National Research University Higher School of Economics

Elderly Passion Fruit

Alexandr Nekrasov, Iurii Pustovalov, Igor Markelov

November 13, 2022

Contest (1)

template.cpp

34 lines

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

using namespace std;

using ll = long long;
using ld = long double;
using ull = unsigned long long;

#define pbc push_back
#define mp make_pair
#define all(a) (a).begin(), (a).end()
#define vin(a)      \
  for (auto& i : a) \
  cin >> i

mt19937 rnd(chrono::steady_clock::now().time_since_epoch().count());

template <typename T1, typename T2>
inline void chkmin(T1& x, const T2& y) {
  if (y < x)
    x = y;
}

template <typename T1, typename T2>
inline void chkmax(T1& x, const T2& y) {
  if (x < y)
    x = y;
}

signed main() {
  cin.tie(0)->sync_with_stdio(0);
  cout.precision(20), cout.setf(ios::fixed);
  return 0;
}
```

genfolders.sh

6 lines

```
for f in {a..z}
do
    mkdir $f
    cp template.cpp $f/$f.cpp
    touch $f/in
done
```

hash.sh

3 lines

```
# Hashes a file, ignoring all whitespace and comments. Use for
# verifying that code was correctly typed.
cpp -dD -P -fpreprocessed | tr -d '[:space:]'| md5sum |cut -c-6
```

C++ (2)

GpHashtable.cpp

Description: Hash map with mostly the same API as unordered_map, but ~3x faster. Uses 1.5x memory. Initial capacity must be a power of 2 (if provided).

e44914, 12 lines

```
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;

const int RANDOM = chrono::high_resolution_clock::now().time_since_epoch()
    .count();
struct hasher {
  int operator()(int x) const {
    return x ^ RANDOM;
  }
};

gp_hash_table<int, int, hasher> table;
```

OrderedSet.cpp

Description: A set (not multiset!) with support for finding the n'th element, and finding the index of an element. To get a map, change null_type.

Time: $\mathcal{O}(\log N)$

b4103c, 28 lines

```
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>

typedef __gnu_pbds::tree<int, __gnu_pbds::null_type, std::less<int>,
    __gnu_pbds::rb_tree_tag,
                         __gnu_pbds::tree_order_statistics_node_update>
    oset;

#include <iostream>

int main() {
  oset X;
  X.insert(1);
  X.insert(2);
  X.insert(4);
  X.insert(8);
  X.insert(16);
  std::cout << *X.find_by_order(1) << std::endl;
                                  // 2
  std::cout << *X.find_by_order(2) << std::endl;
                                  // 4
  std::cout << *X.find_by_order(4) << std::endl;
                                  // 16
  std::cout << std::boolalpha << (end(X) == X.find_by_order(6)) << std::
      endl;  // true

  std::cout << X.order_of_key(-5) << std::endl;   // 0
  std::cout << X.order_of_key(1) << std::endl;    // 0
  std::cout << X.order_of_key(3) << std::endl;    // 2
  std::cout << X.order_of_key(4) << std::endl;    // 2
  std::cout << X.order_of_key(400) << std::endl;  // 5
}
```

Geometry (3)

HalfPlaneIntersection.cpp

Description: Finding the intersection of half-planes.
Time: $\mathcal{O}(n \cdot \log(n))$

87f1c8, 108 lines

