```cpp
/* 2-SAT */

const int N = 1e5 + 5;
int n, m;
vector<int> v[2*N];

int norm(int x) {
        if(x < 0) return (-x - 1) * 2 + 1;
        return 2 * (x - 1);
}

void add_edge(int x, int y) {
        x = norm(x);
        y = norm(y);
        v[x].pb(y);
}

void add_alt(int x, int y) {
        add_edge(-x, y);
        add_edge(-y, x);
}

stack<int> S;
bool inv1[2*N], inv2[2*N];
int spoj[2*N], spojwsk;

void dfs1(int x) {
        inv1[x] = 1;
        for(auto y : v[x]) if(!inv1[y]) dfs1(y);
        S.push(x);
}

void dfs2(int x) {
        inv2[x] = 1;
        spoj[x] = spojwsk;
        for(auto y : v[x^1]) if(!inv2[y^1]) dfs2(y ^ 1);
}

bool sat() {
        for(int i=0;i<2*n;i++) if(!inv1[i]) dfs1(i);
        while(!S.empty()) {
                int x = S.top();
                S.pop();
                if(!inv2[x]) {
                        spojwsk++;
                        dfs2(x);
                }
        }

        for(int i=0;i<2*n;i++) if(spoj[i] == spoj[i^1]) return 0;
        return 1;
}

vector<int> spojne[2*N];
bool ozn[2*N];
bool rozw[N];

void roz() {
        for(int i=0;i<2*n;i++) spojne[spoj[i]].pb(i);
        for(int i=2*n;i>=1;i--) for(auto x : spojne[i]) {
                if(!ozn[x^1]) ozn[x] = 1;
        }

        for(int i=0;i<n;i++) if(ozn[2*i]) rozw[i] = 1;
}

/* AHO-CORAICK */

const int N = 2e5 + 5, S = 26;
int n, trie[N][S], siz = 1, cnt[N], pref[N], suf[N];
int fail[N];
string t, s[N];

void add(string s) {
        int v = 1;
        for(int i=0;i<s.length();i++) {
                int let = s[i] - 'a';
                if(trie[v][let] == 0) {
                        siz++;
                        trie[v][let] = siz;
                }
                v = trie[v][let];
        }
        cnt[v]++;
}

void build() {
        queue<int> q;
        while(!q.empty()) q.pop();
```

```
        fail[1] = 1;
        for(int i=0;i<S;i++) if(trie[1][i] == 0) {
                trie[1][i] = 1;
        }
        else {
                fail[trie[1][i]] = 1;
                q.push(trie[1][i]);
        }

        while(!q.empty()) {
                int v = q.front();
                q.pop();
                for(int i=0;i<S;i++) {
                        int x = trie[v][i];
                        if(x == 0) {
                                trie[v][i] = trie[fail[v]][i];
                        }
                        else {
                                q.push(x);
                                fail[x] = trie[fail[v]][i];
                        }
                }
                cnt[v] += cnt[fail[v]];
        }
}

/* Dinic */

const int N = 2e3 + 5;
const long long INF2 = 2e18, INF = 2e9 + 5;
const long long LOGQ = 30;

class Graph {
        public:

        int n, t, s;
        map<int, long long> cap[N];
        vector<int> NG[N];
        int odl[N];
        bool taken[N];
        long long flow;

        Graph(int n, int s, int t) {
                this->n = n;
                this->s = s;
                this->t = t;
                for(int i=0;i<=n;i++) cap[i].clear();
                flow = 0;
        }

        void add_edge(int x, int y, long long w) {
                cap[x][y] = w;
                cap[y][x] = 0;
        }

        long long dfs(int x, long long fl) {
                if(x == t) return fl;
                long long ans = 0;
                while(!NG[x].empty() && fl > 0) {
                        int y = NG[x].back();
                        long long re = 0;
                        long long val = cap[x][y];
                        if(val > 0) re = dfs(y, min(fl, val));
                        if(re) {
                                fl -= re;
                                ans += re;
                                cap[x][y] -= re;
                                cap[y][x] += re;
                        }
                        NG[x].pop_back();
                }
                return ans;
