

Contents

1 Basic	1
1.1 vimrc	1
1.2 Black Magic	1
1.3 readchar	1
1.4 Pragma Optimization	1
1.5 readchar	1
1.6 Shell script	1
1.7 Default code	1
2 Graph	2
2.1 BCC Vertex*	2
2.2 Bridge*	2
2.3 SCC*	2
2.4 2SAT*	2
2.5 Virtual Tree*	2
2.6 Dominator Tree*	2
3 Data Structure	3
3.1 Discrete Trick	3
3.2 BIT kth*	3
3.3 DSU	3
3.4 Interval Container*	3
3.5 Heavy light Decomposition*	3
3.6 Centroid Decomposition*	4
3.7 LiChaoST*	4
3.8 Link cut tree*	4
3.9 Treap	5
3.10 Sparse table	5
4 Flow/Matching	5
4.1 Bipartite Matching*	5
4.2 Dinic	6
4.3 Kuhn Munkres*	6
5 String	7
5.1 KMP	7
5.2 Z-value*	7
5.3 Manacher*	7
5.4 Suffix Array	7
5.5 De Bruijn sequence*	7
5.6 Main Lorentz	7

6 Math	8
6.1 ax+by=gcd(only exgcd*)	8
6.2 Floor and Ceil	8
6.3 Floor Enumeration	8
6.4 Gaussian integer gcd	8
6.4.1 Construction	8
6.5 chineseRemainder	8
6.6 Primes	8
6.7 Estimation	8
6.8 Euclidean Algorithms	8
6.9 General Purpose Numbers	9
6.10 Tips for Generating Functions	9
7 Polynomial	9
7.1 Fast Fourier Transform	9
7.2 Number Theory Transform*	9
7.3 Newton's Method	9
8 Geometry	9
8.1 Default Code	9
8.2 PointSegDist*	10
8.3 Convex hull*	10
8.4 PointInConvex*	10
8.5 Intersection of line and convex	10
8.6 VectorInPoly*	10
8.7 PolyUnion*	10
8.8 Polar Angle Sort*	10
8.9 Half plane intersection*	11
8.10 RotatingSweepLine	11
8.11 Minkowski Sum*	11
9 Else	11
9.1 Cyclic Ternary Search*	11
9.2 Mo's Algorithm(With modification)	11
9.3 Mo's Algorithm On Tree	11
9.4 Additional Mo's Algorithm Trick	12
9.5 All LCS*	12
9.6 Tree Hash*	12
9.7 Min Plus Convolution*	12
9.8 Bitset LCS	12
10 Python	12
10.1 Misc	12

1 Basic

1.1 vimrc

```
"This file should be placed at ~/.vimrc"
se nu ai hls et ru ic is sc cul
se re=1 ts=4 sts=4 sw=4 ls=2 mouse=a
syntax on
hi cursorline cterm=none ctermbg=89
set bg=dark
inoremap {<CR> {<CR>}<Esc>ko<tab>
"Select
    region and then type :Hash to hash your selection."
"Useful for verifying that there aren't mistypes."
ca Hash w !cpp -dD -P -fpreprocessed
    \ | tr -d '[:space:]' \ | md5sum \ | cut -c-6
```

1.2 Black Magic [d41d8c]

```
/*先編譯成執行檔 good, bad, 然後寫好生測資用的東西
(用 python 或 c++ 都可) 後, 把這個檔案存成 run.sh
#!/usr/bin/env bash
i=0
while true
do
    echo $i
    ((++i))
    ./gen > in
    ./good < in > out1
    ./bad < in > out2
    diff out1 out2 || break
done
之後再 terminal 打 chmod +x run.sh 後 ./run.sh 就好了*/

/*把執行視窗改成 terminal
去 Settings
>Environment>"Terminal to launch console programs"
把 xterm
-T $TITLE -e 改成gnome-terminal --title=$TITLE -x
自訂編譯參數
去 Settings
>Compiler>Compiler Settings>Other Compiler Options
```

和 Settings
>Compiler>Linker Settings>Other Linker Options
都加上
-Wall -Wextra -Wshadow -Wconversion
-fsanitize=address,undefined */

1.3 readchar [8c6b69]

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define F first
#define S second
#define all(x) x.begin(),x.end()
#define pii pair<int,int>
#define pb push_back
#define sz(x) (int)(x.size())
#define chmin(x,y) x=min(x,y)
#define chmax(x,y) x=max(x,y)
#define vi vector<int>
#define vp vector<pii>
#define vvi vector<vi>
#define ykh mt19937_64 rng(time(NULL))
#define _ ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
//0-base
signed main(){
    _;
}
/*
input:
*/
```

1.4 Pragma Optimization [6006f6]

```
#pragma GCC optimize("Ofast,no-stack-protector")
#pragma GCC optimize("no-math-errno,unroll-loops")
#pragma GCC target("sse,sse2,sse3,ssse3,sse4")
#pragma GCC target("popcnt,abm,mmx,avx,arch=skylake")
__builtin_ia32_ldmxcsr(__builtin_ia32_stmxcsr()|0x8040)
```

1.5 readchar [a419b9]

```
inline char readchar() {
    static const size_t bufsize = 65536;
    static char buf[bufsize];
    static char *p = buf, *end = buf;
    if (p == end) end = buf +
        fread_unlocked(buf, 1, bufsize, stdin), p = buf;
    return *p++;
}
```

1.6 Shell script [3b2450]

```
g++ -O2 -std=c++17 -Dbbq -Wall -Wextra -Wshadow -o $1 $1.cpp
chmod +x compile.sh
g++ -o good a.cpp
```

1.7 Default code [8c6b69]

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define F first
#define S second
#define all(x) x.begin(),x.end()
#define pii pair<int,int>
#define pb push_back
#define sz(x) (int)(x.size())
#define chmin(x,y) x=min(x,y)
#define chmax(x,y) x=max(x,y)
#define vi vector<int>
#define vp vector<pii>
#define vvi vector<vi>
#define ykh mt19937_64 rng(time(NULL))
#define _ ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
//0-base
signed main(){
    _;
}
/*
input:
*/
```

2 Graph

2.1 BCC Vertex* [740acb]

```
struct BCC { // 0-base
    int n, dft, nbcc;
    vector<int> low, dfn, bln, stk, is_ap, cir;
    vector<vector<int>> G, bcc, nG;
    void make_bcc(int u) {
        bcc.emplace_back(1, u);
        for (; stk.back() != u; stk.pop_back())
            bln[stk.back()] = nbcc, bcc[nbcc].pb(stk.back());
        stk.pop_back(), bln[u] = nbcc++;
    }
    void dfs(int u, int f) {
        int child = 0;
        low[u] = dfn[u] = ++dft, stk.pb(u);
        for (int v : G[u])
            if (!dfn[v]) {
                dfs(v, u), ++child;
                low[u] = min(low[u], low[v]);
                if (dfn[u] <= low[v]) {
                    is_ap[u] = 1, bln[u] = nbcc;
                    make_bcc(v), bcc.back().pb(u);
                }
            } else if (dfn[v] < dfn[u] && v != f)
                low[u] = min(low[u], dfn[v]);
        if (f == -1 && child < 2) is_ap[u] = 0;
        if (f == -1 && child == 0) make_bcc(u);
    }
    BCC(int _n): n(_n), dft(),
                nbcc(), low(n), dfn(n), bln(n), is_ap(n), G(n) {}
    void add_edge(int u, int v) {
        G[u].pb(v), G[v].pb(u);
    }
    void solve() {
        for (int i = 0; i < n; ++i)
            if (!dfn[i]) dfs(i, -1);
    }
    void block_cut_tree() {
        cir.resize(nbcc);
        for (int i = 0; i < n; ++i)
            if (is_ap[i])
                bln[i] = nbcc++;
        cir.resize(nbcc, 1), nG.resize(nbcc);
        for (int i = 0; i < nbcc && !cir[i]; ++i)
            for (int j : bcc[i])
                if (is_ap[j])
                    nG[i].pb(bln[j]), nG[bln[j]].pb(i);
    } // up to 2 * n - 2 nodes!! bln[i] for id
};
```

2.2 Bridge* [4da29a]

```
struct ECC { // 0-base
    int n, dft, ecnt, necc;
    vector<int> low, dfn, bln, is_bridge, stk;
    vector<vector<pii>> G;
    void dfs(int u, int f) {
        dfn[u] = low[u] = ++dft, stk.pb(u);
        for (auto [v, e] : G[u])
            if (!dfn[v])
                dfs(v, e), low[u] = min(low[u], low[v]);
            else if (e != f)
                low[u] = min(low[u], dfn[v]);
        if (low[u] == dfn[u]) {
            if (f != -1) is_bridge[f] = 1;
            for (; stk.back() != u; stk.pop_back())
                bln[stk.back()] = necc;
            bln[u] = necc++, stk.pop_back();
        }
    }
    ECC(int _n): n(_n), dft(),
                ecnt(), necc(), low(n), dfn(n), bln(n), G(n) {}
    void add_edge(int u, int v) {
        G[u].pb(pii(v, ecnt)), G[v].pb(pii(u, ecnt++));
    }
    void solve() {
        is_bridge.resize(ecnt);
        for (int i = 0; i < n; ++i)
            if (!dfn[i]) dfs(i, -1);
    }
}; // ecc_id(i): bln[i]
```

2.3 SCC* [4057dc]

```
struct SCC { // 0-base
```

```
int n, dft, nscc;
vector<int> low, dfn, bln, instack, stk;
vector<vector<int>> G;
void dfs(int u) {
    low[u] = dfn[u] = ++dft;
    instack[u] = 1, stk.pb(u);
    for (int v : G[u])
        if (!dfn[v])
            dfs(v), low[u] = min(low[u], low[v]);
        else if (instack[v] && dfn[v] < dfn[u])
            low[u] = min(low[u], dfn[v]);
    if (low[u] == dfn[u]) {
        for (; stk.back() != u; stk.pop_back())
            bln[stk.back()] = nscc, instack[stk.back()] = 0;
        instack[u] = 0, bln[u] = nscc++, stk.pop_back();
    }
}
SCC(int _n): n(_n), dft(), nscc(),
             low(n), dfn(n), bln(n), instack(n), G(n) {}
void add_edge(int u, int v) {
    G[u].pb(v);
}
void solve() {
    for (int i = 0; i < n; ++i)
        if (!dfn[i]) dfs(i);
}; // scc_id(i): bln[i]
```

