

Contents

1	Setups	1	3.2	CHT	3
1.1	vimrc	1	3.3	Treap	3
1.2	pbds	1	4	Geometry	4
1.3	terminal	1	4.1	Point	4
1.4	debug	1	4.2	Convex Hull	4
1.5	template	1	4.3	Minkowski sum	4
2	Graph	1	5	String	4
2.1	Dominator Tree	1	5.1	Suffix Array (SAIS)	4
2.2	Incremental SCC	1	5.2	ACautomaton	5
2.3	Block-Cut Tree	2	6	Math	5
2.4	Euler Tour	2	6.1	module int	5
3	Data Structure	3	6.2	FFT	6
3.1	LiChao Tree	3	6.3	FWT	6

1 Setups

1.1 vimrc [6c9876]

```
se nu ai rnu cin ts=4 sw=4 | sy on
inoremap {<CR> {<CR><Esc>O
inoremap jk <Esc>
```

1.2 pbds [9f7c3e]

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

template <typename T>
using ordered_set = tree<T, null_type, less
    <T>, rb_tree_tag, tree_order_statistics_node_update>;

int main(){
    ordered_set<int> st;
    st.insert(1);
    st.find_by_order(0); //iterator to 1
    st.order_of_key(1); //returns 0
}
```

1.3 terminal [46fc34]

```
-- terminal --
$ setxkbmap -option caps:swapescape
```

1.4 debug [33c0d3]

```
#ifdef MIKU
string
    dbmc = "\033[1;38;2;57;197;187m", dbrs = \033[0m;
#define debug
    (x...) cout << dbmc << "[" << #x << "]" : ", dout(x)
void dout() { cout << dbrs << endl; }
template <typename T, typename ...U>
void dout(T t, U ...u) { cout
    << t << sizeof...(u) ? ", " : ""; dout(u...); }
#else
#define debug(...) 39
#endif

int main(){
    int a = 49;
    char c = '8';
    debug(a); // outputs "[a] : 49"
    debug(a, c); // outputs "[a, c] : 49, 8"
    debug("PCCORZ"); // outputs "[\"PCCORZ\"] : PCCORZ"
    debug(); // outputs "[ ] : "
}
```

1.5 template [5116af]

```
#include <bits/stdc++.h>
using namespace std;
#define fs first
#define sc second
#define F first
#define S second
#define FOR(i, j, k) for (int i = j, Z = k; i < Z; i++)
using ll = long long;
using lll = __int128_t;
typedef pair<int, int> pii;
typedef tuple<int, int, int> tiii;
typedef pair<ll, ll> pll

int main(){
}
```

2 Graph

2.1 Dominator Tree [3b89c3]

```
struct DominatorTree{
    //1-indexed
    //not reachable from s -> not on tree
    int n;
    vector<vector<int>> G, rG;
    vector<int> pa, dfn, id;
    int dfnCnt;
    vector<int> semi, idom, best;
    vector<vector<int>> ret;
    void init(int _n){
        n=_n;
        G = rG = ret = vector<vector<int>>(n+1);
        pa = dfn = id = vector<int>(n+1, -1);
        dfnCnt = 0;
        semi = idom = best = vector<int>(n+1, -1);
    }
    void add_edge(int u, int v){
        G[u].push_back(v);
        rG[v].push_back(u);
    }
    void dfs(int u){
        id[dfn[u]] = ++dfnCnt;
        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){
        dfnCnt = 0;
        for(int i = 1; i <= n; ++i){
            dfn[i] = idom[i] = 0;
            ret[i].clear();
            best[i] = semi[i] = i;
        }
        dfs(root);
        for(int i = dfnCnt; 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]]);
            }
            ret[semi[i]].push_back(i);
            for(auto v:ret[pa[i]]){
                find(v, pa[i]);
                idom[v]
                    = semi[best[v]] == pa[i] ? pa[i] : best[v];
            }
            ret[pa[i]].clear();
        }
        for(int i = 2; i <= dfnCnt; ++i){
            if(idom[i] != semi[i]) idom[i] = idom[idom[i]];
            ret[id[idom[i]]].push_back(id[i]);
        }
    }
    vector<vector<int>> solve(int s){
        tarjan(s);
        return ret;
    }
};
```

2.2 Incremental SCC [d8b556]

