Contents

1. Contest

1.1. commands

```sh
alias c='g++ -g --std=c++17 -O2 -Wall -Wextra -Wshadow -D_GLIBCXX_DEBUG
-fsanitize=address -fsanitize=undefined'
```

1.2. template

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

using namespace std;

#define rep(i, a, b) for (int i = a; i < (b); ++i)
#define all(x) begin(x), end(x)
#define sz(x) (int)(x).size()
typedef long long ll;
typedef pair<int, int> pii;
typedef vector<int> vi;

void solve() {}

int main() {
  cin.tie(0)->sync_with_stdio(0);
  cin.exceptions(cin.failbit);
  int tc = 1;
//   cin >> tc;
  for (int i = 1; i <= tc; ++i) {
    solve();
  }
}
```

2. Mathematics

2.1. MillerRabin

```cpp
bool isPrime(ull n) {
  if (n < 2 || n % 6 % 4 != 1) return (n | 1) == 3;
  ull A[] = {2, 325, 9375, 28178, 450775, 9780504, 1795265022},
      s = __builtin_ctzll(n - 1), d = n >> s;
  for (ull a : A) {  // ^ count trailing zeroes
    ull p = modpow(a % n, d, n), i = s;
    while (p != 1 && p != n - 1 && a % n && i--) p = modmul(p, p, n);
    if (p != n - 1 && i != s) return 0;
```

```
  }
  return 1;
}
```

2.2. ModMulPow

```cpp
typedef unsigned long long ull;
ull modmul(ull a, ull b, ull M) {
  ll ret = a * b - M * ull(1.L / M * a * b);
  return ret + M * (ret < 0) - M * (ret >= (ll)M);
}
ull modpow(ull b, ull e, ull mod) {
  ull ans = 1;
  for (; e; b = modmul(b, b, mod), e /= 2)
    if (e & 1) ans = modmul(ans, b, mod);
  return ans;
}
```

2.3. ModLog

```cpp
ll modLog(ll a, ll b, ll m) {
  ll n = (ll)sqrt(m) + 1, e = 1, f = 1, j = 1;
  unordered_map<ll, ll> A;
  while (j <= n && (e = f = e * a % m) != b % m) A[e * b % m] = j++;
  if (e == b % m) return j;
  if (gcd(m, e) == gcd(m, b))
    rep(i, 2, n + 2) if (A.count(e = e * f % m)) return n * i - A[e];
  return -1;
}
```

2.4. ModSQRT

```cpp
ll sqrt(ll a, ll p) {
  a %= p;
  if (a < 0) a += p;
  if (a == 0) return 0;
  assert(modpow(a, (p - 1) / 2, p) == 1);  // else no solution
  if (p % 4 == 3) return modpow(a, (p + 1) / 4, p);
  // a^(n+3)/8 or 2^(n+3)/8 * 2^(n-1)/4 works if p % 8 == 5
  ll s = p - 1, n = 2;
  int r = 0, m;
  while (s % 2 == 0) ++r, s /= 2;
  /// find a non-square mod p
  while (modpow(n, (p - 1) / 2, p) != p - 1) ++n;
  ll x = modpow(a, (s + 1) / 2, p);
  ll b = modpow(a, s, p), g = modpow(n, s, p);
  for (;; r = m) {
    ll t = b;
    for (m = 0; m < r && t != 1; ++m) t = t * t % p;
    if (m == 0) return x;
    ll gs = modpow(g, 1LL << (r - m - 1), p);
    g = gs * gs % p;
    x = x * gs % p;
    b = b * g % p;
  }
}
```

2.5. Factor

```cpp
ull pollard(ull n) {
  ull x = 0, y = 0, t = 30, prd = 2, i = 1, q;
  auto f = [&](ull x) { return modmul(x, x, n) + i; };
  while (t++ % 40 || __gcd(prd, n) == 1) {
    if (x == y) x = ++i, y = f(x);
    if ((q = modmul(prd, max(x, y) - min(x, y), n))) prd = q;
    x = f(x), y = f(f(y));
  }
  return __gcd(prd, n);
}
vector<ull> factor(ull n) {
  if (n == 1) return {};
  if (isPrime(n)) return {n};
  ull x = pollard(n);
  auto l = factor(x), r = factor(n / x);
  l.insert(l.end(), all(r));
  return l;
```