2.4 2SAT* [f5630a]

```
struct SAT { // 0-base
    int n;
    vector<bool> istrue;
    SCC scc;
    SAT(int _n): n(_n), istrue(n + n), scc(n + n) {}
    int rv(int a) {
        return a >= n ? a - n : a + n;
    }
    void add_clause(int a, int b) {
        scc.add_edge(rv(a), b), scc.add_edge(rv(b), a);
    }
    bool solve() {
        scc.solve();
        for (int i = 0; i < n; ++i) {
            if (scc.bln[i] == scc.bln[i + n]) return false;
            istrue[i] = scc.bln[i] < scc.bln[i + n];
            istrue[i + n] = !istrue[i];
        }
        return true;
    }
};
```

2.5 Virtual Tree* [1b641b]

```
vector<int> vG[N];
int top, st[N];

void insert(int u) {
    if (top == -1) return st[++top] = u, void();
    int p = LCA(st[top], u);
    if (p == st[top]) return st[++top] = u, void();
    while (top >= 1 && dep[st[top - 1]] >= dep[p])
        vG[st[top - 1]].pb(st[top]), --top;
    if (st[top] != p)
        vG[p].pb(st[top]), --top, st[++top] = p;
    st[++top] = u;
}

void reset(int u) {
    for (int i : vG[u]) reset(i);
    vG[u].clear();
}

void solve(vector<int> &v) {
    top = -1;
    sort(ALL(v), [&](int a, int b) { return dfn[a] < dfn[b]; });
    for (int i : v) insert(i);
    while (top > 0) vG[st[top - 1]].pb(st[top]), --top;
    // do something
    reset(v[0]);
}
```

2.6 Dominator Tree* [2b8b32]

```
struct dominator_tree { // 1-base
```

```

vector<int> G[N], rG[N];
int n, pa[N], dfn[N], id[N], Time;
int semi[N], idom[N], best[N];
vector<int> tree[N]; // dominator_tree
void init(int _n) {
    n = _n;
    for (int i = 1; i <= n; ++i)
        G[i].clear(), rG[i].clear();
}
void add_edge(int u, int v) {
    G[u].pb(v), rG[v].pb(u);
}
void dfs(int u) {
    id[dfn[u] = ++Time] = u;
    for (auto v : G[u])
        if (!dfn[v]) dfs(v), pa[dfn[v]] = dfn[u];
}
int find(int y, int x) {
    if (y <= x) return y;
    int tmp = find(pa[y], x);
    if (semi[best[y]] > semi[best[pa[y]]])
        best[y] = best[pa[y]];
    return pa[y] = tmp;
}
void tarjan(int root) {
    Time = 0;
    for (int i = 1; i <= n; ++i) {
        dfn[i] = idom[i] = 0;
        tree[i].clear();
        best[i] = semi[i] = i;
    }
    dfs(root);
    for (int i = Time; i > 1; --i) {
        int u = id[i];
        for (auto v : rG[u])
            if (v == dfn[v]) {
                find(v, i);
                semi[i] = min(semi[i], semi[best[v]]);
            }
        tree[semi[i]].pb(i);
        for (auto v : tree[pa[i]]) {
            find(v, pa[i]);
            idom[v] =
                semi[best[v]] == pa[i] ? pa[i] : best[v];
        }
        tree[pa[i]].clear();
    }
    for (int i = 2; i <= Time; ++i) {
        if (idom[i] != semi[i]) idom[i] = idom[idom[i]];
        tree[id[idom[i]]].pb(id[i]);
    }
}
};

```

3 Data Structure

3.1 Discrete Trick

```

vector<int> val;
// build
sort(ALL
    (val)), val.resize(unique(ALL(val)) - val.begin());
// index of x
upper_bound(ALL(val), x) - val.begin();
// max idx <= x
upper_bound(ALL(val), x) - val.begin();
// max idx < x
lower_bound(ALL(val), x) - val.begin();

```

3.2 BIT kth* [e39485]

```

int bit[N + 1]; // N = 2 ^ k
int query_kth(int k) {
    int res = 0;
    for (int i = N >> 1; i >= 1; i >>= 1)
        if (bit[res + i] < k)
            k -= bit[res + i];
    return res + 1;
}

```

3.3 DSU [b248db]

```

struct DSU {
    vector<int> to, num;
    int cnt;
    DSU(int n = 0): to(n), num(n) {
        cnt = n;
    }
};

```

```

for (int i = 0; i < n; i++) {
    to[i] = i;
    num[i] = 1;
}
int find(int x) {
    return x == to[x] ? x : to[x] = find(to[x]);
}
bool un(int x, int y) {
    x = find(x), y = find(y);
    if (x == y) return 0;
    cnt--;
    if (num[x] > num[y]) swap(x, y);
    to[x] = y;
    num[y] += num[x];
    return 1;
}
};

```

3.4 Interval Container* [c54d29]

```

/* Add and
   remove intervals from a set of disjoint intervals.
   * Will merge the added interval with
   any overlapping intervals in the set when adding.
   * Intervals are [inclusive, exclusive). */
set<pii>::
    iterator addInterval(set<pii>& is, int L, int R) {
        if (L == R) return is.end();
        auto it = is.lower_bound({L, R}), before = it;
        while (it != is.end() && it->X <= R) {
            R = max(R, it->Y);
            before = it = is.erase(it);
        }
        if (it != is.begin() && (--it)->Y >= L) {
            L = min(L, it->X);
            R = max(R, it->Y);
            is.erase(it);
        }
        return is.insert(before, pii(L, R));
    }
void removeInterval(set<pii>& is, int L, int R) {
    if (L == R) return;
    auto it = addInterval(is, L, R);
    auto r2 = it->Y;
    if (it->X == L) is.erase(it);
    else (int&)it->Y = L;
    if (R != r2) is.emplace(R, r2);
}

```

3.5 Heavy light Decomposition* [b004ae]

```

struct Heavy_light_Decomposition { // 1-base
    int n, ulink[N], deep[N], mxson[N], w[N], pa[N];
    int t, pl[N], data[N], val[N]; // val: vertex data
    vector<int> G[N];
    void init(int _n) {
        n = _n;
        for (int i = 1; i <= n; ++i)
            G[i].clear(), mxson[i] = 0;
    }
    void add_edge(int a, int b) {
        G[a].pb(b), G[b].pb(a);
    }
    void dfs(int u, int f, int d) {
        w[u] = 1, pa[u] = f, deep[u] = d++;
        for (int &i : G[u])
            if (i != f) {
                dfs(i, u, d), w[u] += w[i];
                if (w[mxson[u]] < w[i]) mxson[u] = i;
            }
    }
    void cut(int u, int link) {
        data[pl[u] = ++t] = val[u], ulink[u] = link;
        if (!mxson[u]) return;
        cut(mxson[u], link);
        for (int i : G[u])
            if (i != pa[u] && i != mxson[u])
                cut(i, i);
    }
    void build() { dfs(1, 1, 1), cut(1, 1), /*build*/; }
    int query(int a, int b) {
        int ta = ulink[a], tb = ulink[b], res = 0;
        while (ta != tb) {
            if (deep
                [ta] > deep[tb]) swap(ta, tb), swap(a, b);
            // query(pl[tb], pl[b])
        }
    }
};

```

```

    tb = ulink[b = pa[tb]];
}
if (pl[a] > pl[b]) swap(a, b);
// query(pl[a], pl[b])
}
};

```

3.6 Centroid Decomposition* [5a24da]

```

struct Cent_Dec { // 1-base
    vector<pll> G[N];
    pll info[N]; // store info. of itself
    pll upinfo[N]; // store info. of climbing up
    int n, pa[N], layer[N], sz[N], done[N];
    ll dis[__lg(N) + 1][N];
    void init(int _n) {
        n = _n, layer[0] = -1;
        fill_n(pa + 1, n, 0), fill_n(done + 1, n, 0);
        for (int i = 1; i <= n; ++i) G[i].clear();
    }
    void add_edge(int a, int b, int w) {
        G[a].pb(pll(b, w)), G[b].pb(pll(a, w));
    }
    void get_cent(
        int u, int f, int &mx, int &c, int num) {
        int mxsz = 0;
        sz[u] = 1;
        for (pll e : G[u])
            if (!done[e.X] && e.X != f) {
                get_cent(e.X, u, mx, c, num);
                sz[u] += sz[e.X], mxsz = max(mxsz, sz[e.X]);
            }
        if (mx > max(mxsz, num - sz[u]))
            mx = max(mxsz, num - sz[u]), c = u;
    }
    void dfs(int u, int f, ll d, int org) {
        // if required, add self info or climbing info
        dis[layer[org]][u] = d;
        for (pll e : G[u])
            if (!done[e.X] && e.X != f)
                dfs(e.X, u, d + e.Y, org);
    }
    int cut(int u, int f, int num) {
        int mx = 1e9, c = 0, lc;
        get_cent(u, f, mx, c, num);
        done[c] = 1, pa[c] = f, layer[c] = layer[f] + 1;
        for (pll e : G[c])
            if (!done[e.X]) {
                if (sz[e.X] > sz[c])
                    lc = cut(e.X, c, num - sz[c]);
                else lc = cut(e.X, c, sz[e.X]);
                upinfo[lc] = pll(), dfs(e.X, c, e.Y, c);
            }
        return done[c] = 0, c;
    }
    void build() { cut(1, 0, n); }
    void modify(int u) {
        for (int a = u, ly = layer[a]; a;
             a = pa[a], --ly) {
            info[a].X += dis[ly][u], ++info[a].Y;
            if (pa[a])
                upinfo[a].X += dis[ly - 1][u], ++upinfo[a].Y;
        }
    }
    ll query(int u) {
        ll rt = 0;
        for (int a = u, ly = layer[a]; a;
             a = pa[a], --ly) {
            rt += info[a].X + info[a].Y * dis[ly][u];
            if (pa[a])
                rt -=
                    upinfo[a].X + upinfo[a].Y * dis[ly - 1][u];
        }
        return rt;
    }
};

