

Contents

1	Setups	1	3.3	Treap	5
1.1	vimrc	1	3.4	quadrangle	6
1.2	pbds	1	3.5	Splay	6
1.3	terminal	1	3.6	Link-Cut Tree	7
1.4	debug	1			
1.5	template	1	4	Geometry	7
2	Graph	1	4.1	Point	7
2.1	Dominator Tree	1	4.2	Convex Hull	7
2.2	Incremental SCC	1	4.3	Minkowski sum	7
2.3	Block-Cut Tree	2			
2.4	Euler Tour	2	5	String	8
2.5	Dinic	3	5.1	KMP	8
2.6	Gomory-Hu Tree	3	5.2	Zalgorithm	8
2.7	Min-Cost-Max-Flow	3	5.3	manacher	8
2.8	KM	4	5.4	Suffix Array (SAIS)	8
2.9	Max Clique	4	5.5	AC automaton	8
3	Data Structure	5	6	Math	9
3.1	Li Chao Tree	5	6.1	FFT	9
3.2	Dynamic Convex Hull	5	6.2	FWT	9
			6.3	NTT	9
			6.4	Chinese Remainder Theo-	10
			6.5	Pollard Rho	10

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]=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){
        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);
    }
    return scc_id;
}
vector<int> mapping;
void dc(int l,int r,vector<tiii> &edges){
    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)mapping[i] = -1;

    auto scc_id = SCC(n,paths);
    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 [22e960]

```

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

struct EulerTour{
    //undirected graph,0-indexed,fails if doesn't exist
    //returns the order of edges
    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);
    reverse(re.begin(), re.end());
    return re;
}
};

```

2.5 Dinic [360712]

```

struct Dinic{//0-indexed
    struct E{
        int t, f, c;
        E(int tt
            = 0, int cc = 0, int ff = 0):t(tt), c(cc), f(ff){}
    };
    vector<vector<int>> paths;
    vector<int> ptr, lvl;
    vector<E> e;
    queue<int> q;
    Dinic(int _n = 0){
        paths = vector<vector<int>>(_n);
        ptr = lvl = vector<int>(_n);
    }
    void add_edge(int a, int b, int c, int d = 0){
        paths[a].push_back(e.size());
        e.push_back(E(b, c));
        paths[b].push_back(e.size());
        e.push_back(E(a, d));
    }
    bool bfs(int s, int t){
        fill(lvl.begin(), lvl.end(), -1);
        q.push(s);
        lvl[s] = 0;
        while(!q.empty()){
            auto now = q.front(); q.pop();
            for(auto &eid: paths[now]){
                if(e[eid].f == e[eid].c) continue;
                if(lvl[e[eid].t] == -1){
                    lvl[e[eid].t] = lvl[now] + 1;
                    q.push(e[eid].t);
                }
            }
        }
        return lvl[t] != -1;
    }
    int dfs(int now, int t, int flow){
        if(now == t) return flow;
        for(int &i = ptr[now]; i < paths[now].size(); i++){
            int eid = paths[now][i];
            if(e[eid].f == e[eid].c) continue;
            if(int re =
                dfs(e[eid].t, t, min(flow, e[eid].c - e[eid].f))){
                e[eid].f += re;
                e[eid^1].f -= re;
                return re;
            }
        }
        return 0;
    }
};

```

```

}
int flow(int s, int t){
    int ans = 0;
    while(bfs(s, t)){
        fill(ptr.begin(), ptr.end(), 0);
        while(auto re = dfs(s, t, INT_MAX)){
            ans += re;
        }
    }
    return ans;
}
bool inScut(int k){
    return lvl[k] != -1;
}
};

```

2.6 Gomory-Hu Tree [3ab29a]

```

struct GomoryHuTree{//0-indexed
#define pii pair<int, int>
#define tiii tuple<int, int, int>
    vector<tiii> edges;
    vector<vector<pii>> tr;
    vector<int> p;
    int n;
    GomoryHuTree(int _n = 0){
        n = _n;
        p = vector<int>(_n, 0);
        tr = vector<vector<pii>>(_n);
    }
    void add_edge(int a, int b, int c){
        edges.push_back(tiii(a, b, c));
    }
    vector<vector<pii>> make_tree(){
        fill(p.begin(), p.end(), 0);
        tr = vector<vector<pii>>(_n);
        for(int i = 1; i < p.size(); i++){
            Dinic din(n);
            for(auto &[a, b, w]: edges){
                din.add_edge(a, b, w);
            }
            int w = din.flow(i, p[i]);
            tr[i].push_back(pii(p[i], w));
            tr[p[i]].push_back(pii(i, w));
            for(int j = i + 1; j < n; j++){
                if(p[j] == p[i] && din.inScut(j)) p[j] = i;
            }
        }
        return tr;
    }
#undef pii
#undef tiii
};