        }

        bool Dinic(long long lim) {
                for(int i=0;i<=n;i++) {
                        odl[i] = INF;
                        NG[i].clear();
                        taken[i] = 0;
                }

                odl[s] = 0;
                taken[s] = 1;
                vector<int> q = {s};

                for(int i=0;i<q.size();i++) {
                        int x = q[i];
                        for(auto y : cap[x]) if(y.second >= lim && odl[y.first] ==
                                INF) {
                                odl[y.first] = odl[x] + 1;
                                q.pb(y.first);
```

```
                        }
                }

                if(odl[t] == INF) return 0;

                for(auto x : q)
                for(auto p : cap[x]) {
                        int y = p.first;
                        long long d = p.second;
                        if(odl[y] == odl[x] + 1 && d >= lim && taken[x]) {
                                NG[x].pb(y);
                                taken[y] = 1;
                        }
                }

                flow += dfs(s, INF2);
                return 1;
        }

        long long maxflow() {
                for(int i=LOGQ;i>=0;i--) while(Dinic((1<<i)));
                return flow;
        }
};

/* FFT */

const long double PI = 2LL * acos(-1);

struct cmx {
        long double x, y;
};
cmx operator +(cmx &a, cmx &b) { return {a.x + b.x, a.y + b.y}; }
cmx operator -(cmx &a, cmx &b) { return {a.x - b.x, a.y - b.y}; }
cmx operator *(cmx &a, cmx &b) { return {a.x * b.x - a.y * b.y, a.x * b.y + a.y * b.
    x}; }
cmx ll(long long x) {return {(long double)x, 0LL}; }

typedef vector<long long> VI;
typedef vector<cmx> CI;

int bit_reverse(int x, int logn) {
        int n = 0;
        for(int i=0;i<logn;i++) {
                n <<= 1;
                n |= (x & 1);
                x >>= 1;
        }
        return n;
}

CI FFT(CI &A, bool num) {
        int n = A.size();

        int logn = 0;
        int nn = n;
        while(nn) {
                nn /= 2;
                logn++;
        }
        logn--;

        CI B = {};
        B.resize(n);

        for(int i=0;i<n;i++) {
                int rev = bit_reverse(i, logn);
                B[i] = A[rev];
        }

        for(int s=1;s<=logn;s++) {
                int m = 1 << s;
                int m2 = m / 2;
                cmx omega = {1, 0};
                cmx pom = {cos(PI / (long double)m), sin(PI / (long double)m)};
                if(num) pom.y = -pom.y;
                for(int j=0;j<m2;j++) {
                        for(int k=j;k<n;k+=m) {
                                cmx t = omega * B[k + m2];
                                cmx u = B[k];
                                B[k] = u + t;
                                B[k + m2] = u - t;
                        }
                        omega = omega * pom;
                }
        }

        return B;
}

CI &to_size(CI &A, int size) {
```

```
        A.resize(size);
        return A;
}

VI operator *(VI &A, VI &B) {
        CI AC = {};
        CI BC = {};

        for(auto x : A) AC.pb(ll(x));
        for(auto x : B) BC.pb(ll(x));

        int size = A.size() + B.size() - 1;
        int p = 1;
        while(p < size) p *= 2;
        size = p;

        AC = to_size(AC, size);
        BC = to_size(BC, size);

        AC = FFT(AC, false);
        BC = FFT(BC, false);

        CI CC;
        CC.resize(size);
        for(int i=0;i<size;i++) CC[i] = AC[i] * BC[i];

        CC = FFT(CC, true);
        long double t = size;

        VI C;
        C.resize(size);
        for(int i=0;i<size;i++) C[i] = round(CC[i].x / t);

        while(C.back() == 0) C.pop_back();

        return C;
}

/* NTT */

const long long MOD = 998244353;
const long long G = 3;

typedef vector<int> VI;

long long pot(long long a, long long p) {