```
const ld EPS = 1e-9;
ld sq(ld a) {
  return a * a;
}
struct Point {
  ld x, y;
  Point() {
  }
  Point(ld _x, ld _y) {
    x = _x;
    y = _y;
  }
  Point operator-(const Point &other) const {
    return Point(x - other.x, y - other.y);
  }
  ld operator^(const Point &other) const {
    return x * other.y - y * other.x;
  }
  ld len2() const {
    return sq(x) + sq(y);
  }
  ld len() const {
    return sqrt(len2());
  }
};
#define Vec Point
struct line {
  ld a, b, c;
  line() {
  }
  // All points on the left of xy lie in a halfplane
  line(Point x, Point y) : a(y.y - x.y), b(x.x - y.x), c(x.y * y.x - x.x *
       y.y) {
    ld d = Vec(a, b).len();
    a /= d;
    b /= d;
    c /= d;
  }
};
Point cross(line l, line m) {
  ld d = l.b * m.a - l.a * m.b;
  ld dx = l.c * m.b - l.b * m.c;
  ld dy = l.a * m.c - l.c * m.a;
  return Point(dx / d, dy / d);
}
Vec getPoint(line l) {
  return Vec(-l.b, l.a);
}
ld eval(line l, Point a) {
  return l.a * a.x + l.b * a.y + l.c;
}
bool bad(line a, line b, line c) {
  Point x = cross(b, c);
  return eval(a, x) > 0;
}
// Do not forget about the bounding box
vector<Point> hpi(vector<line> lines) {
  sort(all(lines), [](line al, line bl) -> bool {
    Point a = getPoint(al);
    Point b = getPoint(bl);
    if (a.y >= 0 && b.y < 0)
      return 1;
    if (a.y < 0 && b.y >= 0)
      return 0;
    if (a.y == 0 && b.y == 0)
      return a.x > 0 && b.x < 0;
    return (a ^ b) > 0;
  });

  vector<pair<line, int> > st;
  for (int it = 0; it < 2; it++) {
    for (int i = 0; i < lines.size(); i++) {
      bool flag = false;
      while (!st.empty()) {
        if ((getPoint(st.back().first) - getPoint(lines[i])).len() < EPS)
            {
          if (lines[i].c <= st.back().first.c) {
            flag = true;
            break;
          } else {
            st.pop_back();
          }
        } else if ((getPoint(st.back().first) ^ getPoint(lines[i])) < EPS
            / 2) {
          return {};
        } else if (st.size() >= 2 &&
                   bad(st[st.size() - 2].first, st[st.size() - 1].first,
                       lines[i])) {
          st.pop_back();
        } else {
          break;
        }
      }
      if (!flag)
        st.push_back({lines[i], i});
    }
  }

  vector<int> en(lines.size(), -1);
  vector<Point> ans;
  for (int i = 0; i < st.size(); i++) {
    if (en[st[i].second] == -1) {
      en[st[i].second] = i;
      continue;
    }
    for (int j = en[st[i].second]; j < i; j++) {
      ans.push_back(cross(st[j].first, st[j + 1].first));
    }
    break;
  }
  return ans;
}
```

Strings (4)

SuffixArray.cpp

Description: Build suffix array
Time: $\mathcal{O}(n \log(n))$

3caefc, 44 lines

```
vector<int> buildSuffixArray(string& s) {
  // Remove, if you want to sort cyclic shifts
  s += "$";
  int n = s.size();
  vector<int> a(n);
  iota(all(a), 0);
  stable_sort(all(a), [&](int i, int j) { return s[i] < s[j]; });
  vector<int> c(n);
  int cc = 0;
  for (int i = 0; i < n; i++) {
    if (i == 0 || s[a[i]] != s[a[i - 1]]) {
      c[a[i]] = cc++;
    } else {
      c[a[i]] = c[a[i - 1]];
    }
  }
  for (int l = 1; l < n; l *= 2) {
    vector<int> cnt(n);
    for (auto i : c) {
      cnt[i]++;
    }
    vector<int> pref(n);
    for (int i = 1; i < n; i++) {
      pref[i] = pref[i - 1] + cnt[i - 1];
    }
    vector<int> na(n);
    for (int i = 0; i < n; i++) {
      int pos = (a[i] - l + n) % n;
      na[pref[c[pos]]++] = pos;
    }
    a = na;
    vector<int> nc(n);
    cc = 0;
    for (int i = 0; i < n; i++) {
      if (i == 0 || c[a[i]] != c[a[i - 1]] || c[(a[i] + l) % n] != c[(a[i
          - 1] + l) % n]) {
        nc[a[i]] = cc++;
      } else {
        nc[a[i]] = nc[a[i - 1]];
      }
    }
    c = nc;
  }
  return a;
}
```

Lcp.cpp

Description: lcp array
Time: $\mathcal{O}(n)$

fa8216, 26 lines