```

2.7 Min-Cost-Max-Flow [80eed6]

```

#define T ll
const T inf = 1e12;
struct MCMF{//TC: O(VEF)
    struct E{
        int t, f;
        T c, w;
        E(int tt, T cap, T wei):t(tt), c(cap), w(wei), f(0){}
    };
    vector<E> e;
    vector<vector<int>> paths;
    vector<T> dis;
    vector<int> pre;
    vector<bool> inq;
    queue<int> q;
    int n;
    MCMF(int _n = 0){
        n = _n;
        paths = vector<vector<int>>(_n);
        e.clear();
        pre = vector<int>(_n);
        dis = vector<T>(_n);
        inq = vector<bool>(_n);
    }
    void add_edge
        (int a, int b, int c, int d){//from, to, cap, wei
        paths[a].push_back(e.size());
        e.push_back(E(b, c, d));
        paths[b].push_back(e.size());
        e.push_back(E(a, 0, -d));
    }
    bool SPFA(int s, int t){

```

```

fill(dis.begin(),dis.end(),inf);
fill(pre.begin(),pre.end(),-1);
dis[s] = 0;
q.push(s);inq[s] = true;
while(!q.empty()){
    auto now = q.front();q.pop();
    inq[now] = false;
    //assert(dis[now]>=0);
    for(auto &eid:paths[now]){
        if(e[eid].f == e[eid].c)continue;
        int nxt = e[eid].t;
        if(dis[nxt]>dis[now]+e[eid].w){
            pre[nxt] = eid;
            dis[nxt] = dis[now]+e[eid].w;
            if(!inq[nxt]){
                inq[nxt] = true;
                q.push(nxt);
            }
        }
    }
}
return dis[t] != inf;
}
T flow(int s,int
t,int cnt = INT_MAX){//cnt is the number of flows
T ans = 0;
while(cnt--&&SPFA(s,t)){
    ans += dis[t];
    int now = t;
    while(pre[now] != -1){
        int eid = pre[now];
        e[eid].f++;
        e[eid^1].f--;
        now = e[eid^1].t;
    }
}
return ans;
}
};
#undef T

```

2.8 KM [2bf044]

```

// Kuhn-Munkres :
Bipartite matching with "maximum" weight in  $O(n^3)$ 
struct KM{
    const static int
        M = 500; // modify maximum number of vertices
    int n;
    ll ans = 0;
    // 0-base
    vector
        <vector<ll>> w; // input weighted edges w[x][y]
    vector<int> match; // match[y] = x
    vector<ll> lx, ly, slack;
    bitset<M> visx, visy; // initialize with all zero

    // abbr
# define forx for(int x=0; x<n; x++)
# define fory for(int y=0; y<n; y++)
# define z match[y]

    bool dfs(int x){
        visx[x] = 1;
        fory{
            if(visy[y]) continue;
            ll d = lx[x]+ly[y]-w[x][y];
            if(!d){
                visy[y] = 1;
                if(z== -1 || (!visx[z] && dfs(z))){
                    z = x;
                    return 1;
                }
            }
            else if(d<slack[y]) slack[y] = d;
        }
        return 0;
    }

    bool augment(){
        fory if(!visy[y] && !slack[y]){
            visy[y] = 1;
            if(z== -1) return 1;
            else if(!visx[z] && dfs(z)){
                z = -1;
                return 1;
            }
        }
    }
}

