Notebook - Maratona de Programação

Tiago de Souza Fernandes

Sumário

1 Algoritmos

1.1 Meet-in-the-middle

```cpp
// Subsequence with the biggest sum%m value O(2^(n/2)
    *n)

int n, m, a[40];

void comb(int l, int r, vi &v){
    int sz = r-l+1;
    for(int i=0;i<(1<<sz);i++){
        int sum = 0;
        for(int j=0;j<sz;j++)
            if(i & (1<<j))
                sum = (sum + a[l+j])%m;
        v.pb(sum);
    }

    sort(v.begin(), v.end());
}

int merge(vi &x, vi &y){
    int k=y.size()-1, ans=0;
    for(auto v: x){
        while(k>0 and v+y[k]>=m)
            k--;
        ans = max(ans, v+y[k]);
    }
    return ans;
}


int main()
{sws;

    vi x, y;
    cin >> n >> m;

    for(int i=0;i<n;i++)
        cin >> a[i];

    comb(0, n/2, x);
    comb(n/2 + 1, n-1, y);
    cout << merge(x, y) << endl;

    return 0;
}
```

1.2 Iterative-BS

```cpp
int l=1, r=N;
int res=-1;

while(l<=r){
    int m = (l+r)/2;
    if(!ver(m)){
        l = m+1;
    }
    else{
        res = m;
        r = m-1;
    }
}
cout << res << endl;
```

2 Grafos

2.1 BFS-01

```cpp
vector<int> d(n, INF);
deque<int> q;

void bfs(int x){
    d[x] = 0;
    q.push_front(x);
    while(!q.empty()){
        int u = q.front();
        q.pop_front();
        for(auto e: grafo[u]){
            int v = edge.ff;
            int w = edge.ss;
            if(d[v] > d[u] + w){
                d[v] = d[u] + w;
                if(w == 1)
                    q.push_back(v);
                else
                    q.push_front(v);
            }
        }
    }
}
```

2.2 BFS

```cpp
queue<int> q;
vector<bool> used(n);
vi d(n), p(n);

void bfs(int x){
    q.push(x);
    used[x] = true;
    p[x] = -1;
    while(!q.empty()){
        int u = q.front();
        q.pop();
        for(int v: adj[u]) {
            if(!used[v]){
                used[v] = true;
                q.push(v);
                d[v] = d[u] + 1;
                p[v] = u;
            }
        }
    }
}

// Restore
if(!used[u])
    cout << "No path!";
else{
    vi path;
    for(int v = u; v != -1; v = p[v])
        path.push_back(v);
    reverse(path.begin(), path.end());
    cout << "Path: ";
    for (int v : path)
        cout << v << " ";
}
```

2.3 2SAT

```cpp
vector<int> g[MAX], gt[MAX], S; int vis[MAX], cor[MAX
    ];

int val(int n, bool tvalue) {
    if(tvalue) return 2*n;
    return 2*n +1;
}

void dfs(int u) {
    vis[u] = 1; for(int v : g[u]) if(!vis[v]) dfs(v);
    S.push_back(u);
}

void dfst(int u, int e) {
    cor[u] = e;
    for(int v : gt[u]) if(!cor[v]) dfst(v, e);
}

void kosaraju(int n) {
    for(int i = 0; i <= n; i++) if(!vis[i]) dfs(i);
    for(int i = 0; i <= n; i++) for(int j : g[i])
        gt[j].push_back(i);
    int e = 0; reverse(S.begin(), S.end());
    for(int u : S) if(!cor[u]) dfst(u, ++e);
}

// antes de chamar essa funcao, colocar as arestas do grafo
bool solve(int n, vi &res) {
    kosaraju(2*n); // MAX > 2*N
    vi r;

    forn(i, n) {
        int t = val(i, true), f = val(i, false);
        if(cor[t] == cor[f]) {
            return false;
        }
        else {
            if(cor[t] > cor[f])
                r.pb(1);
            else
                r.pb(0);
        }
    }
    swap(r, res);
    return true;
}
```

