Содержание

1 algo/math/fft_recursive.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;
#define forn(i, n) for (int i = 0; i < (int)(n); ++i)
typedef long long i64;

typedef double ld;

struct base {
    ld re, im;
    base(){}
    base(ld re) : re(re), im(0) {}
    base(ld re, ld im) : re(re), im(im) {}

    base operator+(const base& o) const { return {re+o.re, im+o.im}; }
    base operator-(const base& o) const { return {re-o.re, im-o.im}; }
    base operator*(const base& o) const {
        return {
            re*o.re - im*o.im,
            re*o.im + im*o.re
        };
    }
};

const int sz = 1<<20;

int revb[sz];
vector<base> ang[21];

void init(int n) {
    int lg = 0;
    while ((1<<lg) != n) {
        ++lg;
    }
    forn(i, n) {
        revb[i] = (revb[i>>1]>>1)^((i&1)<<(lg-1));
    }

    ld e = M_PI * 2 / n;
    ang[lg].resize(n);
    forn(i, n) {
        ang[lg][i] = { cos(e * i), sin(e * i) };
    }

    for (int k = lg - 1; k >= 0; --k) {
        ang[k].resize(1 << k);
        forn(i, 1<<k) {
            ang[k][i] = ang[k+1][i*2];
        }
    }
}

void fft_rec(base *a, int lg, bool rev) {
    if (lg == 0) {
        return;
    }
```

```
    int len = 1 << (lg - 1);
    fft_rec(a, lg-1, rev);
    fft_rec(a+len, lg-1, rev);

    forn(i, len) {
        base w = ang[lg][i];
        if (rev) w.im *= -1;
        base u = a[i];
        base v = a[i+len] * w;
        a[i] = u + v;
        a[i+len] = u - v;
    }
}

void fft(base *a, int n, bool rev) {
    forn(i, n) {
        int j = revb[i];
        if (i < j) swap(a[i], a[j]);
    }
    int lg = 0;
    while ((1<<lg) != n) {
        ++lg;
    }
    fft_rec(a, lg, rev);
    if (rev) forn(i, n) {
        a[i] = a[i] * (1.0 / n);
    }
}

const int maxn = 1050000;

int n;
base a[maxn];
base b[maxn];

void test() {
    int n = 1<<19;
    mt19937 rr(55);
    forn(i, n) a[i] = rr() % 10000;
    forn(j, n) b[j] = rr() % 10000;

    int N = 1;
    while (N < 2*n) N *= 2;

    clock_t start = clock();
    init(N);
    cerr << "init time: " << (clock()-start) / 1000 << " ms" << endl;
    fft(a, N, 0);
    fft(b, N, 0);
    forn(i, N) a[i] = a[i] * b[i];
    fft(a, N, 1);
    clock_t end = clock();

    ld err = 0;
    forn(i, N) {
        err = max(err, (ld)fabsl(a[i].im));
        err = max(err, (ld)fabsl(a[i].re - (i64(a[i].re + 0.5))));
```

```
    }

    cerr << "Time: " << (end - start) / 1000 << " ms, err = " << err << endl;
}

int main() {
#ifdef LOCAL
    freopen("input.txt", "r", stdin);
#endif

    test();

#ifdef LOCAL
    cerr << "Time elapsed: " << clock() / 1000 << " ms" << endl;
#endif
    return 0;
}
```

2 algo/strings/automaton_is.cpp

```cpp
//real    4m27.689s
#include <bits/stdc++.h>
using namespace std;
#define forn(i, n) for (int i = 0; i < (int)(n); ++i)
#define fore(i, b, e) for (int i = (int)(b); i <= (int)(e); ++i)
#define ford(i, n) for (int i = (int)(n) - 1; i >= 0; --i)
#define mp make_pair
#define pb push_back
#define fi first
#define se second
#define all(x) (x).begin(), (x).end()
typedef vector<int> vi;
typedef pair<int, int> pii;
typedef long long i64;
typedef unsigned long long u64;
const int inf = 1e9+100500;

const int maxn = 100500;

int t[maxn][26], lnk[maxn], len[maxn];
int sz;
int last;

void init() {
    sz = 3;
    last = 1;
    forn(i, 26) t[2][i] = 1;
    len[2] = -1;
    lnk[1] = 2;
}