```

3.7 LiChaoST* [4a4bee]

```

struct L {
    ll m, k, id;
    L() : id(-1) {}
    L(ll a, ll b, ll c) : m(a), k(b), id(c) {}
    ll at(ll x) { return m * x + k; }
};
class LiChao { // maintain max
private:

```

```

    int n; vector<L> nodes;
    void insert(int l, int r, int rt, L ln) {
        int m = (l + r) >> 1;
        if (nodes[rt].id == -1)
            return nodes[rt] = ln, void();
        bool atLeft = nodes[rt].at(l) < ln.at(l);
        if (nodes[rt].at(m) < ln.at(m))
            atLeft ^= 1, swap(nodes[rt], ln);
        if (r - l == 1) return;
        if (atLeft) insert(l, m, rt << 1, ln);
        else insert(m, r, rt << 1 | 1, ln);
    }
    ll query(int l, int r, int rt, ll x) {
        int m = (l + r) >> 1; ll ret = -INF;
        if (nodes[rt].id != -1) ret = nodes[rt].at(x);
        if (r - l == 1) return ret;
        if (x
            < m) return max(ret, query(l, m, rt << 1, x));
        return max(ret, query(m, r, rt << 1 | 1, x));
    }
public:
    LiChao(int n_) : n(n_), nodes(n * 4) {}
    void insert(L ln) { insert(0, n, 1, ln); }
    ll query(ll x) { return query(0, n, 1, x); }
};

```

3.8 Link cut tree* [a35b5d]

```

struct Splay { // xor-sum
    static Splay nil;
    Splay *ch[2], *f;
    int val, sum, rev, size;
    Splay(int
        _val = 0) : val(_val), sum(_val), rev(0), size(1)
    { f = ch[0] = ch[1] = &nil; }
    bool isr()
    { return f->ch[0] != this && f->ch[1] != this; }
    int dir()
    { return f->ch[0] == this ? 0 : 1; }
    void setCh(Splay *c, int d) {
        ch[d] = c;
        if (c != &nil) c->f = this;
        pull();
    }
    void give_tag(int r) {
        if (r) swap(ch[0], ch[1]), rev ^= 1;
    }
    void push() {
        if (ch[0] != &nil) ch[0]->give_tag(rev);
        if (ch[1] != &nil) ch[1]->give_tag(rev);
        rev = 0;
    }
    void pull() {
        // take care of the nil!
        size = ch[0]->size + ch[1]->size + 1;
        sum = ch[0]->sum ^ ch[1]->sum ^ val;
        if (ch[0] != &nil) ch[0]->f = this;
        if (ch[1] != &nil) ch[1]->f = this;
    }
} Splay::nil;
Splay *nil = &Splay::nil;
void rotate(Splay *x) {
    Splay *p = x->f;
    int d = x->dir();
    if (!p->isr()) p->f->setCh(x, p->dir());
    else x->f = p->f;
    p->setCh(x->ch[!d], d);
    x->setCh(p, !d);
    p->pull(), x->pull();
}
void splay(Splay *x) {
    vector<Splay*> splayVec;
    for (Splay *q = x;; q = q->f) {
        splayVec.pb(q);
        if (q->isr()) break;
    }
    reverse(ALL(splayVec));
    for (auto it : splayVec) it->push();
    while (!x->isr()) {
        if (x->f->isr()) rotate(x);
        else if (x->dir() == x->f->dir())
            rotate(x->f), rotate(x);
        else rotate(x), rotate(x);
    }
}
Splay* access(Splay *x) {
    Splay *q = nil;

```

```

for (; x != nil; x = x->f)
    splay(x), x->setCh(q, 1), q = x;
return q;
}
void root_path(Splay *x) { access(x), splay(x); }
void chroot(Splay *x){
    root_path(x), x->give_tag(1);
    x->push(), x->pull();
}
void split(Splay *x, Splay *y) {
    chroot(x), root_path(y);
}
void link(Splay *x, Splay *y) {
    root_path(x), chroot(y);
    x->setCh(y, 1);
}
void cut(Splay *x, Splay *y) {
    split(x, y);
    if (y->size != 5) return;
    y->push();
    y->ch[0] = y->ch[0]->f = nil;
}
Splay* get_root(Splay *x) {
    for (root_path(x); x->ch[0] != nil; x = x->ch[0])
        x->push();
    splay(x);
    return x;
}
bool conn(Splay *x, Splay *y) {
    return get_root(x) == get_root(y);
}
Splay* lca(Splay *x, Splay *y) {
    access(x), root_path(y);
    if (y->f == nil) return y;
    return y->f;
}
void change(Splay *x, int val) {
    splay(x), x->val = val, x->pull();
}
int query(Splay *x, Splay *y) {
    split(x, y);
    return y->sum;
}
}

```

3.9 Treap [5ab1a1]

```

struct node {
    int data, sz;
    node *l, *r;
    node(int k) : data(k), sz(1), l(0), r(0) {}
    void up() {
        sz = 1;
        if (l) sz += l->sz;
        if (r) sz += r->sz;
    }
    void down() {}
};
int sz(node *a) { return a ? a->sz : 0; }
node *merge(node *a, node *b) {
    if (!a || !b) return a ? a : b;
    if (rand() % (sz(a) + sz(b)) < sz(a))
        return a->down(), a->r = merge(a->r, b), a->up(),
            a;
    return b->down(), b->l = merge(a, b->l), b->up(), b;
}
void split(node *o, node *&a, node *&b, int k) {
    if (!o) return a = b = 0, void();
    o->down();
    if (o->data <= k)
        a = o, split(o->r, a->r, b, k), a->up();
    else b = o, split(o->l, a, b->l, k), b->up();
}
void split2(node *o, node *&a, node *&b, int k) {
    if (sz(o) <= k) return a = o, b = 0, void();
    o->down();
    if (sz(o->l) + 1 <= k)
        a = o, split2(o->r, a->r, b, k - sz(o->l) - 1);
    else b = o, split2(o->l, a, b->l, k);
    o->up();
}
node *kth(node *o, int k) {
    if (k <= sz(o->l)) return kth(o->l, k);
    if (k == sz(o->l) + 1) return o;
    return kth(o->r, k - sz(o->l) - 1);
}
int Rank(node *o, int key) {
    if (!o) return 0;
}

```

```

if (o->data < key)
    return sz(o->l) + 1 + Rank(o->r, key);
else return Rank(o->l, key);
}
bool erase(node *&o, int k) {
    if (!o) return 0;
    if (o->data == k) {
        node *t = o;
        o->down(), o = merge(o->l, o->r);
        delete t;
        return 1;
    }
    node *&t = k < o->data ? o->l : o->r;
    return erase(t, k) ? o->up(), 1 : 0;
}
void insert(node *&o, int k) {
    node *a, *b;
    split(o, a, b, k),
        o = merge(a, merge(new node(k), b));
}
void interval(node *&o, int l, int r) {
    node *a, *b, *c;
    split2(o, a, b, l - 1), split2(b, b, c, r);
    // operate
    o = merge(a, merge(b, c));
}
}

```

3.10 Sparse table [c135a1]

```

struct Sparse_table {
    int st[__lg(MAXN) + 1][MAXN], n;
    void init(int _n, int *data) {
        n = _n;
        for (int i = 0; i < n; ++i) st[0][i] = data[i];
        for (int i = 1, t = 2; t < n; t <= 1, i++)
            for (int j = 0; j + t <= n; j++)
                st[i][j]
                    = max(st[i - 1][j], st[i - 1][j + t / 2]);
    }
    int query(int a, int b) {
        int t = __lg(b - a + 1);
        return max(st[t][a], st[t][b - (1 << t) + 1]);
    }
};

```

4 Flow/Matching

4.1 Bipartite Matching* [784535]

```

struct Bipartite_Matching { // 0-base
    int mp[N], mq[N], dis[N + 1], cur[N], l, r;
    vector<int> G[N + 1];
    bool dfs(int u) {
        for (int &i = cur[u]; i < SZ(G[u]); ++i) {
            int e = G[u][i];
            if (mq[e] == l
                || (dis[mq[e]] == dis[u] + 1 && dfs(mq[e])))
                return mp[mq[e] = u] = e, 1;
        }
        return dis[u] = -1, 0;
    }
    bool bfs() {
        queue<int> q;
        fill_n(dis, l + 1, -1);
        for (int i = 0; i < l; ++i)
            if (!mp[i])
                q.push(i), dis[i] = 0;
        while (!q.empty()) {
            int u = q.front();
            q.pop();
            for (int e : G[u])
                if (!dis[mq[e]])
                    q.push(mq[e]), dis[mq[e]] = dis[u] + 1;
        }
        return dis[l] != -1;
    }
    int matching() {
        int res = 0;
        fill_n(mp, l, -1), fill_n(mq, r, l);
        while (bfs()) {
            fill_n(cur, l, 0);
            for (int i = 0; i < l; ++i)
                res += (!mp[i] && dfs(i));
        }
        return res; // (i, mp[i] != -1)
    }
    void add_edge(int s, int t) { G[s].pb(t); }
}

```

```

void init(int _l, int _r) {
    l = _l, r = _r;
    for (int i = 0; i <= l; ++i)
        G[i].clear();
}
};