        if(p == 0) return 1;
        long long w = pot(a, p / 2);
        w = w * w % MOD;
        if(p % 2) w = (w * a) % MOD;
        return w;
}

int bit_reverse(int x, int logn) {
        int n = 0;
        for(int i=0;i<logn;i++) {
                n <<= 1;
                n |= (x & 1);
                x >>= 1;
        }
        return n;
}

VI NTT(VI &A, bool num) {
        int n = A.size();
        int logn = 0;
        int nn = n;
        while(nn) {
                nn /= 2;
                logn++;
        }
        logn--;

        VI B = {};
        B.resize(n);

        for(int i=0;i<n;i++) {
                int rev = bit_reverse(i, logn);
                B[i] = A[rev];
        }

        for(int s=1;s<=logn;s++) {
                int m = 1 << s;
                int m2 = m / 2;
                long long root = 1;
                long long p = (MOD - 1) / (long long)m;
                long long pom = pot(G, p);
                if(num) pom = pot(pom, MOD - 2);

                for(int j=0;j<m2;j++) {
                        for(int k=j;k<n;k+=m) {
```

```
                                        long long t = root * (long long)B[k+m2];
                                        long long u = B[k];
                                        B[k] = (u + t) % MOD;
                                        B[k+m2] = (u - t) % MOD;
                                        if(B[k+m2] < 0) B[k+m2] += MOD;
                                }
                                root = (root * pom) % MOD;
                        }
                }

                return B;
}

VI &to_size(VI &A, int size) {
        A.resize(size);
        return A;
}

VI operator *(VI &A, VI &B) {
        int size = A.size() + B.size() - 1;
        int p = 1;
        while(p < size) p *= 2;
        size = p;

        A = to_size(A, size);
        B = to_size(B, size);

        VI AC = NTT(A, false);
        VI BC = NTT(B, false);

        VI C;
        C.resize(size);
        for(int i=0;i<size;i++) C[i] = ((long long)AC[i] * (long long)BC[i]) % MOD;
        C = NTT(C, true);
        long long inv = pot(size, MOD - 2);
        for(int i=0;i<size;i++) C[i] = ((long long)C[i] * inv) % MOD;
        while(C.back() == 0) C.pop_back();

        return C;
}

/* Hungarian */

#define REP(i, n) for (int i = 0; i < int(n); ++i)
vector<int> hungarian2(const vector<vector<int>> &w) {
  const int n = w.size();
  vector<int> one(n, -1), two(n, -1), L(n), R(n), par(n);
  REP(i, n) L[i] = *max_element(w[i].begin(), w[i].end());
  REP(rep, n) {
    vector<bool> left(n), right(n);
    vector<int> slack(n, INT_MAX), q;
    int x = -1;
    REP(i, n) if (one[i] == -1) q.push_back(i);
    while (x == -1) {
      REP(z, q.size()) {
        int a = q[z];
        left[a] = true;
        REP(b, n) {
          int tmp = L[a] + R[b] - w[a][b];
          if (!right[b] && tmp < slack[b]) {
            par[b] = a;
            slack[b] = tmp;
            if (tmp == 0) {
              right[b] = true;
              if (two[b] != -1)
                q.push_back(two[b]);
              else {
                x = b;
                goto koniec;
              }
            }
          }
        }
      }
      int val = INT_MAX;
      REP(i, n) if (!right[i]) val = min(val, slack[i]);
      REP(i, n) {
        if (left[i])
          L[i] -= val;
        if (right[i])
          R[i] += val;
        else if ((slack[i] -= val) == 0) {
          right[i] = true;
          if (two[i] != -1)
            q.push_back(two[i]);
          else
            x = i;
        }
      }
    }
  koniec:
```

```
    while (x != -1) {
      int tmp = one[par[x]];
      one[par[x]] = x;
      two[x] = par[x];
      x = tmp;
    }
  }
  return one;
}

/* Link-Cut */

struct Splay {
  Splay *l = nullptr, *r = nullptr, *p = nullptr;
  bool flip = false;
  int roz = 1;
  int axroz = 1; // SUBTREE Pomocniczny rozmiar poddrzewa.