void addchar(int c) {
    int nlast = sz++;
    len[nlast] = len[last] + 1;
    int p = last;
    for (; !t[p][c]; p = lnk[p]) {
        t[p][c] = nlast;
    }
    int q = t[p][c];
    if (len[p] + 1 == len[q]) {
        lnk[nlast] = q;
    } else {
        int clone = sz++;
        len[clone] = len[p] + 1;
        lnk[clone] = lnk[q];
        lnk[q] = lnk[nlast] = clone;
        forn(i, 26) t[clone][i] = t[q][i];
        for (; t[p][c] == q; p = lnk[p]) {
            t[p][c] = clone;
        }
    }
    last = nlast;
}

bool check(const string& s) {
```

```
    int v = 1;
    for (int c: s) {
        c -= 'a';
        if (!t[v][c]) return false;
        v = t[v][c];
    }
    return true;
}

int main() {
#ifdef HOME
    freopen("input.txt", "r", stdin);
#endif

    string s;
    cin >> s;
    init();
    for (int i: s) {
        addchar(i-'a');
    }
    forn(i, s.length()) {
        assert(check(s.substr(i)));
    }
    cout << sz << endl;


#ifdef HOME
    cerr << "Time elapsed: " << clock() / 1000 << " ms" << endl;
#endif
    return 0;
}
```

3 algo/strings/suffix_array_is.cpp

```
#include <iostream>
#include <string>
#include <algorithm>
using namespace std;
#define forn(i, n) for (int i = 0; i < (int)(n); ++i)

const int maxn = 100500;

string s;
int n;
int sa[maxn], new_sa[maxn], cls[maxn], new_cls[maxn], cnt[maxn], lcp[maxn];
int n_cls;

void build() {
    n_cls = 256;
    forn(i, n) {
        sa[i] = i;
        cls[i] = s[i];
    }
    for (int d = 0; d < n; d = d ? d*2 : 1) {

        forn(i, n) new_sa[i] = (sa[i] - d + n) % n;
        forn(i, n_cls) cnt[i] = 0;
        forn(i, n) ++cnt[cls[i]];
        forn(i, n_cls) cnt[i+1] += cnt[i];
        for (int i = n-1; i >= 0; --i) sa[--cnt[cls[new_sa[i]]]] = new_sa[i];

        n_cls = 0;
        forn(i, n) {
            if (i && (cls[sa[i]] != cls[sa[i-1]] ||
                      cls[(sa[i] + d) % n] != cls[(sa[i-1] + d) % n])) {
                ++n_cls;
            }
            new_cls[sa[i]] = n_cls;
        }
        ++n_cls;
        forn(i, n) cls[i] = new_cls[i];
    }

    // cls is also a reverse permutation of sa if a string is not cyclic
    // (i.e. a position of i-th lexicographical suffix)
    int val = 0;
    forn(i, n) {
        if (val) --val;
        if (cls[i] == n-1) continue;
        int j = sa[cls[i] + 1];
        while (i + val != n && j + val != n && s[i+val] == s[j+val]) ++val;
        lcp[cls[i]] = val;
    }
}

