Contents

Compilation date: 15.12.2025

FFT

```
const int MOD = 998244353;
const int LOG_SZ = 18;
const int SZ = (1 << LOG_SZ);

void init_fft(int log_sz) {
    int root = deg(3, (MOD - 1) >> log_sz);
    assert(deg(root, 1 << (log_sz)) == 1 && deg(root, 1 << (log_sz - 1)) !=
1);

    int sz = (1 << LOG_SZ);
    for (int i = 0, r = 1; i < sz; i++, r = mul(r, root))
        roots[i] = r;
    for (int i = 1; i < sz; i++)
        perm[i] = (perm[i >> 1] >> 1) | ((i & 1) << (log_sz - 1));
}

void fft(int arr[], int sz) {
    for (int i = 0; i < sz; i++) {
        if (i < perm[i]) swap(arr[i], arr[perm[i]]);
    }
    for (int i = 1, shift = sz / 2; i < sz; i *= 2, shift /= 2) {
        for (int j = 0; j < sz; j += i * 2) {
            for (int k = 0, pos = 0; k < i; k++, pos += shift) {
                int val = mul(arr[i + j + k], roots[pos]);
                arr[i + j + k] = dec(arr[j + k], val);
                arr[j + k] = add(arr[j + k], val);
            }
        }
    }
}

int fftRes[SZ];

void mult(int a[], int b[], int sz) {

    fill(arra, arra + sz, 0);
    for (int i = 0; i < sz; i++) arra[i] = a[i];
    fft(arra, sz);

    fill(arrb, arrb + sz, 0);
    for (int i = 0; i < sz; i++) arrb[i] = b[i];
    fft(arrb, sz);

    for (int i = 0; i < sz; i++) arra[i] = mul(arra[i], arrb[i]);
    fft(arra, sz);

    reverse(arra + 1, arra + sz);
    vector<int> ans;
    int revsz = deg(sz, MOD - 2);
    for (int i = 0; i < sz; i++) fftRes[i] = mul(arra[i], revsz);
}
```

Потоки. Диниц

```
const int MAXV = 1e5, MAXE = 1e6, INF = 1e9;

struct Edge{
    int from, to, cap, flow;

    Edge(int from = 0, int to = 0, int cap = 0, int flow = 0):
        from(from), to(to), cap(cap), flow(flow) {}
}edge[MAXE];

int dst[MAXV];

vector<int> g[MAXV];

bool bfs(int start, int finish) {
    fill(dst, dst + MAXV, INF);
    dst[start] = 0;
    queue<int> q;
    q.push(start);
    while (!q.empty()) {
        int v = q.front();
        q.pop();
        for (int i : g[v]) {
            if (dst[edge[i].to] == INF && edge[i].flow < edge[i].cap) {
                dst[edge[i].to] = dst[v] + 1;
                q.push(edge[i].to);
            }
        }
    }
    return dst[finish] != INF;
}

int nxt[MAXV];

int dfs(int v, int flow, int finish) {
    if (v == finish) return flow;
    for (int j = nxt[v]; j < sz(g[v]); j++) {
        int i = g[v][j];
        if (edge[i].flow < edge[i].cap && dst[edge[i].to] == dst[v] + 1) {
            int delta = dfs(edge[i].to, min(flow, edge[i].cap - edge[i].flow),
finish);
            if (delta > 0) {
                edge[i].flow += delta;
                edge[i ^ 1].flow -= delta;
                return delta;
            }
        }
    }
    return 0;
}
```

```
int dinic(int cntv) {
    int ans = 0;
    while (bfs(start, finish)) {
        fill(nxt, nxt + cntv, 0);
        int flow = dfs(start, INF, finish);
        while (flow) {
            ans += flow;
            flow = dfs(start, INF, finish);
        }
    }
    return ans;
}
```

