maxn / 10], pptr = 1;
isp[0] = isp[1] = 1;
p[0] = 2;
for (int i = 3; i < maxn; i += 2)
{
    if (!isp[i])
    {
        p[pptr++] = i;
        for (ll j = 1ll * i * i; j < maxn; j += i)
            isp[j] = 1;
    }
}
```

最小生成树(n=5e3,m=2e5)

Kruskal(21/93/72)

```
struct edge
{
    ll u, v, cost;
    bool operator<(const edge &t) const { return cost < t.cost; }
};
vector<edge> a;
// union-find set init
ll ans = 0;
sort(a.begin(), a.end());
for (int i = 0; i < a.size(); i++)
    if (findfa(a[i].u) != findfa(a[i].v))
    {
        unite(a[i].u, a[i].v);
        ans += a[i].cost;
    }
```

n2 prim(84/284/72)

```
vector<pair<ll, ll>> edge[maxn];
ll dis[maxn], used[maxn];
memset(dis, 0x3f, sizeof(dis));
dis[1] = 0;
ll ans = 0;
while (true)
{
    ll v = -1;
    for (int i = 1; i <= n; i++)
        if (!used[i] && (v == -1 || dis[i] < dis[v]))
            v = i;
    if (v == -1)
        break;
    used[v] = 1;
    ans += dis[v];
    for (pair<ll, ll> e : edge[v])
        dis[e.first] = min(dis[e.first], e.second);
}
```

mlogn prim(22/101/72)

```
vector<pair<ll, ll>> edge[maxn];
ll dis[maxn], used[maxn];
priority_queue<pair<ll, ll>, vector<pair<ll, ll>>, greater<pair<ll, ll>>> q;
memset(dis, 0x3f, sizeof(dis));
dis[1] = 0;
q.push({0, 1});
ll ans = 0;
while (!q.empty())
{
    ll v = q.top().second;
    q.pop();
    if (used[v])
        continue;
    used[v] = 1;
    ans += dis[v];
    for (pair<ll, ll> e : edge[v])
        if (!used[e.first] && e.second < dis[e.first])
        {
            dis[e.first] = e.second;
            q.push({dis[e.first], e.first});
        }
}
```

快速幂(3/24/24)

```
ll ksm(ll a, ll b)
{
```

```
    ll rtn = 1;
    while (b)
    {
        if (b & 1)
            rtn = rtn * a % p;
        a = a * a % p;
        b >>= 1;
    }
    return rtn;
}
```

普及/提高-

负环(从s可达)(bellman-ford)(n=2e3,m=6e3)(86/275/57)

```
struct edge
{
    ll u, v, cost;
};
vector<edge> a;
ll dis[maxn];
ll ans = 0;
memset(dis, 0x3f, sizeof(dis));
dis[1] = 0;
for (int i = 0; i < n - 1; i++)
    for (edge e : a)
        dis[e.v] = min(dis[e.v], dis[e.u] + e.cost);
for (edge e : a)
    if (dis[e.v] < 1e9 && dis[e.v] > dis[e.u] + e.cost)
        ans = 1;
```

附:全图负环

```
ll ans = 0;
memset(dis, 0, sizeof(dis));
for (int i = 0; i < n - 1; i++)
    for (edge e : a)
        dis[e.v] = min(dis[e.v], dis[e.u] + e.cost);
for (edge e : a)
    if (dis[e.v] > dis[e.u] + e.cost)
        ans = 1;
```

ST表(n=1e5,q=2e6)(178/612/489)

```
```

```
ll st[18][maxn];
for (int i = 0; i < n; i++)
    st[0][i] = IStream::Get_Int();
for (int i = 1; i < 18; i++)
    for (int j = 0; j < n - (1 << i) + 1; j++)
        st[i][j] = max(st[i - 1][j], st[i - 1][j + (1 << i - 1)]);
while (m--)
{
    x = IStream::Get_Int();
    y = IStream::Get_Int();
    ll ptr = 32 - __builtin_clz(y - x + 1) - 1;
    cout << max(st[ptr][x - 1], st[ptr][y - (1 << ptr)]) << endl;
}
```

字典树(n=1e5,q=1e5,sumlen=3e6)(347/1028/488)

```
int last;
char s[maxn];
int trie[maxn], ptr[maxn][62];
void ins(char *s)
{
    ll len = strlen(s);
    ll now = 0;
    trie[0]++;
    for (int i = 0; i < len; i++)
    {
        if (!ptr[now][trans(s[i])])
            ptr[now][trans(s[i])] = ++last;
        now = ptr[now][trans(s[i])];
        trie[now]++;
    }
}
ll query(char *s)
{
    ll len = strlen(s);
    ll now = 0;
    for (int i = 0; i < len; i++)
    {
        if (!ptr[now][trans(s[i])])
            return 0;
        now = ptr[now][trans(s[i])];
    }
    return trie[now];
}
```

逆元(n=3e6)(167/288/163)

```
inv[1] = 1;
for (int i = 2; i <= n; i++)
    inv[i] = (p - p / i) * inv[p % i] % p;
```

gcd&exgcd(3/34/33)

```
ll gcd(ll a, ll b) { return b ? gcd(b, a % b) : a; }
ll exgcd(ll a, ll b, ll &x, ll &y)
{
    ll d = a;
    if (b)
    {
        d = exgcd(b, a % b, y, x);
        y -= (a / b) * x;
    }
    else
    {
        x = 1;
        y = 0;
    }
    return d;
}
```

线段树(区间加,区间和)(n=1e5,op=1e5)(49/188/63)