```
}
```

2.6. CRT

```cpp
template <typename T>
struct CRT {
  T res;
  CRT() { res = 0, prd = 1; }
  // Add condition: res % p == r
  void add(T p, T r) {
    res += mul(r - res % p + p, euclid(prd, p).first + p, p) * prd;
    prd *= p;
    if (res >= prd) res -= prd;
  }

 private:
  T prd;
  T mul(T a, T b, T p) {
    a %= p, b %= p;
    T q = (T)((long double)a * b / p);
    T r = a * b - q * p;
    while (r < 0) r += p;
    while (r >= p) r -= p;
    return r;
  }
  pair<T, T> euclid(T a, T b) {
    if (!b) return make_pair(1, 0);
    pair<T, T> r = euclid(b, a % b);
    return make_pair(r.second, r.first - a / b * r.second);
  }
};
```

2.7. DivModSum

```cpp
typedef unsigned long long ull;
ull sumsq(ull to) { return to / 2 * ((to - 1) | 1); }
/// ^ written in a weird way to deal with overflows correctly

ull divsum(ull to, ull c, ull k, ull m) {
  ull res = k / m * sumsq(to) + c / m * to;
  k %= m;
  c %= m;
  if (!k) return res;
  ull to2 = (to * k + c) / m;
  return res + (to - 1) * to2 - divsum(to2, m - 1 - c, m, k);
}

ll modsum(ull to, ll c, ll k, ll m) {
  c = ((c % m) + m) % m;
  k = ((k % m) + m) % m;
  return to * c + k * sumsq(to) - m * divsum(to, c, k, m);
}
```

2.8. LinearRec

```cpp
ll linearRec(Poly S, Poly tr, ll k) {
  int n = sz(tr);

  auto combine = [&](Poly a, Poly b) {
    Poly res(n * 2 + 1);
    rep(i, 0, n + 1) rep(j, 0, n + 1) res[i + j] =
        (res[i + j] + a[i] * b[j]) % mod;
    for (int i = 2 * n; i > n; --i)
      rep(j, 0, n) res[i - 1 - j] = (res[i - 1 - j] + res[i] * tr[j]) % mod;
    res.resize(n + 1);
    return res;
  };

  Poly pol(n + 1), e(pol);
  pol[0] = e[1] = 1;

  for (++k; k; k /= 2) {
    if (k % 2) pol = combine(pol, e);
    e = combine(e, e);
  }
```

```
  ll res = 0;
  rep(i, 0, n) res = (res + pol[i + 1] * S[i]) % mod;
  return res;
}
```

3. Geometry

4. Data Structures

4.1. RMQ

cpp
```
template <class T>
struct RMQ {
  vector<vector<T>> jmp;
  RMQ(const vector<T>& V) : jmp(1, V) {
    for (int pw = 1, k = 1; pw * 2 <= sz(V); pw *= 2, ++k) {
      jmp.emplace_back(sz(V) - pw * 2 + 1);
      rep(j, 0, sz(jmp[k])) jmp[k][j] = min(jmp[k - 1][j], jmp[k - 1][j +
pw]);
    }
  }
  T query(int a, int b) {
    assert(a < b);  // or return inf if a == b
    int dep = 31 - __builtin_clz(b - a);
    return min(jmp[dep][a], jmp[dep][b - (1 << dep)]);
  }
};
```

4.2. DSURollback

cpp
```
struct DSURollback {
  vi e;
  vector<pii> st;
  DSURollback(int n) : e(n, -1) {}
  int size(int x) { return -e[find(x)]; }
  int find(int x) { return e[x] < 0 ? x : find(e[x]); }
  int time() { return sz(st); }
  void rollback(int t) {
    for (int i = time(); i-- > t;) e[st[i].first] = st[i].second;
    st.resize(t);
  }
  bool join(int a, int b) {
    a = find(a), b = find(b);
    if (a == b) return false;
    if (e[a] > e[b]) swap(a, b);
    st.push_back({a, e[a]});
    st.push_back({b, e[b]});
    e[a] += e[b];
    e[b] = a;
    return true;
  }
};
```

5. Graph

5.1. Biconnected

cpp
```
vi num, st;
vector<vector<pii>> ed;
int Time;
template <class F>
int dfs(int at, int par, F& f) {
  int me = num[at] = ++Time, top = me;
  for (auto [y, e] : ed[at])
    if (e != par) {
      if (num[y]) {
        top = min(top, num[y]);
        if (num[y] < me) st.push_back(e);
      } else {
        int si = sz(st);
        int up = dfs(y, e, f);
        top = min(top, up);
        if (up == me) {
          st.push_back(e);
          f(vi(st.begin() + si, st.end()));
```

```
          st.resize(si);
        } else if (up < me)
          st.push_back(e);
        else { /* e is a bridge */
        }
      }
    }
  return top;
}

template <class F>
void bicomps(F f) {
  num.assign(sz(ed), 0);
  rep(i, 0, sz(ed)) if (!num[i]) dfs(i, -1, f);
}
```

5.2. PushRelabel

