

## Содержание

<b>1 Настройка CLion</b>	<b>1</b>
<b>2 Теория чисел</b>	<b>1</b>
2.1 КТО	1
2.2 Алгоритм Миллера — Рабина	1
2.3 Алгоритм Берлекэмпа — Мессис	2
<b>3 Графы</b>	<b>2</b>
3.1 SCC и 2-SAT	2
3.2 Эйлеров цикл	2
3.3 Компоненты рёберной двусвязности	2
3.4 DCP offline	3
3.5 Взвешенное паросочетание	3
<b>4 Свёртки</b>	<b>4</b>
4.1 AND, OR, XOR свёртки	4
4.2 FFT & co	5
4.3 Быстрое FFT	6
4.4 FFT в double'ax	7
<b>5 Структуры данных</b>	<b>8</b>
5.1 Дерево Фенвика	8
5.2 Дерево отрезков в точке	8
5.3 Массовое дерево отрезков	9
5.4 Битовый бор	10
5.5 Ordered set	10
5.6 Convex hull trick	11
5.7 Центроиды	11
5.8 Дерево Ли Чао	11
5.9 Min-Kinetic Segment Tree	11
<b>6 Строковые алгоритмы</b>	<b>12</b>
6.1 Префикс-функция	12
6.2 Z-функция	12
6.3 Алгоритм Манакера	13
6.4 Суфмассив	13
6.5 Алгоритм Ахо — Корасик	14
6.6 Алгоритм Ахо Корасик	14
6.7 Дерево палиндромов	14
6.8 Дерево палиндромов	14
<b>7 Потoki</b>	<b>14</b>
7.1 Алгоритм Диница	14
7.2 Mincost k-flow	15
<b>8 Гамильтоновы путь и цикл</b>	<b>16</b>
8.1 Link-cut tree	16
8.2 Undirected case	16
8.3 Directed case	17
<b>9 Геометрия</b>	<b>18</b>
9.1 Примитивы	18
9.2 Выпуклая оболочка	18
9.3 Точка внутри многоугольника	18
9.4 Касательные	18
<b>10 Цепные дроби</b>	<b>19</b>
10.1 Поиск нижней огибающей, сумма и минимум по модулю	19
10.2 Простая рекурсия	19
<b>11 Разное</b>	<b>20</b>
11.1 Компараторы	20
11.2 Трюки от Сергея Копелиовича	20
11.2.1 Быстрый ввод	20
11.2.2 Быстрый аллокатор	20
11.3 Флаги компиляции	20
11.3.1 Сетка в vim	20
11.4 Что сделать на пробном туре	20

## 1 Настройка CLion

- В файле CMakeLists.txt дописать строку `add_compile_definitions(LOCAL)`. Нажать появившуюся опцию в правом верхнем углу `enable auto-reload`.

- Вбить шаблон в main.cpp:

```
#ifndef LOCAL
#define _GLIBCXX_DEBUG
#endif
#include<bits/stdc++.h>

using namespace std;

#define int long long
#define app push_back
#define all(x) x.begin(), x.end()
#ifdef LOCAL
#define debug(...) [](auto...a){ ((cout << a << ' '
), ...) << endl; }(#__VA_ARGS__, ": ",
__VA_ARGS__)
#define debugv(v) do { cout << #v << ": "; for (
auto x : v) cout << x << ' '; cout << endl; }
while(0)
#else
#define debug(...)
#define debugv(v)
#endif

int32_t main() {
    cin.tie(0); ios_base::sync_with_stdio(0);
    int n = 2; vector<int> a(n, n);
    debug(n); debugv(a);
}
```

Скомпилировать, чтобы проверить отсутствие опечаток.

- Запустить терминал (`ctrl + alt + T`)

```
$ cd workspace/CLionProjects
$ for c in {A..Z}; do cp main.cpp $c.cpp && echo "
add_executable($c $c.cpp)" >> CMakeLists.txt;
done
```

Далее отключаем подсветку и форматирование в настройках (`ctrl+alt+S`)

- Editor → Code Style → Formatter → Do not format прописать \*
- Editor → Inspections → C/C++ → static analysis tools → CLang-Tidy отключить
- Editor → Inlay Hints → отключаем всё (достаточно первых трёх — code vision, parameter names, types).

Тёмная тема отключается в Appearance & Behavior → Appearance.

Чтобы добавить санитайзеры, надо дописать в CMakeLists.txt `set(CMAKE_CXX_FLAGS "-fsanitize=address -fsanitize=undefined")`

## 2 Теория чисел

### 2.1 КТО

```
int gcd(int a, int b, int &x, int &y) {
    if (b==0) { x = 1; y = 0; return a; }
    int d = gcd(b, a%b, y, x);
    y-=a/b*x;
    return d;
}

int inv(int r, int m) {
    int x, y;
    gcd(r, m, x, y);
    return (x+m)%m;
}

int crt(int r, int n, int c, int m) { return r + ((c -
r) % m + m) * inv(n, m) % m * n; }
```

### 2.2 Алгоритм Миллера — Рабина

```

__int128 one=1;
int po(int a,int b,int p)
{
    int res=1;
    while(b & 1) {res=(res*one*a)%p;--b;} else {a
        =(a*one*a)%p;b>=>1;} return res;
}
bool chprime(int n) ///miller-rabin
{
    if(n==2) return true;
    if(n<=1 || n%2==0) return false;
    int h=n-1;int d=0;while(h%2==0) {h/=2;++d;}
    for(int a:{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31,
        37})
    {
        if(a==n) return true;
        int u=po(a,h,n);bool ok=0;
        if(u%n==1) continue;
        for(int c=0;c<d;++c)
        {
            if((u+1)%n==0) {ok=1;break;}
            u=(u*one*u)%n;
        }
        if(!ok) return false;
    }
    return true;
}

```

### 2.3 Алгоритм Берлекэмп — Мессе

<https://mzhang2021.github.io/cp-blog/berlekamp-massey/>

```

template<typename T>
vector<T> berlekampMassey(const vector<T> &s) {
    int n = s.size(), l = 0, m = 1;
    vector<T> b(n), c(n);
    T ld = b[0] = c[0] = 1;
    for (int i=0; i<n; i++, m++) {
        T d = s[i];
        for (int j=1; j<=l; j++)
            d += c[j] * s[i-j];
        if (d == 0) continue;
        vector<T> temp = c;
        T coef = d / ld;
        for (int j=m; j<n; j++) c[j] -= coef * b[j-m];
        if (2 * l <= i) {
            l = i + 1 - l;
            b = temp;
            ld = d;
            m = 0;
        }
    }
    c.resize(l + 1);
    c.erase(c.begin());
    for (T &x : c)
        x = -x;
    return c;
}

```

## 3 Графы

### 3.1 SCC и 2-SAT

Алгоритм ищет сильносвязные компоненты в графе  $g$ , если есть путь  $i \rightarrow j$ , то  $scc[i] \leq scc[j]$

```

vector<int> find_scc(vector<vector<int>> g) {
    vector<vector<int>> r(g.size());
    for (int i = 0; i < g.size(); ++i) {
        for (int j : g[i]) r[j].push_back(i);
    }
    vector<int> used(g.size()), tout(g.size());
    int time = 0;
    auto dfs = [&](auto dfs, int cur) -> void {
        if (used[cur]) return;
        used[cur] = 1;
        for (int nxt : g[cur]) {
            dfs(dfs, nxt);
        }
        tout[cur] = time++;
    };
}

```

```

for (int i = 0; i < g.size(); ++i) if (!used[i]) dfs
    (dfs, i);
vector<int> ind(g.size());
iota(ind.begin(), ind.end(), 0);
sort(all(ind), [&](int i, int j){return tout[i] >
    tout[j];});
vector<int> scc(g.size(), -1);
auto go = [&](auto go, int cur, int color) -> void {
    if (scc[cur] != -1) return;
    scc[cur] = color;
    for (int nxt : r[cur]) {
        go(go, nxt, color);
    }
};
int color = 0;
for (int i : ind) {
    if (scc[i] == -1) go(go, i, color++);
}
return scc;
}

```

Чтобы решать 2-SAT, надо создать граф на  $2n$  вершинах, рёбра  $i \Rightarrow j$  и  $(j \oplus 1) \Rightarrow (i \oplus 1)$  должны быть добавлены одновременно. После этого если  $scc[2 * i] = scc[2 * i + 1]$ , то решения нет; если  $scc[2 * i + 0] < scc[2 * i + 1]$ , то присутствует импликация  $\neg i \Rightarrow i$ , надо назначить  $i = \text{true}$ .

### 3.2 Эйлеров цикл

```

vector<int> euler(vector<vector<pair<int, int>>> g,
    int m, int src) { // g[cur][i] = pair{nxt, idx}
    int n = g.size();
    vector<int> used(m), it(n), cycle;
    auto dfs = [&](auto dfs, int cur) -> void {
        while (true) {
            while (it[cur] < g[cur].size() && used[g[cur][it
                [cur]].second]) it[cur]++;
            if (it[cur] == g[cur].size()) return;
            auto [nxt, idx] = g[cur][it[cur]];
            used[idx] = true;
            dfs(dfs, nxt);
            cycle.push_back(idx); // or {cur, nxt}
        }
    };
    dfs(dfs, src);
    reverse(cycle.begin(), cycle.end());
    if (cycle.size() != m) return {}; // check that all
        edges are present in the cycle, fail otherwise
    return cycle;
}

```

### 3.3 Компоненты рёберной двусвязности

```

int n, m;
cin >> n >> m;
vector <vector <int> > g(n + 1);
map <pair <int, int>, int> comp, col;
for (int i = 0; i < m; ++i) {
    int u, v, c; cin >> u >> v >> c; c--;
    col[{u,v}]=col[{v,u}]=c;
    g[u].push_back(v);
    g[v].push_back(u);
}
vector <int> used(n + 1);
vector <int> newCompWithoutParent(n + 1), h(n + 1), up
    (n + 1);
auto findCutPoints = [&](auto self, int u, int p) ->
    void {
    used[u] = 1;
    up[u] = h[u];
    for (int v : g[u]) {
        if (!used[v]) {
            h[v] = h[u] + 1;
            self(self, v, u);
            up[u] = min(up[u], up[v]);
            if (up[v] >= h[u]) {
                newCompWithoutParent[v] = 1;
            }
        }
    }
    else {

```

```

        up[u] = min(up[u], h[v]);
    }
}
};
for (int u = 1; u <= n; ++u) {
    if (!used[u]) {
        findCutPoints(findCutPoints, u, u);
    }
}
int ptr = 0;
vector<map<int, int>> colors(m);
auto markComponents = [&] (auto self, int u, int cur)
-> void {
    used[u] = 1;
    for (int v : g[u]) {
        if (!used[v]) {
            if (newCompWithoutParent[v]) {
                ptr++;
                self(self, v, ptr - 1);
            }
            else {
                self(self, v, cur);
            }
        }
        else if (h[v] < h[u]) {
            comp[{u,v}] = comp[{v,u}] = cur;
            int c = col[{u,v}];
            colors[cur][u] |= 1 << c;
            colors[cur][v] |= 1 << c;
        }
    }
};
used.assign(n + 1, 0);
for (int u = 1; u <= n; ++u) {
    if (!used[u]) {
        markComponents(markComponents, u, -1);
    }
}
for (int comp = 0; comp < m; ++comp) {
    vector<int> cnt(4);
    int tot = 0;
    for (auto [u, mask] : colors[comp]) {
        tot |= mask;
        cnt[bp(mask)]++;
    }
    if (bp(tot) < 3) {
        continue;
    }
    if (cnt[2] || cnt[3] > 2) {
        cout << "Yes" << endl;
        return;
    }
}
cout << "No" << endl;

```

### 3.4 DCP offline

```

struct Dsu {
    int n;
    vector<pair<int &, int>> s;
    vector<int> p, sz;
    // other info

    Dsu(int n) : n(n), p(n), sz(n, 1) {
        iota(all(p), 0);
    }

    int get(int u) {
        while (u != p[u]) u = p[u];
        return u;
    }

    bool merge(int u, int v) {
        u = get(u), v = get(v);
        if (u == v) return false;
        if (sz[v] < sz[u]) swap(u, v);
        s.append({p[u], p[u]});
        s.append({sz[v], sz[v]});
        // app other info like s.append({comp, comp});
        p[u] = v;
        sz[v] += sz[u];
    }
};