```
ll tree[maxn * 4], lazy[maxn * 4];
void pushup(int rt, int l, int r)
{
    tree[rt] = tree[ls] + tree[rs];
}
void pushdown(int rt, int l, int r)
{
    ll mid = (l + r) / 2;
    lazy[ls] += lazy[rt];
    lazy[rs] += lazy[rt];
    tree[ls] += lazy[rt] * (mid - l);
    tree[rs] += lazy[rt] * (r - mid);
    lazy[rt] = 0;
}
void build(int rt, int l, int r)
{
    lazy[rt] = 0;
    if (l + 1 == r)
    {
        tree[rt] = a[l];
        return;
    }
    int mid = l + r >> 1;
```

```
    build(ls, l, mid);
    build(rs, mid, r);
    pushup(rt, l, r);
}
void upd(int rt, int l, int r, int lll, int rrr, ll v)
{
    if (lll <= l && rrr >= r)
    {
        lazy[rt] += v;
        tree[rt] += (r - l) * v;
        return;
    }
    pushdown(rt, l, r);
    int mid = l + r >> 1;
    if (lll < mid)
        upd(ls, l, mid, lll, rrr, v);
    if (rrr > mid)
        upd(rs, mid, r, lll, rrr, v);
    pushup(rt, l, r);
}
ll query(int rt, int l, int r, int lll, int rrr)
{
    if (lll <= l && rrr >= r)
    {
        return tree[rt];
    }
    pushdown(rt, l, r);
    int mid = l + r >> 1;
    ll ans = 0;
    if (lll < mid)
        ans += query(ls, l, mid, lll, rrr);
    if (rrr > mid)
        ans += query(rs, mid, r, lll, rrr);
    return ans;
}
```

附:树状数组实现(16/76/63)

```
ll bit[2][maxn + 1];
ll sum(int ptr, int i)
{
    ll s = 0;
    while (i > 0)
    {
        s += bit[ptr][i];
        i -= i & -i;
    }
    return s;
}
void add(int ptr, int i, ll v)
{
    while (i < maxn)
```

```
    {
        bit[ptr][i] += v;
        i += i & -i;
    }
}
//sum(i)=i*bit1[i]+bit0[i]
add(0, x, -(x - 1) * z);
add(0, y + 1, y * z);
add(1, x, z);
add(1, y + 1, -z);
```

三分法(3/33/30)

```
//find max
while (r - l > 1e-8)
{
    double midl = (l + l + r) / 3, midr = (l + r + r) / 3;
    double fl = f(midl), fr = f(midr);
    if (fl > fr)
        r = midr;
    else
        l = midl;
}
```

矩阵快速幂(矩阵类)(n=100,k=1e12)(131/522/108)

```
struct matrix
{
    ll n, m;
    ll **data;
    matrix(ll nn, ll mm) : n(nn), m(mm)
    {
        ll *tmp = new ll[n * m]();
        data = new ll *[n];
        for (int i = 0; i < n; i++)
            data[i] = tmp + m * i;
    }
    matrix(ll nn) : n(nn), m(nn)
    {
        ll *tmp = new ll[n * m]();
        data = new ll *[n];
        for (int i = 0; i < n; i++)
            data[i] = tmp + m * i;
        for (int i = 0; i < n; i++)
            data[i][i] = 1;
    }
    matrix(const matrix &b) : n(b.n), m(b.m)
    {
        ll *tmp = new ll[n * m]();
```

```
        data = new ll *[n];
        for (int i = 0; i < n; i++)
            data[i] = tmp + m * i;
        for (int i = 0; i < n; i++)
            for (int j = 0; j < m; j++)
                data[i][j] = b.data[i][j];
    }
    matrix operator=(const matrix &b)
    {
        if(&b==this)
            return *this;
        n = b.n;
        m = b.m;
        delete[] data[0];
        delete[] data;
        ll *tmp = new ll[n * m]();
        data = new ll *[n];
        for (int i = 0; i < n; i++)
            data[i] = tmp + m * i;
        for (int i = 0; i < n; i++)
            for (int j = 0; j < m; j++)
                data[i][j] = b.data[i][j];
        return *this;
    }
    matrix operator+(const matrix &b) const
    {
        assert(n == b.n && m == b.m);
        matrix rtn(n, m);
        for (int i = 0; i < n; i++)
            for (int j = 0; j < m; j++)
                rtn.data[i][j] = (data[i][j] + b.data[i][j]) % p;
        return rtn;
    }
    matrix operator-(const matrix &b) const
    {
        assert(n == b.n && m == b.m);
        matrix rtn(n, m);
        for (int i = 0; i < n; i++)
            for (int j = 0; j < m; j++)
                rtn.data[i][j] = (data[i][j] - b.data[i][j] + p) % p;
        return rtn;
    }
    matrix operator-() const
    {
        matrix rtn(n, m);
        for (int i = 0; i < n; i++)
            for (int j = 0; j < m; j++)
                rtn.data[i][j] = (p - data[i][j]) % p;
        return rtn;
    }
    matrix operator*(const matrix &b) const
    {
        assert(m == b.n);
        matrix rtn(n, b.m);
```

```
        for (int i = 0; i < n; i++)
            for (int j = 0; j < b.m; j++)
                for (int k = 0; k < m; k++)
                    rtn.data[i][j] = (rtn.data[i][j] + data[i][k] * b.data[k][j])
% p;
        return rtn;
    }
    matrix ksm(ll e) const
    {
        assert(n == m);
        matrix rtn(n), tmp(*this);
        while (e)
        {
            if (e % 2)
                rtn = rtn * tmp;
            e /= 2;
            tmp = tmp * tmp;
        }
        return rtn;
    }
    matrix trans() const
    {
        matrix rtn(m, n);
        for (int i = 0; i < n; i++)
            for (int j = 0; j < m; j++)
                rtn.data[j][i] = data[i][j];
        return rtn;
    }
    ~matrix()
    {
        delete[] data[0];
        delete[] data;
    }
};
```

KMP(n=m=1e6)(24/15/11)