```
struct IncrementalSCC{
#define pii pair<int, int>
#define fs first
#define sc second
#define tiii tuple<int, int, int>
    //if u == v : ans[i] = -1
    //if not connected : ans[i] = m
    //all 0-indexed
    int n;
    vector<int> ans;
    int m;
    vector<tiii> all;
    vector<int> SCC(int n, vector<vector<int>>& paths){
        vector<int> scc_id(n, -1), idx(n, -1), low(n, -1), st;
        int cnt = 0, gcnt = 0;
```

```

function<void(int)> dfs = [&](int now)->void{
    low[now] = idx[now] = cnt++;
    st.push_back(now);
    for(auto nxt:paths[now]){
        if(scc_id[nxt] != -1)continue;
        if(idx[nxt] == -1){
            dfs(nxt);
            low[now] = min(low[now],low[nxt]);
        }
        else{
            low[now] = min(low[now],idx[nxt]);
        }
    }
    if(low[now] == idx[now]){
        int id = -1;
        while(id != now){
            id = st.back();
            st.pop_back();
            scc_id[id] = gcnt;
        }
        gcnt++;
    }
};
for(int i = 0;i<n;i++){
    if(scc_id[i] == -1)dfs(i);
}
//cerr<<"SCC: "<<n<<": ";for(int
    i = 0;i<n;i++)cerr<<scc_id[i]<<',';cerr<<endl;
return scc_id;
}
vector<int> mapping;
void dc(int l,int r,vector<tiii> &edges){
    //cerr<<l<<' '<<r<<": "<<endl;
    if(l == r){
        for(auto
            &[id,_,_]:edges)ans[id] = min(ans[id],l);
        return;
    }
    int mid = (l+r)>>1;
    int cnt = 0;
    for(auto &[t,u,v]:edges){
        if(mapping[u] == -1)mapping[u] = cnt++;
        if(mapping[v] == -1)mapping[v] = cnt++;
    }
    n = cnt;
    vector<vector<int>> paths(n);
    vector<int> vv;
    for(auto &[t,u,v]:edges){
        vv.push_back(u);
        vv.push_back(v);
        u = mapping[u],v = mapping[v];
        if(t<=mid)paths[u].push_back(v);
    }
    //for(auto &i:vv)cerr<<i<<',';cerr<<endl;
    for(auto &i:vv)mapping[i] = -1;

    auto scc_id = SCC(n,paths);
    //for(auto
        [t,u,v]:edges)cerr<<t<<','<<u<<','<<v<<endl;
    //cerr<<endl;
    vector<tiii> vl,vr;
    for(auto &[t,u,v]:edges){
        if(scc_id[u] == scc_id[v]){
            ans[t] = min(ans[t],mid);
            vl.push_back(tiii(t,u,v));
        }
        else{
            u = scc_id[u],v = scc_id[v];
            vr.push_back(tiii(t,u,v));
        }
    }
    vector<tiii>().swap(edges);
    dc(l,mid,vl);
    dc(mid+1,r,vr);
    return;
}
void add_edge(int u,int v){
    all.push_back(tiii(all.size(),u,v));
}
vector<tiii> solve(){//[time,u,v]
    m = all.size();
    vector<tiii> ret(m);
    for(auto [t,u,v]:all)ret[t] = tiii(m,u,v);
    for(auto [t,u,v]:all)n = max({n,u,v});
    n++;
    ans = vector<int>(m,m);

```

```

    for(auto [t,u,v]:all){
        if(u == v)ans[t] = -1;
    }
    mapping = vector<int>(n,-1);
    dc(0,m,all);
    for(int i = 0;i<m;i++)get<0>(ret[i]) = ans[i];
    return ret;
}
IncrementalSCC(){
    ans.clear();
    n = m = 0;
}
}
#undef tiii
#undef pii
#undef fs
#undef sc
};

```

2.3 Block-Cut Tree [f44682]