Потоки. Min cost max flow

```
template<typename T>
using normal_queue = priority_queue<T, vector<T>, greater<T> >;

const int MAXV = 1e6, MAXE = 1e6, INF = 1e9;

struct Edge{
    int from, to, cap, flow, w;

    Edge(int from = 0, int to = 0, int cap = 0, int w = 0):
        from(from), to(to), cap(cap), flow(0), w(w) {}
}edge[MAXE];

int cntv, cnte;

int dst[MAXV], pot[MAXV], par[MAXV];

vector<int> g[MAXV];

bool bfs(int start, int finish) {
    fill(dst, dst + cntv, INF);
    fill(par, par + cntv, -1);
    dst[start] = 0;
    normal_queue<pii> q;
    q.push({0, start});
    while (!q.empty()) {
        auto [d, v] = q.top();
        q.pop();
        if (dst[v] < d) continue;
        for (int i : g[v]) {
            int to = edge[i].to, w = edge[i].w + pot[v] - pot[to];
            if (edge[i].flow < edge[i].cap && dst[to] > dst[v] + w) {
                dst[to] = dst[v] + w;
                par[to] = i;
                q.push({dst[to], to});
            }
        }
    }
    return dst[finish] != INF;
}

void ford(int start) {
    fill(pot, pot + cntv, INF);
    pot[start] = 0;
    for (int iter = 0; iter < cntv - 1; iter++) {
        for (int i = 0; i < cnte; i++) {
            int from = edge[i].from, to = edge[i].to, w = edge[i].w;
            if (pot[to] > pot[from] + w) {
                pot[to] = pot[from] + w;
            }
        }
    }
}

ll getFlow() {
    ll ans = 0;
    ford(start);
    while (bfs(start, finish)) {
        for (int i = 0; i < cntv; i++) pot[i] += dst[i];
        ll delta = INF;
        for (v = n - 1; v != 0; v = edge[par[v]].from) {
            int i = par[v];
            delta = min(delta, edge[i].cap - edge[i].flow);
        }
        for (v = n - 1; v != 0; v = edge[par[v]].from) {
            int i = par[v];
            edge[i].flow += delta;
            edge[i ^ 1].flow -= delta;
            ans += delta * edge[i].w;
        }
    }
}
```

Строки. Манакер

```
vector<int> man_odd(string &t) {
    int m = sz(t);
    vector<int> d(m, 1);
    int l = 0, r = 0;
    for (int i = 1; i < m; i++) {
        if (i < r)
            d[i] = min(r - i + 1, d[l + r - i]);
        while (i - d[i] >= 0 && i + d[i] < m && t[i - d[i]] == t[i + d[i]])
            d[i]++;
        if (i + d[i] - 1 > r)
            l = i - d[i] + 1, r = i + d[i] - 1;
    }
    return d;
}

vector<int> man_even(string &t) {
    int m = sz(t);
    vector<int> d(m, 0);
    int l = -1, r = -1;
    for (int i = 0; i < m - 1; i++) {
        if (i < r)
            d[i] = min(r - i, d[l + r - i - 1]);
        while (i - d[i] >= 0 && i + d[i] + 1 < m && t[i - d[i]] == t[i + d[i]
+ 1])
            d[i]++;
        if (i + d[i] > r)
            l = i - d[i] + 1, r = i + d[i];
    }
    return d;
}
```