```
// nxt[i]=j iff a0a1...aj-1=ai-j+1...ai
nxt[0] = 0;
for (int i = 1; i < t.size(); i++)
{
    ll ptr = nxt[i - 1];
    while (ptr && t[i] != t[ptr])
        ptr = nxt[ptr - 1];
    if (t[i] == t[ptr])
        ptr++;
    nxt[i] = ptr;
}
ll ptr = 0;
for (int i = 0; i < s.size(); i++)
{
```

```
    while (ptr && s[i] != t[ptr])
        ptr = nxt[ptr - 1];
    if (s[i] == t[ptr] && ptr == t.size() - 1)
    {
        //matched
        ptr = nxt[ptr - 1];
    }
    if (s[i] == t[ptr])
        ptr++;
}
```

LCA(n=5e5,q=5e5)

倍增(798/2870/597)

```
ll fa[maxn][20], depth[maxn];
depth[rt] = 0;
void dfs(ll i, ll from)
{
    depth[i] = depth[from] + 1;
    fa[i][0] = from;
    for (int j = 1; j < 20; j++)
        fa[i][j] = fa[fa[i][j - 1]][j - 1];
    for (int j : edge[i])
        if (j != from)
            dfs(j, i);
}
ll lca(ll x, ll y)
{
    if (depth[x] < depth[y])
        swap(x, y);
    for (int i = 19; i >= 0; i--)
        if (depth[x] - depth[y] >= (1 << i))
            x = fa[x][i];
    if (x == y)
        return x;
    for (int i = 19; i >= 0; i--)
        if (fa[x][i] != fa[y][i])
        {
            x = fa[x][i];
            y = fa[y][i];
        }
    return fa[x][0];
}
```

tarjan(离线)(536/2450/597)

```
vector<pair<ll, ll>> query[maxn];
ll ans[maxn];
```

```
ll fa[maxn], weight[maxn], visited[maxn];
inline ll findfa(ll x) { return fa[x] == x ? x : fa[x] = findfa(fa[x]); }
void dfs(ll i, ll from)
{
    visited[i] = 1;
    for (ll j : edge[i])
        if (j != from)
        {
            dfs(j, i);
            fa[j] = i;
        }
    for (pair<ll, ll> tmp : query[i])
        if (visited[tmp.first])
            ans[tmp.second] = findfa(tmp.first);
}
```

RMQ(392/1930/597)

```
ll dfsptr = 0, st[20][maxn * 2], place[maxn];
void dfs(ll i, ll from)
{
    st[0][dfsptr] = i;
    place[i] = dfsptr;
    dfsptr++;
    for (int j : edge[i])
        if (j != from)
        {
            dfs(j, i);
            st[0][dfsptr++] = i;
        }
}
dfs(rt, rt);
for (int i = 1; i < 20; i++)
    for (int j = 0; j < dfsptr - (1 << i) + 1; j++)
        st[i][j] = place[st[i - 1][j]] < place[st[i - 1][j + (1 << i - 1)]] ? st[i
- 1][j] : st[i - 1][j + (1 << i - 1)];
while (q--)
{
    x = IO::geti();
    y = IO::geti();
    x = place[x];
    y = place[y];
    if (x > y)
        swap(x, y);
    ll ptr = 32 - __builtin_clz(y - x + 1) - 1;
    IO::puti(place[st[ptr][x]] < place[st[ptr][y - (1 << ptr) + 1]] ? st[ptr][x] :
st[ptr][y - (1 << ptr) + 1]);
    IO::putc('\n');
}
```

dijkstra(n=1e5,m=2e5)(50/227/83)

```
priority_queue<pair<ll, ll>, vector<pair<ll, ll>>, greater<pair<ll, ll>>> q;
memset(dis, 0x3f, sizeof(dis));
dis[s] = 0;
q.push({0, s});
while (!q.empty())
{
    pair<ll, ll> tmp = q.top();
    q.pop();
    if (dis[tmp.second] < tmp.first)
        continue;
    ll u = tmp.second;
    for (pair<ll, ll> t : edge[u])
        if (dis[t.first] > dis[u] + t.second)
        {
            dis[t.first] = dis[u] + t.second;
            q.push({dis[t.first], t.first});
        }
}
```

普及+/提高

SCC(tarjan)(n=1e4,m=1e5)(8/62/52)

```
vector<ll> edge[maxn];
ll dfn[maxn], low[maxn], dfsptr; // low: node with min depth it can visit
stack<ll> s;
ll instack[maxn], scc[maxn], scccnt, sccsize[maxn];
void tarjan(ll u)
{
    low[u] = dfn[u] = ++dfsptr;
    s.push(u);
    instack[u] = 1;
    for (ll v : edge[u])
    {
        if (!dfn[v])
        {
            tarjan(v);
            low[u] = min(low[u], low[v]);
        }
        else if (instack[v])
            low[u] = min(low[u], dfn[v]);
    }
    if (dfn[u] == low[u])
    {
        scccnt++;
        while (s.top() != u)
        {
```

```
            scc[s.top()] = scccnt;
            sccsize[scccnt]++;
            instack[s.top()] = 0;
            s.pop();
        }
        scc[s.top()] = scccnt;
        sccsize[scccnt]++;
        instack[s.top()] = 0;
        s.pop();
    }
}
```

线段树(区间乘,区间加,区间和)(n=1e5,op=1e5)(84/294/259)