```

struct BlockCutTree{
    //0-indexed
    //returns a forest if the graph is not connected
    vector<vector<int>> g;
    vector<vector<int>> groups;
    vector<vector<int>> tr;
    vector<int> idx,low,st;
    int cnt,gcnt;
    int n;
    RoundSquareTree<int> _n = 0){
        cnt = gcnt = 0;
        n = _n;
        g = vector<vector<int>>(n);
    }
    void add_edge(int a,int b){//adds bidirectional edges
        g[a].push_back(b);
        g[b].push_back(a);
    }
    void dfs(int now){
        idx[now] = low[now] = cnt++;
        st.push_back(now);
        for(auto nxt:g[now]){
            if(idx[nxt] == -1){
                dfs(nxt);
                low[now] = min(low[now],low[nxt]);
                if(low[nxt] == idx[now]){
                    int id = -1;
                    tr.push_back(vector<int>());
                    while(id != nxt){
                        id = st.back();st.pop_back();
                        groups[id].push_back(gcnt);
                        tr[id].push_back(gcnt+n);
                        tr[gcnt+n].push_back(id);
                    }
                    groups[now].push_back(gcnt);
                    tr[now].push_back(gcnt+n);
                    tr[gcnt+n].push_back(now);
                    gcnt++;
                }
            }
            else idx[now] = min(idx[now],idx[nxt]);
        }
        return;
    }
    vector<vector<int>> solve(){//
        //returns the tree (round vertices numbered [0,n))
        idx = low = vector<int>(n,-1);
        tr = vector<vector<int>>(n);
        for(int i = 0;i<n;i++){
            if(idx[i] == -1)dfs(i);
        }
        return tr;
    }
};

```

2.4 Euler Tour [a4ce3c]

```

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

struct EulerTour{
    //undirected graph,0-indexed,fails if doesn't exist
    //returns the order of edges
#define pii pair<int,int>
    vector<vector<pii>> g;
    vector<int> ptr;
    vector<bool> vis;

```

```

vector<int> re;
int n, ecnt;
void init(int _n){
    n = _n;
    ecnt = 0;
    g = vector<vector<pii>>(n);
    ptr = vector<int>(n);
}
void add_edge(int a, int b, int id = -1){
    if(id == -1) id = ecnt;
    g[a].push_back(pii(b, id));
    g[b].push_back(pii(a, id));
    ecnt++;
}
void dfs(int now){
    for(int &i = ptr[now]; i < g[now].size(); i++){
        auto [to, eid] = g[now][i];
        if(vis[eid]) continue;
        vis[eid] = true;
        dfs(to);
        re.push_back(eid);
    }
    return;
}
vector<int> solve(int s){
    re.clear();
    vis = vector<bool>(ecnt, 0);
    dfs(s);
    return re;
}
#undef pii
};

```

3 Data Structure

3.1 Li Chao Tree [565209]

```

//range add line get min
//can even be used of modifies isn't range modify
#define ll long long
const ll SZ = 8e6+10;
const ll inf = 3e18;
vector<ll> all;
struct Line{
    ll m, b;
    Line(ll mm = 0, ll bb = 0): m(mm), b(bb){}
    ll operator()(ll k){
        return m*k+b;
    }
};
struct LiChao{
#define ls now*2+1
#define rs now*2+2
#define mid ((l+r)>>1)
    Line seg[SZ];
    LiChao(){
        fill(seg, seg+SZ, Line(0, inf));
    }
    void modify(int now, int l, int r, int s, int e, Line v){
        if(l == r){
            if(seg[now](all[l]) > v(all[l])) swap(seg[now], v);
            return;
        }
        if(l >= s && e >= r){
            if(seg[now](all[mid]) > v(all[mid])) swap(seg[now], v);
            if(seg[now].m < v.m) modify(ls, l, mid, s, e, v);
            else modify(rs, mid+1, r, s, e, v);
        }
        else{
            if(mid >= s) modify(ls, l, mid, s, e, v);
            if(mid < e) modify(rs, mid+1, r, s, e, v);
        }
        return;
    }
    ll getval(int now, int l, int r, int p){
        if(l == r) return seg[now](all[p]);
        if(mid >= p) return
            min(seg[now](all[p]), getval(ls, l, mid, p));
        else return
            min(seg[now](all[p]), getval(rs, mid+1, r, p));
    }
    void add_line(int s, int e, Line v){
        modify(0, 0, all.size()-1, s, e, v);
        return;
    }
    ll getmin(int p){

```

```

        return getval(0, 0, all.size()-1, p);
    }
#undef ls
#undef rs
#undef mid
};
#undef ll long long

```

3.2 CHT [98f67f]

