National Research University Higher School of Economics

Elderly Passion Fruit

Alexandr Nekrasov, Iurii Pustovalov, Igor Markelov

November 9, 2022

Contest (1)

template.cpp

30 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
}
```

Math (3)

GoncharFedor.cpp

Description: Calculating number of points s.t. $x, y \geq 0, Ax + By \leq C$

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

0ef10e, 10 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, 21 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, 22 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
```

Geometry (4)

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 (5)

PrefixZ.cpp

Description: Calculates Prefix,Z-functions

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

1c4e93, 25 lines

```
vector<int> pf(string s) {
    int k = 0;
    vector<int> p(s.size());
    for (int i = 1; i < s.size(); ++i) {
        while (k && s[i] != s[k])
            k = p[k - 1];
        k += (s[i] == s[k]);
        p[i] = k;
    }
    return p;
}
vector<int> zf(string s) {
    int n = s.size();
    vector<int> z(n, 0);
    for (int i = 1, l = 0, r = 0; i < n; ++i) {
        if (i <= r)
            z[i] = min(r - i + 1, z[i - 1]);
        while (i + z[i] < n && s[z[i]] == s[i + z[i]])
            ++z[i];
        if (i + z[i] - 1 > r)
            l = i, r = i + z[i] - 1;
    }
    z[0] = n;
    return z;
}
```

Eertree.cpp

Description: Creates Eertree of string str

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

7924c8, 39 lines

```
struct eertree {
    int len[MAXN], suffLink[MAXN];
    int to[MAXN][26];
    int numV, v;
    void addLetter(int n, string& str) {
        while (str[n - len[v] - 1] != str[n])
            v = suffLink[v];
        int u = suffLink[v];
        while (str[n - len[u] - 1] != str[n])
            u = suffLink[u];
        int u_ = to[u][str[n] - ’a’];
        int v_ = to[v][str[n] - ’a’];
        if (v_ == -1) {
            v_ = to[v][str[n] - ’a’] = numV;
            len[numV++] = len[v] + 2;
            suffLink[v_] = u_;
        }
        v = v_;
    }
    void init() {
        len[0] = -1;
        len[1] = 0;
        suffLink[1] = 0;
        suffLink[0] = 0;
        numV = 2;
        for (int i = 0; i < 26; ++i) {
            to[0][i] = numV++;
            suffLink[numV - 1] = 1;
            len[numV - 1] = 1;
        }
        v = 0;
    }
    void init(int sz) {
        for (int i = 0; i < sz; ++i) {
            len[i] = suffLink[i] = 0;
            for (int j = 0; j < 26; ++j) to[i][j] = -1;
        }
    }
};
```

MinShift.cpp

Description: Calculates min-cyclic-shift of s, Duval decomposition

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

3f0fb9, 18 lines

```
string minshift(string s) {
    int i = 0, ans = 0;
    s += s;
    int n = s.size();
    while (i < n / 2) {
        ans = i;
        int j = i + 1, k = i;
        while (j < n && s[k] <= s[j]) {
            if (s[k] < s[j]) k = i;
            else ++k;
            ++j;
        }
        while (i <= k) {
            i += j - k;
        }
    }
    return s.substr(ans, n / 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					