```
vector<int> buildLCP(string& s, vector<int>& a) {
  int n = s.size();
  vector<int> ra(n);
  for (int i = 0; i < n; i++) {
    ra[a[i]] = i;
  }
  vector<int> lcp(n - 1);
  int cur = 0;
  for (int i = 0; i < n; i++) {
    cur--;
    chkmax(cur, 0);
    if (ra[i] == n - 1) {
      cur = 0;
      continue;
    }
    int j = a[ra[i] + 1];
    while (s[i + cur] == s[j + cur])
      cur++;
    lcp[ra[i]] = cur;
  }
  // for suffixes!!!
  s.pop_back();
  a.erase(a.begin());
  lcp.erase(lcp.begin());
  return lcp;
}
```

Graph (5)

BlossomShrinking.cpp

Description: Maximum matching in general graph
Time: $\mathcal{O}(n^3)$

23839d, 118 lines

```
struct Edge {
  int u, v;
};
const int N = 510;
int n, m;
vector<int> g[N];
vector<Edge> perfectMatching;
int match[N], par[N], base[N];
bool used[N], blossom[N], lcaUsed[N];
int lca(int u, int v) {
  fill(lcaUsed, lcaUsed + n, false);
  while (u != -1) {
    u = base[u];
    lcaUsed[u] = true;
    if (match[u] == -1)
      break;
    u = par[match[u]];
  }
  while (v != -1) {
    v = base[v];
    if (lcaUsed[v])
      return v;
    v = par[match[v]];
  }
  assert(false);
  return -1;
}
void markPath(int v, int myBase, int children) {
  while (base[v] != myBase) {
    blossom[v] = blossom[match[v]] = true;
    par[v] = children;
    children = match[v];
    v = par[match[v]];
  }
}
int findPath(int root) {
  iota(base, base + n, 0);
  fill(par, par + n, -1);
  fill(used, used + n, false);
  queue<int> q;
  q.push(root);
  used[root] = true;
  while (!q.empty()) {
    int v = q.front();
    q.pop();
    for (auto to : g[v]) {
      if (match[v] == to)
        continue;
      if (base[v] == base[to])
        continue;
      if (to == root || (match[to] != -1 && par[match[to]] != -1)) {
        fill(blossom, blossom + n, false);
        int myBase = lca(to, v);
        markPath(v, myBase, to);
        markPath(to, myBase, v);
        for (int u = 0; u < n; ++u) {
          if (!blossom[base[u]])
            continue;
          base[u] = myBase;
          if (used[u])
            continue;
          used[u] = true;
          q.push(u);
        }
      } else if (par[to] == -1) {
        par[to] = v;
        if (match[to] == -1) {
          return to;
        }
        used[match[to]] = true;
        q.push(match[to]);
      }
    }
  }
  return -1;
}
void blossomShrinking() {
  fill(match, match + n, -1);
  for (int v = 0; v < n; ++v) {
    if (match[v] != -1)
      continue;
    int nxt = findPath(v);
    while (nxt != -1) {
      int parV = par[nxt];
      int parParV = match[parV];
      match[nxt] = parV;
      match[parV] = nxt;
      nxt = parParV;
    }
  }
  for (int v = 0; v < n; ++v) {
    if (match[v] != -1 && v < match[v]) {
      perfectMatching.push_back({v, match[v]});
    }
  }
}
signed main() {
  cin >> n;
  int u, v;
  set<pair<int, int>> edges;
  while (cin >> u >> v) {
    --u;
    --v;
    if (u > v)
      swap(u, v);
    if (edges.count({u, v}))
      continue;
    edges.insert({u, v});
    g[u].push_back(v);
    g[v].push_back(u);
  }
  blossomShrinking();
  cout << perfectMatching.size() * 2 << '\n';
  for (auto i : perfectMatching) {
    cout << i.u + 1 << " " << i.v + 1 << "\n";
  }
  return 0;
}
```

Hungarian.cpp

Description: Hungarian algorithm

Time: $\mathcal{O}(n^3)$

5afee5, 41 lines