```

```

        return true;
    }

    void rollback(int sz) {
        while (s.size() != sz) {
            s.back().first = s.back().second;
            s.pop_back();
        }
    }
};

struct DcpOffline {
    int n;
    vector<vector<pair<int, int>>> d;

    void addEdgeOnSegment(int l, int r, int a, int b) {
        for (l += n, r += n; l < r; l /= 2, r /= 2) {
            if (l & 1) d[l++].append({a, b});
            if (r & 1) d[--r].append({a, b});
        }
    }

    template<typename T>
    void dfs(Dsu &dsu, T act) {
        dfs(1, 0, n, dsu, act);
    }

    template<typename T>
    void dfs(int v, int l, int r, Dsu &dsu, T act) {
        int sz = dsu.s.size();
        for (auto [u, v] : d[v]) {
            dsu.merge(u, v);
        }
        if (l + 1 == r) {
            act(l, dsu);
        }
        else {
            int m = (l + r) / 2;
            dfs(v * 2, l, m, dsu, act);
            dfs(v * 2 + 1, m, r, dsu, act);
        }
        dsu.rollback(sz);
    }

    DcpOffline(int maxt) : n(2 << __lg(maxt + 1)), d(2 * n) {}
};

```

### 3.5 Взвешенное паросочетание

<https://judge.yosupo.jp/submission/201334>

```

namespace blossom {
#define d(x) (lab[x.u] + lab[x.v] - 2 * e[x.u][x.v].w)
const int N = 403*2;
const int inf = 1e18;
struct Q { int u, v, w; } e[N][N];
vector<int> p[N];
int n, m = 0, id, h, t, lk[N], sl[N], st[N], f[N],
b[N][N], s[N], ed[N], q[N], lab[N];
void upd(int u, int v) { if (!sl[v] || d(e[u][v]) < d(e[sl[v]][v])) sl[v] = u; }
void ss(int v) {
    sl[v] = 0;
    for (int u = 1; u <= n; ++u) if (e[u][v].w > 0
        && st[u] != v && !s[st[u]]) upd(u, v);
}
void ins(int u) { if (u <= n) q[++t] = u; else for
(int v : p[u]) ins(v); }
void ch(int u, int w) { st[u] = w; if (u > n) for
(int v : p[u]) ch(v, w); }
int gr(int u, int v) {
    if ((v = find(all(p[u]), v) - p[u].begin()) &
1) {
        reverse(1 + all(p[u]));
        return (int)p[u].size() - v;
    }
    return v;
}
void stm(int u, int v) {
    lk[u] = e[u][v].v;
}

```

```

    if (u <= n) return; Q w = e[u][v];
    int x = b[u][w.u], y = gr(u,x);
    for (int i = 0; i < y; ++i) stm(p[u][i], p[u][
i^1]);
    stm(x, v); rotate(p[u].begin(), y+all(p[u]));
}
void aug(int u, int v) {
    int w = st[lk[u]];stm(u, v);if (!w) return;
    stm(w, st[f[w]]);
    aug(st[f[w]], w);
}
int lca(int u, int v) {
    for (id++; u|v; swap(u, v)) {
        if (!u) continue;if(ed[u] == id) return u;
        ed[u] = id; if (u = st[lk[u]]) u = st[f[u
]]; // =, not ==
    }
    return 0;
}
void add(int u, int a, int v) {
    int x = n + 1; while (x <= m && st[x]) ++x;
    if (x > m) ++m;
    lab[x] = s[x] = st[x] = 0;
    lk[x] = lk[a];
    p[x].clear();
    p[x].push_back(a);
#define op(q) for (int i = q, j = 0; i != a; i=st[f[j
]]) p[x].push_back(i), p[x].push_back(j=st[lk[i]])
    , ins(j) // also not ==
    op(u); reverse(l+all(p[x]));op(v);
    ch(x, x); for (int i = 1; i <= m; ++i) e[x][i
].w = e[i][x].w = 0;
    fill(b[x]+1, b[x]+n+1, 0);
    for (int u : p[x]) {
        for (int v = 1; v <= m; ++v) if (!e[x][v].
w || d(e[u][v]) < d(e[x][v])) e[x][v] = e[u][v], e
[v][x] = e[v][u];
        for (int v = 1; v <= n; ++v) if (b[u][v])
b[x][v] = u;
    }
    ss(x);
}
void ex(int u) {
    for (int x : p[u]) ch(x, x);
    int a = b[u][e[u][f[u]].u], r = gr(u, a);
    for (int i = 0; i < r; i += 2) {
        int x = p[u][i], y = p[u][i + 1];
        f[x] = e[y][x].u; s[x] = 1; s[y] = 0; sl[x
] = 0; ss(y); ins(y);
    }
    s[a] = 1; f[a] = f[u];
    for (int i = r + 1; i < p[u].size(); ++i) s[p[
u][i]] = -1, ss(p[u][i]);
    st[u] = 0;
}
bool on(const Q &e) {
    int u = st[e.u], v = st[e.v], a;
    if (s[v] == -1) {
        f[v] = e.u, s[v] = 1, a = st[lk[v]], sl[v]
= sl[a] = s[a] = 0, ins(a);
    } else if (!s[v]) {
        a = lca(u, v); if (!a) return aug(u, v),
aug(v, u), 1; else add(u, a, v);
    }
    return 0;
}
bool bfs() {
    fill(s+1, s+m+1, -1);fill(sl+1, sl+m+1, 0); //
s is filled with -1
    h = 1, t = 0; for (int i = 1; i <= m; ++i) if
(st[i] == i && !lk[i]) f[i] = s[i] = 0, ins(i);
    if (h > t) return 0;
    while (1) {
        while (h <= t) {
            int u = q[h++];
            if (s[st[u]] != 1) {
                for (int v = 1; v <= n; ++v) if (e
[u][v].w > 0 && st[u] != st[v]) {
                    if (d(e[u][v])) upd(u, st[v]);
                    else if (on(e[u][v])) return 1;
                }
            }
        }
    }
}

```

```

    }
    }
    int x = inf;
    for (int i = n+1; i <= m; ++i) if (st[i]
== i && s[i] == 1) x = min(x, lab[i]/2);
    for (int i = 1; i <= m; ++i) if (st[i] ==
i && sl[i] && s[i] != 1) x = min(x, d(e[sl[i]][i])
>>s[i]+1);
    for (int i = 1; i <= n; ++i) if (~s[st[i
]]) if ((lab[i] += (s[st[i]] * 2 - 1) * x) <=0)
return 0;
    for (int i = n + 1; i <= m; ++i) if (st[i]
== i && ~s[st[i]]) lab[i] += (2 - 4 * s[st[i]]) *
x;
    h = 1, t = 0;
    for (int i = 1; i <= m; ++i) if (st[i] ==
i && sl[i] && st[sl[i]] != i && !d(e[sl[i]][i]) &&
on(e[sl[i]][i])) return 1;
    for (int i = n+1; i <= m; ++i) if (st[i]
== i && s[i] == 1 && !lab[i]) ex(i);
}
pair<int, vector<array<int, 2>>> run(int N, vector
<array<int, 3>> edges) {
    for (auto &[u, v, w] : edges) ++u, ++v;
    fill(ed+1, ed+m+1, 0);
    fill(lk+1, lk+m+1, 0);
    n = m = N;
    id = 0;
    iota(st + 1, st + n + 1, 1);
    int wm = 0, weight = 0;
    for (int i = 1; i <= n; ++i) for (int j = 1; j
<= n; ++j) e[i][j] = {i,j,0};
    for (auto [u, v, w] : edges) wm = max(wm, e[v
][u].w = e[u][v].w = max(e[u][v].w, w));
    for (int i = 1; i <= n; ++i) p[i].clear();
    for (int i = 1; i <= n; ++i) for (int j = 1; j
<= n; ++j) b[i][j] = i==j?i:0;
    fill_n(lab+1, n, wm); while (bfs());
    vector<array<int, 2>> matching;
    for (int i = 1; i <= n; ++i) if (i < lk[i])
weight += e[i][lk[i]].w, matching.push_back({i -
1, lk[i] - 1});
    return {weight, matching};
}
}

```

## 4 Свёртки

### 4.1 AND, OR, XOR свёртки

```

const int p = 998244353;
vector<int> band(vector<int> a,vector<int> b)
{
    int n=0;while((1<<n)<a.size()) ++n;
    a.resize(1<<n);b.resize(1<<n);
    for(int i=0;i<n;++i) for(int mask=0;mask<(1<<n);++
mask) if(mask & (1<<i)) {a[mask-(1<<i)]+=a[mask];a
[mask-(1<<i)]%=p;}
    for(int i=0;i<n;++i) for(int mask=0;mask<(1<<n);++
mask) if(mask & (1<<i)) {b[mask-(1<<i)]+=b[mask];b
[mask-(1<<i)]%=p;}
    vector<int> c(1<<n,0);
    for(int mask=0;mask<(1<<n);++mask) {c[mask]=a[mask]*
b[mask];c[mask]%=p;}
    for(int i=0;i<n;++i) for(int mask=0;mask<(1<<n);++
mask) if(!(mask & (1<<i))) {c[mask]-=c[mask+(1<<i)
];c[mask]%=p;}
    return c;
}
vector<int> bor(vector<int> a,vector<int> b)
{
    int n=0;while((1<<n)<a.size()) ++n;
    a.resize(1<<n);b.resize(1<<n);
    for(int i=0;i<n;++i) for(int mask=0;mask<(1<<n);++
mask) if(!(mask & (1<<i))) {a[mask+(1<<i)]+=a[mask
];a[mask+(1<<i)]%=p;}
}

```

```

for(int i=0;i<n;++i) for(int mask=0;mask<(1<<n);++
    mask) if(!(mask & (1<<i))) {b[mask+(1<<i)]+=b[mask
    ];b[mask+(1<<i)]%=p;}
vector<int> c(1<<n,0);
for(int mask=0;mask<(1<<n);++mask) {c[mask]=a[mask]*
    b[mask];c[mask]%=p;}
for(int i=0;i<n;++i) for(int mask=0;mask<(1<<n);++
    mask) if(mask & (1<<i)) {c[mask]-=c[mask-(1<<i)];c
    [mask]%=p;}
return c;
}
vector<int> bxor(vector<int> a,vector<int> b)
{
    assert(p%2==1);int inv2=(p+1)/2;
    int n=0;while((1<<n)<a.size()) ++n;
    a.resize(1<<n);b.resize(1<<n);
    for(int i=0;i<n;++i) for(int mask=0;mask<(1<<n);++
        mask) if(!(mask & (1<<i))) {int u=a[mask],v=a[mask
        +(1<<i)];a[mask+(1<<i)]=(u+v)%p;a[mask]=(u-v)%p;}
    for(int i=0;i<n;++i) for(int mask=0;mask<(1<<n);++
        mask) if(!(mask & (1<<i))) {int u=b[mask],v=b[mask
        +(1<<i)];b[mask+(1<<i)]=(u+v)%p;b[mask]=(u-v)%p;}
    vector<int> c(1<<n,0);
    for(int mask=0;mask<(1<<n);++mask) {c[mask]=a[mask]*
        b[mask];c[mask]%=p;}
    for(int i=0;i<n;++i) for(int mask=0;mask<(1<<n);++
        mask) if(!(mask & (1<<i))) {int u=c[mask],v=c[mask
        +(1<<i)];c[mask+(1<<i)]=(v-u)*inv2%p;c[mask]=(u
        +v)*inv2%p;}
    return c;
}