int main() {
    freopen("input.txt", "r", stdin);
    cin >> s;
```

```
    s += '$';
    n = s.length();
    build();
    forn(i, n) {
        cout << s.substr(sa[i]) << endl;
        cout << lcp[i] << endl;
    }
}
```

4 algo/strings/ukkonen_ks.cpp

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

using namespace std;

#define sz(x) ((int) (x).size())
#define forn(i,n) for (int i = 0; i < int(n); ++i)
#define forab(i,a,b) for (int i = int(a); i < int(b); ++i)

typedef long long ll;
typedef long double ld;

const int INF = 1000001000;
const ll INFL = 2000000000000001000;

const int maxc = 26;
const int maxn = 2000000;

typedef int Link;

string s;

struct Node
{
    int l, r;
    Link to[maxc];
    Link par, link;

    Node(int l, int r): l(l), r(r), par(0), link(0)
    {
        memset(to, 0, sizeof(to));
    }

    Node(): Node(0, 0) {}
};

Node t[maxn];
int nodes = 3;
const Link root = 1;
const Link blank = 2;
Link cur = root;
int pos = 0;

Link newNode(int l, int r)
{
    t[nodes] = Node(l, r);
    return nodes++;
}

int charAt(int pos)
{
    return s[pos] - 'a';
}

void setParent(Link u, Link v)
{
```

```
    if (v)
        t[v].to[charAt(t[u].l)] = u;
    t[u].par = v;
}

void init()
{
    forn (c, maxc)
        t[blank].to[c] = root;
    t[root].par = t[root].link = t[blank].link = blank;
}

bool canGo(int c)
{
    if (pos == t[cur].r)
        return t[cur].to[c];
    return charAt(pos) == c;
}

void goDown(int l, int r)
{
    while (l < r)
    {
        int delta = min(r - l, t[cur].r - pos);
        pos += delta;
        l += delta;
        if (l >= r)
            break;
        assert(canGo(charAt(l)));
        cur = t[cur].to[charAt(l)];
        pos = min(t[cur].l + 1, t[cur].r);
        ++l;
    }
}

void goUp()
{
    if (pos == t[cur].r && t[cur].link)
    {
        cur = t[cur].link;
        pos = t[cur].r;
        return;
    }
    int l = t[cur].l;
    int r = pos;
    cur = t[t[cur].par].link;
    pos = t[cur].r;
    goDown(l, r);
}

void splitNode()
{
    assert(pos != t[cur].r);
    Link middle = newNode(t[cur].l, pos);
    t[cur].l = pos;
    setParent(middle, t[cur].par);
    setParent(cur, middle);
```

```
    cur = middle;
}

void addLeaf(int l)
{
    Link leaf = newNode(l, INF);
    setParent(leaf, cur);
}

void fixLink(Link bad)
{
    if (bad)
        t[bad].link = cur;
}

void addChar(int len)
{
    int c = charAt(len);
    Link badNode = 0;
    while (!canGo(c))
    {
        if (pos != t[cur].r)
        {
            splitNode();
            fixLink(badNode);
            badNode = cur;
        }
        else
        {
            fixLink(badNode);
            badNode = 0;
        }
        addLeaf(len);
        goUp();
    }
    fixLink(badNode);
    goDown(len, len + 1);
}

int main()
{
    init();
    s = "
        qweqhtkijaeidfhjasdfasdfasdflkjlkjaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaababababababcdcdcdcdcdabcaac
        ";
    forn (i, sz(s))
        addChar(i);
    return 0;
}
```

5 algo/structures/ordered_set.cpp

```cpp
#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
}
```

6 algo/structures/splay.cpp

```
#include <iostream>
#include <set>
#include <cstdio>
#include <algorithm>
#include <cassert>
#include <cstdlib>
using namespace std;
#define forn(i, n) for (int i = 0; i < (int)(n); ++i)
#define fore(i, b, e) for (int i = (int)(b); i <= (int)(e); ++i)
#define ford(i, n) for (int i = (int)(n) - 1; i >= 0; --i)
#define mp make_pair
#define pb push_back
#define fi first
#define se second
#define all(x) (x).begin(), (x).end()
typedef vector<int> vi;
typedef pair<int, int> pii;
typedef long long i64;
typedef unsigned long long u64;
const int inf = 1e9+100500;
const int maxn = 100500;

struct node;
void updson(node* p, node* v, node* was);

struct node {
    int val;
    node *l, *r, *p;
    node() {}
    node(int val) : val(val), l(r=p=NULL) {}

    bool isRoot() const { return !p; }
    bool isRight() const { return p && p->r == this; }
    bool isLeft() const { return p && p->l == this; }
    void setLeft(node* t) {
        if (t) t->p = this;
        l = t;
    }
    void setRight(node *t) {
        if (t) t->p = this;
        r = t;
    }
};

void updson(node *p, node *v, node *was) {
    if (p) {
        if (p->l == was) p->l = v;
        else p->r = v;
    }
    if (v) v->p = p;
}

void rightRotate(node *v) {
    assert(v && v->l);
    node *u = v->l;
```

```
    node *p = v->p;
    v->setLeft(u->r);
    u->setRight(v);
    updson(p, u, v);
}

void leftRotate(node *v) {
    assert(v && v->r);
    node *u = v->r;
    node *p = v->p;
    v->setRight(u->l);
    u->setLeft(v);
    updson(p, u, v);
}

void splay(node *v) {
    while (v->p) {
        if (!v->p->p) {
            if (v->isLeft()) rightRotate(v->p);
            else leftRotate(v->p);
        } else if (v->isLeft() && v->p->isLeft()) {
            rightRotate(v->p->p);
            rightRotate(v->p);
        } else if (v->isRight() && v->p->isRight()) {
            leftRotate(v->p->p);
            leftRotate(v->p);
        } else if (v->isLeft()) {
            rightRotate(v->p);
            leftRotate(v->p);
        } else {
            leftRotate(v->p);
            rightRotate(v->p);
        }
    }
    v->p = NULL;
}

node *insert(node *t, node *n) {
    if (!t) return n;
    int x = n->val;
    while (true) {
        if (x < t->val) {
            if (t->l) {
                t = t->l;
            } else {
                t->setLeft(n);
                t = t->l;
                break;
            }
        } else {
            if (t->r) {
                t = t->r;
            } else {
                t->setRight(n);
                t = t->r;
                break;
            }
```

```
        }
    }
    splay(t);
    return t;
}

void print(node* t) {
    if (t) {
        if (t->l) assert(t->l->p == t);
        if (t->r) assert(t->r->p == t);
        print(t->l);
        printf("%d ", t->val);
        print(t->r);
    }
    fflush(stdout);
}

void printCool(node* t, int d = 0) {
    if (t) {
        if (d == 0) assert(!t->p);
        if (t->l) assert(t->l->p == t);
        if (t->r) assert(t->r->p == t);
        printCool(t->r, d+2);
        cout << string(d, ' ') << t->val << endl;
        printCool(t->l, d+2);
    }
}

node *insert(node *t, int x) {
    return insert(t, new node(x));
}

int main() {
#ifdef HOME
//     freopen("input.txt", "r", stdin);
#endif

    node *t = NULL;
    forn(i, 1000000) {
        int x = rand();
        t = insert(t, x);
    }

#ifdef HOME
    cerr << "Time elapsed: " << clock() / 1000 << " ms" << endl;
#endif
    return 0;
}
```