```

```

}
return 0;
}

void relabel(){
    ll d = INT64_MAX;
    fory if(!visy[y]) d = min(d, slack[y]);
    forx if(visx[x]) lx[x] -= d;
    fory{
        if(visy[y]) ly[y] += d;
        else slack[y] -= d;
    }
}

KM(vector<vector<ll>>
&W): n(W.size()), w(W) { // input edges' weight
    //initialize
    slack.resize(n);
    match.assign(n, -1);
    lx.assign(n, INT64_MIN);
    ly.assign(n, 0);
    forx fory lx[x] = max(lx[x], w[x][y]);
    //matching
    forx{
        visx.reset();
        visy.reset();
        visx[x] = 1;
        fory slack[y] = lx[x]+ly[y]-w[x][y];
        while(!augment()) relabel();
        visx.reset();
        visy.reset();
        dfs(x);
    }
    //summing
    forx ans += lx[x];
    fory ans += ly[y];
}

# undef forx
# undef fory
# undef z
};

```

2.9 Max Clique [0de041]

```

constexpr size_t kN = 150; using bits = bitset<kN>;
#define _all(T) T.begin(),T.end()
struct MaxClique {
    bits G[kN], cs[kN];
    int ans, sol[kN], q, cur[kN], d[kN], n;
    void init(int _n) {
        n = _n;
        for (int i = 0; i < n; ++i) G[i].reset();
    }
    void
        add_edge(int u, int v) { G[u][v] = G[v][u] = 1; }
    void pre_dfs(vector<int> &v, int i, bits mask) {
        if (i < 4) {
            for (
                int x : v) d[x] = (int)(G[x] & mask).count();
            sort(_all(v), [&](int x, int y) {
                return d[x] > d[y]; });
        }
        vector<int> c(v.size());
        cs[1].reset(), cs[2].reset();
        int l = max(ans - q + 1, 1), r = 2, tp = 0, k;
        for (int p : v) {
            for (k = 1; (cs[k] & G[p]).any(); ++k);
            if (k >= r) cs[++r].reset();
            cs[k][p] = 1;
            if (k < l) v[tp++] = p;
        }
        for (k = l; k < r; ++k)
            for (auto p = cs[k]._Find_first();
                p < kN; p = cs[k]._Find_next(p))
                v[tp] = (int)p, c[tp] = k, ++tp;
        dfs(v, c, i + 1, mask);
    }
    void dfs(vector<int> &v, vector<int> &c,
        int i, bits mask) {
        while (!v.empty()) {
            int p = v.back(); v.pop_back(); mask[p] = 0;
            if (q + c.back() <= ans) return;
            cur[q++] = p;
            vector<int> nr;
            for (int x : v) if (G[p][x]) nr.push_back(x);
            if (!nr.empty()) pre_dfs(nr, i, mask & G[p]);
        }
    }
}

```

```

    else if (q > ans) ans = q, copy_n(cur, q, sol);
    c.pop_back(); --q;
}
}
int solve() {
    vector<int> v(n); iota(_all(v), 0);
    ans = q = 0; pre_dfs(v, 0, bits(string(n, '1')));
    return ans; // sol[0 ~ ans-1]
}
};

```

3 Data Structure

3.1 Li Chao Tree [565209]

```

//range add line get min
//can even be used if modifies aren'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 Dynamic Convex Hull [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].pl.sum + nd[nd[now].pl].tag * nd[nd[now].pl].sz;
        ll rs = nd[nd[now].pr].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[now].pl.sz+1<=tar){
        a = now;
        split(nd[now]
            ].pr,nd[a].pr,b,tar-(nd[now].pl.sz+1));
    }
    else{
        b = now;
        split(nd[now].pl,a,nd[b].pl,tar);
    }
    pull(a);
    pull(b);
    return;
}
};
Treap T;

```

3.4 quadrangle [1d61e6]

```

struct QUADRANGLE {
    struct TUPLE {
        int l, r, id;
        TUPLE() {}
        TUPLE(int _l,
            int _r, int _id) : l(_l), r(_r), id(_id) {}
    };
    int n, now;
    deque<TUPLE> dq;

    int calc_dp(int id, int i) {
        // ...
    }
    bool cmp(int cid, int pid, int i) {
        // ...
    }
    void init(int _n) {
        n = _n;
        now = 1;
        dq.clear();
    }
    void kill_head() {
        now++;
        if (dq
            .front().l == dq.front().r) dq.pop_front();
        else dq.front().l++;
    }
    void push(int id) {
        while (dq.size()) {
            TUPLE tl = dq.back();
            dq.pop_back();
            if (cmp(id, tl.id, tl.l)) {
                continue;
            }
            int l = tl.l, r = tl.r + 1;
            while (l + 1 < r) {
                int mid = (l + r) >> 1;
                (cmp(id, tl.id, mid) ? r : l) = mid;
            }
            dq.push_back(TUPLE(tl.l, l, tl.id));
            if (r <= n) dq.push_back(TUPLE(r, n, id));
            return;
        }
        dq.push_back(TUPLE(now, n, id));
    }
    int determine(int id) {
        return calc_dp(dq.front().id, id);
    }
};