2.4 Hungarian

```
template<typename T> struct hungarian {
    int n, m;
    vector<vector<T>> a;
    vector<T> u, v;
    vector<int> p, way;
    T inf;

    hungarian(int n_, int m_) : n(n_), m(m_), u(m+1), v(m+1), p(m+1), way(m+1) {
        a = vector<vector<T>>(n, vector<T>(m));
        inf = numeric_limits<T>::max();
    }
    pair<T, vector<int>> assignment() {
        for (int i = 1; i <= n; i++) {
            p[0] = i;
            int j0 = 0;
            vector<T> minv(m+1, inf);
            vector<int> used(m+1, 0);
            do {
                used[j0] = true;
                int i0 = p[j0], j1 = -1;
                T delta = inf;
                for (int j = 1; j <= m; j++) if (!used[j]) {
                    T cur = a[i0-1][j-1] - u[i0] - v[j];
                    if (cur < minv[j]) minv[j] = cur, way[j] = j0;
                    if (minv[j] < delta) delta = minv[j], j1 = j;
                }
                for (int j = 0; j <= m; j++)
                    if (used[j]) u[p[j]] += delta, v[j] -= delta;
                    else minv[j] -= delta;
                j0 = j1;
            } while (p[j0] != 0);
            do {
                int j1 = way[j0];
                p[j0] = p[j1];
                j0 = j1;
            } while (j0);
        }
        vector<int> ans(m);
        for (int j = 1; j <= n; j++) ans[p[j]-1] = j-1;
        return make_pair(-v[0], ans);
    }
};
```

2.5 Find-bridges

```
int n;
vector<vi> adj(n+1, vi());

vector<bool> visited;
vi tin, low;
int timer;

void dfs(int v, int p=-1){
    visited[v] = true;
    tin[v] = low[v] = timer++;
    for (int to: adj[v]){
        if(to == p) continue;
        if(visited[to])
            low[v] = min(low[v], tin[to]);
        else{
            dfs(to, v);
            low[v] = min(low[v], low[to]);
            if(low[to] > tin[v])
                IS_BRIDGE(v, to);
        }
    }
}

void find_bridges(){
    timer = 0;
    visited.assign(n, false);
    tin.assign(n, -1);
    low.assign(n, -1);
    for(int i=0;i<n;i++)
        if(!visited[i])
            dfs(i);
}
```

2.6 HLD-Vertice

```
struct Hld {
    Segtree st;
    int n;
    vector<vi> g;
    vi pos, sz, peso, pai, h, v;
    int t;

    Hld(int n){
        this->n=n;
        st = Segtree(n);
        g.assign(n, vi());
        pos.assign(n, 0);sz.assign(n, 0);
        peso.assign(n, 0);pai.assign(n, 0);
        h.assign(n, 0);v.assign(n, 0);
    }

    void build_hld(int k, int p = -1, int f = 1){
        v[pos[k] = t++] = peso[k]; sz[k] = 1;
        for(auto &i: g[k]) if(i!=p){
            pai[i] = k;
            h[i] = (i==g[k][0] ? h[k]:i);
            build_hld(i, k, f); sz[k]+=sz[i];

            if(sz[i]>sz[g[k][0]] or g[k][0]==p) swap(i, g[k][0]);
        }
        if(p*f == -1) build_hld(h[k] = k, -1, t = 0);
    }
    void build(int root = 0){
        t = 0;
        build_hld(root);
        for(int i=0;i<n;i++) st.seg[i+n]=v[i];
        st.build();
    }
    ll query_path(int a, int b){
        if(pos[a]<pos[b]) swap(a, b);

        if(h[a]==h[b]) return st.query(pos[b], pos[a]);
        return st.query(pos[h[a]], pos[a]) + query_path(pai[h[a]], b);
    }
    void update_path(int a, int b, int x){
        if(pos[a]<pos[b]) swap(a, b);

        if(h[a]==h[b]) return (void)st.update(pos[b], pos[a], x);
        st.update(pos[h[a]], pos[a], x); update_path(pai[h[a]], b, x);
    }
    ll query_subtree(int a){
        return st.query(pos[a], pos[a]+sz[a]-1);
    }
    void update_subtree(int a, int x){
        st.update(pos[a], pos[a]+sz[a]-1, x);
    }
    int lca(int a, int b){
        if(pos[a]<pos[b]) swap(a, b);
        return (h[a]==h[b] ? b:lca(pai[h[a]], b));
    }
};
```