```
int n, m;
vector<vector<int>> a;
vector<int> u(n + 1), v(m + 1), p(m + 1), way(m + 1);
for (int i = 1; i <= n; ++i) {
  p[0] = i;
  int j0 = 0;
  vector<int> minv(m + 1, INF);
  vector<char> used(m + 1, false);
  do {
    used[j0] = true;
    int i0 = p[j0], delta = INF, j1;
    for (int j = 1; j <= m; ++j)
      if (!used[j]) {
        int cur = a[i0][j] - u[i0] - v[j];
        if (cur < minv[j])
          minv[j] = cur, way[j] = j0;
        if (minv[j] < delta)
          delta = minv[j], j1 = j;
      }
    for (int j = 0; j <= m; ++j)
      if (used[j])
        u[p[j]] += delta, v[j] -= delta;
      else
        minv[j] -= delta;
    j0 = j1;
  } while (p[j0] != 0);
  do {
    int j1 = way[j0];
    p[j0] = p[j1];
    j0 = j1;
  } while (j0);
}

// matching
vector<int> ans(n + 1);
for (int j = 1; j <= m; ++j) {
  ans[p[j]] = j;
}

// cost
int cost = -v[0];
```

Lct.cpp

Description: link cut tree?

Time: $\mathcal{O}(n\log(n))$?

3d8a3f, 142 lines

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

const int MAXN = 1e5 + 228;

struct node {
  node *ch[2];
  node *p;
  bool rev;
  int sz;

  node() {
    ch[0] = ch[1] = p = NULL;
    rev = false;
    sz = 1;
  }
};

int getsz(node *n) {
  return (n == NULL) ? 0 : n->sz;
}

void pull(node *n) {
  n->sz = getsz(n->ch[0]) + getsz(n->ch[1]) + 1;
}

void push(node *n) {
  if (n->rev) {
    if (n->ch[0]) {
      n->ch[0]->rev ^= 1;
    }
    if (n->ch[1]) {
      n->ch[1]->rev ^= 1;
    }
    swap(n->ch[0], n->ch[1]);
    n->rev = 0;
  }
}

bool isRoot(node *n) {
  return n->p == NULL || (n->p->ch[0] != n && n->p->ch[1] != n);
}

int chnum(node *n) {
  return n->p->ch[1] == n;
}

void attach(node *n, node *p, int num) {
  if (n != NULL)
    n->p = p;
  if (p != NULL)
    p->ch[num] = n;
}

void rotate(node *n) {
  int num = chnum(n);
  node *p = n->p;
  node *b = n->ch[1 - num];
  n->p = p->p;
  if (!isRoot(p)) {
    p->p->ch[chnum(p)] = n;
  }
  attach(p, n, 1 - num);
  attach(b, p, num);
  pull(p);
  pull(n);
}

node *qq[MAXN];

void splay(node *n) {
  node *nn = n;
  int top = 0;
  qq[top++] = nn;
  while (!isRoot(nn)) {
    nn = nn->p;
    qq[top++] = nn;
  }
  while (top) {
    push(qq[--top]);
  }
  while (!isRoot(n)) {
    if (!isRoot(n->p)) {
      if (chnum(n) == chnum(n->p)) {
        rotate(n->p);
      } else {
        rotate(n);
      }
    }
    rotate(n);
  }
}

void expose(node *n) {
  splay(n);
  n->ch[1] = NULL;
  pull(n);
  while (n->p != NULL) {
    splay(n->p);
    attach(n, n->p, 1);
    pull(n->p);
    splay(n);
  }
}

void makeRoot(node *n) {
  expose(n);
  n->rev ^= 1;
}

node *nodes[MAXN];

int main() {
  int n;
  cin >> n;
  for (int i = 0; i <= n; i++) {
    nodes[i] = new node();
  }
  int q;
  cin >> q;
  while (q--) {
    string s;
    cin >> s;
    int u, v;
    cin >> u >> v;
    makeRoot(nodes[u]);
    makeRoot(nodes[v]);
    if (s == "get") {
      if (isRoot(nodes[u]) && u != v) {
        cout << "-1" << endl;
      } else {
        cout << getsz(nodes[v]) - 1 << endl;
      }
    } else if (s == "link") {
      nodes[v]->p = nodes[u];
    } else {
      push(nodes[v]);
      nodes[v]->ch[1] = NULL;
      nodes[u]->p = NULL;
    }
  }
}
```

Pushrelabel.cpp

Description: maxflow?

Time: ?