```
struct PushRelabel {                                                    cpp
  struct Edge {
    int dest, back;
    ll f, c;
  };
  vector<vector<Edge>> g;
  vector<ll> ec;
  vector<Edge*> cur;
  vector<vi> hs;
  vi H;
  PushRelabel(int n) : g(n), ec(n), cur(n), hs(2 * n), H(n) {}

  void addEdge(int s, int t, ll cap, ll rcap = 0) {
    if (s == t) return;
    g[s].push_back({t, sz(g[t]), 0, cap});
    g[t].push_back({s, sz(g[s]) - 1, 0, rcap});
  }

  void addFlow(Edge& e, ll f) {
    Edge& back = g[e.dest][e.back];
    if (!ec[e.dest] && f) hs[H[e.dest]].push_back(e.dest);
    e.f += f;
    e.c -= f;
    ec[e.dest] += f;
    back.f -= f;
    back.c += f;
    ec[back.dest] -= f;
  }
  ll calc(int s, int t) {
    int v = sz(g);
    H[s] = v;
    ec[t] = 1;
    vi co(2 * v);
    co[0] = v - 1;
    rep(i, 0, v) cur[i] = g[i].data();
    for (Edge& e : g[s]) addFlow(e, e.c);

    for (int hi = 0;;) {
      while (hs[hi].empty())
        if (!hi--) return -ec[s];
      int u = hs[hi].back();
      hs[hi].pop_back();
      while (ec[u] > 0)  // discharge u
        if (cur[u] == g[u].data() + sz(g[u])) {
          H[u] = 1e9;
          for (Edge& e : g[u])
            if (e.c && H[u] > H[e.dest] + 1) H[u] = H[e.dest] + 1, cur[u] =
&e;
          if (++co[H[u]], !--co[hi] && hi < v)
            rep(i, 0, v) if (hi < H[i] && H[i] < v)-- co[H[i]], H[i] = v +
1;
          hi = H[u];
        } else if (cur[u]->c && H[u] == H[cur[u]->dest] + 1)
          addFlow(*cur[u], min(ec[u], cur[u]->c));
        else
          ++cur[u];
    }
  }
```

```
  bool leftOfMinCut(int a) { return H[a] >= sz(g); }
};
```

5.3. GomoryHu

```cpp
typedef array<ll, 3> Edge;
vector<Edge> gomoryHu(int N, vector<Edge> ed) {
  vector<Edge> tree;
  vi par(N);
  rep(i, 1, N) {
    PushRelabel D(N);  // Dinic also works
    for (Edge t : ed) D.addEdge(t[0], t[1], t[2], t[2]);
    tree.push_back({i, par[i], D.calc(i, par[i])});
    rep(j, i + 1, N) if (par[j] == par[i] && D.leftOfMinCut(j)) par[j] = i;
  }
  return tree;
}
```

5.4. MinCostMaxFlow

```cpp
const ll INF = numeric_limits<ll>::max() / 4;

struct MCMF {
  struct edge {
    int from, to, rev;
    ll cap, cost, flow;
  };
  int N;
  vector<vector<edge>> ed;
  vi seen;
  vector<ll> dist, pi;
  vector<edge*> par;

  MCMF(int N) : N(N), ed(N), seen(N), dist(N), pi(N), par(N) {}

  void addEdge(int from, int to, ll cap, ll cost) {
    if (from == to) return;
    ed[from].push_back(edge{from, to, sz(ed[to]), cap, cost, 0});
    ed[to].push_back(edge{to, from, sz(ed[from]) - 1, 0, -cost, 0});
  }

  void path(int s) {
    fill(all(seen), 0);
    fill(all(dist), INF);
    dist[s] = 0;
    ll di;

    __gnu_pbds::priority_queue<pair<ll, int>> q;
    vector<decltype(q)::point_iterator> its(N);
    q.push({0, s});

    while (!q.empty()) {
      s = q.top().second;
      q.pop();
      seen[s] = 1;
      di = dist[s] + pi[s];
      for (edge& e : ed[s])
        if (!seen[e.to]) {
          ll val = di - pi[e.to] + e.cost;
          if (e.cap - e.flow > 0 && val < dist[e.to]) {
            dist[e.to] = val;
            par[e.to] = &e;
            if (its[e.to] == q.end())
              its[e.to] = q.push({-dist[e.to], e.to});
            else
              q.modify(its[e.to], {-dist[e.to], e.to});
          }
        }
    }
    rep(i, 0, N) pi[i] = min(pi[i] + dist[i], INF);
  }

  pair<ll, ll> maxflow(int s, int t) {
    ll totflow = 0, totcost = 0;
    while (path(s), seen[t]) {
      ll fl = INF;
```