```

3.5 Splay [977ab6]

```

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

```

```

#define ll long long
const int SZ = 2e5+10;
//1-indexed, 0 used for nullptr
//range reverse range sum
struct Splay{
#define ls ch[now][0]
#define rs ch[now][1]
    ll val[SZ];
    ll sum[SZ];
    int ch[SZ][2], fa[SZ], cnt, rev[SZ], sz[SZ];
    void pull(int now){
        if(!now) return;
        sum[now] = sum[ls]+sum[rs]+val[now];
        sz[now] = sz[ls]+sz[rs]+1;
        return;
    }
    void push(int now){
        if(!now) return;
        if(rev[now]){
            swap(ls,rs);
            rev[ls] ^= 1;
            rev[rs] ^= 1;
            rev[now] = 0;
        }
        pull(now);
        return;
    }
    Splay(){
        fill(sz+1,sz+SZ,1);
        return;
    }
    int newnode(){
        return ++cnt;
    }
    int dir(int now){//is ls or rs
        return ch[fa[now]][1] == now;
    }
    bool isroot(int k){//for LCT
        return !fa[k] || ch[fa[k]][dir(k)] != k;
    }
    void rot(int now){
        assert(now);
        assert(fa[now]);
        int p = fa[now];
        int g = fa[p];
        push(g);
        push(p);
        push(now);
        int d = dir(now);
        if(!isroot(p)) ch[g][dir(p)] = now;
        fa[ch[now][d^1]] = p;
        ch[p][d] = ch[now][d^1];
        fa[now] = g;
        fa[p] = now;
        ch[now][d^1] = p;
        pull(p);
        pull(now);
        return;
    }
    void splay(int now){
        if(!now) return;
        while(!isroot(now)){
            push(fa[fa[now]]);
            push(fa[now]);
            push(now);
            if(!isroot(fa[now])){
                if(dir(fa[now]) == dir(now)) rot(fa[now]);
                else rot(now);
            }
            rot(now);
        }
        push(now);
        return;
    }
    int get_sz(int now,int tar){
        push(now);
        while(now&&sz[ls]+1 != tar){
            if(sz[ls]>=tar) now = ls;
            else{
                tar -= sz[ls]+1;
                now = rs;
            }
        }
        push(now);
        return now;
    }
}

```

```

void merge(int a, int b){
    if(!a || !b) return;
    splay(a); splay(b);
    a = get_sz(a, sz[a]);
    b = get_sz(b, 1);
    splay(a); splay(b);
    ch[a][1] = b;
    fa[b] = a;
    pull(a);
    return;
}
pair<int, int> split(int a, int tar){
    splay(a);
    if(!tar) return make_pair(0, a);
    int b = get_sz(a, tar);
    splay(b);
    pair<int, int> re;
    re.first = b;
    re.second = ch[b][1];
    fa[ch[b][1]] = 0;
    ch[b][1] = 0;
    pull(b);
    return re;
}
#undef ls
#undef rs
};

```

3.6 Link-Cut Tree [085466]

```

#define ll long long
//needs splay
//vertex add paths sum link-cut
struct LCT{
    Splay sp;
    void access(int x){
        sp.splay(x);
        sp.ch[x][1] = 0;
        sp.pull(x);
        while(sp.fa[x]){
            int u = sp.fa[x];
            sp.splay(u);
            sp.push(u);
            sp.ch[u][1] = x;
            sp.pull(u);
            sp.splay(x);
        }
    }
    void makeroot(int x){
        access(x); sp.splay(x);
        sp.rev[x] ^= 1;
    }
    void link(int u, int v){
        makeroot(u);
        sp.splay(u);
        sp.fa[u] = v;
    }
    void cut(int u, int v){
        makeroot(u);
        access(v);
        sp.splay(v);
        int lc = sp.ch[v][0];
        sp.fa[lc] = 0;
        sp.ch[v][0] = 0;
        sp.pull(v);
    }
    ll path_sum(int u, int v){
        makeroot(u);
        access(v);
        sp.splay(v);
        return sp.sum[v];
    }
    void addval(int u, int val){
        sp.splay(u);
        sp.val[u] += val;
        sp.pull(u);
        return;
    }
    int find(int p){
        access(p);
        return sp.get_sz(p, 1);
    }
};