1dbe57, 87 lines

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

typedef long long ll;

struct MaxFlow {
  static const ll INF = 1e18 + 228;  // maybe int?
  struct edge {
    int to, rev;
    ll cap;  // maybe int?
  };

  int n;
  vector<vector<edge>> g;
  vector<ll> ex;  // maybe int?
  vector<int> q;

  ll flow(int t) {  // maybe int?
    while (true) {
      vector<int> dist(n, n);
      dist[t] = 0;
      int l = 0;
      int r = 1;
      q[0] = t;
      while (l != r) {
        int v = q[l++];
        for (auto e : g[v]) {
          if (g[e.to][e.rev].cap > 0 && dist[e.to] > dist[v] + 1) {
            dist[e.to] = dist[v] + 1;
            q[r++] = e.to;
          }
        }
      }
      ll was = ex[t];
      for (int ind = r - 1; ind >= 0; ind--) {
        int v = q[ind];
        if (ex[v] == 0)
          continue;
        for (auto &e : g[v]) {
          if (dist[e.to] + 1 == dist[v] && e.cap > 0) {
            auto f = min(ex[v], e.cap);
            e.cap -= f;
            ex[e.to] += f;
            ex[v] -= f;
            g[e.to][e.rev].cap += f;
          }
        }
      }
      if (was == ex[t]) {
        break;
      }
    }
    return ex[t];
  }
  MaxFlow(int n) : n(n) {
    g.resize(n);
    ex.resize(n);
    q.resize(n);
  }
  ll run(int s, int t) {  // maybe int?
    ex[s] = INF;
    return flow(t);
  }
  void add_edge(int a, int b, int c, int cr = 0) {
    int sza = g[a].size();
    int szb = g[b].size();
    g[a].push_back({b, szb, c});
    g[b].push_back({a, sza, cr});
  }
};

int main() {
  int n;
  cin >> n;
  MaxFlow mf(n);
  int s = 0, t = n - 1;
  int m;
  cin >> m;
  for (int i = 0; i < m; i++) {
    int a, b, c;
    cin >> a >> b >> c;
    a--;
    b--;
    mf.add_edge(a, b, c);
  }
  cout << mf.run(s, t) << endl;
}
```

GlobalMincut.cpp

Description: ?
Time: ?

7b8a6b, 35 lines

```
const int MAXN = 500;
int n, g[MAXN][MAXN];
int best_cost = 1000000000;
vector<int> best_cut;
void mincut() {
  vector<int> v[MAXN];
  for (int i = 0; i < n; ++i)
    v[i].assign(1, i);
  int w[MAXN];
  bool exist[MAXN], in_a[MAXN];
  memset(exist, true, sizeof exist);
  for (int ph = 0; ph < n - 1; ++ph) {
    memset(in_a, false, sizeof in_a);
    memset(w, 0, sizeof w);
    for (int it = 0, prev; it < n - ph; ++it) {
      int sel = -1;
      for (int i = 0; i < n; ++i)
        if (exist[i] && !in_a[i] && (sel == -1 || w[i] > w[sel]))
          sel = i;
      if (it == n - ph - 1) {
        if (w[sel] < best_cost)
          best_cost = w[sel], best_cut = v[sel];
        v[prev].insert(v[prev].end(), v[sel].begin(), v[sel].end());
        for (int i = 0; i < n; ++i)
          g[prev][i] = g[i][prev] += g[sel][i];
        exist[sel] = false;
      } else {
        in_a[sel] = true;
        for (int i = 0; i < n; ++i)
          w[i] += g[sel][i];
        prev = sel;
      }
    }
  }
}
```

Math (6)

GoncharFedor.cpp

Description: Calculating number of points s.t. $x, y \geq 0, Ax + By \leq C$
Time: $\mathcal{O}(\log(C))$

0ef10e, 11 lines

```
ll solve_triangle(ll A, ll B, ll C) {  // x, y >=0, Ax+By<=C
  if (C < 0)
    return 0;
  if (A > B)
    swap(A, B);
  ll p = C / B;
  ll k = B / A;
  ll d = (C - p * B) / A;
  return solve_triangle(B - k * A, A, C - A * (k * p + d + 1)) + (p + 1) *
       (d + 1) +
         k * p * (p + 1) / 2;
}
```

PrimalityTest.cpp

Description: Checking primality of p
Time: $\mathcal{O}(\log(C))$

af473a, 32 lines