```

4.2 Dinic [98fb3a]

```

struct MaxFlow { // 0-base
    struct edge {
        int to, cap, flow, rev;
    };
    vector<edge> G[MAXN];
    int s, t, dis[MAXN], cur[MAXN], n;
    int dfs(int u, int cap) {
        if (u == t || !cap) return cap;
        for (int &i = cur[u]; i < (int)G[u].size(); ++i) {
            edge &e = G[u][i];
            if (dis[e.to] == dis[u] + 1 && e.flow != e.cap) {
                int df = dfs(e.to, min(e.cap - e.flow, cap));
                if (df) {
                    e.flow += df;
                    G[e.to][e.rev].flow -= df;
                    return df;
                }
            }
        }
        dis[u] = -1;
        return 0;
    }
    bool bfs() {
        fill_n(dis, n, -1);
        queue<int> q;
        q.push(s), dis[s] = 0;
        while (!q.empty()) {
            int tmp = q.front();
            q.pop();
            for (auto &u : G[tmp])
                if (!dis[u.to] && u.flow != u.cap) {
                    q.push(u.to);
                    dis[u.to] = dis[tmp] + 1;
                }
        }
        return dis[t] != -1;
    }
    int maxflow(int _s, int _t) {
        s = _s, t = _t;
        int flow = 0, df;
        while (bfs()) {
            fill_n(cur, n, 0);
            while ((df = dfs(s, INF))) flow += df;
        }
        return flow;
    }
    void init(int _n) {
        n = _n;
        for (int i = 0; i < n; ++i) G[i].clear();
    }
    void reset() {
        for (int i = 0; i < n; ++i)
            for (auto &j : G[i]) j.flow = 0;
    }
    void add_edge(int u, int v, int cap) {
        G[u].pb(edge{v, cap, 0, (int)G[v].size()});
        G[v].pb(edge{u, 0, 0, (int)G[u].size() - 1});
    }
};

```

4.3 Kuhn Munkres* [4b3863]

```

struct KM { // 0-base, maximum matching
    ll w[N][N], hl[N], hr[N], slk[N];
    int fl[N], fr[N], pre[N], qu[N], ql, qr, n;
    bool vl[N], vr[N];
    void init(int _n) {
        n = _n;
        for (int i = 0; i < n; ++i)
            fill_n(w[i], n, -INF);
    }
    void add_edge(int a, int b, ll wei) {
        w[a][b] = wei;
    }
    bool Check(int x) {
        if (vl[x] = 1, ~fl[x])
            return vr[qu[qr++] = fl[x]] = 1;
        while (~x) swap(x, fr[fl[x] = pre[x]]);
    }
};

```

```

return 0;
}
void bfs(int s) {
    fill_n(slk, n, INF), fill_n(vl, n, 0), fill_n(vr, n, 0);
    ql = qr = 0, qu[qr++] = s, vr[s] = 1;
    for (ll d;;) {
        while (ql < qr)
            for (int x = 0, y = qu[ql++]; x < n; ++x)
                if (!vl[x] && slk
                    [x] >= (d = hl[x] + hr[y] - w[x][y])) {
                    if (pre[x] = y, d) slk[x] = d;
                    else if (!Check(x)) return;
                }
        d = INF;
        for (int x = 0; x < n; ++x)
            if (!vl[x] && d > slk[x]) d = slk[x];
        for (int x = 0; x < n; ++x) {
            if (vl[x]) hl[x] += d;
            else slk[x] -= d;
            if (vr[x]) hr[x] -= d;
        }
        for (int x = 0; x < n; ++x)
            if (!vl[x] && !slk[x] && !Check(x)) return;
    }
}
ll solve() {
    fill_n(fl, n, -1), fill_n(fr, n, -1), fill_n(hr, n, 0);
    for (int i = 0; i < n; ++i)
        hl[i] = *max_element(w[i], w[i] + n);
    for (int i = 0; i < n; ++i) bfs(i);
    ll res = 0;
    for (int i = 0; i < n; ++i) res += w[i][fl[i]];
    return res;
}
};

```

- Maximum/Minimum flow with lower bound / Circulation problem
 - Construct super source S and sink T .
 - For each edge (x, y, l, u) , connect $x \rightarrow y$ with capacity $u - l$.
 - For each vertex v , denote by $in(v)$ the difference between the sum of incoming lower bounds and the sum of outgoing lower bounds.
 - If $in(v) > 0$, connect $S \rightarrow v$ with capacity $in(v)$, otherwise, connect $v \rightarrow T$ with capacity $-in(v)$.
 - To maximize, connect $t \rightarrow s$ with capacity ∞ (skip this in circulation problem), and let f be the maximum flow from S to T . If $f \neq \sum_{v \in V, in(v) > 0} in(v)$, there's no solution. Otherwise, the maximum flow from s to t is the answer.
 - To minimize, let f be the maximum flow from S to T . Connect $t \rightarrow s$ with capacity ∞ and let the flow from S to T be f' . If $f + f' \neq \sum_{v \in V, in(v) > 0} in(v)$, there's no solution. Otherwise, f' is the answer.
 - The solution of each edge e is $l_e + f_e$, where f_e corresponds to the flow of edge e on the graph.
- Construct minimum vertex cover from maximum matching M on bipartite graph (X, Y)
 - Redirect every edge: $y \rightarrow x$ if $(x, y) \in M$, $x \rightarrow y$ otherwise.
 - DFS from unmatched vertices in X .
 - $x \in X$ is chosen iff x is unvisited.
 - $y \in Y$ is chosen iff y is visited.
- Minimum cost cyclic flow
 - Construct super source S and sink T
 - For each edge (x, y, c) , connect $x \rightarrow y$ with $(cost, cap) = (c, 1)$ if $c > 0$, otherwise connect $y \rightarrow x$ with $(cost, cap) = (-c, 1)$
 - For each edge with $c < 0$, sum these cost as K , then increase $d(y)$ by 1, decrease $d(x)$ by 1
 - For each vertex v with $d(v) > 0$, connect $S \rightarrow v$ with $(cost, cap) = (0, d(v))$
 - For each vertex v with $d(v) < 0$, connect $v \rightarrow T$ with $(cost, cap) = (0, -d(v))$
 - Flow from S to T , the answer is the cost of the flow $C + K$
- Maximum density induced subgraph
 - Binary search on answer, suppose we're checking answer T
 - Construct a max flow model, let K be the sum of all weights
 - Connect source $s \rightarrow v, v \in G$ with capacity K
 - For each edge (u, v, w) in G , connect $u \rightarrow v$ and $v \rightarrow u$ with capacity w
 - For $v \in G$, connect it with sink $v \rightarrow t$ with capacity $K + 2T - (\sum_{e \in E(v)} w(e)) - 2w(v)$
 - T is a valid answer if the maximum flow $f < K|V|$
- Minimum weight edge cover
 - For each $v \in V$ create a copy v' , and connect $u' \rightarrow v'$ with weight $w(u, v)$.
 - Connect $v \rightarrow v'$ with weight $2\mu(v)$, where $\mu(v)$ is the cost of the cheapest edge incident to v .
 - Find the minimum weight perfect matching on G' .

- Project selection problem

1. If $p_v > 0$, create edge (s, v) with capacity p_v ; otherwise, create edge (v, t) with capacity $-p_v$.
 2. Create edge (u, v) with capacity w with w being the cost of choosing u without choosing v .
 3. The mincut is equivalent to the maximum profit of a subset of projects.
- Dual of minimum cost maximum flow
1. Capacity c_{uv} , Flow f_{uv} , Cost w_{uv} , Required Flow difference for vertex b_u .
 2. If all w_{uv} are integers, then optimal solution can happen when all p_u are integers.

$$\min \sum_{uv} w_{uv} f_{uv} \quad \min \sum_u b_u p_u + \sum_{uv} c_{uv} \max(0, p_v - p_u - w_{uv})$$

$$-f_{uv} \geq -c_{uv} \Leftrightarrow \sum_v f_{vu} - \sum_v f_{uv} = -b_u \quad p_u \geq 0$$

5 String

5.1 KMP [5a0728]

```
int F[MAXN];
vector<int> match(string A, string B) {
    vector<int> ans;
    F[0] = -1, F[1] = 0;
    for (int i = 1, j = 0; i < SZ(B); F[++i] = ++j) {
        if (B[i] == B[j]) F[i] = F[j]; // optimize
        while (j != -1 && B[i] != B[j]) j = F[j];
    }
    for (int i = 0, j = 0; i < SZ(A); ++i) {
        while (j != -1 && A[i] != B[j]) j = F[j];
        if (++j == SZ(B)) ans.pb(i + 1 - j), j = F[j];
    }
    return ans;
}
```

5.2 Z-value* [b47c17]

```
int z[MAXN];
void make_z(const string &s) {
    int l = 0, r = 0;
    for (int i = 1; i < SZ(s); ++i) {
        for (z[i] = max(0, min(r - i + 1, z[i - l]));
             i + z[i] < SZ(s) && s[i + z[i]] == s[z[i]];
             ++z[i])
            ;
        if (i + z[i] - 1 > r) l = i, r = i + z[i] - 1;
    }
}
```

5.3 Manacher* [1ad8ef]

```
int z[MAXN]; // 0-base
/* center i: radius z[i * 2 + 1] / 2
   center i, i + 1: radius z[i * 2 + 2] / 2
   both aba, abba have radius 2 */
void Manacher(string tmp) {
    string s = "%";
    int l = 0, r = 0;
    for (char c : tmp) s.pb(c), s.pb('%');
    for (int i = 0; i < SZ(s); ++i) {
        z[i] = r > i ? min(z[2 * l - i], r - i) : 1;
        while (i - z[i] >= 0 && i + z[i] < SZ(s)
               && s[i + z[i]] == s[i - z[i]]) ++z[i];
        if (z[i] + i > r) r = z[i] + i, l = i;
    }
}
```