7 algo/structures/treap_fast_insert.cpp

```cpp
#include <iostream>
#include <unordered_set>
#include <set>
#include <cstdio>
#include <algorithm>
#include <cassert>
#include <cstdlib>
using namespace std;
#define forn(i, n) for (int i = 0; i < (int)(n); ++i)
#define fore(i, b, e) for (int i = (int)(b); i <= (int)(e); ++i)
#define ford(i, n) for (int i = (int)(n) - 1; i >= 0; --i)
#define mp make_pair
#define pb push_back
#define fi first
#define se second
#define all(x) (x).begin(), (x).end()
typedef vector<int> vi;
typedef pair<int, int> pii;
typedef long long i64;
typedef unsigned long long u64;
const int inf = 1e9+100500;
const int maxn = 100500;

struct node {
    int x, y;
    node *l, *r;
    node(int x) : x(x), y(rand()), l(r=NULL) {}
};

void split(node *t, node *&l, node *&r, int x) {
    if (!t) return (void)(l=r=NULL);
    if (x <= t->x) {
        split(t->l, l, t->l, x), r = t;
    } else {
        split(t->r, t->r, r, x), l = t;
    }
}

node *merge(node *l, node *r) {
    if (!l) return r;
    if (!r) return l;
    if (l->y > r->y) {
        l->r = merge(l->r, r);
        return l;
    } else {
        r->l = merge(l, r->l);
        return r;
    }
}

node *insert(node *t, node *n) {
    node *l, *r;
    split(t, l, r, n->x);
    return merge(l, merge(n, r));
}
```

```
node *insert(node *t, int x) {
    return insert(t, new node(x));
}

node *fast_insert(node *t, node *n) {
    if (!t) return n;
    node *root = t;
    while (true) {
        if (n->x < t->x) {
            if (!t->l || t->l->y < n->y) {
                split(t->l, n->l, n->r, n->x), t->l = n;
                break;
            } else {
                t = t->l;
            }
        } else {
            if (!t->r || t->r->y < n->y) {
                split(t->r, n->l, n->r, n->x), t->r = n;
                break;
            } else {
                t = t->r;
            }
        }
    }
    return root;
}

node *fast_insert(node *t, int x) {
    return fast_insert(t, new node(x));
}

void print(node* t) {
    if (t) {
        print(t->l);
        printf("%d ", t->x);
        print(t->r);
    }
    fflush(stdout);
}

int main() {
    node *t = NULL;
    unordered_set<int> q;
    forn(i, 1000000) {
        int x = rand();
        q.insert(x);
//          t = insert(t, x);
//          t = fast_insert(t, x);
    }

#ifdef HOME
    cerr << "Time elapsed: " << clock() / 1000 << " ms" << endl;
#endif
}
```