```

## 4.2 FFT & co

```

typedef long long ll;
const int p=998244353;
int po(int a,int b) {if(b==0) return 1; if(b==1)
    return a; if(b%2==0) {int u=po(a,b/2);return (u*1
    LL*u)%p;} else {int u=po(a,b-1);return (a*1LL*u)%p
    ;}}
int inv(int x) {return po(x,p-2);}
template<int M, int K, int G> struct Fft {
    // 1, 1/4, 1/8, 3/8, 1/16, 5/16, 3/16, 7/16, ...
    int g[1 << (K - 1)];
    Fft() : g() { //if t1 constexpr...
        static_assert(K >= 2, "Fft: K >= 2 must hold");
        g[0] = 1;
        g[1 << (K - 2)] = G;
        for (int l = 1 << (K - 2); l >= 2; l >>= 1) {
            g[l >> 1] = (g[l] * 1LL * g[l]) % M;
        }
        assert((g[1]*1LL * g[1]) % M == M - 1);
        for (int l = 2; l <= 1 << (K - 2); l <<= 1) {
            for (int i = 1; i < l; ++i) {
                g[l + i] = (g[l] * 1LL * g[i]) % M;
            }
        }
    }
}
void fft(vector<int> &x) const {
    const int n = x.size();
    assert(n <= 1 << K);
    for (int h = __builtin_ctz(n); h--; ) {
        const int l = (1 << h);
        for (int i = 0; i < n >> (h+1); ++i) {
            for (int j = i << (h+1); j < (((i << 1) + 1)
            << h); ++j) {
                const int t = (g[i] * 1LL * x[j | l]) % M;
                x[j | l] = x[j] - t;
                if (x[j|l] < 0) x[j | l] += M;
                x[j]+=t;
                if (x[j] >= M) x[j] -= M;
            }
        }
    }
    for (int i = 0, j = 0; i < n; ++i) {
        if (i < j) std::swap(x[i], x[j]);
        for (int l = n; (l >>= 1) && !((j ^= 1) & 1); )
        {}
    }
}

```

```

vector<int> convolution(vector<int> a, vector<int> b
    ) const {
    if(a.empty() || b.empty()) return {};
    for(int& x:a) {x%=p;if(x>=p) x-=p; if(x<0) x+=p;}
    for(int& x:b) {x%=p;if(x>=p) x-=p; if(x<0) x+=p;}
    const int na = a.size(), nb = b.size();
    int n, invN = 1;
    for (n = 1; n < na + nb - 1; n <= 1) invN = ((
    invN & 1) ? (invN + M) : invN) >> 1;
    vector<int> x(n, 0), y(n, 0);
    std::copy(a.begin(), a.end(), x.begin());
    std::copy(b.begin(), b.end(), y.begin());
    fft(x);
    fft(y);
    for (int i = 0; i < n; ++i) x[i] = (((static_cast<
    long long>(x[i]) * y[i]) % M) * invN) % M;
    std::reverse(x.begin() + 1, x.end());
    fft(x);
    x.resize(na + nb - 1);
    return x;
}
};
Fft<998244353,23,31> muls;

vector<int> form(vector<int> v,int n)
{
    while(v.size()<n) v.push_back(0);
    while(v.size()>n) v.pop_back();
    return v;
}
vector<int> operator *(vector<int> v1,vector<int> v2)
{
    return muls.convolution(v1,v2);
}
vector<int> operator +(vector<int> v1,vector<int> v2)
{
    while(v2.size()<v1.size()) v2.push_back(0); while(v1
    .size()<v2.size()) v1.push_back(0);
    for(int i=0;i<v1.size();++i) {v1[i]+=v2[i];if(v1[i
    ]>=p) v1[i]-=p; else if(v1[i]<0) v1[i]+=p;}
    return v1;
}
vector<int> operator -(vector<int> v1,vector<int> v2)
{
    int sz=max(v1.size(),v2.size());while(v1.size()<sz)
    v1.push_back(0); while(v2.size()<sz) v2.push_back
    (0);
    for(int i=0;i<sz;++i) {v1[i]-=v2[i];if(v1[i]<0) v1[i
    ]+=p; else if(v1[i]>=p) v1[i]-=p;} return v1;
}
vector<int> trmi(vector<int> v)
{
    for(int i=1;i<v.size();i+=2) {if(v[i]>0) v[i]=p-v[i
    ]; else v[i]=(-v[i]);}
    return v;
}
vector<int> deriv(vector<int> v)
{
    if(v.empty()) return{};
    vector<int> ans(v.size()-1);
    for(int i=1;i<v.size();++i) ans[i-1]=(v[i]*1LL*i)%p;
    return ans;
}
vector<int> integ(vector<int> v)
{
    vector<int> ans(v.size()+1);ans[0]=0;
    for(int i=1;i<v.size();++i) ans[i-1]=(v[i]*1LL*i)%p;
    return ans;
}
vector<int> mul(vector<vector<int>> > v)
{
    if(v.size()==1) return v[0];
    vector<vector<int>> > v1,v2;for(int i=0;i<v.size()
    /2;++i) v1.push_back(v[i]); for(int i=v.size()/2;i
    <v.size();++i) v2.push_back(v[i]);
    return muls.convolution(mul(v1),mul(v2));
}
vector<int> invl(vector<int> v,int n)
{
    assert(v[0]!=0);

```

```

int sz=1;v=form(v,n);vector<int> a={inv(v[0])};
while(sz<n)
{
    vector<int> vsz;for(int i=0;i<min(n,2*sz);++i) vsz
    .push_back(v[i]);
    vector<int> b=((vector<int>) {1})-muls.convolution
    (a,vsz);
    for(int i=0;i<sz;++i) assert(b[i]==0);
    b.erase(b.begin(),b.begin()+sz);
    vector<int> c=muls.convolution(b,a);
    for(int i=0;i<sz;++i) a.push_back(c[i]);
    sz*=2;
}
return form(a,n);
}

```

### 4.3 Быстрое FFT

- Solution based on <https://codeforces.com/blog/entry/117947>
- Iterative and in-place version.
- Uses signed montgomery
- Optimized to minimize memory usage

```

const int MOD = 998244353;
const long long MOD2 = (long long) MOD * MOD;
const int root = 3;
const int alim = 64; // Bound for using O(n^2)
                        polynomial mult

int modpow(int b, int e) {
    int ans = 1;
    for (; e; b = (long long) b * b % MOD, e /= 2)
        if (e & 1) ans = (long long) ans * b % MOD;
    return ans;
}

const int MODinv = 2 - MOD; // pow(-MOD, -1, 2**32)
inline int m_reduce(long long x) {
    int m = x * MODinv;
    return (x>>32) - (((long long) m * MOD) >> 32);
}

const int r2 = modpow(2, 64);
inline int m_transform(int x) {
    return m_reduce((long long)x * r2);
}

inline int m_add(int x, int y) {
    int z = x + y;
    return z < 0 ? z + MOD : z - MOD;
}

inline int m_sub(int x, int y) {
    int z = x - y;
    return z < 0 ? z + MOD : z - MOD;
}

inline int m_mult(int x, int y) {
    return m_reduce((long long) x * y);
}

vector<int> rt = {1};
vector<int> transformed_rt;
vector<int> transformed_rt2;

template<int a>
void transform(vector<int> &P) {
    int m = P.size();
    int n = m / a;

    int size = rt.size();
    while (2 * size < n) {
        rt.resize(n / 2);
        int r = modpow(root, MOD / (4 * size));
        for (int i = 0; i < size; ++i)
            rt[i + size] = (long long) r * rt[i] % MOD
    }
    size *= 2;
}

```

```

}

// For montgomery
for (int i = transformed_rt.size(); i < rt.size();
    ++i) {
    transformed_rt.resize(rt.size());
    transformed_rt[i] = m_transform(rt[i]);
    transformed_rt2.resize(rt.size());
    transformed_rt2[i] = (unsigned int) MODinv *
    transformed_rt[i];
}

int k = n;
while (k >= 4) k /= 4;

if (k == 2) {
    int step = n * a;
    int half_step = step / 2;
    for (int j1 = 0; j1 < half_step; ++j1) {
        int j2 = j1 + half_step;

        int diff = m_sub(P[j1], P[j2]);
        P[j1] = m_add(P[j1], P[j2]);
        P[j2] = diff;
    }
    k = n/2;
} else {
    k = n;

    for (; k > 1; k /= 4) {
        for (int i = 0; i < n/k; ++i) {
            int step = k * a;
            int half_step = step / 2;
            int quarter_step = half_step / 2;

            int R20 = transformed_rt2[2 * i];
            int RR0 = transformed_rt[2 * i];

            int R21 = transformed_rt2[2 * i + 1];
            int RR1 = transformed_rt[2 * i + 1];

            int R2 = transformed_rt2[i];
            int RR = transformed_rt[i];

            int j1 = i * step;
            int j2 = j1 + quarter_step;
            int j3 = j2 + quarter_step;
            int j4 = j3 + quarter_step;

            for (int j = 0; j < quarter_step; ++j, ++
                j1, ++j2, ++j3, ++j4) {
                int z0;
                {
                    int z = P[j3];
                    int m = (unsigned int) R2 * z;
                    z0 = ((long long) z * RR - (long
                        long) m * MOD) >> 32;
                }

                int z1;
                {
                    int z = P[j4];
                    int m = (unsigned int) R2 * z;
                    z1 = ((long long) z * RR - (long
                        long) m * MOD) >> 32;
                }

                int sum0 = m_add(P[j1], z0);
                int diff0 = m_sub(P[j1], z0);
                int sum1 = P[j2] + z1;
                int diff1 = P[j2] - z1;

                // [sum0, sum1, diff0, diff1]

                int zz0;
                {
                    int z = sum1;
                    int m = (unsigned int) R20 * z;
                    zz0 = ((long long) z * RR0 - (long

```



```

    long) m * MOD) >> 32;
    }

    int zz1;
    {
        int z = diff1;
        int m = (unsigned int) R21 * z;
        zz1 = ((long long) z * RR1 - (long
long) m * MOD) >> 32;
    }

    P[j1] = m_add(sum0, zz0);
    P[j2] = m_sub(sum0, zz0);
    P[j3] = m_add(diff0, zz1);
    P[j4] = m_sub(diff0, zz1);
    }
}

for (int i = 0; i < m; ++i)
    if (P[i] < 0) P[i] += MOD;
}

template<int a>
void inverse_transform(vector<int> &P) {
    int m = P.size();
    int n = m / a;
    int n_inv = m_transform(modpow(n, MOD - 2));

    vector<int> rev(n);
    for (int i = 1; i < n; ++i) {
        rev[i] = rev[i / 2] / 2 + (i & 1) * n / 2;
    }

    // P = [p * n_inv for p in P]
    for (int i = 0; i < m; ++i)
        P[i] = m_mult(n_inv, P[i]);

    // P = [P[a * rev[i // a] + (i % a)] for i in
range(m)]
    for (int i = 1; i < n; ++i)
        if (i < rev[i])
            swap_ranges(P.begin() + a * i, P.begin() +
a * i + a, P.begin() + a * rev[i]);

    // P = [P[-a * (i // a) + (i % a)] for i in range(
m)]
    for (int i = 1; i < n/2; ++i)
        swap_ranges(P.begin() + a * i, P.begin() + a *
i + a, P.begin() + a * (n - i));

    transform<a>(P);

    // P = [P[a * rev[i // a] + (i % a)] for i in
range(m)]
    for (int i = 1; i < n; ++i)
        if (i < rev[i])
            swap_ranges(P.begin() + a * i, P.begin() +
a * i + a, P.begin() + a * rev[i]);
}

template<int a>
void fast_polymult_mod(vector<int> &P, vector<int> &Q)
{
    int m = P.size();
    int n = m / a;

    transform<a>(P);
    transform<a>(Q);

    vector<int> &PQ = P;
    for (int i = 0; i < n; ++i) {
        vector<unsigned long long> res(2 * a);
        for (int j = 0; j < a; ++j) {
            if (j >= 10 && j % 9 == 8)
                for (int k = j; k < j + a - 10; ++k)
                    res[k] -= (res[k] >> 63) * 9 *
MOD2;
            for (int k = 0; k < a; ++k)
                res[j + k] += (long long) P[i * a + j]