```

4 Geometry

4.1 Point [47044e]

```

template<typename T = int>
struct Pt{
    T x, y;
    Pt (T xx = (T)(0), T yy = (T)(0)): 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])
                ^ (u.end()[-2]-u.end()[-1])) >= 0) u.pop_back();
            while(d.size() > 1 && ((i-d.end())[-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 KMP [bb2a1b]

```

template <typename T>
struct KMP {
    void operator()(T a, int n, int *pi) {
        pi[0] = -1, pi[1] = 0;
        for (int i = 1; i < n; i++) {
            int j = pi[i];
            while (j >= 0 && a[i] != a[j]) j = pi[j];
            pi[i + 1] = j + 1;
        }
    }
};

```

5.2 Zalgorithm [6a03da]

```

template <typename T>
struct Z_alg {
    void operator()(T *a, int n, int *z) {
        fill(z, z + n + 1, 0);
        int l = 0;
        for (int i = 1; i <= n; i++) {
            if (i >= l + z[l]) {
                while (i + z[i] < n
                    && a[i + z[i]] == a[z[i]]) z[i]++;
                l = i;
                continue;
            }
            int i_ = i - l;
            if (i_ + z[i_] == z[l]) {
                z[i] = z[i_];
                while (i + z[i] < n
                    && a[i + z[i]] == a[z[i]]) z[i]++;
                l = i;
                continue;
            }
            z[i] = (
                i_ + z[i_] < z[l] ? z[i_] : z[l] - i_);
        }
    }
};

```

5.3 manacher [c7bcd5]

```

template <typename T>
struct MANACHER {
    void operator()(T a, int n, int *mn) {
        fill(mn, mn + n, 0);
        int l = 0;
        for (int i = 1; i < n; i++) {
            if (i > l + mn[l]) {
                while (i - mn[i] - 1 >= 0
                    && i + mn[i] + 1 < n && a[i - mn[i]]
                    - 1 == a[i + mn[i] + 1]) mn[i]++;
                l = i;
                continue;
            }
            int i_ = 2 * l - i;
            if (i_ - mn[i_] == l - mn[l]) {
                mn[i] = mn[i_];
                while (i - mn[i] - 1 >= 0
                    && i + mn[i] + 1 < n && a[i - mn[i]]
                    - 1 == a[i + mn[i] + 1]) mn[i]++;
                l = i;
                continue;
            }
            mn[i] = (i_ - mn[i_]
                > l - mn[l] ? mn[i_] : i_ - l + mn[l]);
        }
    }
};

```

5.4 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[s[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);
}

```

5.5 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 FFT [9ff2a9]

```

#define TYPE double
typedef complex<TYPE> cd;
#undef TYPE

struct FFT {
    const double pi = acos(-1);
    cd cis(double theta) {
        return cd(cos(theta), sin(theta));
    }
    cd OMEGA(int n, int k) {
        return cis(pi * 2 * k / n);
    }
    void operator()(cd *a, int N, bool inv) {
        auto REV = [&](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 = REV(i);
            if (i < r) swap(a[i], a[r]);
        }
        for (int w = 1; w < N; w <= 1) {
            int on = w < 1;
            for (int ok = 0; ok < w; ok++) {
                cd omega = OMEGA(on, (inv ? -1 : 1) * ok);
                for (int s = 0; s < N; s += on) {
                    cd &L = a[s + ok], &R = a[s + ok + w];
                    cd l = L, r = omega * R;
                    L = l + r;
                    R = l - r;
                }
            }
        }
        if (inv) {
            for (int i = 0; i < N; i++) a[i] /= N;
        }
    }
};
fft;

```

6.2 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*(mint 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_;
        }
    }
};

```

6.3 NTT [13fd5f]

```

#define FOR(i, j, k) for (int i = j, Z = k; i < Z; i++)

struct NTT {
    const static int LG = 20;
    int mod;
    int o[(1 << LG) + 1];
    int ADD(int a, int b) {

```