```

//reference : 8BCube
#define ll long long
// only works for integer coordinates!! maintain max

struct Line {
    mutable ll a, b, p;
    bool operator
        <(const Line &rhs) const { return a < rhs.a; }
    bool operator<(ll x) const { return p < x; }
};
struct CHT : multiset<Line, less<>> {
    static const ll kInf = 1e18;
    ll Div(ll a,
        ll b) { return a / b - ((a ^ b) < 0 && a % b); }
    bool isect(iterator x, iterator y) {
        if (y == end()) { x->p = kInf; return 0; }
        if (x
            ->a == y->a) x->p = x->b > y->b ? kInf : -kInf;
        else x->p = Div(y->b - x->b, x->a - y->a);
        return x->p >= y->p;
    }
    void addline(ll a, ll b) {
        auto z = insert({a, b, 0}), y = z++, x = y;
        while (isect(y, z)) z = erase(z);
        if (x != begin
            ()) && isect(--x, y)) isect(x, y = erase(y));
        while ((y = x) != begin
            ()) && (--x)->p >= y->p) isect(x, erase(y));
    }
    ll query(ll x) {
        auto l = *lower_bound(x);
        return l.a * x + l.b;
    }
};

```

3.3 Treap [ff4001]

```

#define ll long long
//range reverse range add range sum
//need to push before using the info on node
struct node{
    int pri;
    int pl, pr;
    ll sum, tag, val;
    int sz;
    int rev;
    node(){
        pl = pr = sum = tag = 0;
        sz = 0;
        rev = 0;
        pri = rand();
    }
};
const int SZ = 2e5+10;
struct Treap{
    node nd[SZ];
    int cnt = 0;
    Treap(){
        cnt = 0;
    }
    int newnode(){
        cnt++;
        nd[cnt].sz = 1;
        return cnt;
    }
    void pull(int now){
        if(!now) return;
        nd[now].sz = nd[nd[now].pr].sz + nd[nd[now].pl].sz + 1;
        ll ls = nd[nd[now].pl].sum + nd[nd[now].pl].tag * nd[nd[now].pl].sz;
        ll rs = nd[nd[now].pr].sum + nd[nd[now].pr].tag * nd[nd[now].pr].sz;
        nd[now].sum = nd[now].val + ls + rs;
        return;
    }
    void push(int now){

```

```

if(!now) return;
if(nd[now].rev){
    swap(nd[now].pl, nd[now].pr);
    if(nd[now].pl) nd[nd[now].pl].rev ^= 1;
    if(nd[now].pr) nd[nd[now].pr].rev ^= 1;
    nd[now].rev = 0;
}
int tl = nd[now].pl, tr = nd[now].pr;
nd[now].val += nd[now].tag;
if(tl) nd[tl].tag += nd[now].tag;
if(tr) nd[tr].tag += nd[now].tag;
nd[now].tag = 0;
pull(now);
}
int merge(int a, int b){
    if(!a) return b;
    if(!b) return a;
    if(nd[a].pri > nd[b].pri){
        push(a);
        nd[a].pr = merge(nd[a].pr, b);
        pull(a);
        return a;
    }
    else{
        push(b);
        nd[b].pl = merge(a, nd[b].pl);
        pull(b);
        return b;
    }
}
void split(int now, int &a, int &b, int tar){
    if(!now){
        a = b = 0;
        return;
    }
    push(now);
    if(nd[nd[now].pl].sz + 1 <= tar){
        a = now;
        split(nd[now].pl, nd[a].pr, b, tar - (nd[nd[now].pl].sz + 1));
    }
    else{
        b = now;
        split(nd[now].pl, a, nd[b].pl, tar);
    }
    pull(a);
    pull(b);
    return;
}
};
Treap T;

```

4 Geometry

4.1 Point [d6339e]

```

template<typename T = int>
struct Pt{
    T x, y;
    Pt(T xx = T(), T yy = T()): x(xx), y(yy){}
    Pt operator+(Pt b) const { return Pt(x+b.x, y+b.y); }
    Pt operator-(Pt b) const { return Pt(x-b.x, y-b.y); }
    T operator*(Pt b) const { return x*b.x + y*b.y; }
    T operator^(Pt b) const { return x*b.y - y*b.x; }
    T operator/(Pt b) const { return x*b.y - y*b.x; }
    bool operator
    <(Pt b) const { return x == b.x ? y < b.y : x < b.x; }