```

```

        * Q[i * a + k];
    }

    int c = rt[i/2];
    if (i & 1) c = MOD - c;
    for (int j = 0; j < a; ++j)
        PQ[i * a + j] = (res[j] + c * (res[j + a]
% MOD)) % MOD;
}

inverse_transform<a>(PQ);
}

template <size_t... N>
void work(std::index_sequence<N...>, int x, std:::
vector<int>& a, std::vector<int>& b) {
    static void (*ptrs[]) (std::vector<int>&, std:::
vector<int>&) = {&fast_polymult_mod<N+1>...};
    ptrs[x - 1](a, b);
}

void fast_polymult(vector<int> &P, vector<int> &Q) {
    int m1 = P.size();
    int m2 = Q.size();
    int res_len = m1 + m2 - 1;

    int b = 1;
    while ((alim << b) < res_len) ++b;
    int a = ((res_len - 1) >> b) + 1;
    int m = a << b;

    P.resize(m);
    Q.resize(m);

    // Call fast_polymult_mod<a>(P, Q);
    work(std::make_index_sequence<alim>{}, a, P, Q);

    P.resize(res_len);
}

```

#### 4.4 FFT в double'ax

```

using cd = complex<double>;
const double PI = acos(-1);

void fft(vector<cd> &a, bool invert) {
    int n = a.size();

    for (int i = 1, j = 0; i < n; i++) {
        int bit = n >> 1;
        for (; j & bit; bit >>= 1)
            j ^= bit;
        j ^= bit;

        if (i < j)
            swap(a[i], a[j]);
    }

    for (int len = 2; len <= n; len <= 1) {
        double ang = 2 * PI / len * (invert ? -1 : 1);
        cd wlen(cos(ang), sin(ang));
        for (int i = 0; i < n; i += len) {
            cd w(1);
            for (int j = 0; j < len / 2; j++) {
                cd u = a[i+j], v = a[i+j+len/2] * w;
                a[i+j] = u + v;
                a[i+j+len/2] = u - v;
                w *= wlen;
            }
        }
    }

    if (invert) {
        for (cd &x : a)
            x /= n;
    }
}

vector<int> multiply(vector<int> const& a, vector<int>
const& b) {

```

```

vector<cd> fa(a.begin(), a.end()), fb(b.begin(), b.
    end());
int n = 1;
while (n < a.size() + b.size())
    n <= 1;
fa.resize(n);
fb.resize(n);

fft(fa, false);
fft(fb, false);
for (int i = 0; i < n; i++)
    fa[i] *= fb[i];
fft(fa, true);

vector<int> result(n);
for (int i = 0; i < n; i++)
    result[i] = round(fa[i].real());
while(!result.empty() && !result.back()) result.
    pop_back();
return result;
}

```

## 5 Структуры данных

### 5.1 Дерево Фенвика

```

int fe[maxn];
void pl(int pos, int val) {while(pos<maxn) {fe[pos]+=
    val;pos+=(pos+1);}}
int get(int pos) {int ans=0;while(pos>=0) {ans+=fe[pos
    ];pos&=(pos+1);--pos;} return ans;} /// [0,pos] -
    vkluchitelno!!!
int get(int l, int r) {return get(r-1)-get(l-1);} ///
    sum of [l,r)

```

### 5.2 Дерево отрезков в точке

```

template<typename T, typename U>
struct SegmentTree {
    int h, n;
    T neutral;
    U unite;
    vector<T> data;

    template<typename I>
    SegmentTree(int sz, T neutral, U unite, I init) :
        h(__lg(sz) + 1), n(1 << h), neutral(neutral),
        unite(unite), data(2 * n) {
        for (int i = 0; i < sz; ++i) data[i + n] =
            init(i);
        for (int i = n - 1; i > 0; --i) data[i] =
            unite(data[2 * i], data[2 * i + 1]);
    }

    SegmentTree(int sz, T neutral, U unite) : h(__lg(
        sz) + 1), n(1 << h), neutral(neutral), unite(unite
        ), data(2 * n, neutral) {}

    void set(int i, T x) {
        data[i += n] = x;
        for (i /= 2; i > 0; i /= 2) data[i] = unite(
            data[2 * i], data[2 * i + 1]);
    }

    T get(int l, int r) {
        T leftRes = neutral, rightRes = neutral;
        for (l += n, r += n; l < r; l /= 2, r /= 2) {
            if (l & 1) leftRes = unite(leftRes, data[l
                ++]);
            if (r & 1) rightRes = unite(data[--r],
                rightRes);
        }
        return unite(leftRes, rightRes);
    }

    int left(int i) {
        int lvl = __lg(i);
        return (i & ((1 << lvl) - 1)) * (1 << (h - lvl
            ));
    }
    int right(int i) {

```

```

        int lvl = __lg(i);
        return ((i & ((1 << lvl) - 1)) + 1) * (1 << (h
            - lvl));
    }

    // l \in [0; n) && ok(get(l, l), l);
    // returns last r: ok(get(l, r), r)
    template<typename C>
    int lastTrue(int l, C ok) {
        T cur = neutral;
        l += n;
        do {
            l >>= __builtin_ctz(l);
            T withl = unite(cur, data[l]);
            if (ok(withl, right(l))) {
                cur = withl;
                ++l;
            } else {
                while (l < n) {
                    T with2 = unite(cur, data[2 * l]);
                    if (ok(with2, right(2 * l))) {
                        cur = with2;
                        l = 2 * l + 1;
                    } else {
                        l = 2 * l;
                    }
                }
                return l - n;
            }
        } while (l & (1 - 1));
        return n;
    }

    // r \in [0; n) && ok(get(r, r), r);
    // returns first l: ok(get(l, r), l)
    template<typename C>
    int firstTrue(int r, C ok) {
        T cur = neutral;
        r += n;
        while (r & (r - 1)) {
            r >>= __builtin_ctz(r);
            T withl = unite(data[--r], cur);
            if (ok(withl, left(r))) {
                cur = withl;
            } else {
                while (r < n) {
                    T with2 = unite(data[2 * r + 1],
                        cur);
                    if (ok(with2, left(2 * r + 1))) {
                        cur = with2;
                        r = 2 * r;
                    } else {
                        r = 2 * r + 1;
                    }
                }
                return r - n + 1;
            }
        }
        return 0;
    }
};

void example () {
    // max
    SegmentTree<int> segtree(n, -(int)1e18, [](int x,
        int y) { return max(x, y); });

    // sum
    SegmentTree<int> ones(n, 0LL, [](int x, int y) {
        return x + y; });

    auto left_zero = [&](int r) { // nearest zero
        strictly to the left
        return ones.firstTrue(r, [r](int sum, int l){
            return r - l == sum; }) - 1;
    };
    auto right_zero = [&](int l) { // nearest zero
        strictly to the right
        return ones.lastTrue(l + 1, [l](int sum, int r){
            return r - (l + 1) == sum; });
    };
}

```



```
};
}
```

### 5.3 Массовое дерево отрезков

```
#ifndef LOCAL
int __lg(int x) { return 63 - __builtin_clzll(x); }
#endif

template<typename Data, typename Mod, typename
    UniteData, typename UniteMod, typename Apply>
struct MassSegmentTree {
    int h, n;
    Data zd;
    Mod zm;
    vector<Data> data;
    vector<Mod> mod;

    UniteData ud; // Data (Data, Data)
    UniteMod um; // Mod (Mod, Mod);
    Apply a; // Data (Data, Mod, int); last argument is
    // the length of current segment (could be used for
    // range += and sum counting, for instance)

    template<typename I>
    MassSegmentTree(int sz, Data zd, Mod zm, UniteData
        ud, UniteMod um, Apply a, I init) : h(__lg(sz > 1
        ? sz - 1 : 1) + 1), n(1 << h), zm(zm), zd(zd),
        data(2 * n, zd), mod(n, zm), ud(ud), um(um), a(a)
        {
            for (int i = 0; i < sz; ++i) data[i + n] = init(i);
            for (int i = n - 1; i > 0; --i) data[i] = ud(data
                [2 * i], data[2 * i + 1]);
        }

    MassSegmentTree(int sz, Data zd, Mod zm, UniteData
        ud, UniteMod um, Apply a) : h(__lg(sz > 1 ? sz - 1
        : 1) + 1), n(1 << h), zm(zm), zd(zd), data(2 * n,
        zd), mod(n, zm), ud(ud), um(um), a(a) {}

    void push(int i) {
        if (mod[i] == zm) return;
        apply(2 * i, mod[i]);
        apply(2 * i + 1, mod[i]);
        mod[i] = zm;
    }

    // is used only for apply
    int length(int i) { return 1 << (h - __lg(i)); }

    // used only for descent
    int left(int i) {
        int lvl = __lg(i);
        return (i & ((1 << lvl) - 1)) * (1 << (h - lvl));
    }

    // used only for descent
    int right(int i) {
        int lvl = __lg(i);
        return ((i & ((1 << lvl) - 1)) + 1) * (1 << (h -
            lvl));
    }

    template<typename S>
    void apply(int i, S x) {
        data[i] = a(data[i], x, length(i));
        if (i < n) mod[i] = um(mod[i], x);
    }

    void update(int i) {
        if (mod[i] != zm) return;
        data[i] = ud(data[2 * i], data[2 * i + 1]);
    }

    template<typename S>
    void update(int l, int r, S x) { // [l; r)
        l += n, r += n;
        for (int shift = h; shift > 0; --shift) {
            push(l >> shift);
            push((r - 1) >> shift);
        }
    }
};
```

```

    }
    for (int lf = 1, rg = r; lf < rg; lf /= 2, rg /=
        2) {
        if (lf & 1) apply(lf++, x);
        if (rg & 1) apply(--rg, x);
    }
    for (int shift = 1; shift <= h; ++shift) {
        update(l >> shift);
        update((r - 1) >> shift);
    }
}

Data get(int l, int r) { // [l; r)
    l += n, r += n;
    for (int shift = h; shift > 0; --shift) {
        push(l >> shift);
        push((r - 1) >> shift);
    }
    Data leftRes = zd, rightRes = zd;
    for (; l < r; l /= 2, r /= 2) {
        if (l & 1) leftRes = ud(leftRes, data[l++]);
        if (r & 1) rightRes = ud(data[--r], rightRes);
    }
    return ud(leftRes, rightRes);
}

// l \in [0; n) && ok(get(l, l), l);
// returns last r: ok(get(l, r), r)
template<typename C>
int lastTrue(int l, C ok) {
    l += n;
    for (int shift = h; shift > 0; --shift) push(l >>
        shift);
    Data cur = zd;
    do {
        l >>= __builtin_ctz(l);
        Data withl;
        withl = ud(cur, data[l]);
        if (ok(withl, right(l))) {
            cur = withl;
            ++l;
        } else {
            while (l < n) {
                push(l);
                Data with2;
                with2 = ud(cur, data[2 * l]);
                if (ok(with2, right(2 * l))) {
                    cur = with2;
                    l = 2 * l + 1;
                } else {
                    l = 2 * l;
                }
            }
            return l - n;
        }
    } while (l & (l - 1));
    return n;
}

// r \in [0; n) && ok(get(r, r), r);
// returns first l: ok(get(l, r), l)
template<typename C>
int firstTrue(int r, C ok) {
    r += n;
    for (int shift = h; shift > 0; --shift) push((r -
        1) >> shift);
    Data cur = zd;
    while (r & (r - 1)) {
        r >>= __builtin_ctz(r);
        Data withl;
        withl = ud(data[--r], cur);
        if (ok(withl, left(r))) {
            cur = withl;
        } else {
            while (r < n) {
                push(r);
                Data with2;
                with2 = ud(data[2 * r + 1], cur);
                if (ok(with2, left(2 * r + 1))) {
                    cur = with2;
                }
            }
        }
    }
}
```

```

        r = 2 * r;
    } else {
        r = 2 * r + 1;
    }
    return r - n + 1;
}
return 0;
};

void example () {
    // max and +=
    MassSegmentTree segtree(n, 0LL, 0LL,
    [](int x, int y) { return max(x, y); },
    [](int x, int y) { return x + y; },
    [](int x, int y, int len) { return x + y; });

    // sum and +=
    MassSegmentTree segtree(n, 0LL, 0LL,
    [](int x, int y) { return x + y; },
    [](int x, int y) { return x + y; },
    [](int x, int y, int len) { return x + y * len; });

    // sum and assignment
    MassSegmentTree segtree(n, 0LL, -1LL,
    [](int x, int y) { return x + y; },
    [](int x, int y) { return y; },
    [](int x, int y, int len) { return y * len; });
}