Строки. Суфмас + LCP

```
vector<int> suffix_array(string t) {
    t += '$';
    int m = sz(t);
    vector<int> suff(m);
    iota(all(suff), 0);
    sort(all(suff), [&](int i1, int i2) {
        return t[i1] < t[i2];
    });
    vector<int> color(m);
    for (int i = 1; i < m; ++i) {
        color[suff[i]] = color[suff[i - 1]] + (t[suff[i]] != t[suff[i - 1]]);
    }
    auto add = [&](int x, int y) {
        int ans = x + y;
        while (ans >= m) ans -= m;
        return ans;
    };
    auto sub = [&](int x, int y) {
        int ans = x - y;
        while (ans < 0) ans += m;
        return ans;
    };
    vector<int> cnt(m), head(m), new_suff(m), new_color(m);
    for (int len = 1; len < m; len *= 2) {
        fill(all(cnt), 0);
        for (int i = 0; i < m; ++i) cnt[color[i]]++;
        fill(all(head), 0);
        for (int i = 1; i < m; ++i) head[i] = head[i - 1] + cnt[i - 1];
        fill(all(new_suff), 0);
        for (int i = 0; i < m; ++i) {
            new_suff[head[color[sub(suff[i], len)]]++] =
                    sub(suff[i], len);
        }
        suff.swap(new_suff);
        fill(all(new_color), 0);
        for (int i = 1; i < m; ++i) {
            new_color[suff[i]] = new_color[suff[i - 1]];
            if (color[suff[i]] != color[suff[i - 1]] ||
                color[add(suff[i], len)] != color[add(suff[i - 1], len)]) {
                new_color[suff[i]]++;
            }
        }
        color.swap(new_color);
    }
    return vector(suff.begin() + 1, suff.end());
}

const int inf = 1e9;

vector<int> build_lcp(string &t, vector<int> &suff) {
    int m = sz(t);
    vector<int> pos(m);
    for (int i = 0; i < m; ++i) {
        pos[suff[i]] = i;
    }
    vector<int> lcp(m);
    int curr = 0;
    for (int i = 0; i < m; ++i) {
        if (curr > 0) --curr;
        if (pos[i] == m - 1) {
            lcp[pos[i]] = curr;
            continue;
        }
        int j = suff[pos[i] + 1];
        while (i + curr < m &&
               j + curr < m &&
               t[i + curr] == t[j + curr]) {
            ++curr;
        }
        lcp[pos[i]] = curr;
    }
    return lcp;
}
```

Строки. Суфдерево

```
const int MAX_N = 1e5 + 5;
const int MAX_T = 2 * MAX_N;
const int MAX_A = 27;

struct Node {
    int next[MAX_A]{};
    int l{}, r{};
    int par{}, suf{};

    Node() = default;
    Node(int l, int r, int par): l(l), r(r), par(par) {
      fill(next, next + MAX_A, -1);
      suf = -1;
    }
} t[MAX_T];

int ptr_node = 0;

int new_node(int l, int r, int par = -1) {
  int v = ptr_node++;
  t[v] = Node(l, r, par);
  return v;
}

int root;

int n;
string s;

int get(char x) {
  if (x == '$') return 0;
  return x - 'a' + 1;
}

void build() {
  // s should ends with $!
  root = new_node(0, 0);

  int v = root, m = 0;
  for (int i = 0; i < n; i++) {
    int p = -1;
    while (true) {
      if (m < t[v].r) {
        if (s[i] == s[m]) {
          m += 1;
          break;
        } else {
          int w = new_node(t[v].l, m, t[v].par);

          t[t[v].par].next[get(s[t[v].l])] = w;
          t[w].next[get(s[m])] = v;

          t[v].l = m;
          t[v].par = w;

          v = w;
        }
      }

      if (m == t[v].r) {
        if (p != -1)
          t[p].suf = v;
        p = v;

        if (t[v].next[get(s[i])] != -1) {
          v = t[v].next[get(s[i])];
          m = t[v].l + 1;
          break;
        } else {
          int u = new_node(i, n, v);
          t[v].next[get(s[i])] = u;

          if (v == root)
            break;

          int len = m - t[v].l;
          v = t[v].par;

          if (v == root)
            len -= 1;
          else
            v = t[v].suf;

          m = t[v].l;
          while (len > 0) {
            v = t[v].next[get(s[i - len])];
            m = t[v].l;

            int sub = min(len, t[v].r - t[v].l);
            len -= sub;
            m += sub;
          }
        }
      }
    }
  }
}

int ptr = 0;

void dfs(int v, int p = -1) {
  int id = ptr++;
  if (p != -1)
    cout << p << " " << t[v].l << " " << t[v].r << "\n";
  for (int i = 0; i < MAX_A; i++) {
    int u = t[v].next[i];
    if (u != -1)
      dfs(u, id);
  }
}
```