5.4 Suffix Array [0093e4]

```
struct suffix_array {
    int box[MAXN], tp[MAXN], m;
    bool not_equ(int a, int b, int k, int n) {
        return ra[a] != ra[b] || a + k >= n ||
            b + k >= n || ra[a + k] != ra[b + k];
    }
    void radix(int *key, int *it, int *ot, int n) {
        fill_n(box, m, 0);
        for (int i = 0; i < n; ++i) ++box[key[i]];
        partial_sum(box, box + m, box);
        for (int i = n - 1; i >= 0; --i)
            ot[--box[key[it[i]]]] = it[i];
    }
    void make_sa(const string &s, int n) {
        int k = 1;
        for (int i = 0; i < n; ++i) ra[i] = s[i];
        do {
```

```
        iota(tp, tp + k, n - k), iota(sa + k, sa + n, 0);
        radix(ra + k, sa + k, tp + k, n - k);
        radix(ra, tp, sa, n);
        tp[sa[0]] = 0, m = 1;
        for (int i = 1; i < n; ++i) {
            m += not_equ(sa[i], sa[i - 1], k, n);
            tp[sa[i]] = m - 1;
        }
        copy_n(tp, n, ra);
        k *= 2;
    } while (k < n && m != n);
}
void make_he(const string &s, int n) {
    for (int j = 0, k = 0; j < n; ++j) {
        if (ra[j])
            for (; s[j + k] == s[sa[ra[j] - 1] + k]; ++k)
                he[ra[j]] = k, k = max(0, k - 1);
    }
}
int sa[MAXN], ra[MAXN], he[MAXN];
void build(const string &s) {
    int n = SZ(s);
    fill_n
        (sa, n, 0), fill_n(ra, n, 0), fill_n(he, n, 0);
    fill_n(box, n, 0), fill_n(tp, n, 0), m = 256;
    make_sa(s, n), make_he(s, n);
}
};
```

5.5 De Bruijn sequence* [a09470]

```
constexpr int MAXC = 10, MAXN = 1e5 + 10;
struct DBSeq {
    int C, N, K, L, buf[MAXC * MAXN]; // K <= C^N
    void dfs(int *out, int t, int p, int &ptr) {
        if (ptr >= L) return;
        if (t > N) {
            if (N % p) return;
            for (int i = 1; i <= p && ptr < L; ++i)
                out[ptr++] = buf[i];
        } else {
            buf[t] = buf[t - p], dfs(out, t + 1, p, ptr);
            for (int j = buf[t - p] + 1; j < C; ++j)
                buf[t] = j, dfs(out, t + 1, t, ptr);
        }
    }
    void solve(int _c, int _n, int _k, int *out) {
        int p = 0;
        C = _c, N = _n, K = _k, L = N + K - 1;
        dfs(out, 1, 1, p);
        if (p < L) fill(out + p, out + L, 0);
    }
} dbs;
```

5.6 Main Lorentz [615b8f]

```
vector<pair<int, int>> rep[kN]; // 0-base [l, r]
void main_lorentz(const string &s, int sft = 0) {
    const int n = s.size();
    if (n == 1) return;
    const int nu = n / 2, nv = n - nu;
    const string u = s.substr(0, nu), v = s.substr(nu),
        ru(u.rbegin(), u.rend()), rv(v.rbegin(), v.rend());
    main_lorentz(u, sft), main_lorentz(v, sft + nu);
    const auto z1 = Zalgo(ru), z2 = Zalgo(v + '#' + u),
        z3 = Zalgo(ru + '#' + rv), z4 = Zalgo(v);
    auto get_z = [](const vector<int> &z, int i) {
        return
            (0 <= i and i < (int)z.size()) ? z[i] : 0;
    };
    auto add_rep
        = [&](bool left, int c, int l, int k1, int k2) {
            const
                int L = max(1, l - k2), R = min(l - left, k1);
            if (L > R) return;
            if (left)
                rep[l].emplace_back(sft + c - R, sft + c - L);
            else rep[l].emplace_back
                (sft + c - R - l + 1, sft + c - L - l + 1);
        };
    for (int cntr = 0; cntr < n; cntr++) {
        int l, k1, k2;
        if (cntr < nu) {
            l = nu - cntr;
            k1 = get_z(z1, nu - cntr);
            k2 = get_z(z2, nv + 1 + cntr);
```

```

    } else {
        l = cntr - nu + 1;
        k1 = get_z(z3, nu + 1 + nv - 1 - (cntr - nu));
        k2 = get_z(z4, (cntr - nu) + 1);
    }
    if (k1 + k2 >= l)
        add_rep(cntr < nu, cntr, l, k1, k2);
}
// p |in [l, r] => s[p, p + i] = s[p + i, p + 2i]

```

6 Math

6.1 ax+by=gcd(only exgcd *) [7b833d]

```

pll exgcd(ll a, ll b) {
    if (b == 0) return pll(1, 0);
    ll p = a / b;
    pll q = exgcd(b, a % b);
    return pll(q.Y, q.X - q.Y * p);
}
/* ax+by=res, let x be minimum non-negative
g, p = gcd(a, b), exgcd(a, b) * res / g
if p.X < 0: t = (abs(p.X) + b / g - 1) / (b / g)
else: t = -(p.X / (b / g))
p += (b / g, -a / g) * t * /

```

6.2 Floor and Ceil [692c04]

```

int floor(int a, int b)
{ return a / b - (a % b && (a < 0) ^ (b < 0)); }
int ceil(int a, int b)
{ return a / b + (a % b && (a < 0) ^ (b > 0)); }

```

6.3 Floor Enumeration [7cbcdf]

```

// enumerating x = floor(n / i), [l, r]
for (int l = 1, r; l <= n; l = r + 1) {
    int x = n / l;
    r = n / x;
}

```

6.4 Gaussian integer gcd [e637cd]

```

double a[110][110];
const double eps = 1e-7;
void solve(){
    int n;
    cin >> n;
    for(int i = 1; i <= n; i++){
        for(int j = 1; j <= n + 1; j++){
            cin >> a[i][j];
        }
    }
    for(int i = 1; i <= n; i++){
        int mx = i;
        for(int j = i + 1; j <= n; j++){
            if(fabs(a[j][i]) > fabs(a[mx][i])) mx = j;
        }
        swap(a[i], a[mx]);
        if(fabs(a[i][i]) < eps){
            continue;
        }
        for(int j = n + 1; j >= i; j--){
            a[i][j] /= a[i][i];
        }
        for(int j = i + 1; j <= n; j++){
            for(int k = n + 1; k >= i; k--){
                a[j][k] -= a[i][k] * a[j][i];
            }
        }
    }
    for(int i = n; i >= 1; i--){
        for(int j = i + 1; j <= n; j++){
            a[i][n + 1] -= a[i][j] * a[j][n + 1];
        }
        if(a[i][i] == 0){
            cout << "No Solution";
            return;
        }
        //a[i][i] = 0 and a[i][n + 1] == 0 無限多解
        //a[i][i] = 0 and a[i][n + 1] != 0 無解
    }
    cout << fixed << setprecision(2);
    for(int i = 1; i <= n; i++){
        cout << a[i][n + 1] << ' ';
    }
}

```

6.4.1 Construction

Primal	Dual
Maximize $c^T x$ s.t. $Ax \leq b, x \geq 0$	Minimize $b^T y$ s.t. $A^T y \geq c, y \geq 0$
Maximize $c^T x$ s.t. $Ax \leq b$	Minimize $b^T y$ s.t. $A^T y = c, y \geq 0$
Maximize $c^T x$ s.t. $Ax = b, x \geq 0$	Minimize $b^T y$ s.t. $A^T y \geq c$

\bar{x} and \bar{y} are optimal if and only if for all $i \in [1, n]$, either $\bar{x}_i = 0$ or $\sum_{j=1}^m A_{ji} \bar{y}_j = c_i$ holds and for all $i \in [1, m]$ either $\bar{y}_i = 0$ or $\sum_{j=1}^n A_{ij} \bar{x}_j = b_j$ holds.

- In case of minimization, let $c'_i = -c_i$
- $\sum_{1 \leq i \leq n} A_{ji} x_i \geq b_j \rightarrow \sum_{1 \leq i \leq n} -A_{ji} x_i \leq -b_j$
- $\sum_{1 \leq i \leq n} A_{ji} x_i = b_j$
 - $\sum_{1 \leq i \leq n} A_{ji} x_i \leq b_j$
 - $\sum_{1 \leq i \leq n} A_{ji} x_i \geq b_j$
- If x_i has no lower bound, replace x_i with $x_i - x'_i$

6.5 chineseRemainder [a53b6d]

```

ll solve(ll x1, ll m1, ll x2, ll m2) {
    ll g = gcd(m1, m2);
    if ((x2 - x1) % g) return -1; // no sol
    m1 /= g; m2 /= g;
    pll p = exgcd(m1, m2);
    ll lcm = m1 * m2 * g;
    ll res = p.first * (x2 - x1) * m1 + x1;
    // be careful with overflow
    return (res % lcm + lcm) % lcm;
}

```

6.6 Primes

```

/* 12721 13331 14341 75577 123457 222557
556679 999983 1097774749 1076767633 100102021
999997771 1001010013 1000512343 987654361 999991231
999888733 98789101 987777733 999991921 1010101333
1010102101 10000000000039 1000000000000037
2305843009213693951 4611686018427387847
9223372036854775783 18446744073709551557 */

```

6.7 Estimation

n	2	3	4	5	6	7	8	9	20	30	40	50	100		
$p(n)$	2	3	5	7	11	15	22	30	627	5604	4e4	2e5	2e8		
n	100	1e3	1e6	1e9	1e12	1e15	1e18								
$d(i)$	12	32	240	1344	6720	26880	103680								
n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\binom{2n}{n}$	2	6	20	70	252	924	3432	12870	48620	184756	7e5	2e6	1e7	4e7	1.5e8
n	2	3	4	5	6	7	8	9	10	11	12	13			
B_n	2	5	15	52	203	877	4140	21147	115975	7e5	4e6	3e7			