```

    // help yourself
}
int SUB(int a, int b) {
    // help yourself
}
int MUL(int a, int b) {
    // help yourself
}
int POW(int a, int b) {
    // help yourself
}
NTT(int g, int gap, int _mod) {
    mod = _mod;
    o[0] = 1;
    int pp = POW(g, gap);
    FOR(i,
        1, (1 << LG) + 1) o[i] = MUL(o[i - 1], pp);
}
void operator()(int *a, int n, bool inv) {
    auto REV = [&](int x) -> int {
        int ans = 0;
        for (int w = 1; w < n; w <= 1) {
            ans = (ans << 1) | (x & 1);
            x >>= 1;
        }
        return ans;
    };
    FOR(i, 0, n) {
        int j = REV(i);
        if (i < j) swap(a[i], a[j]);
    }
    for (int w = 1; w < n; w <= 1) {
        int owo = 1 << (LG - __lg(w) - 1), oid = 0;
        FOR(i, 0, w) {
            int omega
                = o[inv ? (1 << LG) - oid : oid];
            for (int s = 0; s < n; s += (w <= 1)) {
                int &L
                    = a[s + i], &R = a[s + w + i];
                int l = L, r = MUL(omega, R);
                L = ADD(l, r);
                R = SUB(l, r);
            }
            oid += owo;
        }
    }
    if (inv) {
        int x = POW(n, mod - 2);
        FOR(i, 0, n) a[i] = MUL(a[i], x);
    }
}
};

NTT ntt1(3, 952, 998244353);
NTT ntt2(3, 100, 104857601);
NTT ntt3(3, 160, 167772161);

```

6.4 Chinese Remainder Theorem [6fdd6f]

```

using lll = __int128_t;

struct ICRT {
    lll p1, p2, p3;
    lll c1, c2, c3;
    ICRT() {}
    ICRT(lll _p1,
        lll _p2, lll _p3) : p1(_p1), p2(_p2), p3(_p3) {
        auto POW = [&](lll a, lll b, lll mod) -> lll {
            lll ans = 1;
            while (b) {
                if (b & 1) ans = ans * a % mod;
                b >>= 1;
                a = a * a % mod;
            }
            return ans;
        };
        c1 = POW(p2 * p3 % p1, p1 - 2, p1) * p2 * p3;
        c2 = POW(p3 * p1 % p2, p2 - 2, p2) * p3 * p1;
        c3 = POW(p1 * p2 % p3, p3 - 2, p3) * p1 * p2;
    }
    lll operator()(int r1, int r2, int r3) {
        return (c1
            * r1 + c2 * r2 + c3 * r3) % (p1 * p2 * p3);
    }
};

ICRT icrt(998244353, 104857601, 167772161);

```

6.5 Pollard Rho [b24d9c]

```

//needs mad,mub,mul,pw with changable mod
//!!!use int128 for pw and mul

bool isprime(ll x) {
    if (x <= 2 || ~x & 1) return x == 2;
    auto witn = [&](ll a, int t) {
        for (ll a2; t-- && (a2 = mul(a, a, x)); a = a2)
            if (a2 == 1 && a != 1 && a != x - 1) return true;
        return a > 1;
    };
    int t = __builtin_ctzll(x-1); ll odd = (x-1) >> t;
    for (ll m:
        {2, 325,
         9375, 28178, 450775, 9780504, 1795265022})
        if (witn(pw(m % x, odd, x), t)) return false;
    return true;
}

ll pollard_rho(ll n) {
    static mt19937_64 rnd(120821011);
    if (!(n & 1)) return 2;
    ll y = 2, z = y, c = rnd() % n, p = 1, i = 0, t;
    auto f = [&](ll x) {
        return mad(mul(x, x, n), c, n);
    };
    do {
        p = mul(mub(z = f(f(z)), y = f(y), n), p, n);
        if (++i &= 63) if (i == (i & -i)) t = gcd(p, n);
    } while (t == 1);
    return t == n ? pollard_rho(n) : t;
}

vector<ll> factorize(ll k){
    if(k == 1) return {};
    else if(isprime(k)) return {k};
    else{
        vector<ll> re;
        function<void>(ll) dc = [&](ll k){
            if(isprime(k)){
                re.push_back(k);
                return;
            }
            ll x = pollard_rho(k);
            dc(x);dc(k/x);
        };
        dc(k);
        sort(re.begin(), re.end());
        return re;
    }
}

```