    friend int dir(Pt a, Pt b) { // returns sign(a ^ b)
        auto re = a ^ b;
        return re < 0 ? -1 : re > 0 ? 1 : 0;
    }
    friend bool onseg(Pt x, Pt s, Pt e){
        if(((e-x)^(s-x)) != 0) return false;
        else if((s-x)*(e-x) > 0) return false;
        return true;
    }
    friend int
    intersect(Pt s1, Pt e1, Pt s2, Pt e2) { // returns 0
        if doesn't intersect, 1 if intersect, 2 if on line
        if(onseg(s1, s2, e2) || onseg(e1, s2,
            e2) || onseg(s2, s1, e1) || onseg(e2, s1, e1)) return 2;
        if(dir(s1-s2, e2-s2)*dir(e1-s2, e2-s2) < 0 &&
            dir(s2-s1, e1-s1)*dir(e2-s1, e1-s1) < 0) return 1;
        return 0;
    }
};

```

```

};

```

4.2 Convex Hull [2a54da]

```

//needs Point.cpp
template<typename T = int>
struct ConvexHull { // returns in clockwise direction
    vector<Pt<T>> solve(vector<Pt<T>> v){
        sort(v.begin(), v.end());
        vector<Pt<T>> u, d;
        for(auto &i: v){
            while(u.size() > 1 && ((i-u.end()[1]-1])
                ^ (u.end()[2]-u.end()[1])) >= 0) u.pop_back();
            while(d.size() > 1 && ((i-d.end()[1]-1])
                ^ (d.end()[2]-d.end()[1])) <= 0) d.pop_back();
            u.push_back(i);
            d.push_back(i);
        }
        for(int i = 1; i+1 < d.size(); i++) u.push_back(d.end()[1-i]);
        return u;
    }
};

```

4.3 Minkowski sum [9db95e]

```

//needs Point template
template<typename T>
vector<Pt
    <T>> minkowski(vector<Pt<T>> va, vector<Pt<T>> vb){
    deque<Pt<T>> a, b;
    for(auto &i: va) a.push_back(i);
    for(auto &i: vb) b.push_back(i);
    Pt head = *min_element(a.begin(), a.end());
    while(a[0].x != head.x || a[0].y != head.y){
        a.push_back(a[0]);
        a.pop_front();
    }
    head = *min_element(b.begin(), b.end());
    while(b[0].x != head.x || b[0].y != head.y){
        b.push_back(b[0]);
        b.pop_front();
    }
    a.push_back(a[0]);
    b.push_back(b[0]);
    int p1 = 0, p2 = 0;
    vector<Pt<T>> re;
    while(p1 < a.size() && p2 < b.size()){
        //cerr<<a
        .size() << ', '<<b.size() << ": "<<p1 << ' '<<p2 << endl;
        int dir = 0;
        re.push_back(a[p1]+b[p2]);
        if(p1+1 == a.size()) dir = 1;
        else if(p2+1 == b.size()) dir = 0;
        else
            if(((a[p1+1]-a[p1])^(b[p2+1]-b[p2])) > 0) dir = 0;
        else dir = 1;
        if(dir == 0) p1++;
        else p2++;
    }
    return re;
}

```

5 String

5.1 Suffix Array (SAIS) [a683f1]

```

int SA[MXN * 2], H[MXN], RA[MXN];
namespace SAIS {
    bool _t[MXN * 2];
    int _s[MXN * 2], _c[MXN * 2], x[MXN], _p[MXN], _q[MXN * 2];
    void pre(int *sa, int *c, int n, int z){
        fill_n(sa, n, 0);
        copy_n(c, z, x);
    }
    void induce(int
        *sa, int *c, int *s, bool *t, int n, int z){
        copy_n(c, z - 1, x + 1);
        FOR(i, 0, n){
            if(sa[i] && !t[sa[i] - 1]){
                sa[x[sa[sa[i] - 1]]++] = sa[i] - 1;
            }
        }
        copy_n(c, z, x);
        for(int i = n - 1; i >= 0; i--){
            if(sa[i] && t[sa[i] - 1]){

```