6.8 Euclidean Algorithms

- $m = \lfloor \frac{an+b}{c} \rfloor$
- Time complexity: $O(\log n)$

$$f(a, b, c, n) = \sum_{i=0}^n \left\lfloor \frac{ai+b}{c} \right\rfloor$$

$$= \begin{cases} \left\lfloor \frac{a}{c} \right\rfloor \cdot \frac{n(n+1)}{2} + \left\lfloor \frac{b}{c} \right\rfloor \cdot (n+1) \\ + f(a \bmod c, b \bmod c, c, n), & a \geq c \vee b \geq c \\ 0, & n < 0 \vee a = 0 \\ nm - f(c, c-b-1, a, m-1), & \text{otherwise} \end{cases}$$

$$g(a, b, c, n) = \sum_{i=0}^n i \left\lfloor \frac{ai+b}{c} \right\rfloor$$

$$= \begin{cases} \left\lfloor \frac{a}{c} \right\rfloor \cdot \frac{n(n+1)(2n+1)}{6} + \left\lfloor \frac{b}{c} \right\rfloor \cdot \frac{n(n+1)}{2} \\ + g(a \bmod c, b \bmod c, c, n), & a \geq c \vee b \geq c \\ 0, & n < 0 \vee a = 0 \\ \frac{1}{2} \cdot (n(n+1)m - f(c, c-b-1, a, m-1)) \\ - h(c, c-b-1, a, m-1), & \text{otherwise} \end{cases}$$

$$h(a, b, c, n) = \sum_{i=0}^n \left\lfloor \frac{ai+b}{c} \right\rfloor^2$$

$$= \begin{cases} \left\lfloor \frac{a}{c} \right\rfloor^2 \cdot \frac{n(n+1)(2n+1)}{6} + \left\lfloor \frac{b}{c} \right\rfloor^2 \cdot (n+1) \\ + 2 \left\lfloor \frac{a}{c} \right\rfloor \cdot \left\lfloor \frac{b}{c} \right\rfloor \cdot n(n+1) \\ + h(a \bmod c, b \bmod c, c, n) \\ + 2 \left\lfloor \frac{b}{c} \right\rfloor \cdot g(a \bmod c, b \bmod c, c, n) \\ + 2 \left\lfloor \frac{b}{c} \right\rfloor \cdot f(a \bmod c, b \bmod c, c, n), & a \geq c \vee b \geq c \\ 0, & n < 0 \vee a = 0 \\ nm(m+1) - 2g(c, c-b-1, a, m-1) \\ - 2f(c, c-b-1, a, m-1) - f(a, b, c, n), & \text{otherwise} \end{cases}$$

6.9 General Purpose Numbers

- Bernoulli numbers

$$B_0=1, B_1^{\pm}=\pm\frac{1}{2}, B_2=\frac{1}{6}, B_3=0$$

$$\sum_{j=0}^m \binom{m+1}{j} B_j = 0, \text{ EGF is } B(x) = \frac{x}{e^x - 1} = \sum_{n=0}^{\infty} B_n \frac{x^n}{n!}.$$

$$S_m(n) = \sum_{k=1}^n k^m = \frac{1}{m+1} \sum_{k=0}^m \binom{m+1}{k} B_k^+ n^{m+1-k}$$

- Stirling numbers of the second kind Partitions of n distinct elements into exactly k groups.

$$S(n, k) = S(n-1, k-1) + kS(n-1, k), S(n, 1) = S(n, n) = 1$$

$$S(n, k) = \frac{1}{k!} \sum_{i=0}^k (-1)^{k-i} \binom{k}{i} i^n$$

$$x^n = \sum_{i=0}^n S(n, i) (x)_i$$

- Pentagonal number theorem

$$\prod_{n=1}^{\infty} (1 - x^n) = 1 + \sum_{k=1}^{\infty} (-1)^k \left(x^{k(3k+1)/2} + x^{k(3k-1)/2} \right)$$

- Catalan numbers

$$C_n^{(k)} = \frac{1}{(k-1)n+1} \binom{kn}{n}$$

$$C^{(k)}(x) = 1 + x[C^{(k)}(x)]^k$$

- Eulerian numbers

Number of permutations $\pi \in S_n$ in which exactly k elements are greater than the previous element. k j's s.t. $\pi(j) > \pi(j+1)$, $k+1$ j's s.t. $\pi(j) \geq j$, k j's s.t. $\pi(j) > j$.

$$E(n, k) = (n-k)E(n-1, k-1) + (k+1)E(n-1, k)$$

$$E(n, 0) = E(n, n-1) = 1$$

$$E(n, k) = \sum_{j=0}^k (-1)^j \binom{n+1}{j} (k+1-j)^n$$

6.10 Tips for Generating Functions

- Ordinary Generating Function $A(x) = \sum_{i \geq 0} a_i x^i$

- $A(rx) \Rightarrow r^n a_n$
- $A(x) + B(x) \Rightarrow a_n + b_n$
- $A(x)B(x) \Rightarrow \sum_{i=0}^n a_i b_{n-i}$
- $A(x)^k \Rightarrow \sum_{i_1+i_2+\dots+i_k=n} a_{i_1} a_{i_2} \dots a_{i_k}$
- $x A(x)' \Rightarrow n a_n$
- $\frac{A(x)}{1-x} \Rightarrow \sum_{i=0}^n a_i$

- Exponential Generating Function $A(x) = \sum_{i \geq 0} \frac{a_i}{i!} x^i$

- $A(x) + B(x) \Rightarrow a_n + b_n$
- $A^{(k)}(x) \Rightarrow a_{n+k}$
- $A(x)B(x) \Rightarrow \sum_{i=0}^n \binom{n}{i} a_i b_{n-i}$
- $A(x)^k \Rightarrow \sum_{i_1+i_2+\dots+i_k=n} \binom{n}{i_1, i_2, \dots, i_k} a_{i_1} a_{i_2} \dots a_{i_k}$
- $x A(x) \Rightarrow n a_n$

- Special Generating Function

- $(1+x)^n = \sum_{i \geq 0} \binom{n}{i} x^i$
- $\frac{1}{(1-x)^n} = \sum_{i \geq 0} \binom{n-1}{i} x^i$

7 Polynomial

7.1 Fast Fourier Transform [56bdd7]

```
template<int MAXN>
struct FFT {
    using val_t = complex<double>;
    const double PI = acos(-1);
    val_t w[MAXN];
    FFT() {
        for (int i = 0; i < MAXN; ++i) {
            double arg = 2 * PI * i / MAXN;
            w[i] = val_t(cos(arg), sin(arg));
        }
    }
    void bitrev(val_t *a, int n); // see NTT
    void trans
        (val_t *a, int n, bool inv = false); // see NTT;
        // remember to replace LL with val_t
};
```

7.2 Number Theory Transform* [f68103]

```
//(2^16)+1, 65537, 3
//7*17*(2^23)+1, 998244353, 3
//1255*(2^20)+1, 1315962881, 3
//51*(2^25)+1, 1711276033, 29
template<int MAXN, ll P, ll RT> //MAXN must be 2^k
struct NTT {
    ll w[MAXN];
    ll mpow(ll a, ll n);
    ll minv(ll a) { return mpow(a, P - 2); }
    NTT() {
        ll dw = mpow(RT, (P - 1) / MAXN);
        w[0] = 1;
        for (int
            i = 1; i < MAXN; ++i) w[i] = w[i - 1] * dw % P;
```

```

    }
    void bitrev(ll *a, int n) {
        int i = 0;
        for (int j = 1; j < n - 1; ++j) {
            for (int k = n >> 1; (i ^ k) < k; k >>= 1);
            if (j < i) swap(a[i], a[j]);
        }
    }
    void operator()(
        ll *a, int n, bool inv = false) { //0 <= a[i] < P
        bitrev(a, n);
        for (int L = 2; L <= n; L <= 1) {
            int dx = MAXN / L, dl = L >> 1;
            for (int i = 0; i < n; i += L) {
                for (int
                    j = i, x = 0; j < i + dl; ++j, x += dx) {
                    ll tmp = a[j + dl] * w[x] % P;
                    if ((a[j
                        + dl] = a[j] - tmp) < 0) a[j + dl] += P;
                    if ((a[j] += tmp) >= P) a[j] -= P;
                }
            }
        }
        if (inv) {
            reverse(a + 1, a + n);
            ll invn = minv(n);
            for (int
                i = 0; i < n; ++i) a[i] = a[i] * invn % P;
        }
    }
};
```