8 bugs.cpp

```
//IOI 2015 scales
-    bool ok = false;
+    bool ok = true;
     forn (q, 3)
         if (!calc(b[sizes[q].second], bound - 1))
         {
             ok = false;
             break;
         }


//CF 315C
-    dfs1(i);
+    forn (i, N)
+        if (!used[i])
+            dfs1(i);


//CF 315C
-    forn (i, N)
+    for (int i: ord)
     {
         ++COL;
         if (!used[i])
         dfs2(i);
     }


//IOI 2015 horses
     forn (i, n)
         if (x[i] > 1)
-            goodx.insert(x[i]);
+            goodx.insert(i);


//Petrsu summer camp 2015 day1 A

         //Mergeable maps
-        if (sz(a[i]) > sz(a[j]))
-            swap(i, j);
+        bool swapped = false;
+        if (sz(a[i]) > sz(a[j]))
+            swap(i, j), swapped = true;

         for (auto p: a[i].pref)
         {
             int x = p.first + a[i].bal;
             ll cnt = p.second;
             res += cnt * a[j].getsuf(x);
         }
         for (auto p: a[i].suf)
         {
             int x = p.first - a[i].bal;
             ll cnt = p.second;
             res += cnt * a[j].getpref(x);
```

```
        }
-       if (s[u] == '(')
-           a[i].open();
-       else
-           a[i].close();
+       if (s[u] == '(')
+           a[swapped ? i : j].open();
+       else
+           a[swapped ? i : j].close();


//Petrsu summer camp 2015 day1 D
    //Segment tree
+   fill(upd, upd + base * 2, 0);


//Petrsu summer camp 2015 day1 J
    ld qwer = (x * cosl(phi) + y * sinl(phi)) / dist;
-   ld dphi = acosl(min(1.0L, max(-1.0L, qwer)));
+   ld dphi = acosl(qwer);


//Petrsu summer camp 2015 day3 G
    int getId(Node *t)
    {
-       int id = 0;
+       int id = cnt(t->l);
        while (t->p)
        {
            if (t->p->r == t)
                id += cnt(t->p->l) + 1;
            t = t->p;
        }
        return id;
    }


//Petrsu summer camp 2015 day4 C
    forn (i, m)
        {
            scanf("%d", c + i + n);
            s.emplace(c[i + n], i + n);
-           ans[i] = -1;
+           ans[i + n] = -1;
        }


//Petrsu summer camp 2015 day5 A
    forn (i, n - 1)
    {
        int p;
        cin >> p;
        g[p].push_back(i + 1);
-       ::p[i][0] = p;
+       ::p[i + 1][0] = p;
    }
```

```
//Petrsu summer camp 2015 day5 A
-       ans += min(R + 1, p.second) - max(L, p.first);
+       ans += max(0ll, min(R + 1, p.second) - max(L, p.first));


//Petrsu summer camp 2015 day2 J
    cnt[u] = 1;
    for (int v: g[u])
        if (v != prev)
        {
            dfs(v, u, ch + 1);
+           cnt[u] += cnt[v];
        }


//CF 327C
-   if (res == INF)
+   if (res >= INF / 2)
        res = -1;


//Opentrains 010026 F
    struct E {
        int v, c, w;
-   } es[maxn];
+   } es[2 * maxn];


//Opentrains 010026 H
        fill(mask, mask + n, 0);
-       fill(init, mask + n, ll(-1));
+       fill(init, init + n, ll(-1));


//Opentrains 010026 H
-       mask[v] |= (init[u] & mask[v]);
+       mask[v] |= (init[v] & mask[u]);


//Opentrains 001367 H
    forn (i, n)
-       forn (j, n)
+       forn (j, i)


//Opencup GP of Yekaterinburg
    edge[e].f++;
-   edge[e ^ 1].f++;
+   edge[e ^ 1].f--;


//Hackerrank testing
//Fenwick tree
-   return get(xr, yr) + get(xl, yl) - get(xr, yl) - get(yr, xl);
+   return get(xr, yr) + get(xl, yl) - get(xr, yl) - get(xl, yr);
```

```
//Opencup GP of Siberia 4
+    cout.precision(10);


//Opencup GP of Siberia 3
-    if (mask[u] | (ull(1) << (j - i)))
+    if (mask[u] & (ull(1) << (j - i)))


//Opencup GP of Siberia 6
     return pos.x * cos(ang) + pos.y * -sin(ang)
-           pos.x * sin(ang) + pos.x *  cos(ang);
+           pos.x * sin(ang) + pos.y *  cos(ang);


//MIPT Fall Day1 J
ld d[1 << maxn][maxn]; // some dynamic
...
int dist[maxn][maxn]; // distance matrix
...
     forn (k, n)
         forn (i, n)
             forn (j, n)
-                d[i][j] = min(d[i][j], d[i][k] + d[k][j]);
+                dist[i][j] = min(dist[i][j], dist[i][k] + dist[k][j]);


//Some task on PE
int add(int x, int y) {
  x += y;
  if (x >= mod) {
-   x -= y;
+   x -= mod;
  }
  return x;
}


//CF 335C
//Before building a convex hull
     sort(p, p + n, cmp);
+    n = unique(p, p + n) - p;


//CF WunderFund Round
     t[v * 2] *= upd[v];
     t[v * 2 + 1] *= upd[v];
+    upd[v * 2] *= upd[v];
+    upd[v * 2 + 1] *= upd[v];
     upd[v] = 1;


//CF 8VC Cup
-    int ans = max(a.back(), b.back());
+    int ans = 0;
+    if (!a.empty())
```

```
+        ans = max(ans, a.back());
+    if (!b.empty())
+        ans = max(ans, b.back());
```