```
const int iters = 8;  // can change
bool isprime(ll p) {
  if (p == 1 || p == 4)
    return 0;
  if (p == 2 || p == 3)
    return 1;
  for (int it = 0; it < iters; ++it) {
    ll a = rnd() % (p - 2) + 2;
    ll nw = p - 1;
    while (nw % 2 == 0)
      nw /= 2;
    ll x = binpow(a, nw, p);  // int128
    if (x == 1)
      continue;
    ll last = x;
    nw *= 2;
    while (nw <= p - 1) {
      x = (__int128_t)x * x % mod;
      if (x == 1) {
        if (last != p - 1) {
          return 0;
        }
        break;
      }
      last = x;
      nw *= 2;
    }
    if (x != 1)
      return 0;
  }
  return 1;
}
```

Factorization.cpp

Description: Factorizing a number real quick

Time: $\mathcal{O}\left(n^{\frac{1}{4}}\right)$

f0d7c6, 53 lines

```
ll gcd(ll a, ll b) {
  while (b)
    a %= b, swap(a, b);
  return a;
}

ll f(ll a, ll n) {
  return ((__int128_t)a * a % n + 1) % n;
}

vector<ll> factorize(ll n) {
  if (n <= 1e6) {  // can add primality check for speed?
    vector<ll> res;
    for (ll i = 2; i * i <= n; ++i) {
      while (n % i == 0) {
        res.pbc(i);
        n /= i;
      }
    }
    if (n != 1)
      res.pbc(n);
    return res;
  }
  ll x = rnd() % (n - 1) + 1;
  ll y = x;
  ll tries = 10 * sqrt(sqrt(n));
  const int C = 60;
  for (ll i = 0; i < tries; i += C) {
    ll xs = x;
    ll ys = y;
    ll m = 1;
    for (int k = 0; k < C; ++k) {
      x = f(x, n);
      y = f(f(y, n), n);
      m = (__int128_t)m * abs(x - y) % n;
    }
    if (gcd(n, m) == 1)
      continue;
    x = xs, y = ys;
    for (int k = 0; k < C; ++k) {
      x = f(x, n);
      y = f(f(y, n), n);
      ll res = gcd(n, abs(x - y));
      if (res != 1 && res != n) {
        vector<ll> v1 = factorize(res), v2 = factorize(n / res);
        for (auto j : v2)
          v1.pbc(j);
        return v1;
      }
    }
  }
  return {n};
}
```

XorConvolution.cpp

Description: Calculating xor-convolution of 2 vectors modulo smth

Time: $\mathcal{O}(n\log(n))$

454afd, 23 lines

```
void fwht(vector<int>& a) {
  int n = a.size();
  for (int l = 1; l < n; l <<= 1) {
    for (int i = 0; i < n; i += 2 * l) {
      for (int j = 0; j < l; ++j) {
        int u = a[i + j], v = a[i + j + l];
        a[i + j] = add(u, v), a[i + j + l] = sub(u, v);
      }
    }
  }
}  // https://judge.yosupo.jp/problem/bitwise_xor_convolution
vector<int> xorconvo(vector<int> a, vector<int> b) {
  int n = 1;
  while (n < max(a.size(), b.size()))
    n *= 2;
  a.resize(n), b.resize(n);
  fwht(a), fwht(b);
  int in = inv(n);
  for (int i = 0; i < n; ++i)
    a[i] = mul(a[i], mul(b[i], in));
  fwht(a);
  return a;
}
```

AndConvolution.cpp

Description: Calculating and-convolution modulo smth

Time: $\mathcal{O}(n\log(n))$

5dedf4, 24 lines