  void update() {
    assert(!flip and (!l or !l->flip) and (!r or !r->flip));
    axroz = roz;
    if (l)
      axroz += l->axroz; // SUBTREE
    if (r)
      axroz += r->axroz; // SUBTREE
  }
  void touch() {
    if (flip) {
      swap(l, r);
      if (l)
        l->flip = !l->flip;
      if (r)
        r->flip = !r->flip;
      flip = false;
    }
  }
  bool sroot() { return !p or (p->l != this and p->r != this); }
  void connect(Splay *c, bool left) {
    (left ? l : r) = c;
    if (c)
      c->p = this;
  }
  void rotate() {
    Splay *f = p;
    Splay *t = f->p;
    const bool isr = f->sroot();
    const bool left = (this == f->l);
    f->connect(left ? r : l, left);
    connect(f, !left);
    if (isr)
      p = t;
    else
      t->connect(this, f == t->l);
    f->update();
  }
  void push() {
    sroot() ? touch() : p->push();
    if (l)
      l->touch();
    if (r)
      r->touch();
  }
  void splay() {
    push();
    while (!sroot()) {
      Splay *x = p->p;
      if (!p->sroot())
        (((p->l == this) == (x->l == p)) ? p : this)->rotate();
      rotate();
    }
    update();
  }
  Splay *expose() {
    Splay *q = this, *x = nullptr;
    while (q) {
      q->splay();
      if (q->r)
        q->roz += q->r->axroz; // SUBTREE
      if (x)
        q->roz -= x->axroz;
      q->r = x;
      q->update();
      x = q;
      q = q->p;
    }
    splay();
    return x;
  }
  // Zwraca roota drzewowego (nie splejowego!).
  Splay *root() {
    expose();
    Splay *s = this;
```

```
    while (s->touch(), s->l)
      s = s->l;
    s->splay();
    return s;
  }
  void cut() {
    expose();
    assert(l /* Nie jest rootem. */);
    Splay *s = l;
    while (s->touch(), s->r)
      s = s->r;
    s->splay();
    s->r->p = nullptr;
    s->r = nullptr;
  }
  void link(Splay *to) {
    expose();
    assert(!l /* Jest rootem. */);
    p = to;
    p->expose();
    p->roz += axroz;
    p->axroz += axroz; // SUBTREE
  }
  void make_root() {
    expose();
    flip = !flip;
    touch();
  }
};

/* Manacher */

pair< vector<int>, vector<int> > manacher(string &s) {
        int n = s.length();

        vector<int> d1 (n);
        int l=0, r=-1;
        for (int i=0; i<n; ++i) {
                int k = (i>r ? 0 : min (d1[l+r-i], r-i)) + 1;
                while (i+k < n && i-k >= 0 && s[i+k] == s[i-k])  ++k;
                d1[i] = k--;
                if (i+k > r)
                        l = i-k,  r = i+k;
        }

        vector<int> d2 (n);
        l=0, r=-1;
        for (int i=0; i<n; ++i) {
                int k = (i>r ? 0 : min (d2[l+r-i+1], r-i+1)) + 1;
                while (i+k-1 < n && i-k >= 0 && s[i+k-1] == s[i-k])  ++k;
                d2[i] = --k;
                if (i+k-1 > r)
                        l = i-k,  r = i+k-1;
        }

        return mp(d1, d2);
}

/* Massey */

const int mod = 1e9 + 7;
void add_self(int &a, int b) {
        a += b;
        a %= mod;
}

void sub_self(int &a, int b) {
        a -= b;
        a %= mod;
        if(a < 0) a += mod;
}

long long mul(long long a, long long b) {
        return (__int128) a * b % mod;
}

long long my_pow(long long a, long long p) {
        if(p == 0) return 1;
        long long w = my_pow(a, p / 2);
        w = w * w % mod;
        if(p % 2) w = w * a % mod;
        return w;
}

long long my_inv(long long a) {
        return my_pow(a, mod - 2);
}

struct Massey {
  vector<int> start, coef; // 3 optional lines
  vector<vector<int>> powers;
```

```
  int memo_inv;
  int L;
  Massey(vector<int> in) { // O(N^2)
    L = 0;
    int N = in.size();
    vector<int> C{1}, B{1};
    for (int n = 0; n < N; ++n) {
      assert(0 <= in[n] && in[n] < mod); // invalid input
      B.insert(B.begin(), 0);
      int d = 0;
      for (int i = 0; i <= L; ++i)
        add_self(d, mul(C[i], in[n - i]));
      if (d == 0)
        continue;
      vector<int> T = C;