```
ll tree[maxn * 4], lazyadd[maxn * 4], lazymul[maxn * 4];
void pushup(int rt, int l, int r)
{
    tree[rt] = tree[ls] + tree[rs];
}
void pushdown(int rt, int l, int r)
{
    ll mid = (l + r) / 2;
    lazymul[ls] *= lazymul[rt];
    lazyadd[ls] = lazyadd[ls] * lazymul[rt] + lazyadd[rt];
    tree[ls] = tree[ls] * lazymul[rt] + lazyadd[rt] * (mid - l);
    lazymul[rs] *= lazymul[rt];
    lazyadd[rs] = lazyadd[rs] * lazymul[rt] + lazyadd[rt];
    tree[rs] = tree[rs] * lazymul[rt] + lazyadd[rt] * (r - mid);
    lazyadd[rt] = 0;
    lazymul[rt] = 1;
}
void build(int rt, int l, int r)
{
    lazyadd[rt] = 0;
    lazymul[rt] = 1;
    if (l + 1 == r)
    {
        tree[rt] = a[l];
        return;
    }
    int mid = l + r >> 1;
    build(ls, l, mid);
    build(rs, mid, r);
    pushup(rt, l, r);
}
void upd(int op, int rt, int l, int r, int lll, int rrr, ll v)
{
    if (lll <= l && rrr >= r)
    {
        if (op == 1) // add
        {
```

```
            lazyadd[rt] += v;
            tree[rt] += (r - l) * v;
        }
        else // mul
        {
            lazyadd[rt] *= v;
            lazymul[rt] *= v;
            tree[rt] *= v;
        }
        return;
    }
    pushdown(rt, l, r);
    int mid = l + r >> 1;
    if (lll < mid)
        upd(op, ls, l, mid, lll, rrr, v);
    if (rrr > mid)
        upd(op, rs, mid, r, lll, rrr, v);
    pushup(rt, l, r);
}
ll query(int rt, int l, int r, int lll, int rrr)
{
    if (lll <= l && rrr >= r)
    {
        return tree[rt];
    }
    pushdown(rt, l, r);
    int mid = l + r >> 1;
    ll ans = 0;
    if (lll < mid)
        ans += query(ls, l, mid, lll, rrr);
    if (rrr > mid)
        ans += query(rs, mid, r, lll, rrr);
    return ans;
}
```

欧拉路径(n=1e5,m=2e5)(56/290/157)

```
ll ddeg[maxn], ptr[maxn];
stack<ll> s;
void dfs(ll i)
{
    while (ptr[i] != edge[i].size())
        dfs(edge[i][ptr[i]++]);
    s.push(i);
}
// check if |ddeg|<=1 and at most one ddeg=1
dfs(begin);
// output s from top to bottom
```

割点(tarjan)(21/217/76)

```
ll dfn[maxn], low[maxn], dfsptr; // low: node with min depth it can visit
ll ans[maxn];
void tarjan(ll u, ll from)
{
    low[u] = dfn[u] = ++dfsptr;
    ll tmp = 0; // count son of root
    for (ll v : edge[u])
    {
        if (!dfn[v])
        {
            tmp++;
            tarjan(v, u);
            low[u] = min(low[u], low[v]);
            if (u != from && low[v] >= dfn[u])
                ans[u] = 1;
        }
        else if (v != from)
            low[u] = min(low[u], dfn[v]);
    }
    if (u == from && tmp >= 2)
        ans[u] = 1;
}
```

附:割边(不含重边)

```
ll dfn[maxn], low[maxn], dfsptr; // low: node with min depth it can visit
ll fa[maxn], ans[maxn];          // ans[i]=1 means i->fa[i] is cut edge
void tarjan(ll u, ll from)
{
    fa[u] = from;
    low[u] = dfn[u] = ++dfsptr;
    for (ll v : edge[u])
    {
        if (!dfn[v])
        {
            tarjan(v, u);
            low[u] = min(low[u], low[v]);
            if (low[v] > dfn[u])
                ans[v] = 1;
        }
        else if (v != from)
            low[u] = min(low[u], dfn[v]);
    }
}
```

附:割边(含重边)(链式前向星)

```
#define maxm 2000010
ll head[maxn], to[maxm * 2], nxt[maxm * 2], ptr = 1; // init 1 to find inverse
edge by i^1
ll dfn[maxn], low[maxn], dfsptr;                     // low: node with min depth
it can visit
ll ans[maxn];                                        // ans[i]=1 means i->fa[i] is
cut edge
void addedge(ll u, ll v)
{
    to[++ptr] = v;
    nxt[ptr] = head[u];
    head[u] = ptr;
}
void tarjan(ll u, ll inedge)
{
    low[u] = dfn[u] = ++dfsptr;
    for (int i = head[u]; i; i = nxt[i])
    {
        ll v = to[i];
        if (!dfn[v])
        {
            tarjan(v, i);
            low[u] = min(low[u], low[v]);
            if (low[v] > dfn[u])
                ans[v] = 1;
        }
        else if (i != (inedge ^ 1))
            low[u] = min(low[u], dfn[v]);
    }
}
for (int i = 1; i <= n; i++)
    if (!dfn[i])
        tarjan(i, 0);
```

全源最短路(n=3e3,m=6e3)(617/2350/778)

```
ll dis[maxn];
ll ans[maxn][maxn];
priority_queue<pair<ll, ll>, vector<pair<ll, ll>>, greater<pair<ll, ll>>> q;
for (int i = 1; i <= n; i++)
    edge[0].push_back({i, 0});
memset(dis, 0x3f, sizeof(dis));
dis[0] = 0;
for (int i = 0; i < n; i++) // caution: one more point
    for (int j = 0; j <= n; j++)
        for (pair<ll, ll> tmp : edge[j])
            dis[tmp.first] = min(dis[tmp.first], dis[j] + tmp.second);
for (int j = 0; j <= n; j++)
    for (pair<ll, ll> tmp : edge[j])
        if (dis[tmp.first] > dis[j] + tmp.second)
```