7.3 Newton's Method

Given $F(x)$ where

$$F(x) = \sum_{i=0}^{\infty} \alpha_i (x - \beta)^i$$

for β being some constant. Polynomial P such that $F(P) = 0$ can be found iteratively. Denote by Q_k the polynomial such that $F(Q_k) = 0 \pmod{x^{2^k}}$, then

$$Q_{k+1} = Q_k - \frac{F(Q_k)}{F'(Q_k)} \pmod{x^{2^{k+1}}}$$

8 Geometry

8.1 Default Code [7002f8]

```
typedef pair<double, double> pdd;
typedef pair<pdd, pdd> Line;
struct Cir { pdd O; double R; };
const double eps = 1e-8;
pdd operator+(pdd a, pdd b)
{ return pdd(a.X + b.X, a.Y + b.Y); }
pdd operator-(pdd a, pdd b)
{ return pdd(a.X - b.X, a.Y - b.Y); }
pdd operator*(pdd a, double b)
{ return pdd(a.X * b, a.Y * b); }
pdd operator/(pdd a, double b)
{ return pdd(a.X / b, a.Y / b); }
double dot(pdd a, pdd b)
{ return a.X * b.X + a.Y * b.Y; }
double cross(pdd a, pdd b)
{ return a.X * b.Y - a.Y * b.X; }
double abs2(pdd a)
{ return dot(a, a); }
double abs(pdd a)
{ return sqrt(dot(a, a)); }
int sign(double a)
{ return fabs(a) < eps ? 0 : a > 0 ? 1 : -1; }
int ori(pdd a, pdd b, pdd c)
{ return sign(cross(b - a, c - a)); }
bool collinearity(pdd p1, pdd p2, pdd p3)
{ return sign(cross(p1 - p3, p2 - p3)) == 0; }
bool btw(pdd p1, pdd p2, pdd p3) {
    if (!collinearity(p1, p2, p3)) return 0;
    return sign(dot(p1 - p3, p2 - p3)) <= 0;
}
bool seg_intersect(pdd p1, pdd p2, pdd p3, pdd p4) {
    int a123 = ori(p1, p2, p3);
    int a124 = ori(p1, p2, p4);
    int a341 = ori(p3, p4, p1);
    int a342 = ori(p3, p4, p2);
    if (a123 == 0 && a124 == 0)
        return btw(p1, p2, p3) || btw(p1, p2, p4) ||
            btw(p3, p4, p1) || btw(p3, p4, p2);
```

```

    return a123 * a124 <= 0 && a341 * a342 <= 0;
}
pdd intersect(pdd p1, pdd p2, pdd p3, pdd p4) {
    double a123 = cross(p2 - p1, p3 - p1);
    double a124 = cross(p2 - p1, p4 - p1);
    return (p4
        * a123 - p3 * a124) / (a123 - a124); // C^3 / C^2
}
pdd perp(pdd p1)
{ return pdd(-p1.Y, p1.X); }
pdd projection(pdd p1, pdd p2, pdd p3)
{ return p1 + (
    p2 - p1) * dot(p3 - p1, p2 - p1) / abs2(p2 - p1); }
pdd reflection(pdd p1, pdd p2, pdd p3)
{ return p3 + perp(p2 - p1
    ) * cross(p3 - p1, p2 - p1) / abs2(p2 - p1) * 2; }
pdd linearTransformation
    (pdd p0, pdd p1, pdd q0, pdd q1, pdd r) {
    pdd dp = p1 - p0,
        dq = q1 - q0, num(cross(dp, dq), dot(dp, dq));
    return q0 + pdd(
        cross(r - p0, num), dot(r - p0, num)) / abs2(dp);
} // from line p0--p1 to q0--q1, apply to r

```

8.2 PointSegDist* [57b6de]

```

double PointSegDist(pdd q0, pdd q1, pdd p) {
    if (sign(abs(q0 - q1)) == 0) return abs(q0 - p);
    if (sign(dot(q1 - q0,
        p - q0)) >= 0 && sign(dot(q0 - q1, p - q1)) >= 0)
        return fabs(cross(q1 - q0, p - q0) / abs(q0 - q1));
    return min(abs(p - q0), abs(p - q1));
}

```

8.3 Convex hull* [feda6f]

```

void hull(vector<pll> &dots) { // n=1 => ans = {}
    sort(dots.begin(), dots.end());
    vector<pll> ans(1, dots[0]);
    for (int ct = 0; ct < 2; ++ct, reverse(ALL(dots)))
        for (int i = 1,
            t = SZ(ans); i < SZ(dots); ans.pb(dots[i++]))
            while (SZ(ans) > t && ori
                (ans[SZ(ans) - 2], ans.back(), dots[i]) <= 0)
                ans.pop_back();
    ans.pop_back(), ans.swap(dots);
}

```

8.4 PointInConvex* [f86640]

```

bool PointInConvex
    (const vector<pll> &C, pll p, bool strict = true) {
    int a = 1, b = SZ(C) - 1, r = !strict;
    if (SZ(C) == 0) return false;
    if (SZ(C) < 3) return r && btw(C[0], C.back(), p);
    if (ori(C[0], C[a], C[b]) > 0) swap(a, b);
    if (ori
        (C[0], C[a], p) >= r || ori(C[0], C[b], p) <= -r)
        return false;
    while (abs(a - b) > 1) {
        int c = (a + b) / 2;
        (ori(C[0], C[c], p) > 0 ? b : a) = c;
    }
    return ori(C[a], C[b], p) < r;
}

```

8.5 Intersection of line and convex [157258]

```

int TangentDir(vector<pll> &C, pll dir) {
    return cyc_tsearch(SZ(C), [&](int a, int b) {
        return cross(dir, C[a]) > cross(dir, C[b]);
    });
}
#define cml(i) sign(cross(C[i] - a, b - a))
pii lineHull(pll a, pll b, vector<pll> &C) {
    int A = TangentDir(C, a - b);
    int B = TangentDir(C, b - a);
    int n = SZ(C);
    if (cml(A) < 0 || cml(B) > 0)
        return pii(-1, -1); // no collision
    auto gao = [&](int l, int r) {
        for (int t = l; (l + 1) % n != r; ) {
            int m = ((l + r + (l < r ? 0 : n)) / 2) % n;
            (cml(m) == cml(t) ? l : r) = m;
        }
        return (l + !cml(r)) % n;
    };
    pii res = pii(gao(B, A), gao(A, B)); // (i, j)
}

```

```

if (res.X == res.Y) // touching the corner i
    return pii(res.X, -1);
if (!
    cml(res.X) && !cml(res.Y)) // along side i, i+1
    switch ((res.X - res.Y + n + 1) % n) {
        case 0: return pii(res.X, res.X);
        case 2: return pii(res.Y, res.Y);
    }
    /* crossing sides (i, i+1) and (j, j+1)
    crossing corner i is treated as side (i, i+1)
    returned
    in the same order as the line hits the convex */
    return res;
} // convex cut: (r, l]

```

8.6 VectorInPoly* [c6d0fa]

```

// ori(a
    , b, c) >= 0, valid: "strict" angle from a-b to a-c
bool btwangle(pll a, pll b, pll c, pll p, int strict) {
    return
        ori(a, b, p) >= strict && ori(a, p, c) >= strict;
}
// whether vector
    {cur, p} in counter-clockwise order prv, cur, nxt
bool inside
    (pll prv, pll cur, pll nxt, pll p, int strict) {
    if (ori(cur, nxt, prv) >= 0)
        return btwangle(cur, nxt, prv, p, strict);
    return !btwangle(cur, prv, nxt, p, !strict);
}

```

8.7 PolyUnion* [3c9b0b]

```

double rat(pll a, pll b) {
    return sign
        (b.X) ? (double)a.X / b.X : (double)a.Y / b.Y;
} // all poly. should be ccw
double polyUnion(vector<vector<pll>> &poly) {
    double res = 0;
    for (auto &p : poly)
        for (int a = 0; a < SZ(p); ++a) {
            pll A = p[a], B = p[(a + 1) % SZ(p)];
            vector
                <pair<double, int>> segs = {{0, 0}, {1, 0}};
            for (auto &q : poly) {
                if (&p == &q) continue;
                for (int b = 0; b < SZ(q); ++b) {
                    pll C = q[b], D = q[(b + 1) % SZ(q)];
                    int sc = ori(A, B, C), sd = ori(A, B, D);
                    if (sc != sd && min(sc, sd) < 0) {
                        double sa = cross(D
                            - C, A - C), sb = cross(D - C, B - C);
                        segs.emplace_back
                            (sa / (sa - sb), sign(sc - sd));
                    }
                    if (!sc && !sd &&
                        &q < &p && sign(dot(B - A, D - C)) > 0) {
                        segs.emplace_back(rat(C - A, B - A), 1);
                        segs.emplace_back(rat(D - A, B - A), -1);
                    }
                }
            }
            sort(ALL(segs));
            for (auto &s : segs) s.X = clamp(s.X, 0.0, 1.0);
            double sum = 0;
            int cnt = segs[0].second;
            for (int j = 1; j < SZ(segs); ++j) {
                if (!cnt) sum += segs[j].X - segs[j - 1].X;
                cnt += segs[j].Y;
            }
            res += cross(A, B) * sum;
        }
    return res / 2;
}

```

8.8 Polar Angle Sort* [b20533]

```

int cml(pll a, pll b, bool same = true) {
#define is_neg(k) (
    sign(k.Y) < 0 || (sign(k.Y) == 0 && sign(k.X) < 0))
    int A = is_neg(a), B = is_neg(b);
    if (A != B)
        return A < B;
    if (sign(cross(a, b)) == 0)
        return same ? abs2(a) < abs2(b) : -1;
    return sign(cross(a, b)) > 0;
}

```

8.9 Half plane intersection* [3753a5]

```

pll area_pair(Line a, Line b)
{ return pll(cross(a.Y
    - a.X, b.X - a.X), cross(a.Y - a.X, b.Y - a.X)); }
bool isin(Line l0, Line l1, Line l2) {
    // Check inter(l1, l2) strictly in l0
    auto [a02X, a02Y] = area_pair(l0, l2);
    auto [a12X, a12Y] = area_pair(l1, l2);
    if (a12X - a12Y < 0) a12X *= -1, a12Y *= -1;
    return (__int128
        ) a02Y * a12X - (__int128) a02X * a12Y > 0;
}
/* Having solution, check size > 2 */
/* --- Line.X --- Line.Y --- */
vector<Line> halfPlaneInter(vector<Line> arr) {
    sort(ALL(arr), [&](Line a, Line b) -> int {
        if (cmp(a.Y - a.X, b.Y - b.X, 0) != -1)
            return cmp(a.Y - a.X, b.Y - b.X, 0);
        return ori(a.X, a.Y, b.Y) < 0;
    });
    deque<Line> dq(1, arr[0]);
    auto pop_back = [&](int t, Line p) {
        while (SZ(dq)
            ) >= t && !isin(p, dq[SZ(dq) - 2], dq.back())
            dq.pop_back();
    };
    auto pop_front = [&](int t, Line p) {
        while (SZ(dq) >= t && !isin(p, dq[0], dq[1]))
            dq.pop_front();
    };
    for (auto p : arr)
        if (cmp(
            dq.back().Y - dq.back().X, p.Y - p.X, 0) != -1)
            pop_back(2, p), pop_front(2, p), dq.pb(p);
    pop_back(3, dq[0]), pop_front(3, dq.back());
    return vector<Line>(ALL(dq));
}