```

## 5.4 Битовый бор

```

template<unsigned int sz, typename T=int>
struct binarytrie{
    using Bit=typename conditional<sz<=32,unsigned int,
    unsigned long long>::type;
    struct node{
        T cnt;
        array<int,2>nxt;
        node():cnt(0),nxt({-1,-1}){}
    };
    vector<node>v;
    binarytrie(){v.emplace_back();}
    void insert(Bit x){add(x,1);}
    void erase(Bit x){add(x,-1);}
    void add(Bit x,T k)
    {
        assert(0<=x&&(x>>sz)==0);
        int p=0;
        v[p].cnt+=k;
        for(int i=sz;i--;)
        {
            int j=x>>i&1;
            if(v[p].nxt[j]==-1)
            {
                v[p].nxt[j]=v.size();
                v.emplace_back();
            }
            p=v[p].nxt[j];
            v[p].cnt+=k;
        }
    }
    T count(Bit x, Bit xor_val=0) const // [0, x)
    {
        assert(0<=xor_val&&(xor_val>>sz)==0);
        if(x<0) return 0;
        else if(x>>sz) return v[0].cnt;
        T ret=0;
        int p=0;
        for(int i=sz;i--;)
        {
            int j=x>>i&1, k=xor_val>>i&1;
            if(j==0)p=v[p].nxt[k];
            else
            {
                if(v[p].nxt[k]>=0)ret+=v[v[p].nxt[k]].cnt;
                p=v[p].nxt[!k];
            }
            if(p==-1)break;

```

```

        }
        return ret;
    }
    Bit max(Bit xor_val=0) const
    {
        assert(0<=xor_val&&(xor_val>>sz)==0);
        int p=0;
        Bit ret=0;
        if(v[p].cnt==0) return ret;
        for(int i=sz;i--;)
        {
            ret<<=1;
            int k=xor_val>>i&1;
            if(v[p].nxt[!k]>=0&&v[v[p].nxt[!k]].cnt>0)
            {
                p=v[p].nxt[!k];
                ret|=1;
            }
            else p=v[p].nxt[k];
        }
        return ret;
    }
    Bit min(Bit xor_val=0) const
    {
        assert(0<=xor_val&&(xor_val>>sz)==0);
        int p=0;
        Bit ret=0;
        for(int i=sz;i--;)
        {
            ret<<=1;
            int k=xor_val>>i&1;
            if(v[p].nxt[k]>=0&&v[v[p].nxt[k]].cnt>0)p=v[p].
            nxt[k];
            else
            {
                p=v[p].nxt[!k];
                ret|=1;
            }
        }
        return ret;
    }
    Bit find_by_order(T ord, Bit xor_val=0) const
    {
        assert(0<=xor_val&&(xor_val>>sz)==0);
        assert(0<=ord&&ord<v[0].cnt);
        int p=0;
        Bit ret=0;
        for(int i=sz;i--;)
        {
            ret<<=1;
            int k=xor_val>>i&1;
            if(v[p].nxt[k]>=0)
            {
                if(ord>=v[v[p].nxt[k]].cnt)
                {
                    ord-=v[v[p].nxt[k]].cnt;
                    p=v[p].nxt[k];
                    ret|=1;
                }
                else p=v[p].nxt[k];
            }
            else
            {
                p=v[p].nxt[!k];
                ret|=1;
            }
        }
        return ret;
    }
    T order_of_key(Bit x, Bit xor_val=0) const {return
    count(x, xor_val);}
};
binarytrie<32>bt;

```

## 5.5 Ordered set

```

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

using namespace __gnu_pbds;

```

```
using namespace std;

using ordered_set = tree<int, null_type, less<>,
    rb_tree_tag, tree_order_statistics_node_update>;
```

## 5.6 Convex hull trick

```
int div_up(int a, int b) { return a/b+((a^b)>0&&a%b);
} // divide a by b rounded up
const int LQ = ..., RQ = ...; //leftmost query,
    rightmost query
int in(ii L, int x) {
    return L.x * x + L.y;
}
struct Hull {
    vector<pair<int, int>> lines;
    vector<int> borders;
    void push(ii L) {
        while (lines.size() && in(L,borders.back()) < in(
            lines.back(),borders.back())) {
            lines.pop_back();
            borders.pop_back();
        }
        if (lines.empty()) {
            lines = {L};
            borders = {LQ};
        }
        else if (lines.back().x > L.x) {
            int x = div_up(L.y - lines.back().y, lines.
                back().x - L.x);
            if (x <= RQ) {
                lines.app(L);
                borders.app(x);
            }
        }
    }
};
Hull (){}
Hull (vector<ii> a) {
    auto comp = [&] (ii u, ii v) {
        return u.x > v.x || (u.x == v.x && u.y < v.y);
    };
    sort(all(a), comp);
    for (auto L : a) {
        push(L);
    }
}
int get(int x) {
    int pos = upper_bound(all(borders), x) - borders.
        begin();
    assert(pos>0);
    pos--;
    return in(lines[pos],x);
}
};
```

## 5.7 Центроиды

```
vector<int> sz(n), lvl(n, -1);
auto dfs = [&](auto dfs, int cur, int prev) -> int {
    if (lvl[cur] != -1) return 0;
    sz[cur] = 1;
    for (auto [nxt, w] : g[cur]) {
        if (nxt != prev) sz[cur] += dfs(dfs, nxt, cur);
    }
    return sz[cur];
};
auto find = [&](auto find, int cur, int prev, int tot)
    -> int {
    int bch = -1, bsz = 0;
    for (auto [nxt, w] : g[cur]) {
        if (nxt == prev || lvl[nxt] != -1) continue;
        if (sz[nxt] > bsz) {
            bch = nxt;
            bsz = sz[nxt];
        }
    }
    if (bsz + bsz <= tot) return cur;
    return find(find, bch, cur, tot);
};
dfs(dfs, 0, 0);
```

```
auto c = find(find, 0, 0, sz[0]);
vector<pair<int, int>> stack{{c, 0}};
while (!stack.empty()) {
    auto [centroid, l] = stack.back();
    stack.pop_back();
    lvl[centroid] = 1;
    for (auto [nxt, w] : g[centroid]) {
        if (lvl[nxt] != -1) continue;
        dfs(dfs, nxt, centroid);
        int new_centroid = find(find, nxt, centroid,
            sz[nxt]);
        stack.push_back({new_centroid, lvl[centroid] +
            1});
    }
}
```

## 5.8 Дерево Ли Чао

```
struct Line{
    int a, b;
    Line(){}
    Line (int a, int b) : a(a), b(b) {}
    int get(int x) { return a + b * x;}
};

struct Lichao {
    int n;
    vector<int> x;
    vector<Line> t;
    Lichao(){}
    Lichao (int n, vector<int> x) : n(n), t(n << 2,
        Line(0, 0)), x(x) {}

    void put(int v, int l, int r, Line L) {
        if (l + 1 == r) {
            if (L.get(x[l]) < t[v].get(x[l])) {
                t[v] = L;
            }
            return;
        }
        int m = (l + r) / 2;
        if (L.get(x[m]) < t[v].get(x[m])) {
            swap(L, t[v]);
        }
        if (L.b > t[v].b) {
            put(2 * v + 1, l, m, L);
        }
        else {
            put(2 * v + 2, m, r, L);
        }
    }

    int get(int v, int l, int r, int i) {
        if (l + 1 == r) {
            return t[v].get(x[l]);
        }
        int m = (l + r) / 2;
        int ans = t[v].get(x[i]);
        if (i < m) {
            ans = min(ans, get(2 * v + 1, l, m, i));
        }
        else {
            ans = min(ans, get(2 * v + 2, m, r, i));
        }
        return ans;
    }

    void put(Line L) {
        put(0, 0, n, L);
    }

    int get(int i) {
        return get(0, 0, n, i);
    }
};
```

## 5.9 Min-Kinetic Segment Tree

I guess the source is <https://koosaga.com/307>

```
using lint = long long;
const lint inf = 4e18;
const int MAXT = 4100000;
```

```

using pi = array<lint, 2>;

struct line {
    lint A, B;
    int idx;

    lint eval(lint x) { return A * x + B; }

    // returns the x-intercept of intersection "
    // strictly" larger than T
    lint cross_after(line &x, lint T) {
        if (x.A == A) {
            return inf;
        }
        lint up = x.B - B;
        lint dn = A - x.A;
        if (dn < 0) {
            dn *= -1;
            up *= -1;
        }
        lint incep = (up <= 0 ? -((-up) / dn) : (up +
dn - 1) / dn);
        if (incep > T)
            return incep;
        return inf;
    }
};

struct kst { // min kinetic segment tree
    line tree[MAXT];
    lint melt[MAXT], T;
    pi lazy[MAXT];
    int n;

    bool cmp(line &a, line &b) {
        lint l = a.eval(T), r = b.eval(T);
        if (l != r)
            return l > r;
        return a.A > b.A;
    }

    void pull(int p) {
        tree[p] = cmp(tree[2 * p], tree[2 * p + 1]) ?
tree[2 * p + 1] : tree[2 * p];
        melt[p] = min({melt[2 * p], melt[2 * p + 1],
tree[2 * p].cross_after(tree[2 * p + 1], 0)});
    }

    void init(int s, int e, int p, vector<line> &l) {
        if (s == e) {
            tree[p] = l[s];
            melt[p] = inf;
            lazy[p] = {0, 0};
            return;
        }
        lazy[p] = {0, 0};
        int m = (s + e) / 2;
        init(s, m, 2 * p, l);
        init(m + 1, e, 2 * p + 1, l);
        pull(p);
    }

    void lazydown(int p) {
        for (int i = 2 * p; i < 2 * p + 2; i++) {
            lazy[i][0] += lazy[p][0];
            lazy[i][1] += lazy[p][1];
            tree[i].B += lazy[p][0] * tree[i].A + lazy
[p][1];
            melt[i] -= lazy[p][0];
        }
        lazy[p][0] = lazy[p][1] = 0;
    }

    void propagate(int p) {
        if (melt[p] > 0)
            return;
        lazydown(p);
        propagate(2 * p);
        propagate(2 * p + 1);
        pull(p);
    }
};

```

```

}

lint query(int s, int e, int ps, int pe, int p =
1) {
    if (e < ps || pe < s)
        return inf;
    if (s <= ps && pe <= e)
        return tree[p].eval(0);
    int pm = (ps + pe) / 2;
    lazydown(p);
    return min(query(s, e, ps, pm, 2 * p), query(s
, e, pm + 1, pe, 2 * p + 1));
}

void heaten(int s, int e, int ps, int pe, int p,
lint v) {
    if (e < ps || pe < s)
        return;
    if (s <= ps && pe <= e) {
        lazy[p][0] += v;
        tree[p].B += v * tree[p].A;
        melt[p] -= v;
        propagate(p);
        return;
    }
    lazydown(p);
    int pm = (ps + pe) / 2;
    heaten(s, e, ps, pm, 2 * p, v);
    heaten(s, e, pm + 1, pe, 2 * p + 1, v);
    pull(p);
}

void add(int s, int e, int ps, int pe, int p, lint
v) {
    if (e < ps || pe < s)
        return;
    if (s <= ps && pe <= e) {
        lazy[p][1] += v;
        tree[p].B += v;
        return;
    }
    lazydown(p);
    int pm = (ps + pe) / 2;
    add(s, e, ps, pm, 2 * p, v);
    add(s, e, pm + 1, pe, 2 * p + 1, v);
    pull(p);
}

void init(vector<line> &l, lint _T) {
    n = l.size();
    T = _T;
    init(0, n - 1, 1, l);
}
};