```
void conv(vector<int>& a, bool x) {
  int n = a.size();
  for (int j = 0; (1 << j) < n; ++j) {
    for (int i = 0; i < n; ++i) {
      if (!(i & (1 << j))) {
        if (x)
          a[i] = add(a[i], a[i | (1 << j)]);
        else
          a[i] = sub(a[i], a[i | (1 << j)]);
      }
    }
  }
}  // https://judge.yosupo.jp/problem/bitwise_and_convolution
vector<int> andcon(vector<int> a, vector<int> b) {
  int n = 1;
  while (n < max(a.size(), b.size()))
    n *= 2;
  a.resize(n), b.resize(n);
  conv(a, 1), conv(b, 1);
  for (int i = 0; i < n; ++i)
    a[i] = mul(a[i], b[i]);
  conv(a, 0);
  return a;
}
```

NTT.cpp

Description: Calculating FFT modulo MOD

Time: $\mathcal{O}(n\log(n))$

07c259, 75 lines

```
// DONT FORGET TO CALL initNTT() AND CHECK MAXLOG
namespace NTT {
const int MOD = 998244353;
const int MAXLOG = 20;
const int N = (1 << MAXLOG);
const int MAXN = (1 << MAXLOG) + 228;
int rev[MAXN];
int w[MAXN];
int n, m;
int a[MAXN];
int b[MAXN];
int fans[MAXN];
void initNTT() {
  int g = 2;
  for (;; g++) {
    int y = g;
    for (int i = 0; i < MAXLOG - 1; ++i) {
      y = mul(y, y);
    }
    if (y == MOD - 1) {
      break;
    }
  }
  w[0] = 1;
  for (int i = 1; i < N; ++i) {
    w[i] = mul(w[i - 1], g);
  }
  rev[0] = 0;
  for (int i = 1; i < N; ++i) {
    rev[i] = (rev[i >> 1] >> 1) ^ ((i & 1) << (MAXLOG - 1));
  }
}
void NTT(int n, int LOG, int* a) {
  for (int i = 0; i < n; ++i) {
    if (i < (rev[i] >> (MAXLOG - LOG))) {
      swap(a[i], a[(rev[i] >> (MAXLOG - LOG))]);
    }
  }
  for (int lvl = 0; lvl < LOG; lvl++) {
    int len = 1 << lvl;
    for (int st = 0; st < n; st += len << 1) {
      for (int i = 0; i < len; ++i) {
        int x = a[st + i], y = mul(a[st + len + i], w[i << (MAXLOG - 1 -
            lvl)]);
        a[st + i] = add(x, y);
        a[st + i + len] = sub(x, y);
      }
    }
  }
}
void mul() {
  int LOG = 0;
  while ((1 << LOG) < 2 * max(n, m))
    LOG++;
  int sz = 1 << LOG;
  for (int i = n; i < sz; ++i) {
    a[i] = 0;
  }
  for (int i = m; i < sz; ++i) {
    b[i] = 0;
  }
  NTT(sz, LOG, a);
  NTT(sz, LOG, b);
  for (int i = 0; i < sz; ++i) {
    a[i] = mul(a[i], b[i]);
  }
  NTT(sz, LOG, a);
  int inv_sz = inv(sz);
  for (int i = 0; i < sz; ++i) {
    fans[i] = mul(a[i], inv_sz);
  }
  reverse(fans + 1, fans + sz);
}
}  // namespace NTT

// DONT FORGET TO CALL initNTT() AND CHECK MAXLOG
```

FFT.cpp

Description: Calculating product of two polynomials
Time: $\mathcal{O}(n\log(n))$

31c0ce, 60 lines