      C.resize(max(B.size(), C.size()));
      for (int i = 0; i < (int)B.size(); ++i)
        sub_self(C[i], mul(d, B[i]));
      if (2 * L <= n) {
        L = n + 1 - L;
        B = T;
        d = my_inv(d);
        for (int &x : B)
          x = mul(x, d);
      }
    }
    for (int i = 1; i < (int)C.size(); ++i)
      coef.push_back((mod - C[i]) % mod);
    assert((int)coef.size() == L);
    for (int i = 0; i < L; ++i)
      start.push_back(in[i]);
    while (!coef.empty() && !coef.back()) {
      coef.pop_back();
      --L;
    }
    if (!coef.empty())
      memo_inv = my_inv(coef.back());
    powers.push_back(coef);
  }

  vector<int> mul_cut(vector<int> a, vector<int> b) {
    vector<int> r(2 * L - 1);
    for (int i = 0; i < L; ++i)
      for (int j = 0; j < L; ++j)
        add_self(r[i + j], mul(a[i], b[j]));
    while ((int)r.size() > L) {
      int value = mul(r.back(), memo_inv); // div(r.back(), coef.back());
      const int X = r.size();
      add_self(r[X - L - 1], value);
      for (int i = 0; i < L; ++i)
        sub_self(r[X - L + i], mul(value, coef[i]));
      assert(r.back() == 0);
      r.pop_back();
    }
    return r;
  }
  int get(long long k) { // O(L^2 * log(k))
    if (k < (int)start.size())
      return start[k];
    if (L == 0)
      return 0;
    k -= start.size();
    vector<int> vec = coef;
    for (int i = 0; (1LL << i) <= k; ++i) {
      if (i == (int)powers.size())
        powers.push_back(mul_cut(powers.back(), powers.back()));
      if (k & (1LL << i))
        vec = mul_cut(vec, powers[i]);
    }
    int total = 0;
    for (int i = 0; i < L; ++i)
      add_self(total, mul(vec[i], start[(int)start.size() - 1 - i]));
    return total;
  }

};

/* Rho */

vector<long long> witness = {2, 325, 9375, 28178, 450775, 9780504, 1795265022};

long long mul(long long a, long long b, long long mod) {
        return (__int128) a * b % mod;
}

long long pot(long long a, long long p, long long mod) {
        if(p == 0) return 1;
        long long w = pot(a, p / 2, mod);
        w = w * w % mod;
        if(p % 2) w = w * a % mod;
        return w;
```

```
}

bool test(long long n) {
        if(n == 2) return true;
        if(n < 2 || n % 2 == 0) return false;
        long long d = n - 1, s = 0;
        while(d % 2 == 0) {
                d /= 2;
                ++s;
        }
        for(auto i : witness) if(i % n) {
                long long x = pot(i, d, n);
                if(x == 1) continue;
                bool zlozona = true;
                for(int j = 0; j < s; ++j) {
                        if(x == n - 1) {
                                zlozona = false;
                                break;
                        }
                        x = mul(x, x, n);
                }
                if(zlozona) return false;
        }
        return true;
}

long long f(long long x, long long mod, long long c) {
        long long y = mul(x, x, mod) + c;
        if(y > mod) y -= mod;
        return y;
}

void rho(long long n, vector<long long> & w) {
        if(n <= 1) return;
        if(test(n)) {
                w.push_back(n);
                return;
        }
        for(long long c = 1; true; ++c) {
                long long x = 2, y = 2, d = 1;
                while(d == 1) {
                        x = f(x, n, c);
                        y = f( f(y,n,c), n, c);
                        d = __gcd(abs(x - y), n);
                }
                if(d < n) {
                        rho(d, w);
                        rho(n / d, w);
                        return;
                }
        }
}

vector<long long> rozklad(long long n) {
        vector<long long> w;
        for(int i = 2; i <= 100; ++i) while(n % i == 0) {
                n /= i;
                w.push_back(i);
        }
        rho(n, w);
        sort(w.begin(), w.end());
        return w;
}
```