```

## 6 Строковые алгоритмы

### 6.1 Префикс-функция

```

vector<int> prefix_function(string s) {
    vector<int> p(s.size());
    for (int i = 1; i < s.size(); ++i) {
        p[i] = p[i - 1];
        while (p[i] && s[p[i]] != s[i]) p[i] = p[p[i] -
1];
        p[i] += s[i] == s[p[i]];
    }
    return p;
}

```

### 6.2 Z-функция

```

vector<int> z_function (string s) { // z[i] - lcp of s
    and s[i:]
    int n = (int) s.length();
    vector<int> z (n);
    for (int i=1, l=0, r=0; i<n; ++i) {
        if (i <= r)
            z[i] = min (r-i+1, z[i-l]);
    }
}

```

```

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;
}
return z;
}

```

### 6.3 Алгоритм Манакера

```

vector<int> manacher_odd(const string &s) {
    vector<int> man(s.size(), 0);
    int l = 0, r = 0;
    int n = s.size();
    for (int i = 1; i < n; i++) {
        if (i <= r) {
            man[i] = min(r - i, man[l + r - i]);
        }
        while (i + man[i] + 1 < n && i - man[i] - 1 >= 0
            && s[i + man[i] + 1] == s[i - man[i] - 1]) {
            man[i]++;
        }
        if (i + man[i] > r) {
            l = i - man[i];
            r = i + man[i];
        }
    }
    return man;
}
// abacaba : (0 1 0 3 0 1 0)
// abbaa : (0 0 0 0 0)

```

```

vector<int> manacher_even(const string &s) {
    assert(s.size());
    string t;
    for (int i = 0; i + 1 < s.size(); ++i) {
        t += s[i];
        t += '#';
    }
    t += s.back();
    auto odd = manacher_odd(t);
    vector<int> ans;
    for (int i = 1; i < odd.size(); i += 2) {
        ans.push_back((odd[i]+1)/2);
    }
    return ans;
}
// abacaba : (0 0 0 0 0 0)
// abbaa : (0 2 0 1)

```

```

auto pal = [&] (int i, int from, int len) {
    if (len == 0) {
        return true;
    }
    int m = len/2;
    if (len & 1) {
        return o[i][from + m] >= m;
    }
    else {
        return e[i][from + m - 1] >= m;
    }
};

```

### 6.4 Суфмассив

Переработанный китайский суфмассив

```

const int inf = 1e9;
struct rmq {
    int n;
    vector<int> a;
    void build(const vector<int> &x) {
        assert(x.size() == n);
        for (int i = 0; i < n; ++i) a[n + i] = x[i];
        for (int i = n - 1; i > 0; --i) a[i] = min(a[2 * i], a[2 * i + 1]);
    }
    rmq(int n) : n(n), a(2 * n, inf) {}
    void put(int i, int x) {
        a[i + n] = min(a[i + n], x);
        for (i = (i + n) / 2; i > 0; i /= 2) {

```

```

            a[i] = min(a[i * 2], a[i * 2 + 1]);
        }
    }
    int getMin(int l, int r) { //[l;r)
        assert(l < r);
        int res = inf;
        for (l += n, r += n; l < r; l /= 2, r /= 2) {
            if (l & 1) res = min(res, a[l++]);
            if (r & 1) res = min(res, a[--r]);
        }
        return res;
    }
};
template <typename T>
vector<int> SA(const T &a) {
    int m = *max_element(all(a)) + 1, n = a.size();
    vector<int> sa(n), nsa(n), pre(max(n, m)), x(a.begin(), a.end()), y(n);
    for (int e : x) pre[e]++;
    for (int i = 1; i < m; ++i) pre[i] += pre[i - 1];
    for (int i = 0; i < n; ++i) sa[--pre[x[i]]] = i;
    int dif = 1;
    y[sa.front()] = 0;
    for (int i = 1; i < n; ++i) {
        dif += x[sa[i]] != x[sa[i-1]];
        y[sa[i]] = dif - 1;
    }
    x = y;
    for (int h = 1; dif < n; h *= 2) {
        fill(all(pre), 0);
        for (int e : x) pre[e]++;
        for (int i = 1; i < dif; ++i) pre[i] += pre[i - 1];
        for (int t = n; t--;) {
            int i = sa[t];
            if (i >= h) {
                nsa[--pre[x[i-h]]] = i-h;
            }
            else if (i + 1 != h) {
                nsa[--pre[x[i-h+n+1]]] = i-h+n+1;
            }
        }
        nsa[--pre[x[n-h]]] = n-h;
        sa = nsa;
        auto getr = [&] (int i) {
            if (i + h < n) {
                return x[i + h];
            }
            else {
                return x[i + h - n - 1];
            }
        };
        dif = 1;
        y[sa.front()] = 0;
        for (int i = 1; i < n; ++i) {
            if (x[sa[i]] != x[sa[i-1]] || sa[i-1]+h==n) {
                dif++;
            }
            else {
                dif += getr(sa[i]) != getr(sa[i-1]);
            }
            y[sa[i]] = dif-1;
        }
        x = y;
    }
    return sa;
}

```

```

template <typename T>
struct suar {
    vector<int> sa, lcp, pos; rmq t;
    suar(const T &a) : t((int)a.size() - 1) {
        sa = SA(a);
        int n = (int)a.size(), k = 0;
        lcp.resize(n - 1);
        pos.resize(n);
        for (int i = 0; i < n; ++i) pos[sa[i]] = i;
        for (int i = 0; i < n; ++i) {
            if (pos[i+1]<n) {

```

```

        int j = sa[pos[i]+1];
        while (i+k<n&&j+k<n&&a[i+k]==a[j+k])k
    ++;
        lcp[pos[i]]=k;
    }
    if (k) {
        k--;
    }
}
t.build(lcp);
}
int getLcp(int i, int j) {
    i = pos[i]; j = pos[j];
    if (j < i) {
        swap(i, j);
    }
    if (i == j) {
        return inf;
    }
    else {
        return t.getMin(i, j);
    }
}
};

```

## 6.5 Алгоритм Ахо — Корасик

## 6.6 Алгоритм Ахо Корасик

```

struct node{
    int next[alpha] = {}, link[alpha] = {};
    int suf = 0;
    ll visited = 0, ans = 0;
    vector<int> term;
    node() {}
};

vector<node> mem;

int get_next(int nd, char c) {
    if (!mem[nd].next[c - a]) { mem[nd].next[c - a] =
        mem.size(); mem.emplace_back(); }
    return mem[nd].next[c - a];
}

void find(string s, vector<string> t) {
    mem.reserve(1e6 + 100); mem.clear();
    mem.emplace_back(); mem.emplace_back();
    // 0th element is nullptr, 1st is the root
    int q = t.size();
    for (int j = 0; j < q; ++j) {
        int cur = 1;
        for (char c : ts[j]) cur = get_next(cur, c);
        mem[cur].term.push_back(j);
    }
    vector<int> bfs_order;
    queue<int> bfs;
    {
        node &root = mem[1];
        root.suf = 1;
        for (char c = a; c < a + alpha; ++c) {
            root.link[c - a] = (root.next[c - a] ? root.
                next[c - a] : 1);
        }
        bfs.push(1);
    }
    while (!bfs.empty()) {
        int cur_idx = bfs.front();
        bfs.pop();
        node &cur = mem[cur_idx];
        bfs_order.push_back(cur_idx);
        for (char c = a; c < a + alpha; ++c) {
            int nxt_idx = cur.next[c - a];
            if (!nxt_idx) continue;
            node &nxt = mem[nxt_idx];
            nxt.suf = (cur_idx == 1 ? 1 : mem[cur.suf].link[
                c - a]);
            for (char c = a; c < a + alpha; ++c) {
                nxt.link[c - a] = (nxt.next[c - a] ? nxt.next[
                    c - a] : mem[nxt.suf].link[c - a]);
            }
        }
    }
}

```

```

    }
    bfs.push(nxt_idx);
}
}
// do something
}

```

## 6.7 Дерево палиндромов

## 6.8 Дерево палиндромов

```

struct palindromic{
    int n;
    vector<int> p, suf{0, 0}, len{-1, 0};
    vector<array<int, alpha>> to{{}, {}};
    int sz = 2;

    palindromic(const string &s) : n(s.size()), p(n + 1,
        1) {
        suf.reserve(n);
        len.reserve(n);
        for (int i = 0; i < n; ++i) {
            auto check = [&](int l) { return i > l && s[i]
                == s[i - l - 1]; };
            int par = p[i];
            while (!check(len[par])) par = suf[par];
            if (to[par][s[i] - a]) {
                p[i + 1] = to[par][s[i] - a];
                continue;
            }
            p[i + 1] = sz++;
            to[par][s[i] - a] = p[i + 1];
            to.emplace_back();
            len.emplace_back(len[par] + 2);
            do {
                par = suf[par];
            } while (!check(len[par]));
            int link = to[par][s[i] - a];
            if (link == p[i + 1]) link = 1;
            suf.emplace_back(link);
        }
    }
};

```

## 7 Потоки

## 7.1 Алгоритм Диница

```

#define pb push_back
struct Dinic{
    struct edge{
        int to, flow, cap;
    };

    const static int N = 555; //count of vertices

    vector<edge> e;
    vector<int> g[N + 7];
    int dp[N + 7];
    int ptr[N + 7];

    void clear(){
        for (int i = 0; i < N + 7; i++) g[i].clear();
        e.clear();
    }

    void addEdge(int a, int b, int cap){
        g[a].pb(e.size());
        e.pb({b, 0, cap});
        g[b].pb(e.size());
        e.pb({a, 0, 0});
    }

    int minFlow, start, finish;

    bool bfs(){
        for (int i = 0; i < N; i++) dp[i] = -1;
        dp[start] = 0;
        vector<int> st;
        int uk = 0;
        st.pb(start);
    }
}

```



```

while(uk < st.size()){
    int v = st[uk++];
    for (int to : g[v]){
        auto ed = e[to];
        if (ed.cap - ed.flow >= minFlow && dp[ed.to] ==
-1){
            dp[ed.to] = dp[v] + 1;
            st.pb(ed.to);
        }
    }
}
return dp[finish] != -1;
}

int dfs(int v, int flow){
    if (v == finish) return flow;
    for (; ptr[v] < g[v].size(); ptr[v]++){
        int to = g[v][ptr[v]];
        edge ed = e[to];
        if (ed.cap - ed.flow >= minFlow && dp[ed.to] == dp
[v] + 1){
            int add = dfs(ed.to, min(flow, ed.cap - ed.flow)
);
            if (add){
                e[to].flow += add;
                e[to ^ 1].flow -= add;
                return add;
            }
        }
    }
    return 0;
}

int dinic(int start, int finish){
    Dinic::start = start;
    Dinic::finish = finish;
    int flow = 0;
    for (minFlow = (1 << 30); minFlow; minFlow >>= 1){
        while(bfs()){
            for (int i = 0; i < N; i++) ptr[i] = 0;
            while(int now = dfs(start, (int)2e9 + 7)) flow
+= now;
        }
    }
    return flow;
}
} dinic;

```

## 7.2 Mincost k-flow

```

struct edge {
    int next, capacity, cost, flow = 0;

    edge() = default;

    edge(int next, int capacity, int cost) : next(next),
        capacity(capacity), cost(cost) {}

    int rem() const { return capacity - flow; }

    int operator+=(int f) { return flow += f; }

    int operator-=(int f) { return flow -= f; }
};

auto addEdge = [&](auto from, auto next, auto capacity
, int cost) {
    g[from].push_back(e.size());
    e.emplace_back(next, capacity, cost);
    g[next].push_back(e.size());
    e.emplace_back(from, 0, -cost);
};

/* in case of undirected graph use this:
addEdge(u, v, capacity, cost);
addEdge(v, u, capacity, cost);
*/

vector<ll> phi(n, 0);
auto fordBellman = [&](int s, int t) {
    phi.assign(n, 0);
    for (int iter = 0; iter < n; ++iter) {
        bool changed = false;
        for (int u = 0; u < n; ++u) {