Строки. Карась

```
const int T = 1e5 + 5;

struct Node {
    int next[26]{};
    int par = -1;
    int suf = -1;
} trie[T];

int ptr_node = 0, root;

int new_node(int par = -1) {
    int v = ptr_node++;
    fill(trie[v].next, trie[v].next + 26, -1);
    trie[v].par = par;
    return v;
}

void add(const string &t) {
    int v = root;
    for (char x : t) {
        int i = x - 'a';
        if (trie[v].next[i] == -1)
            trie[v].next[i] = new_node(v);
        v = trie[v].next[i];
    }
}

void build() {
    queue<int> q;

    trie[root].suf = root;
    q.push(root);

    while (!q.empty()) {
        int v = q.front();
        q.pop();

        for (int i = 0; i < 26; i++) {
            int &u = trie[v].next[i];
            if (u == -1) {
                u = v == root ? root : trie[trie[v].suf].next[i];
            } else {
                trie[u].suf = v == root ? root : trie[trie[v].suf].next[i];
                q.push(u);
            }
        }
    }
}

void solve() {
    root = new_node();
    // ... (add strings)
    build();
}
```

Огрызок шаблона

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>

//#pragma GCC optimize ("O3")
//#pragma GCC optimize ("Ofast")
//#pragma GCC target ("sse,sse2,sse3,sse4,avx,avx2")
//#pragma comment(linker, "/STACK:160777216")

//before loops:
//#pragma unroll(3)

using namespace std;

//using namespace __gnu_pbds;
//template<typename T>
//using orderd_set = tree<T, null_type, less<T>, rb_tree_tag,
tree_order_statistics_node_update>;

template<typename T1, typename T2>
__attribute__((always_inline)) common_type_t<T1, T2> min(const T1 &a, const T2
&b) {
    return a < b ? a : b;
}

template<typename T3, typename T4>
__attribute__((always_inline)) common_type_t<T3, T4> max(const T3 &a, const T4
&b) {
    return a > b ? a : b;
}
```

Xor-базис

```
int basis[LOG];

void build_basis(vector<int> a) {
    int n = sz(s);
    sort(all(a));

    for (int i = n - 1; i >= 0; i--) {
        int x = a[i];
        for (int j = LOG - 1; j >= 0; j--) {
            if (((x >> j) & 1) == 0) {
                continue;
            }
            if (basis[j] != 0) {
                x ^= basis[j];
            } else {
                basis[j] = x;
                break;
            }
        }
    }

    for (int i = 0; i < LOG; i++) {
        if (!basis[i]) {
            continue;
        }
        for (int j = i + 1; j < LOG; j++) {
            if (!basis[j]) {
                continue;
            }
            if ((basis[j] >> i) & 1) {
                basis[j] ^= basis[i];
            }
        }
    }
}
```

Li-chao

```
struct line {
  int k = 0, m = 0;

  line() {}
  line(int k, int m): k(k), m(m) {}

  int get(int x) {
    return k * x + m;
  }
};

line t[4 * MAXN];

void upd(int v, int tl, int tr, line L) {
  if (tl > tr) {
    return;
  }

  int tm = (tl + tr) / 2;
  bool l = L.get(tl) > t[v].get(tl);
  bool mid = L.get(tm) > t[v].get(tm);

  if (mid) {
    swap(L, t[v]);
  }

  if (l != mid) {
    upd(2 * v, tl, tm - 1, L);
  }

  else {
    upd(2 * v + 1, tm + 1, tr, L);
  }
}

int get(int v, int tl, int tr, int x) {
  if (tl > tr) {
    return 0;
  }

  int tm = (tl + tr) / 2;
    if (x == tm) {
        return t[v].get(x);
    }

    if (x < tm) {
    return max(t[v].get(x), get(2 * v, tl, tm - 1, x));
  }

  else {
    return max(t[v].get(x), get(2 * v + 1, tm + 1, tr, x));
  }
}
```

Простые числа

TODO

Геометрия. Целочисленное