```

        sa[--x[s[sa[i] - 1]]] = sa[i] - 1;
    }
}
void sais(int *s, int *sa, int
    *p, int *q, bool *t, int *c, int n, int z) {
    bool uniq = t[n - 1] = true;
    int nn = 0, nmzx =
        -1, *nsa = sa + n, *ns = s + n, last = -1;
    fill_n(c, z, 0);
    FOR(i, 0, n) uniq &= ++c[s[i]] < 2;
    partial_sum(c, c + z, c);
    if (uniq) {
        FOR(i, 0, n) sa[--c[s[i]]] = i;
        return;
    }
    for (int i = n - 2; i >= 0; i--) {
        t[i] = (s[i] ==
            s[i + 1] ? t[i + 1] : s[i] < s[i + 1]);
    }
    pre(sa, c, n, z);
    FOR(i, 1, n) {
        if (t[i] && !t[i - 1]) {
            sa[--x[s[i]]] = p[q[i] = nn++] = i;
        }
    }
    induce(sa, c, s, t, n, z);
    FOR(i, 0, n) {
        if (sa[i] && t[sa[i]] && !t[sa[i] - 1]) {
            bool neq = last < 0 || !equal(s + sa[
                i], s + p[q[sa[i]] + 1], s + last);
            ns[q[last = sa[i]]] = nmzx += neq;
        }
    }
    sais(ns, nsa,
        p + nn, q + n, t + n, c + z, nn, nmzx + 1);
    pre(sa, c, n, z);
    for (int i = nn - 1; i >= 0; i--) {
        sa[--x[s[p[nsa[i]]]]] = p[nsa[i]];
    }
    induce(sa, c, s, t, n, z);
}
void mkhei(int n) {
    for (int i = 0, j = 0; i < n; i++) {
        if (RA[i]) {
            for (; i + j < n
                && SA[RA[i] - 1] + j < n && _s[i +
                j] == _s[SA[RA[i] - 1] + j]; ++j);
            H[RA[i]] = j, j = max(0, j - 1);
        }
    }
}
void build(int *s, int n, int mxc) {
    copy_n(s, n, _s), _s[n] = 0;
    sais(_s, SA, _p, _q, _t, _c, n + 1, mxc);
    copy_n(SA + 1, n, SA);
    FOR(i, 0, n) RA[SA[i]] = i;
    mkhei(n);
    copy(H + 1, H + n, H);
}
}

```

5.2 AC automaton [c073c7]

```

#define FOR(i, j, k) for (int i = j, Z = k; i < Z; i++)
struct AC {
    int nc;
    char c[MXN];
    int pi[MXN], p[MXN], nxt[MXN][MXC];
    void init() {
        nc = 2;
        fill(nxt[0], nxt[0] + MXC, 1);
        fill(nxt[1], nxt[1] + MXC, -1);
    }
    int add_node(int par, char _c) {
        c[nc] = _c;
        p[nc] = par;
        fill(nxt[nc], nxt[nc] + MXC, -1);
        return nc++;
    }
    int push(string &s) {
        int now = 1;
        for (auto &i : s) {
            if (nxt[now][i - 'a'] == -1)
                nxt[now][i - 'a'] = add_node(now, i);
            now = nxt[now][i - 'a'];
        }
    }
}

```

```

    }
    return now;
}
void build() {
    queue<int> q;
    pi[1] = 0;
    FOR(i, 0, MXC) {
        if (nxt[1][
            i] == -1) nxt[1][i] = nxt[pi[1]][i];
        else q.push(nxt[1][i]);
    }
    while (q.size()) {
        int id = q.front();
        q.pop();
        pi[id] = nxt[pi[p[id]]][c[id] - 'a'];
        FOR(i, 0, MXC) {
            if (nxt[id][i]
                == -1) nxt[id][i] = nxt[pi[id]][i];
            else q.push(nxt[id][i]);
        }
    }
}
};

```

6 Math

6.1 module int [a4a56c]

```

#include <bits/stdc++.h>
using namespace std;
using ll = long long;

template <int mod = 998244353>
struct mint {
    int x;
    mint() : x(0) {}
    mint(int _x) : x((_x % mod + mod) % mod) {}
    operator int() const {
        return x;
    }
    template <typename T>
    mint operator+(T t) {
        mint o = mint(t);
        int y = x + o.x;
        y -= (y >= mod ? mod : 0);
        return mint(y);
    }
    template <typename T>
    mint operator+=(T t) {
        return (*this = operator+(t));
    }
    template <typename T>
    mint operator-(T t) {
        mint o = mint(t);
        int y = x - o.x;
        y += (y < 0 ? mod : 0);
        return mint(y);
    }
    template <typename T>
    mint operator-=(T t) {
        return (*this = operator-(t));
    }
    template <typename T>
    mint operator*(T t) {
        mint o = mint(t);
        return mint((ll) x + o.x % mod);
    }
    template <typename T>
    mint &operator*=(T t) {
        return (*this = operator*(t));
    }
    template <typename T>
    mint POW(T t) {
        int b = int(t);
        mint a(x), ans(1);
        while (b) {
            if (b & 1) ans *= a;
            b >>= 1;
            a *= a;
        }
        return ans;
    }
    template <typename T>
    mint inv() {
        return POW(mod - 2);
    }
    template <typename T>

```