```

```

            for (auto index : g[u]) {
                auto edge = e[index];
                if (edge.rem() > 0 && phi[edge.next] > phi[u]
+ edge.cost) {
                    phi[edge.next] = phi[u] + edge.cost;
                    changed = true;
                }
            }
        }
        if (!changed) break;
    }
};

fordBellman(s, t);
// now shortest path using dijkstra with potentials
vector<ll> dist;
vector<int> from;
vector<bool> cnt;
auto dijkstra = [&](int s, int t) {
    dist.assign(n, 1e18);
    from.assign(n, -1);
    cnt.assign(n, false);
    dist[s] = 0;
    set<pair<int, int>> se;
    se.insert({0, s});
    while ((int)(se.size())) {
        int cur = se.begin()->y;
        se.erase(se.begin());
        cnt[cur] = true;
        for (int index : g[cur]) {
            auto &edge = e[index];
            if (edge.rem() == 0) continue;
            ll weight = edge.cost + phi[cur] - phi[edge.next
];
            if (dist[edge.next] > dist[cur] + weight) {
                se.erase({dist[edge.next], edge.next});
                dist[edge.next] = dist[cur] + weight;
                se.insert({dist[edge.next], edge.next});
                from[edge.next] = cur;
            }
        }
    }
    if (dist[t] == (ll) 1e18) return -1LL;
    ll cost = 0;
    for (int p = t; p != s; p = from[p]) {
        for (auto index : g[from[p]]) {
            auto &edge = e[index];
            ll weight = edge.cost + phi[from[p]] - phi[edge.
next];
            if (edge.rem() > 0 && edge.next == p && dist[
edge.next] == dist[from[p]] + weight) {
                edge += 1;
                e[index ^ 1] -= 1;
                cost += edge.cost;
                break;
            }
        }
    }
    for (int i = 0; i < n; ++i) {
        phi[i] += dist[i];
    }
    return cost;
};

ll cost = 0;
for (int flow = 0; flow < k; ++flow) {
    ll a = dijkstra(s, t);
    if (a == -1) {
        cout << "-1\n";
        return;
    }
    cost += a;
}

// now recover answer
auto findPath = [&](int s, int t) {
    vector<int> ans;
    int cur = s;
    while (cur != t) {
        for (auto index : g[cur]) {
            auto &edge = e[index];
            if (edge.flow <= 0) continue;
            edge -= 1;

```

```

    e[index ^ 1] += 1;
    ans.push_back(index / 4);
    // index / 4 because each edge has 4 copies
    cur = edge.next;
    break;
}
}
return ans;
};
for (int flow = 0; flow < k; ++flow) {
    auto p = findPath(s, t);
    cout << p.size() << ' ';
    for (int x : p) cout << x + 1 << ' ';
    cout << '\n';
}
}

```

## 8 Гамильтоновы путь и цикл

<https://codeforces.com/blog/entry/90513>,  
<https://codeforces.com/blog/entry/90743>.

### 8.1 Link-cut tree

```

namespace LCT {
    vector<vi> ch;
    vi fa, rev;
    void init(int n) {
        ch.resize(n + 1);
        fa.resize(n + 1);
        rev.resize(n + 1);
        for (int i = 0; i <= n; i++)
            ch[i].resize(2),
            ch[i][0] = ch[i][1] = fa[i] = rev[i] = 0;
    }
    bool isr(int a)
    {
        return !(ch[fa[a]][0] == a || ch[fa[a]][1] ==
a);
    }
    void pushdown(int a)
    {
        if (rev[a])
        {
            rev[ch[a][0]] ^= 1, rev[ch[a][1]] ^= 1;
            swap(ch[a][0], ch[a][1]);
            rev[a] = 0;
        }
    }
    void push(int a)
    {
        if (!isr(a)) push(fa[a]);
        pushdown(a);
    }
    void rotate(int a)
    {
        int f = fa[a], gf = fa[f];
        int tp = ch[f][1] == a;
        int son = ch[a][tp ^ 1];
        if (!isr(f))
            ch[gf][ch[gf][1] == f] = a;
        fa[a] = gf;

        ch[f][tp] = son;
        if (son) fa[son] = f;

        ch[a][tp ^ 1] = f, fa[f] = a;
    }
    void splay(int a)
    {
        push(a);
        while (!isr(a))
        {
            int f = fa[a], gf = fa[f];
            if (isr(f)) rotate(a);
            else
            {
                int t1 = ch[gf][1] == f, t2 = ch[f][1]
== a;
                if (t1 == t2) rotate(f), rotate(a);
                else rotate(a), rotate(a);
            }
        }
    }
}

```

```

    }
}
}
void access(int a)
{
    int pr = a;
    splay(a);
    ch[a][1] = 0;
    while (1)
    {
        if (!fa[a]) break;
        int u = fa[a];
        splay(u);
        ch[u][1] = a;
        a = u;
    }
    splay(pr);
}
void makeroot(int a)
{
    access(a);
    rev[a] ^= 1;
}
void link(int a, int b)
{
    makeroot(a);
    fa[a] = b;
}
void cut(int a, int b)
{
    makeroot(a);
    access(b);
    fa[a] = 0, ch[b][0] = 0;
}
int fdr(int a)
{
    access(a);
    while (1)
    {
        pushdown(a);
        if (ch[a][0]) a = ch[a][0];
        else {
            splay(a);
            return a;
        }
    }
}
}

```

### 8.2 Undirected case

```

#include <bits/stdc++.h>
using namespace std;
namespace hamil {
    template <typename T> bool chkmax(T &x, T y) {return
x < y ? x = y, true : false;}
    template <typename T> bool chkmin(T &x, T y) {return
x > y ? x = y, true : false;}
    #define vi vector<int>
    #define pb push_back
    #define mp make_pair
    #define pi pair<int, int>
    #define fi first
    #define se second
    #define ll long long
    using namespace LCT;
    vector<vi> used;
    unordered_set<int> caneg;
    void cut(int a, int b) {
        LCT::cut(a, b);
        for (int s = 0; s < 2; s++) {
            for (int i = 0; i < used[a].size(); i++)
                if (used[a][i] == b) {
                    used[a].erase(used[a].begin() + i);
                    break;
                }
            if (used[a].size() == 1) caneg.insert(a);
            swap(a, b);
        }
    }
    void link(int a, int b) {

```

```

LCT::link(a, b);
for (int s = 0; s < 2; s++) {
    used[a].pb(b);
    if (used[a].size() == 2) caneg.erase(a);
    swap(a, b);
}
}

vi work(int n, vector<pi> eg, ll mx_ch = -1) {
    // mx_ch : max number of adding/replacing
    default is (n + 100) * (n + 50)
    // n : number of vertices. 1-indexed.
    // eg: vector<pair<int, int> > storing all the
    edges.
    // return a vector<int> consists of all
    indices of vertices on the path. return empty list
    if failed to find one.

    LCT::init(n);
    if (mx_ch == -1) mx_ch = 1ll * (n + 100) * (n
+ 50); //default
    used.resize(n + 1);
    caneg.clear();
    for (int i = 1; i <= n; i++) used[i].clear();

    vector<vi> edges(n + 1);
    for (auto v : eg)
        edges[v.fi].pb(v.se),
        edges[v.se].pb(v.fi);

    for (int i = 1; i <= n; i++)
        caneg.insert(i);

    mt19937 x(chrono::steady_clock::now().
time_since_epoch().count());
    int tot = 0;
    while (mx_ch >= 0) {
        // cout << tot << ' ' << mx_ch << endl;
        vector<pi> eg;
        for (auto v : caneg)
            for (auto s : edges[v])
                eg.pb(mp(v, s));

        shuffle(eg.begin(), eg.end(), x);
        if (eg.size() == 0) break;
        for (auto v : eg) {
            mx_ch--;
            int a = v.fi, b = v.se;
            if (used[a].size() < used[b].size())
                swap(a, b);
            if (used[b].size() >= 2) continue;
            if (x() & 1) continue;
            if (LCT::fdr(a) == LCT::fdr(b))
                continue;
            if (used[a].size() < 2 && used[b].size
() < 2)
                tot++;
            if (used[a].size() == 2) {
                int p = used[a][x() % 2];
                cut(a, p);
            }
            link(a, b);
        }
        if (tot == n - 1) {
            vi cur;
            for (int i = 1; i <= n; i++)
                if (used[i].size() <= 1) {
                    int pl = i, ls = 0;
                    while (pl) {
                        cur.pb(pl);
                        int flag = 0;
                        for (auto v : used[pl])
                            if (v != ls) {
                                ls = pl;
                                pl = v;
                                flag = 1;
                                break;
                            }
                    }
                    if (!flag) break;
                }
            break;
        }
    }
}

```

```

    }
    return cur;
}
}
//failed to find a path
return vi();
}
}

```

### 8.3 Directed case

```

namespace hamil {
    template <typename T> bool chkmax(T &x, T y){return
    x<y?x=y,true:false;}
    template <typename T> bool chkmin(T &x, T y){return
    x>y?x=y,true:false;}
    #define vi vector<int>
    #define pb push_back
    #define mp make_pair
    #define pi pair<int, int>
    #define fi first
    #define se second
    #define ll long long
    using namespace LCT;
    vi out, in;
    vi work(int n, vector<pi> eg, ll mx_ch = -1) {
        // mx_ch : max number of adding/replacing
        default is (n + 100) * (n + 50)
        // n : number of vertices. 1-indexed.
        // eg: vector<pair<int, int> > storing all the
        edges.
        // return a vector<int> consists of all
        indices of vertices on the path. return empty list
        if failed to find one.
        out.resize(n + 1), in.resize(n + 1);
        LCT::init(n);
        for (int i = 0; i <= n; i++) in[i] = out[i] =
0;
        if (mx_ch == -1) mx_ch = 1ll * (n + 100) * (n
+ 50); //default
        vector<vi> from(n + 1), to(n + 1);
        for (auto v : eg)
            from[v.fi].pb(v.se),
            to[v.se].pb(v.fi);
        unordered_set<int> canin, canout;
        for (int i = 1; i <= n; i++)
            canin.insert(i),
            canout.insert(i);
        mt19937 x(chrono::steady_clock::now().
time_since_epoch().count());
        int tot = 0;
        while (mx_ch >= 0) {
            // cout << tot << ' ' << mx_ch << endl;
            vector<pi> eg;
            for (auto v : canout)
                for (auto s : to[v])
                    if (in[s] == 0) {
                        assert(canin.count(s));
                        continue;
                    }
                else eg.pb(mp(v, s));
            for (auto v : canin)
                for (auto s : to[v])
                    eg.pb(mp(s, v));
            shuffle(eg.begin(), eg.end(), x);
            if (eg.size() == 0) break;
            for (auto v : eg) {
                mx_ch--;
                if (in[v.se] && out[v.fi]) continue;
                if (LCT::fdr(v.fi) == LCT::fdr(v.se))
                    continue;
                if (in[v.se] || out[v.fi])
                    if (x() & 1) continue;
                if (!in[v.se] && !out[v.fi])
                    tot++;
                if (in[v.se]) {
                    LCT::cut(in[v.se], v.se);
                    canin.insert(v.se);
                    canout.insert(in[v.se]);
                    out[in[v.se]] = 0;
                    in[v.se] = 0;
                }
            }
        }
    }
}

```

```

    }
    if (out[v.fi]) {
        LCT::cut(v.fi, out[v.fi]);
        canin.insert(out[v.fi]);
        canout.insert(v.fi);
        in[out[v.fi]] = 0;
        out[v.fi] = 0;
    }
    LCT::link(v.fi, v.se);
    canin.erase(v.se);
    canout.erase(v.fi);
    in[v.se] = v.fi;
    out[v.fi] = v.se;
}
if (tot == n - 1) {
    vi cur;
    for (int i = 1; i <= n; i++)
        if (!in[i]) {
            int pl = i;
            while (pl) {
                cur.pb(pl);
                pl = out[pl];
            }
            break;
        }
    return cur;
}
}
//failed to find a path
return vi();
}
}