```
        {
            IO::puti(-1);
            return 0;
        }
for (int i = 1; i <= n; i++)
    for (pair<ll, ll> &tmp : edge[i])
        tmp.second += dis[i] - dis[tmp.first];
for (int i = 1; i <= n; i++)
{
    memset(ans[i], 0x3f, sizeof(ans[i]));
    ans[i][i] = 0;
    q.push({0, i});
    while (!q.empty())
    {
        pair<ll, ll> tmp = q.top();
        q.pop();
        if (ans[i][tmp.second] < tmp.first)
            continue;
        ll u = tmp.second;
        for (pair<ll, ll> t : edge[u])
            if (ans[i][t.first] > ans[i][u] + t.second)
            {
                ans[i][t.first] = ans[i][u] + t.second;
                q.push({ans[i][t.first], t.first});
            }
    }
}
for (int i = 1; i <= n; i++)
    for (int j = 1; j <= n; j++)
        ans[i][j] += dis[j] - dis[i];
```

边双连通分量(n=5e5,m=2e6)(536/1830/1340)

```
#define maxm 2000010
ll head[maxn], to[maxm * 2], nxt[maxm * 2], ptr = 1; // init 1 to find inverse
edge by i^1
ll dfn[maxn], low[maxn], dfsptr;                     // low: node with min depth
it can visit
stack<ll> s;
vector<vector<ll>> res;
void addedge(ll u, ll v)
{
    to[++ptr] = v;
    nxt[ptr] = head[u];
    head[u] = ptr;
}
void tarjan(ll u, ll inedge)
{
    low[u] = dfn[u] = ++dfsptr;
    s.push(u);
    for (int i = head[u]; i; i = nxt[i])
```

```
    {
        ll v = to[i];
        if (!dfn[v])
        {
            tarjan(v, i);
            low[u] = min(low[u], low[v]);
            if (low[v] > dfn[u])
            {
                vector<ll> tmp;
                while (s.top() != v)
                {
                    tmp.push_back(s.top());
                    s.pop();
                }
                tmp.push_back(s.top());
                s.pop();
                res.push_back(tmp);
            }
        }
        else if (i != (inedge ^ 1))
            low[u] = min(low[u], dfn[v]);
    }
}
for (int i = 1; i <= n; i++)
    if (!dfn[i])
    {
        tarjan(i, 0);
        vector<ll> tmp;
        while (!s.empty())
        {
            tmp.push_back(s.top());
            s.pop();
        }
        res.push_back(tmp);
    }
```

点双连通分量(n=5e5,m=2e6)(811/2990/1350)

```
ll dfn[maxn], low[maxn], dfsptr; // low: node with min depth it can visit
stack<ll> s;
vector<vector<ll>> res;
void tarjan(ll u, ll from)
{
    ll soncnt = 0;
    low[u] = dfn[u] = ++dfsptr;
    s.push(u);
    for (ll v : edge[u])
    {
        if (!dfn[v])
        {
            soncnt++;
```

```
            tarjan(v, u);
            low[u] = min(low[u], low[v]);
            if (low[v] >= dfn[u])
            {
                vector<ll> tmp;
                while (s.top() != v)
                {
                    tmp.push_back(s.top());
                    s.pop();
                }
                tmp.push_back(s.top());
                s.pop();
                tmp.push_back(u);
                res.push_back(tmp);
            }
        }
        else if (v != from)
            low[u] = min(low[u], dfn[v]);
    }
    if (u == from && soncnt == 0) // isolated point (possibly self loop)
    {
        vector<ll> tmp(1, u);
        res.push_back(tmp);
    }
}
```

最大流(dinic)(二分图最大匹配)(n=500,e=5e4)(8/3/3)

```
const ll inf = 1e12;
struct node
{
    ll to;
    ll cap;
    ll rev;
};
vector<node> edge[maxn];
ll level[maxn];
ll vis[maxn];
void add_edge(ll from, ll to, ll cap)
{
    edge[from].push_back({to, cap, (ll)edge[to].size()});
    edge[to].push_back({from, 0, (ll)edge[from].size() - 1});
}
void bfs(ll s)
{
    memset(level, -1, sizeof(level));
    queue<ll> q;
    level[s] = 0;
    q.push(s);
    while (!q.empty())
    {
```