```

8.10 RotatingSweepLine [af0be4]

```

void rotatingSweepLine(vector<pii> &ps) {
    int n = SZ(ps), m = 0;
    vector<int> id(n), pos(n);
    vector<pii> line(n * (n - 1));
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < n; ++j)
            if (i != j) line[m++] = pii(i, j);
    sort(ALL(line), [&](pii a, pii b) {
        return cmp(ps[a.Y] - ps[a.X], ps[b.Y] - ps[b.X]);
    }); // cmp(): polar angle compare
    iota(ALL(id), 0);
    sort(ALL(id), [&](int a, int b) {
        if (ps[a.Y] != ps[b.Y]) return ps[a.Y] < ps[b.Y];
        return ps[a] < ps[b];
    }); // initial order, since (1, 0) is the smallest
    for (int i = 0; i < n; ++i) pos[id[i]] = i;
    for (int i = 0; i < m; ++i) {
        auto l = line[i];
        // do something
        tie(pos[l.X], pos[l.Y], id[pos[l.X]], id[pos[l.Y]
            ]) = make_tuple(pos[l.Y], pos[l.X], l.Y, l.X);
    }
}

```

8.11 Minkowski Sum* [9fbd05]

```

vector<pll> Minkowski
(vector<pll> A, vector<pll> B) { // |A|, |B| >= 3
    hull(A), hull(B);
    vector<pll> C(1, A[0] + B[0]), s1, s2;
    for (int i = 0; i < SZ(A); ++i)
        s1.pb(A[(i + 1) % SZ(A)] - A[i]);
    for (int i = 0; i < SZ(B); ++i)
        s2.pb(B[(i + 1) % SZ(B)] - B[i]);
    for (int i = 0, j = 0; i < SZ(A) || j < SZ(B);)
        if (j >= SZ
            (B) || (i < SZ(A) && cross(s1[i], s2[j]) >= 0))
            C.pb(B[j % SZ(B)] + A[i++]);
        else
            C.pb(A[i % SZ(A)] + B[j++]);
    return hull(C), C;
}

```

9 Else

9.1 Cyclic Ternary Search* [9017cc]

```

/* bool pred(int a, int b);
f(0) ~ f(n - 1) is a cyclic-shift U-function
return idx s.t. pred(x, idx) is false forall x*/
int cyc_tsearch(int n, auto pred) {
    if (n == 1) return 0;
    int l = 0, r = n; bool rv = pred(1, 0);
    while (r - l > 1) {
        int m = (l + r) / 2;
        if (pred(0, m) ? rv: pred(m, (m + 1) % n)) r = m;
        else l = m;
    }
    return pred(l, r % n) ? l : r % n;
}

```

9.2 Mo's Algorithm(With modification) [f05c5b]

```

/*
Mo's Algorithm With modification
Block: N^{2/3}, Complexity: N^{5/3}
*/
struct Query {
    int L, R, LBid, RBid, T;
    Query(int l, int r, int t):
        L(l), R(r), LBid(l / blk), RBid(r / blk), T(t) {}
    bool operator<(const Query &q) const {
        if (LBid != q.LBid) return LBid < q.LBid;
        if (RBid != q.RBid) return RBid < q.RBid;
        return T < q.T;
    }
};
void solve(vector<Query> query) {
    sort(ALL(query));
    int L=0, R=0, T=-1;
    for (auto q : query) {
        while (T < q.T) addTime(L, R, ++T); // TODO
        while (T > q.T) subTime(L, R, T--); // TODO
        while (R < q.R) add(arr[++R]); // TODO
        while (L > q.L) add(arr[--L]); // TODO
        while (R > q.R) sub(arr[R--]); // TODO
        while (L < q.L) sub(arr[L--]); // TODO
        // answer query
    }
}

```

9.3 Mo's Algorithm On Tree [8331c2]

```

/*
Mo's Algorithm On Tree
Preprocess:
1) LCA
2) dfs with in[u] = dft++, out[u] = dft++
3) ord[in[u]] = ord[out[u]] = u
4) bitset<MAXN> inset
*/
struct Query {
    int L, R, LBid, lca;
    Query(int u, int v) {
        int c = LCA(u, v);
        if (c == u || c == v)
            q.lca = -1, q.L = out[c ^ u ^ v], q.R = out[c];
        else if (out[u] < in[v])
            q.lca = c, q.L = out[u], q.R = in[v];
        else
            q.lca = c, q.L = out[v], q.R = in[u];
        q.Lid = q.L / blk;
    }
    bool operator<(const Query &q) const {
        if (LBid != q.LBid) return LBid < q.LBid;
        return R < q.R;
    }
};
void flip(int x) {
    if (inset[x]) sub(arr[x]); // TODO
    else add(arr[x]); // TODO
    inset[x] = ~inset[x];
}
void solve(vector<Query> query) {
    sort(ALL(query));
    int L = 0, R = 0;
    for (auto q : query) {
        while (R < q.R) flip(ord[++R]);
        while (L > q.L) flip(ord[--L]);
        while (R > q.R) flip(ord[R--]);
        while (L < q.L) flip(ord[L--]);
        if (~q.lca) add(arr[q.lca]);
        // answer query
        if (~q.lca) sub(arr[q.lca]);
    }
}

```

```
}
}
```

9.4 Additional Mo's Algorithm Trick

• Mo's Algorithm With Addition Only

- Sort queries same as the normal Mo's algorithm.
- For each query $[l, r]$:
- If $l/blk = r/blk$, brute-force.
- If $l/blk \neq curL/blk$, initialize $curL := (l/blk + 1) \cdot blk$, $curR := curL - 1$
- If $r > curR$, increase $curR$
- decrease $curL$ to fit l , and then undo after answering

• Mo's Algorithm With Offline Second Time

- Require: Changing answer \equiv adding $f([l, r], r+1)$.
- Require: $f([l, r], r+1) = f([l, r], r+1) - f([l, l], r+1)$.
- Part1: Answer all $f([l, r], r+1)$ first.
- Part2: Store $curR \rightarrow R$ for $curL$ (reduce the space to $O(N)$), and then answer them by the second offline algorithm.
- Note: You must do the above symmetrically for the left boundaries.

9.5 All LCS* [78a378]

```
void all_lcs(string s, string t) { // 0-base
    vector<int> h(SZ(t));
    iota(ALL(h), 0);
    for (int a = 0; a < SZ(s); ++a) {
        int v = -1;
        for (int c = 0; c < SZ(t); ++c)
            if (s[a] == t[c] || h[c] < v)
                swap(h[c], v);
        // LCS(s[0, a], t[b, c]) =
        // c - b + 1 - sum([h[i] >= b] | i <= c)
        // h[i] might become -1 !!
    }
}
```

Start from $S = \emptyset$. In each iteration, let

- $Y_1 = \{x \notin S \mid S \cup \{x\} \in I_1\}$
- $Y_2 = \{x \notin S \mid S \cup \{x\} \in I_2\}$

If there exists $x \in Y_1 \cap Y_2$, insert x into S . Otherwise for each $x \in S, y \notin S$, create edges

- $x \rightarrow y$ if $S - \{x\} \cup \{y\} \in I_1$.
- $y \rightarrow x$ if $S - \{x\} \cup \{y\} \in I_2$.

Find a *shortest* path (with BFS) starting from a vertex in Y_1 and ending at a vertex in Y_2 which doesn't pass through any other vertices in Y_2 , and alternate the path. The size of S will be incremented by 1 in each iteration. For the weighted case, assign weight $w(x)$ to vertex x if $x \in S$ and $-w(x)$ if $x \notin S$. Find the path with the minimum number of edges among all minimum length paths and alternate it.

9.6 Tree Hash* [34aae5]

```
ull seed;
ull shift(ull x) {
    x ^= x << 13;
    x ^= x >> 7;
    x ^= x << 17;
    return x;
}
ull dfs(int u, int f) {
    ull sum = seed;
    for (int i : G[u])
        if (i != f)
            sum += shift(dfs(i, u));
    return sum;
}
```

9.7 Min Plus Convolution* [09b5c3]

```
// a is convex a[i+1]-a[i] <= a[i+2]-a[i+1]
vector<int> min_plus_convolution
(vector<int> &a, vector<int> &b) {
    int n = SZ(a), m = SZ(b);
    vector<int> c(n + m - 1, INF);
    auto dc = [&](auto Y, int l, int r, int jl, int jr) {
        if (l > r) return;
        int mid = (l + r) / 2, from = -1, &best = c[mid];
        for (int j = jl; j <= jr; ++j)
            if (int i = mid - j; i >= 0 && i < n)
                if (best > a[i] + b[j])
                    best = a[i] + b[j], from = j;
        Y(Y, l, mid - 1, jl, from), Y(Y, mid + 1, r, from, jr);
    };
    return dc(dc, 0, n - 1 + m - 1, 0, m - 1), c;
}
```

9.8 Bitset LCS [330ab1]

```
cin >> n >> m;
for (int i = 1, x; i <= n; ++i)
    cin >> x, p[x].set(i);
for (int i = 1, x; i <= m; i++) {
    cin >> x, (g = f) |= p[x];
    f.shiftLeftByOne(), f.set(0);
    ((f = g - f) ^= g) &= g;
}
cout << f.count() << '\n';
```

10 Python

10.1 Misc

```
from decimal import *
setcontext(Context(prec
                 =MAX_PREC, Emax=MAX_EMAX, rounding=ROUND_FLOOR))
print(Decimal(input()) * Decimal(input()))
from fractions import Fraction
Fraction
('3.14159').limit_denominator(10).numerator # 22
```