```

## 9 Геометрия

### 9.1 Примитивы

```

struct Point {
    int x, y;
    Point(){}
    Point(int x_, int y_) {
        x = x_; y = y_;
    }
    Point operator + (Point p) {
        return Point(x+p.x,y+p.y);
    }
    Point operator - (Point p) {
        return Point(x - p.x, y - p.y);
    }
    int operator * (Point p) {
        return x * p.y - y * p.x;
    }
    int operator % (Point p) {
        return x * p.x + y * p.y;
    }
    bool operator < (Point v) {
        return (*this) * v > 0;
    }
    bool operator > (Point v) {
        return v < (*this);
    };
    bool operator <= (Point v) {
        return (*this) * v >= 0;
    }
};
bool line(Point a, Point b, Point c) {
    return (b-a)*(c-b)==0;
}
bool ord(Point a, Point p, Point b) {
    return (p - a)*(p - b)<0;
}

int hp(Point a) {
    if (a.y == 0) return a.x >= 0;
    return a.y > 0;
}

bool comp(Point a, Point b) {
    if (hp(a) != hp(b)) return hp(a) < hp(b);
    return a.x * b.y - a.y * b.x > 0;
}

```

### 9.2 Выпуклая оболочка

```

using pt = pair<int, int>;
#define x first
#define y second

int cross(pt p, pt q) {
    return p.x * q.y - p.y * q.x;
}
int scalar(pt p, pt q) {
    return p.x * q.x + p.y * q.y;
}
pt operator-(pt a, pt b) { return {a.x - b.x, a.y - b.y}; }
vector<pt> convex(vector<pt> a) {
    sort(all(a));
    if (a.size() == 2 && a[0] == a[1]) return {a[0]};
    if (a.size() <= 1) return a;
    vector<pt> h;
    for (int t = 0; t < 2; ++t) {
        int sz = h.size() - t;
        for (auto p: a) {
            while (h.size() >= sz + 2 && cross(p - h.end()[-1], h.end()[-2] - h.end()[-1]) <= 0) h.pop_back();
            h.push_back(p);
        }
        reverse(all(a));
    }
    return h; // h is circular: h.front() == h.back()
}

```

### 9.3 Точка внутри многоугольника

```

auto inT = [&] (Point a, Point b, Point c, Point p) {
    a = a-p; b = b-p; c = c-p;
    return abs(a*b)+abs(b*c)+abs(c*a) == abs(a*b+b*c+c*a);
};
auto inP = [&] (Point p) { //a must be in counterclockwise order!
    int l = 1, r = n - 1;
    while (l < r - 1) {
        int m = (l + r) / 2;
        if ((a[m] - a[0]) < (p - a[0])) {
            l = m;
        }
        else {
            r = m;
        }
    }
    return inT(a[l], a[0], a[r], p);
};

```

### 9.4 Касательные

```

auto max = [&] (auto cmp) {
    int k = 0;
    for (int lg = 18; lg >= 0; --lg) {
        int i = k + (1 << lg), j = k - (1 << lg);
        i = (i % n + n) % n;
        j = (j % n + n) % n;
        array<int, 3> ind{i, j, k};
        sort(all(ind), cmp);
        k = ind[2];
    }
    return k;
};
auto uppert = [&] (Point p) { //last vertex in counterclockwise order about p
    auto cmp = [&] (int i, int j) {return (a[i] - p) < (a[j] - p); };
    return max(cmp);
};
auto lower = [&] (Point p) { //first vertex in counterclockwise order about p
    auto cmp = [&] (int i, int j) {return (a[i] - p) > (a[j] - p); };
}

```

```

        return max(cmp);
    };
    auto uppertinf = [&](Point p) { //upper tangent
        line parallel to vector p
        swap(p.x, p.y);
        p.x = -p.x;
        auto cmp = [&] (int i, int j) { return a[i] %
p < a[j] % p; };
        return max(cmp);
    };
    auto lowertinf = [&](Point p) { //lower tangent
        line parallel to vector p
        swap(p.x, p.y);
        p.x = -p.x;
        auto cmp = [&] (int i, int j) { return a[i] %
p > a[j] % p; };
        return max(cmp);
    };

```

## 10 Цепные дроби

<https://cp-algorithms.com/algebra/continued-fractions.html>

### 10.1 Поиск нижней огибающей, сумма и минимум по модулю

```

int floor(int a, int b) {
    return a / b - ((a ^ b) < 0 && a % b);
}
vector<int> decompose(int p, int q) {
    vector<int> f;
    while (q != 0) {
        f.push_back(floor(p, q));
        p -= q * f.back();
        swap(p, q);
    }
    return f;
}

using matrix = array<int, 4>;

matrix operator*(matrix a, matrix b) {
    matrix c{0,0,0,0};
    for (int i = 0; i < 2; ++i) {
        for (int j = 0; j < 2; ++j) {
            for (int k = 0; k < 2; ++k) {
                c[2 * i + k] += a[2 * i + j] * b[2 * j + k];
            }
        }
    }
    return c;
}

#define x first
#define y second

// computes lower convex hull for 0 <= x <= N, 0 <= y
// <= (ax + b) / c
vector<pair<int, int>> lower_convex_hull(int a, int b,
    int c, int n) {
    matrix m = {1, 0, 0, 1};
    auto f = decompose(a, c);
    vector<pair<int, int>> conv{{1, 0}, {0, 1}};
    for (int x : f) {
        m = m * matrix{x, 1, 1, 0};
        conv.emplace_back(m[2], m[0]);
        if (m[2] > n) break; // there should be one (if
any) with .x > n
    }
    auto diff = [&](int x, int y) {
        return c * y - a * x;
    };
    int x = 0, y = b / c;
    vector<pair<int, int>> res{{x, y}};
    int i;
    for (i = 2; i + 1 < conv.size(); i += 2) {
        while (diff(x + conv[i + 1].x, y + conv[i + 1].y)
<= b) {
            int t = 1 + (diff(x + conv[i - 1].x, y + conv[i
- 1].y) - b - 1) / abs(diff(conv[i].x, conv[i].y))

```

```

;
        auto [dx, dy] = tuple{conv[i - 1].x + t * conv[i
].x, conv[i - 1].y + t * conv[i].y};
        int k = (n - x) / dx;
        if (k == 0) break;
        if (diff(dx, dy)) k = min(k, (b - diff(x, y)) /
diff(dx, dy));
        x += k * dx, y += k * dy;
        res.push_back({x, y});
    }
}
if (i >= conv.size()) i -= 2;
for (; i > 0; i -= 2) {
    auto [dx1, dy1] = conv[i];
    if (x + dx1 > n) continue;
    x += dx1, y += dy1;
    if (i + 1 < conv.size()) {
        auto [dx2, dy2] = conv[i + 1];
        int k = (n - x) / dx2;
        x += k * dx2;
        y += k * dy2;
    }
    res.emplace_back(x, y);
    int k = (n - x) / dx1;
    if (k == 0) continue;
    x += k * dx1;
    y += k * dy1;
    res.emplace_back(x, y);
}
return res;
}

// number of (x, y) under pq line such that p.x <= x <
q.x && 0 < y
int area(auto p, auto q) {
    int integers = gcd(q.x - p.x, q.y - p.y);
    return ((p.y + q.y - 1) * (q.x - p.x + 1) + integers
+ 1) / 2 - q.y;
}

// sum of (ax + b) / c for 0 <= x < n
int get_area(int a, int b, int c, int n) { // SUM (ax
+ b) / c for 0 <= x <= n
    auto ch = lower_convex_hull(a, b, c, n + 1);
    int sum = 0;
    for (int i = 0; i + 1 < ch.size(); ++i) {
        sum += area(ch[i], ch[i + 1]);
    }
    return sum;
}

// min of (ax + b) % c for 0 <= x <= n
int get_min(int a, int b, int c, int n) {
    auto ch = lower_convex_hull(a, b, c, n);
    // in fact, here we need only the last point of the
first half of the algo (that is going up)
    int mn = c;
    for (auto [x, y] : ch) mn = min(mn, (a * x + b) % c);
    return mn;
}

```

### 10.2 Простая рекурсия

Число точек  $(x, y) : 0 \leq x < n, 0 < y \leq (kx + b)/d$ . То есть  $\sum_{x=0}^{n-1} \lfloor \frac{kx+b}{d} \rfloor$ .

```

int cnt (int n, int k, int b, int d)
{
    if (k == 0) return (b / d) * n;
    if (k >= d || b >= d) {
        return (k / d) * n * (n - 1) / 2 + (b / d) * n
+ cnt(n, k % d, b % d, d);
    }
    return cnt((k * n + b) / d, d, (k * n + b) % d, k)
;
}

```

## 11 Разное

### 11.1 Компараторы

```
bool cmp1(int x, int y) { return x > y; }

struct cmp2{
    bool operator()(int x, int y) const { return x > y; }
};

int32_t main() {
    set<int, decltype(cmp1)*> s1({1, 2, 3}, cmp1);
    for (int x : s1) cout << x << ' '; cout << '\n';
    set<int, cmp2> s2({4, 5, 6});
    for (int x : s2) cout << x << ' '; cout << '\n';
    auto cmp3 = [&](int x, int y) { return x > y; };
    set<int, decltype(cmp3)> s3({7, 8, 9}, cmp3); //
    second cmp3 could be omitted if cmp3 = [](...) {
    ... }
    for (int x : s3) cout << x << ' '; cout << '\n';

    vector<int> v{3, 2, 1};
    cout << lower_bound(all(v), 2, cmp1) - v.begin()
    << '\n';
    cout << lower_bound(all(v), 2, cmp2()) - v.begin()
    << '\n';
    cout << lower_bound(all(v), 2, cmp3) - v.begin()
    << '\n';
}
```

### 11.2 Трюки от Сергея Копелиовича

#### 11.2.1 Быстрый ввод

[https://acm.math.spbu.ru/~sk1/algo/input-output/fread\\_write.cpp.html](https://acm.math.spbu.ru/~sk1/algo/input-output/fread_write.cpp.html)

```
const int buf_size = 4096;

int getChar() {
    static char buf[buf_size];
    static int len = 0, pos = 0;
    if (pos == len)
        pos = 0, len = fread(buf, 1, buf_size, stdin);
    if (pos == len)
        return -1;
    return buf[pos++];
}

int readChar() {
    while (1) {
        int c = getChar();
        if (c > 32) return c;
    }
}

int readInt() {
    int s = 1, c = readChar(), x = 0;
    if (c == '-')
        s = -1, c = getChar();
    while (isdigit(c))
        x = x * 10 + c - '0', c = getChar();
    return s * x;
}
```

<https://acm.math.spbu.ru/~sk1/algo/memory.cpp.html>

#### 11.2.2 Быстрый аллокатор

```
const int MAX_MEM = 1e8;
int mpos = 0;
char mem[MAX_MEM];
inline void * operator new (size_t n) {
    assert((mpos += n) <= MAX_MEM);
    return (void *) (mem + mpos - n);
}
void operator delete (void *) noexcept { } // must
have!
void operator delete (void *, size_t) noexcept { } //
must have!
```

### 11.3 Флаги компиляции

```
-DLOCAL -Wall -Wextra -pedantic -Wshadow -Wformat=2
-Wfloat-equal -Wconversion -Wlogical-op -Wshift-
overflow=2 -Wduplicated-cond -Wcast-qual -Wcast-
align -D_GLIBCXX_DEBUG -D_GLIBCXX_DEBUG_PEDANTIC
-D_FORTIFY_SOURCE=2 -fsanitize=address -
fsanitize=undefined -fno-sanitize-recover -fstack-
protector -std=c++2a
```

#### 11.3.1 Сеточка в vim

<https://codeforces.com/blog/entry/122540>

```
i|<esc>25A |<esc>
o+<esc>25A---+<esc>
Vky35Pdd
```

### 11.4 Что сделать на пробном туре

- Убедиться, что работают все IDE. Разобраться, как настраивать в них LOCAL.
- В системе ML — это ML или RE?
- Максимальный размер файла
- Можно посмотреть на время работы серверов позапусков Флойда — Варшалла
- Посмотреть, насколько быстр быстрый ввод