```
            ll v = q.front();
            q.pop();
            for (int i = 0; i < edge[v].size(); i++)
            {
                if (edge[v][i].cap > 0 && level[edge[v][i].to] < 0)
                {
                    level[edge[v][i].to] = level[v] + 1;
                    q.push(edge[v][i].to);
                }
            }
        }
    }
    ll dfs(ll v, ll t, ll f)
    {
        if (v == t)
            return f;
        for (; vis[v] < edge[v].size(); vis[v]++)
        {
            node &e = edge[v][vis[v]];
            if (e.cap > 0 && level[v] < level[e.to])
            {
                ll d = dfs(e.to, t, min(f, e.cap));
                if (d > 0)
                {
                    e.cap -= d;
                    edge[e.to][e.rev].cap += d;
                    return d;
                }
            }
        }
        return 0;
    }
    ll max_flow(ll s, ll t)
    {
        ll flow = 0;
        for (;;)
        {
            bfs(s);
            if (level[t] < 0)
                return flow;
            memset(vis, 0, sizeof(vis));
            ll f;
            while ((f = dfs(s, t, inf)) > 0)
                flow += f;
        }
    }
```

提高+/省选-

To be done...

未整理

计算几何

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

const double eps = 1e-8;
const double pi = acos(-1.0);
int sgn(double x)
{
    if (fabs(x) < eps)
        return 0;
    if (x < 0)
        return -1;
    else
        return 1;
}
double sq(double x)
{
    return x * x;
}

struct Point
{
    double x, y;
    Point() {}
    Point(double _x, double _y)
    {
        x = _x;
        y = _y;
    }
    bool operator==(const Point &b) const
    {
        return sgn(x - b.x) == 0 && sgn(y - b.y) == 0;
    }
    Point operator+(const Point &b) const
    {
        return Point(x + b.x, y + b.y);
    }
    Point operator-(const Point &b) const
    {
        return Point(x - b.x, y - b.y);
    }
    Point operator*(const double b) const
    {
        return Point(x * b, y * b);
    }
    Point rot(double arc)
    { // counterclockwise
        return Point(x * cos(arc) - y * sin(arc),
```

```
                    x * sin(arc) + y * cos(arc));
    }
    double norm() const
    {
        return sqrt(x * x + y * y);
    }
    double operator^(const Point &b) const
    {
        return x * b.y - b.x * y;
    }
    double operator*(const Point &b) const
    {
        return x * b.x + y * b.y;
    }
    bool operator<(const Point &t)
    {
        if (!sgn(y - t.y))
            return x < t.x;
        return y < t.y;
    }
};

struct Line
{
    Point s, e;
    double k;
    Line() {}
    Line(Point _s, Point _e)
    {
        s = _s;
        e = _e;
        k = atan2(e.y - s.y, e.x - s.x);
    }
    // intersection
    pair<int, Point> operator&(const Line &b) const
    {
        Point res = s;
        if (sgn((s - e) ^ (b.s - b.e)) == 0)
        {
            if (sgn((s - b.e) ^ (b.s - b.e)) == 0)
                return make_pair(0, res); // same
            else
                return make_pair(1, res); // parallel
        }
        double t = ((s - b.s) ^ (b.s - b.e)) / ((s - e) ^ (b.s - b.e));
        res = res + (e - s) * t;
        return make_pair(2, res);
    }
};

struct Circle
{
    Point o;
    double r;
```

```
    Circle() {}
    Circle(Point _o, double _r) : o(_o), r(_r) {}
    bool operator<(const Circle &t) const
    {
        return r < t.r;
    }
    bool in(Circle t)
    {
        return sgn((o - t.o).norm() + (r - t.r)) <= 0;
    }
    bool operator==(const Circle &t) const
    {
        return o == t.o && sgn(r - t.r) == 0;
    }
    double area()
    {
        return pi * r * r;
    }
};

// intersect
bool inter(Line l1, Line l2)
{
    return max(l1.s.x, l1.e.x) >= min(l2.s.x, l2.e.x) &&
           max(l2.s.x, l2.e.x) >= min(l1.s.x, l1.e.x) &&
           max(l1.s.y, l1.e.y) >= min(l2.s.y, l2.e.y) &&
           max(l2.s.y, l2.e.y) >= min(l1.s.y, l1.e.y) &&
           sgn((l2.s - l1.e) ^ (l1.s - l1.e)) * sgn((l2.e - l1.e) ^ (l1.s - l1.e))
<= 0 &&
           sgn((l1.s - l2.e) ^ (l2.s - l2.e)) * sgn((l1.e - l2.e) ^ (l2.s - l2.e))
<= 0;
}
// line l1,segment l2
bool Line_inter_seg(Line l1, Line l2)
{
    return sgn((l2.s - l1.e) ^ (l1.s - l1.e)) * sgn((l2.e - l1.e) ^ (l1.s - l1.e))
<= 0;
}
// closest point to p on l
Point PointToLine(Point P, Line L)
{
    Point result;
    double t = ((P - L.s) * (L.e - L.s)) / ((L.e - L.s) * (L.e - L.s));
    result = L.s + (L.e - L.s) * t;
    return result;
}
// closest point to p on l
Point NearestPointToSeg(Point P, Line L)
{
    Point result;
    double t = ((P - L.s) * (L.e - L.s)) / ((L.e - L.s) * (L.e - L.s));
    if (t >= 0 && t <= 1)
        result = L.s + (L.e - L.s) * t;
    else if ((P - L.s).norm() < (P - L.e).norm())
```

```
        result = L.s;
    else
        result = L.e;
    return result;
}
// p on l
bool OnSeg(Point P, Line L)
{
    return sgn((L.s - P) ^ (L.e - P)) == 0 &&
           sgn((P.x - L.s.x) * (P.x - L.e.x)) <= 0 &&
           sgn((P.y - L.s.y) * (P.y - L.e.y)) <= 0;
}
// 两圆交点
int CirCirIntersect(Circle c1, Circle c2, Point &p1, Point &p2)
{
    Point pcrs;
    double d = (c1.o - c2.o).norm();
    if (d > c1.r + c2.r + eps || d < fabs(c1.r - c2.r) - eps)
        return 0;
    if (fabs(d) < eps)
        return 3; // same center
    double dt = (sq(c1.r) - sq(c2.r)) / d;
    double d1 = (d + dt) / 2;
    Point dir = c2.o - c1.o;
    dir = dir * (1 / dir.norm());
    pcrs = c1.o + dir * d1;
    dt = sqrt(sq(c1.r) - sq(d1));
    dir = dir.rot(pi / 2);
    p1 = pcrs + dir * dt;
    p2 = pcrs - dir * dt;
    if (sgn((p1 - c1.o) ^ (c2.o - c1.o)) < 0)
        swap(p1, p2);
    if (p1 == p2)
        return 1;
    else
        return 2;
}

// integral of \int_{-r}^r 2*sqrt(r^2-x^2)
double func(double r, double x)
{
    return x * sqrt(r * r - x * x) + r * r * asin(x / r);
}

double ArchArea(double r, double h)
{
    return func(r, -r + h) - func(r, -r);
}

// 两圆面积并
double CircleUnionArea(Circle &c1, Circle &c2)
{
    double ds = (c1.o - c2.o).norm();
    if (c1.r + c2.r < ds - eps)
```

```
        return c1.area() + c2.area();
    if (ds < fabs(c1.r - c2.r) + eps)
        return max(c1.area(), c2.area());
    double dk = (sq(c1.r) - sq(c2.r)) / ds;
    double d1, d2; // d1+d2 = ds, d1-d2 = dk;
    d1 = (ds + dk) / 2;
    d2 = ds - d1;
    return ArchArea(c1.r, c1.r + d1) + ArchArea(c2.r, c2.r + d2);
}
// 两圆面积交
double CircleCommonArea(Circle &c1, Circle &c2)
{
    return c1.area() + c2.area() - CircleUnionArea(c1, c2);
}

// 求圆外一点到圆的两个切点
void OutTangentPoint(Point out, Circle cir, Point &p1, Point &p2)
{
    double d2 = sqrt((out - cir.o).norm() - sq(cir.r));
    Circle c;
    c.o = out;
    c.r = d2;
    CirCirIntersect(c, cir, p1, p2);
}

//convex
Point p[maxn], res[maxn];
bool cmp(const Point &a, const Point &b)
{
    double tem = (p[0] - a) ^ (p[0] - b);
    if (!sgn(tem))
        return sgn((a - p[0]).norm() - (b - p[0]).norm()) < 0;
    else
        return sgn(tem) > 0;
}
ll Graham()
{
    int size = 0;
    for (int i = 1; i < n; i++)
        if (p[i] < p[0])
            swap(p[i], p[0]);
    sort(p + 1, p + n, cmp);
    res[size++] = p[0];
    for (int i = 1; i < n; i++)
    {
        while (size > 1 && sgn((res[size - 1] - res[size - 2]) ^ (p[i] - res[size
- 1])) <= 0)
            size--;
        res[size++] = p[i];
    }
    res[size] = res[0];
    return size;
}
```