2.7 Kahn

```
vi g[MAX];
int in[MAX], cor[MAX];
void kahn(int n) {
    int label = 1;
    priority_queue<int, vector<int>, greater<int>> pq; // trocar por queue para O(n)
    for(int i = 1; i <= n; i++) {
        if(in[i] == 0) {
            pq.push(i);
        }
    }

    while(pq.size()) {
        int u = pq.top(); pq.pop();
        cor[u] = label++;
        for(auto prox : g[u]) {
            in[prox]--;
            if(in[prox] == 0) {
                pq.push(prox);
            }
        }
    }
}
```

2.8 Dijkstra

```
// Dijkstra - Shortest Path

vector<vii> g(MAX+1, vii());
vi d(MAX+1, INF);
priority_queue< pii, vii, greater<pii> > fila;

void dijkstra(int k){
    d[k]=0;
    fila.push({0, k});

    while(!fila.empty()){
        int w=fila.top().ff, u=fila.top().ss;
        fila.pop();
        if(w>d[u]) continue;

        for(auto [v, w]: g[u]){
            if(d[v]>d[u]+w){
                d[v]=d[u]+w;
                fila.push({d[v], v});
            }
        }
    }
}
```

2.9 LCA

```
template<typename T> struct rmq {
    vector<T> v;
    int n; static const int b = 30;
    vector<int> mask, t;

    int op(int x, int y) { return v[x] < v[y] ? x : y; }
    int msb(int x) { return __builtin_clz(1)-__builtin_clz(x); }
    rmq() {}
    rmq(const vector<T>& v_) : v(v_), n(v.size()), mask(n), t(n) {
        for (int i = 0, at = 0; i < n; mask[i++] = at |= 1) {
            at = (at<<1)&((1<<b)-1);
            while (at and op(i, i-msb(at&-at)) == i) at ^= at&-at;
        }
        for (int i = 0; i < n/b; i++) t[i] = b*i+b-1-msb(mask[b*i+b-1]);
        for (int j = 1; (1<<j) <= n/b; j++) for (int i = 0; i+(1<<j) <= n/b; i++)
            t[n/b*j+i] = op(t[n/b*(j-1)+i], t[n/b*(j-1)+i+(1<<(j-1))]);
    }
    int small(int r, int sz = b) { return r-msb(mask[r]&((1<<sz)-1)); }
    T query(int l, int r) {
        if (r-l+1 <= b) return small(r, r-l+1);
        int ans = op(small(l+b-1), small(r));
        int x = l/b+1, y = r/b-1;
        if (x <= y) {
            int j = msb(y-x+1);
            ans = op(ans, op(t[n/b*j+x], t[n/b*j+y-(1<<j)+1]));
        }
        return ans;
    }
};

namespace lca {
    vector<int> g[MAX];
    int v[2*MAX], pos[MAX], dep[2*MAX];
    int t;
    rmq<int> RMQ;

    void dfs(int i, int d = 0, int p = -1) {
```

```cpp
        v[t] = i, pos[i] = t, dep[t++] = d;
        for (int j : g[i]) if (j != p) {
            dfs(j, d+1, i);
            v[t] = i, dep[t++] = d;
        }
    }
    void build(int n, int root) {
        t = 0;
        dfs(root);
        RMQ = rmq<int>(vector<int>(dep, dep+2*n-1));
    }
    int lca(int a, int b) {
        a = pos[a], b = pos[b];
        return v[RMQ.query(min(a, b), max(a, b))];
    }
    int dist(int a, int b) {
        return dep[pos[a]] + dep[pos[b]] - 2*dep[pos[lca(a, b)]];
    }
}
```