```
      for (edge* x = par[t]; x; x = par[x->from])
        fl = min(fl, x->cap - x->flow);

      totflow += fl;
      for (edge* x = par[t]; x; x = par[x->from]) {
        x->flow += fl;
        ed[x->to][x->rev].flow -= fl;
      }
    }
    rep(i, 0, N) for (edge& e : ed[i]) totcost += e.cost * e.flow;
    return {totflow, totcost / 2};
  }

  // If some costs can be negative, call this before maxflow:
  void setpi(int s) {  // (otherwise, leave this out)
    fill(all(pi), INF);
    pi[s] = 0;
    int it = N, ch = 1;
    ll v;
    while (ch-- && it--)
      rep(i, 0, N) if (pi[i] !=
                       INF) for (edge& e :
                                 ed[i]) if (e.cap) if ((v = pi[i] + e.cost)
<
                                                       pi[e.to]) pi[e.to] =
v,
                                                                 ch = 1;
    assert(it >= 0);  // negative cost cycle
  }
};
```

6. Strings

6.1. Z

```
vi Z(const string& S) {                                                cpp
  vi z(sz(S));
  int l = -1, r = -1;
  rep(i, 1, sz(S)) {
    z[i] = i >= r ? 0 : min(r - i, z[i - l]);
    while (i + z[i] < sz(S) && S[i + z[i]] == S[z[i]]) z[i]++;
    if (i + z[i] > r) l = i, r = i + z[i];
  }
  return z;
}
```

6.2. MinRotation

```
int minRotation(string s) {                                            cpp
  int a = 0, N = sz(s);
  s += s;
  rep(b, 0, N) rep(k, 0, N) {
    if (a + k == b || s[a + k] < s[b + k]) {
      b += max(0, k - 1);
      break;
    }
    if (s[a + k] > s[b + k]) {
      a = b;
      break;
    }
  }
  return a;
}
```

6.3. SuffixArray

```
struct SuffixArray {                                                   cpp
  vi sa, lcp;
  SuffixArray(string& s, int lim = 256) {  // or basic_string<int>
    int n = sz(s) + 1, k = 0, a, b;
    vi x(all(s)), y(n), ws(max(n, lim));
    x.push_back(0), sa = lcp = y, iota(all(sa), 0);
    for (int j = 0, p = 0; p < n; j = max(1, j * 2), lim = p) {
      p = j, iota(all(y), n - j);
      rep(i, 0, n) if (sa[i] >= j) y[p++] = sa[i] - j;
      fill(all(ws), 0);
```

```
      rep(i, 0, n) ws[x[i]]++;
      rep(i, 1, lim) ws[i] += ws[i - 1];
      for (int i = n; i--;) sa[--ws[x[y[i]]]] = y[i];
      swap(x, y), p = 1, x[sa[0]] = 0;
      rep(i, 1, n) a = sa[i - 1], b = sa[i],
                   x[b] = (y[a] == y[b] && y[a + j] == y[b + j]) ? p - 1 :
p++;
    }
    for (int i = 0, j; i < n - 1; lcp[x[i++]] = k)
      for (k &&k--, j = sa[x[i] - 1]; s[i + k] == s[j + k]; k++);
  }
};
```

7. Misc

7.1. pbds

cpp

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

using namespace __gnu_pbds;

template <typename T>
using ordered_set = tree<T, null_type, less<T>, rb_tree_tag,
tree_order_statistics_node_update>;

const int RANDOM =
chrono::high_resolution_clock::now().time_since_epoch().count();
struct chash {
    int operator()(int x) const { return x ^ RANDOM; }
};
using fast_map = gp_hash_table<int, int, chash>;
```

7.2. LineContainer

cpp

```
#pragma once

struct Line {
  mutable ll k, m, p;
  bool operator<(const Line& o) const { return k < o.k; }
  bool operator<(ll x) const { return p < x; }
};

struct LineContainer : multiset<Line, less<>> {
  // (for doubles, use inf = 1/.0, div(a,b) = a/b)
  static const ll inf = LLONG_MAX;
  ll div(ll a, ll b) {  // floored division
    return a / b - ((a ^ b) < 0 && a % b);
  }
  bool isect(iterator x, iterator y) {
    if (y == end()) return x->p = inf, 0;
    if (x->k == y->k)
      x->p = x->m > y->m ? inf : -inf;
    else
      x->p = div(y->m - x->m, x->k - y->k);
    return x->p >= y->p;
  }
  void add(ll k, ll m) {
    auto z = insert({k, m, 0}), y = z++, x = y;
    while (isect(y, z)) z = erase(z);
    if (x != begin() && isect(--x, y)) isect(x, y = erase(y));
    while ((y = x) != begin() && (--x)->p >= y->p) isect(x, erase(y));
  }
  ll query(ll x) {
    assert(!empty());
    auto l = *lower_bound(x);
    return l.k * x + l.m;
  }
};
```