最小费用最大流

```
//negative edges only, not allowed negative circles
const int inf = 1e8;
struct node
{
    int to;
    int cap;
    int cost;
    int rev;
};

vector<node> edge[maxn];
int h[maxn];
int dis[maxn];
pair<int, int> pre[maxn];
int vis[maxn];

void add_edge(int from, int to, int cap, int cost)
{
    edge[from].push_back({to, cap, cost, (int)edge[to].size()});
    edge[to].push_back({from, 0, -cost, (int)edge[from].size() - 1});
}

void spfa(int s)
{
    memset(vis, 0, sizeof(vis));
    memset(h, 0x7f, sizeof(h));
    queue<int> q;
    h[s] = 0;
    vis[s] = 1;
    q.push(s);
    while (!q.empty())
    {
        int v = q.front();
        q.pop();
        vis[v] = 0;
        for (int i = 0; i < edge[v].size(); i++)
        {
            node e = edge[v][i];
            if (e.cap && h[e.to] > h[v] + e.cost)
            {
                h[e.to] = h[v] + e.cost;
                if (!vis[e.to])
                {
                    vis[e.to] = 1;
                    q.push(e.to);
                }
            }
        }
    }
```

```
}

bool dijkstra(int s, int t)
{
    priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int,
int>>> q;
    fill(dis, dis + maxn, inf);
    memset(vis, 0, sizeof(vis));
    dis[s] = 0;
    q.push(make_pair(0, s));
    while (!q.empty())
    {
        int v = q.top().second;
        q.pop();
        if (vis[v])
            continue;
        vis[v] = 1;
        for (int i = 0; i < edge[v].size(); i++)
        {
            node e = edge[v][i];
            if (e.cap && dis[e.to] > dis[v] + e.cost + h[v] - h[e.to])
            {
                dis[e.to] = dis[v] + e.cost + h[v] - h[e.to];
                pre[e.to] = make_pair(v, i);
                if (!vis[e.to])
                    q.push(make_pair(dis[e.to], e.to));
            }
        }
    }
    return dis[t] != inf;
}

pair<int, int> min_cost_max_flow(int s, int t)
{
    spfa(s);
    int maxf = 0, minc = 0;
    while (dijkstra(s, t))
    {
        int minf = inf;
        for (int i = 0; i < maxn; i++)
            h[i] += dis[i];
        for (int i = t; i != s; i = pre[i].first)
            minf = min(minf, edge[pre[i].first][pre[i].second].cap);
        for (int i = t; i != s; i = pre[i].first)
        {
            edge[pre[i].first][pre[i].second].cap -= minf;
            edge[i][edge[pre[i].first][pre[i].second].rev].cap += minf;
        }
        maxf += minf;
        minc += minf * h[t];
    }
    return make_pair(maxf, minc);
}
```

min25筛

```
#include <bits/stdc++.h>
#define ll long long
#define maxn 20000010 // 2sqrtn
using namespace std;

const ll p = 1000000007;
ll ksm(ll a, ll b)
{
    ll rtn = 1;
    while (b)
    {
        if (b & 1)
            rtn = rtn * a % p;
        a = a * a % p;
        b >>= 1;
    }
    return rtn;
}

ll n, N;
vector<ll> pr;
bool isp[maxn];
int id1[maxn], id2[maxn], m;
ll g[maxn], v[maxn];

inline int get(ll x) { return x < N ? id1[x] : id2[n / x]; }
inline ll fp(ll x) { return 1; }
inline ll presum(ll x) { return x; } // presum of f viewed as poly limited on
prime
inline ll fpk(ll x, ll e) { return ksm(x, e / 2 * 2); }

void init()
{
    pr.push_back(0);
    isp[0] = isp[1] = 1;
    for (ll i = 2; i <= N; i++)
        if (!isp[i])
        {
            pr.push_back(i);
            for (ll j = i * i; j <= N; j += i)
                isp[j] = 1;
        }
}

void getg()
{
    // g(v[j],0),v[j]=all n/i
    for (ll l = 1, r; l <= n; l = r + 1)
    {
```

```
        r = n / (n / l);
        v[++m] = n / l;
        if (v[m] < N)
            id1[v[m]] = m;
        else
            id2[n / v[m]] = m;
        g[m] = (presum(v[m]) - 1 + p) % p; // presum expect 1
    }
    // g(v[j],inf)
    for (int i = 1; i < pr.size(); i++)
    {
        for (int j = 1; j <= m; j++)
        {
            if (pr[i] * pr[i] > v[j])
                break;
            g[j] = (g[j] - fp(pr[i]) * (g[get(v[j] / pr[i])] - g[get(pr[i - 1])] +
p) % p + p) % p;
        }
    }
}

ll gets(ll x, ll y)
{
    if (pr[y] >= x)
        return 0;
    ll res = (g[get(x)] - g[get(pr[y])] + p) % p;
    for (int i = y + 1; i < pr.size() && pr[i] <= x / pr[i]; i++)
    {
        ll w = pr[i];
        for (int j = 1; w <= x / pr[i]; j++, w = w * pr[i]) // because x/w in the
next line, %p not allowed
            res = (res + fpk(pr[i], j) * gets(x / w, i) % p + fpk(pr[i], j + 1)) %
p;
    }
    return res;
}

int main()
{
    cin >> n;
    N = sqrt(n);
    init();
    getg();
    cout << (gets(n, 0) + 1) % p << endl;
}
```