9 gvimrc.vim

```
syntax on
set ai
set si
set nu
set et
set ts=4
set sts=4
set sw=4
set tm=300
set cin
set cinw+=forn,forab

imap jk <esc>:w<cr>
imap jj <esc>
map <c-j> 5j
map <c-k> 5k
map gc ^i//<esc>
map gu ^xx
map H ^
map L $
imap {<cr> {<cr>}<esc>O

map <f9> :w<cr>:!g++ -O2 -DLOCAL -Wno-unused-result -std=c++11 -Wall -Wextra -o
    %< %<cr>
map <f5> :w<cr>:!%<<cr>

set guifont=Consolas:cRUSSIAN
```

10 strategies.txt

- Проверить руками сэмплы
- Подумать как дебагать после написания
- Выписать сложные формулы и все +-1

- Проверить имена файлов
- Прогнать сэмплы
- Переполнения int, переполнения long long
- Выход за границу массива: _GLIBCXX_DEBUG
- Переполнения по модулю: в псевдоонлайнгенераторе--, в функцияхобертках-
- Проверить мультитест на разных тестах
- Прогнать минимальный по каждому параметру тест
- Прогнать псевдомаксимальный- тестнемного(чисел, но очень большие или очень маленькие)
- Представить что не зайдет и заранее написать assertы', прогнать слегка модифицированные тесты
- cout.precision: в том числе в интерактивных задачах
- Удалить debug-output, отсечения для тестов, вернуть оригинальный maxn, удалить _GLIBCXX_DEBUG

- Вердикт может врать
- Если много тестов(>3), дописать в конец каждого теста ответ, чтобы не забыть
- (WA) Потестить не только ответ, но и содержимое значимых массивов, переменных
- (WA) Изменить тест так, чтобы ответ не менялся: поменять координаты местами, сжатьрастянуть/ координаты, поменять ROOT дерева
- (WA) Подвигать размер блока в корневой или битсете
- (WA) Поставить assertы', возможно написать чекер с assertом'
- (WA) Проверить, что программа не печатает чтолибо- неожиданное, что должно попадать под PE: inf - 2, не лекс. мин. решение, одинаковые числа вместо разных, неправильное количество чисел, пустой ответ, перечитать output format
- (TL) cin -> scanf -> getchar
- (TL) Упихать в кэш большие массивы, поменять местами forы' или измерения массива
- (RE) Проверить формулы на деление на 0, выход за область определения(sqrt(-eps), acos(1 + eps))

11 template.cpp

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

#define forn(i, n) for (int i = 0; i < int(n); ++i)
#define forab(i, a, b) for (int i = int(a); i < int(b); ++i)
#define sz(x) ((int) (x).size())

typedef long long ll;
typedef long long i64;
typedef long double ld;
typedef pair<int, int> pii;

const int inf = int(1e9) + 100;
const ll infl = ll(2e18) + 100;

#define pb push_back
#define eb emplace_back
#define mp make_pair

int main() {
#ifdef LOCAL
    assert(freopen("", "r", stdin));
#endif

}
```

12 vimrc.vim

```
syntax on
set ai
set si
set nu
set et
set ts=4
set sts=4
set sw=4
set tm=300
set cin
set cinw+=forn,forab

imap jk <esc>:w<cr>
imap jj <esc>
map <c-j> 5j
map <c-k> 5k
map gc ^i//<esc>
map gu ^xx
map H ^
map L $
imap {<cr> {<cr>}<esc>O

let $CPPFLAGS='-O2 -DLOCAL -Wall -Wextra -Wno-unused-result -std=c++11'
map <f9> :w<cr>:make %:r<cr>
map <f5> :w<cr>:!./%<<cr>
```