```
struct Point {
    int x, y;
    Point(int x1 = 0, int y1 = 0) {
        x = x1;
        y = y1;
    }

    bool friend operator== (const Point& a, const Point& b) {
        return a.x == b.x && a.y == b.y;
    }

    bool friend operator!= (const Point& a, const Point& b) {
        return !(a == b);
    }
};

ll dst2(const Point& a, const Point& b) {
    ll dx = a.x - b.x;
    ll dy = b.y - b.y;
    return dx * dx + dy * dy;
}

ld dst(const Point& a, const Point& b) {
    return sqrt(dst2(a, b));
}

struct Vec {
    int x, y;

    Vec(int x = 0, int y = 0): x(x), y(y) {}

    Vec(const Point& a, const Point& b): x(b.x - a.x), y(b.y - a.y) {}

    ll len2() const {
        return 1ll * x * x + 1ll * y * y;
    }

    ld len() const {
        return sqrt(len2());
    }

    ll cross(Vec v) const {
        return 1ll * x * v.y - 1ll * y * v.x;
    }

    ll dot(Vec v) const {
        return 1ll * x * v.x + 1ll * y * v.y;
    }
};
```

```
struct Line {
    int a, b, c;

    Line(int a = 0, int b = 0, int c = 0): a(a), b(b), c(c) {}

    Line(const Point& f, const Point& s) {
        a = f.y - s.y;
        b = s.x - f.x;
        c = -a * f.x - b * f.y;
    }

    Vec get_direct_vector() const {
        return Vec(-b, a);
    }
};

bool comp(const Vec& v1, const Vec& v2) {
    bool f1 = v1.y > 0 || (v1.y == 0 && v1.x > 0);
    bool f2 = v2.y > 0 || (v2.y == 0 && v2.x > 0);
    if (f1 != f2) {
        return f1;
    }
    return v1.cross(v2) >= 0;
}

struct Polygon {
    vector<Point> points;
    vector<Vec> vectors;
    int n;

    Polygon(const vector<Point>& a) {
        n = a.size();
        int ind_d = 0;
        for (int i = 1; i < n; i++) {
            if(a[i].y < a[ind_d].y || (a[i].y == a[ind_d].y && a[i].x <
a[ind_d].x)) {
                ind_d = i;
            }
        }
        for(int i = ind_d; i < n; i++) {
            points.push_back(a[i]);
        }
        for(int i = 0; i < ind_d; i++) {
            points.push_back(a[i]);
        }
        vectors.resize(n);
        for(int i = 1; i < n + 1; i++) {
            vectors[i - 1] = Vec(points[i - 1], points[i % n]);
        }
    }
    Line get_tangent(Vec v) {
        if(vectors[0].cross(v) <= 0 && vectors.back().cross(v) >= 0) {
            Line ans;
            ans.a = -v.y;
            ans.b = v.x;
            ans.c = -ans.a * points[0].x - ans.b * points[0].y;
            return ans;
        }
        int i = lower_bound(vectors.begin(), vectors.end(), v, comp) -
vectors.begin();
        Line ans;
        ans.a = -v.y;
        ans.b = v.x;
        ans.c = -ans.a * points[i].x - ans.b * points[i].y;
        return ans;
    }
};

vector<Point> get_hull(vector<Point> a) {
    Point down = a[0];
    for (auto x : a) {
        if (x.x < down.x || (x.x == down.x && x.y < down.y)) {
            down = x;
        }
    }
    auto cmp = [&](Point a, Point b) {
        if (a == down) {
            return false;
        }
        if (b == down) {
            return true;
        }
        if (Vec(down, a).cross(Vec(down, b)) == 0) {
            return Vec(down, a).len() < Vec(down, b).len();
        }
        return Vec(down, a).cross(Vec(down, b)) < 0;
    };

    sort(a.begin(), a.end(), cmp);
    vector<Point> st;
    st.push_back(down);
    for (auto x : a) {
        while (st.size() > 1 && st[st.size() - 2] != x && Vec(st[st.size() -
2], st.back()).cross(Vec(st[st.size() - 2], x)) >= 0) {
            st.pop_back();
        }
        st.push_back(x);
    }
    st.pop_back();
    reverse(st.begin(), st.end());
    return st;
}
```

Compilation date: 15.12.2025