NTT

```
const ll p = 998244353;
ll ksm(ll a, ll b)
{
```

```
    ll rtn = 1;
    while (b)
    {
        if (b & 1)
            rtn = rtn * a % p;
        a = a * a % p;
        b >>= 1;
    }
    return rtn;
}

ll inv3 = ksm(3, p - 2);
ll invi[maxn * 4], invlim;
void NTT(vector<ll> &a, int lim, int type)
{
    for (int i = 0; i < lim; ++i)
        if (i < invi[i])
            swap(a[i], a[invi[i]]);
    for (int i = 2; i <= lim; i <<= 1)
    {
        int mid = i >> 1;
        ll Wn = ksm(~type ? 3 : inv3, (p - 1) / i), t, w;
        for (int j = 0; j < lim; j += i)
        {
            ll w = 1;
            for (int k = 0; k < mid; ++k, w = w * Wn % p)
                a[j + k + mid] = (a[j + k] - (t = w * a[j + k + mid] % p) + p) %
p,
                            a[j + k] = (a[j + k] + t) % p;
        }
    }
    if (type == -1)
        for (int i = 0; i < lim; ++i)
            a[i] = a[i] * invlim % p;
}

void mul(vector<ll> &a, vector<ll> b)
{
    int lim = 1, len = 0, initsize = a.size() + b.size();
    while (lim <= initsize)
        lim <<= 1, len++;
    a.resize(lim);
    b.resize(lim);
    invlim = ksm(lim, p - 2);
    for (int i = 1; i < lim; ++i)
        invi[i] = (invi[i >> 1] >> 1) | ((i & 1) << len - 1);
    NTT(a, lim, 1);
    NTT(b, lim, 1);
    for (int i = 0; i < lim; ++i)
        a[i] = a[i] * b[i] % p;
    NTT(a, lim, -1);
    a.resize(initsize - 1);
}
```

```
void polyksm(vector<ll> &a, ll e)
{
    int lim = 1, len = 0, initsize = a.size() * e;
    while (lim <= initsize)
        lim <<= 1, len++;
    a.resize(lim);
    invlim = ksm(lim, p - 2);
    for (int i = 1; i < lim; ++i)
        invi[i] = (invi[i >> 1] >> 1) | ((i & 1) << len - 1);
    NTT(a, lim, 1);
    for (int i = 0; i < lim; ++i)
        a[i] = ksm(a[i], e);
    NTT(a, lim, -1);
    a.resize(initsize - e + 1);
}
```

rabbin+rho

```
ll gcd(ll a, ll b) { return b ? gcd(b, a % b) : a; }
ll qmul(ll a, ll b, ll p)
{
    a %= p;
    b %= p;
    ll ans = 0;
    while (b)
    {
        if (b & 1)
        {
            ans = (ans + a) % p;
        }
        a = (a <<= 1) % p;
        b >>= 1;
    }
    return ans % p;
}
ll ksm(ll a, ll b, ll p)
{
    ll rtn = 1;
    a %= p;
    while (b)
    {
        if (b & 1)
            rtn = qmul(rtn, a, p);
        a = qmul(a, a, p);
        b >>= 1;
    }
    return rtn;
}
#define C 2730
#define S 3
ll n, m;
```

```
vector<ll> ans;

bool check(ll a, ll p)
{
    ll m = p - 1, x, y;
    int i, j = 0;
    while (!(m & 1))
        m >>= 1, j++;
    x = ksm(a, m, p);
    for (i = 1; i <= j; x = y, i++)
    {
        y = qmul(x, x, p);
        if ((y == 1) && (x != 1) && (x != p - 1))
            return 1;
    }
    return y != 1;
}
bool miller_rabin(int times, ll n)
{
    if (n == 1)
        return 0;
    if (n == 2)
        return 1;
    if (!(n & 1))
        return 0;
    while (times--)
        if (check(rand() % (n - 1) + 1, n))
            return 0;
    return 1;
}
ll pollard_rho(ll n, int c)
{
    ll i = 1, k = 2, x = rand() % n, y = x, d;
    while (1)
    {
        i++, x = (qmul(x, x, n) + c) % n, d = gcd((ll)fabs(double(y - x)), n);
        if (d > 1 && d < n)
            return d;
        if (y == x)
            return n;
        if (i == k)
            y = x, k <<= 1;
    }
}
void findfac(ll n, int c)
{
    if (n == 1)
        return;
    if (miller_rabin(S, n))
    {
        ans.push_back(n);
        return;
    }
    ll m = n;
```

```
    while (m == n)
        m = pollard_rho(n, c--);
    findfac(m, c), findfac(n / m, c);
}
```