```
// DONT FORGET TO INITFFT() AND CHECK MAXLOG
namespace FFT {
const int MAXLOG = 20;
const ld PI = acos(-1);
using cd = complex<long double>;
const int N = (1 << MAXLOG);
const int MAXN = (1 << MAXLOG) + 228;
int rev[MAXN];
cd w[MAXN];
int n, m;
cd a[MAXN], b[MAXN];
int fans[MAXN];
void initFFT() {
  for (int i = 0; i < N; i++) {
    w[i] = cd(cos(2 * PI * i / N), sin(2 * PI * i / N));
  }
  rev[0] = 0;
  for (int i = 1; i < N; i++) {
    rev[i] = (rev[i >> 1] >> 1) ^ ((i & 1) << (MAXLOG - 1));
  }
}
void FFT(int n, int LOG, cd* a) {
  for (int i = 0; i < n; i++) {
    if (i < (rev[i] >> (MAXLOG - LOG))) {
      swap(a[i], a[(rev[i] >> (MAXLOG - LOG))]);
    }
  }
  for (int lvl = 0; lvl < LOG; lvl++) {
    int len = 1 << lvl;
    for (int st = 0; st < n; st += len << 1) {
      for (int i = 0; i < len; i++) {
        cd x = a[st + i], y = a[st + len + i] * w[i << (MAXLOG - 1 - lvl)
          ];
        a[st + i] = x + y;
        a[st + i + len] = x - y;
      }
    }
  }
}
void mul() {
  int LOG = 0;
  while ((1 << LOG) < 2 * max(n, m))
    LOG++;
  int sz = 1 << LOG;
  for (int i = n; i < sz; i++)
    a[i] = 0;
  for (int i = m; i < sz; ++i)
    b[i] = 0;
  FFT(sz, LOG, a);
  FFT(sz, LOG, b);
  for (int i = 0; i < sz; i++) {
    a[i] *= b[i];
  }
  FFT(sz, LOG, a);
  for (int i = 0; i < sz; i++) {
    fans[i] = (int)(a[i].real() / sz + 0.5);
  }
  reverse(fans + 1, fans + sz);
}
}  // namespace FFT
// DONT FORGET TO INITFFT() AND CHECK MAXLOG
```

6.1 Fun things

$ClassesCount = \dfrac{1}{|G|}\sum_{\pi\in G} I(\pi)$

$ClassesCount = \dfrac{1}{|G|}\sum_{\pi\in G} k^{C(\pi)}$

Stirling 2kind - count of partitions of n objects into k nonempty sets:

$S(n,k) = S(n-1,k-1) + kS(n-1,k)$

$S(n,k) = \sum_{j=0}^{n-1}\binom{n-1}{j}S(j,k-1)$

$S(n,k) = \frac{1}{k!}\sum_{j=0}^{k}(-1)^{k+j}\binom{k}{j}j^n$

$\binom{n}{k} \equiv \prod_i \binom{n_i}{k_i}$, n_i, k_i - digits of n, k in p-adic system

$\int_a^b f(x)dx \approx \dfrac{b-a}{6}(f(a) + 4f(\dfrac{a+b}{2}) + f(b))$

$x_{n+1} = x_n - \dfrac{f(x_n)}{f'(x_n)}$, $O(loglog)$

$G(n) = n \oplus (n >> 1)$

$g(n) = \sum_{d|n} f(d) \Rightarrow f(n) = \sum_{d|n} g(d)\mu(\frac{n}{d})$

$\sum_{d|n} \mu(d) = [n = 1], \mu(1) = 1, \mu(p) = -1, \mu(p^k) = 0$

$\sin(a \pm b) = \sin a \cos b \pm \sin b \cos a$

$\cos(a \pm b) = \cos a \cos b \mp \sin a \sin b$

$$\operatorname{tg}(a \pm b) = \frac{\operatorname{tg} a \pm \operatorname{tg} b}{1 \mp \operatorname{tg} a \operatorname{tg} b}$$

$$\operatorname{ctg}(a \pm b) = \frac{\operatorname{ctg} a \operatorname{ctg} b \mp 1}{\operatorname{ctg} b \pm \operatorname{ctg} a}$$

$$\sin\frac{a}{2} = \pm\sqrt{\frac{1 - \cos a}{2}}$$

$$\cos\frac{a}{2} = \pm\sqrt{\frac{1 + \cos a}{2}}$$

$$\operatorname{tg}\frac{a}{2} = \frac{\sin a}{1 - \cos a} = \frac{1 - cosa}{sina}$$

$$\sin a \sin b = \frac{\cos(a - b) - \cos(a + b)}{2}$$

$$\sin a \cos b = \frac{\sin(a - b) + \sin(a + b)}{2}$$

$$\cos a \cos b = \frac{\cos(a - b) + \cos(a + b)}{2}$$

Problem	Status	Comment	Iurii	Alex	Igor
A - 1					
B - 2					
C - 3					
D - 4					
E - 5					
F - 6					
G - 7					
H - 8					
I - 9					
J - 10					
K - 11					
L - 12					
M - 13					
N - 14					
O - 15					