```

    mint operator/(T t) {
        mint o = mint(t);
        return operator*(o.inv());
    }
    template <typename T>
    mint &operator/=(T t) {
        return (*this = operator/(t));
    }
};

```

6.2 FFT [263d47]

```

#define FOR(i, j, k) for (int i = j, Z = k; i < Z; i++)
template <typename T>
struct FFT {
    const T pi = acos(-1);
    complex<T> cis(T theta) {
        return complex<T>(cos(theta), sin(theta));
    }
    complex<T> OMEGA(int n, int k) {
        return cis(pi * 2 * k / n);
    }
    void apply(complex<T> *a, int N, bool inv) {
        auto REVERSE = [&](int x) -> int {
            int ans = 0;
            for (int i = 1; i < N; i <= 1) {
                ans <= 1;
                if (i & x) ans |= 1;
            }
            return ans;
        };
        FOR(i, 0, N) {
            int r = REVERSE(i);
            if (i < r) swap(a[i], a[r]);
        }
        for (int w = 1; w < N; w <= 1) {
            int omega_n = w << 1;
            for (int omega_k = 0; omega_k < w; omega_k++) {
                complex<T> omega = OMEGA(omega_n, (inv ? -1 : 1) * omega_k);
                for (int s = 0; s < N; s += omega_n) {
                    complex<T> &L = a[s + omega_k];
                    complex<T> &R = a[s + omega_k + w];
                    complex<T> l = L, r = omega * R;
                    L = l + r;
                    R = l - r;
                }
            }
        }
        if (inv) {
            FOR(i, 0, N) a[i] /= N;
        }
    }
};
#undef FOR

```

6.3 FWT [a168b9]

```

#include <bits/stdc++.h>
using namespace std;
#define fs first
#define sc second
#define FOR(i, j, k) for (int i = j, Z = k; i < Z; i++)
using ll = long long;
typedef pair<int, int> pii;

typedef pair<pii, pii> MAT;

template <int mod = 998244353>
struct mint {
    int x;
    mint() : x(0) {}
    mint(int _x) : x(_x) {}
    mint operator+(mint o) {
        int y = x + o.x;
        y -= (y >= mod ? mod : 0);
        return mint(y);
    }
    mint operator*(int y) {
        y += (y < 0 ? mod : 0);
        return mint((ll) x * y % mod);
    }
    mint operator*(mint o) {
        return mint((ll) x * o.x % mod);
    }
    mint inv() {

```

```

        int b = mod - 2, a = x;
        int ans = 1;
        while (b) {
            if (b & 1) ans = (ll) ans * a % mod;
            b >>= 1;
            a = (ll) a * a % mod;
        }
        return mint(ans);
    }
};

```

```

template <typename T = int>
struct FWT {
    enum FWT_TYPE {
        AND,
        OR,
        XOR
    };
    const MAT mat[3] = {
        {{1, 1}, {0, 1}},
        {{1, 0}, {1, 1}},
        {{1, 1}, {1, -1}}
    };
    const MAT tam[3] = {
        {{1, -1}, {0, 1}},
        {{1, 0}, {-1, 1}},
        {{1, 1}, {1, -1}}
    };
    FWT() {}
    void btf(T &L, T &R, MAT &m) {
        T l = L, r = R;
        L = l * m.fs.fs + r * m.fs.sc;
        R = l * m.sc.fs + r * m.sc.sc;
    }
    void apply(T *a, int n, bool inv, FWT_TYPE tp) {
        MAT m = (inv ? tam : mat)[tp];
        for (int w = 1; w < n; w <= 1) {
            FOR(i, 0, n) if (i & w) {
                btf(a[i - w], a[i], m);
            }
        }
        if (tp == FWT_TYPE::XOR && inv) {
            T n_ = T(n).inv();
            FOR(i, 0, n) a[i] = a[i] * n_;
        }
    }
};

```