2.10 HLD-Aresta

```cpp
struct Hld {
    Segtree st;
    int n;
    vector<vii> g;
    vi pos, sz, sobe, pai, h, v;
    int t;

    Hld(int n){
        this->n=n;
        st = Segtree(n);
        g.assign(n, vii());
        pos.assign(n, 0);sz.assign(n, 0);
        sobe.assign(n, 0);pai.assign(n, 0);
        h.assign(n, 0);v.assign(n, 0);
    }

    void build_hld(int k, int p = -1, int f = 1){
        v[pos[k] = t++] = sobe[k]; sz[k] = 1;
        for(auto &i: g[k]) if(i.ff != p){
            sobe[i.ff] = i.ss; pai[i.ff] = k;
            h[i.ff] = (i==g[k][0] ? h[k]:i.ff);
            build_hld(i.ff, k, f); sz[k]+=sz[i.ff];

            if(sz[i.ff]>sz[g[k][0].ff] or g[k][0].ff==p) swap(i, g[k][0]);
        }
        if(p*f == -1) build_hld(h[k] = k, -1, t = 0);
    }
    void build(int root = 0){
        t = 0;
        build_hld(root);
        for(int i=0;i<n;i++) st.seg[i+n]=v[i];
        st.build();
    }
    ll query_path(int a, int b){
        if(a==b) return 0;
        if(pos[a]<pos[b]) swap(a, b);

        if(h[a]==h[b]) return st.query(pos[b]+1, pos[a]);
        return st.query(pos[h[a]], pos[a]) + query_path(pai[h[a]], b);
    }
    void update_path(int a, int b, int x){
        if(a==b) return;
        if(pos[a]<pos[b]) swap(a, b);

        if(h[a]==h[b]) return (void)st.update(pos[b]+1, pos[a], x);
        st.update(pos[h[a]], pos[a], x); update_path(pai[h[a]], b, x);
    }
    ll query_subtree(int a){
        if(sz[a]==1) return 0;
        return st.query(pos[a]+1, pos[a]+sz[a]-1);
    }
    void update_subtree(int a, int x){
        if(sz[a]==1) return;
        st.update(pos[a]+1, pos[a]+sz[a]-1, x);
    }
    int lca(int a, int b){
        if(pos[a] < pos[b]) swap(a, b);
        return (h[a]==h[b] ? b:lca(pai[h[a]], b));
    }
};
```

2.11 Floyd-Warshall

```cpp
// Floyd Warshall

int dist[MAX][MAX];

void Floydwarshall()
{
    for(int k = 1;k <= n;k++)
        for(int i = 1;i <= n;i++)
            for(int j = 1;j <= n;j++)
                dist[i][j] = min(dist[i][j], dist[i][k] + dist[k][j]);
}
```

2.12 Kruskal

```cpp
// Uses DSU .join() and .find()
struct Edge {
    int u, v, weight;
    bool operator<(Edge const& other) {
        return weight < other.weight;
    }
};

int n;
DSU dsu(n);
vector<Edge> edges, result;
int cost = 0;

sort(edges.begin(), edges.end());

for(auto e : edges) {
    if (dsu.find(e.u) != dsu.find(e.v)) {
        cost += e.weight;
        result.push_back(e); // vector com as arestas da MST
        dsu.join(e.u, e.v);
    }
}
```

2.13 DFS

```cpp
void DFS(int u, int pai){
    for(auto v: grafo[u]) if(v!=pai){
        DFS(v, u);
    }
}
```

2.14 Topological-sort

```cpp
vector<vi> grafo(MAX, vi());
int grau[MAX]; // Quantas arestas chegam no indice i

```

```
vi topological_sort(int n){
    vi resp;
    for(int i=1;i<=n;i++)
        if(!grau[i])
            resp.push_back(i);

    int k=0;
    while(k < (int)resp.size()){
        int u = resp[k];
        k++;
        for(auto v: grafo[u]){
            grau[v]--;
            if(!grau[v])
                resp.pb(v);
        }
    }

    if((int)resp.size() < n)
        cout << "impossivel\n";

    return resp;
}
```

2.15 Kosaraju

```
int n;
vi g[MAX], gi[MAX]; // grafo invertido
int vis[MAX], comp[MAX]; // componente conexo de cada
     vertice
stack<int> S;

void dfs(int u){
    vis[u] = 1;
    for(auto v: g[u]) if(!vis[v]) dfs(v);
    S.push(u);
}

void scc(int u, int c){
    vis[u] = 1; comp[u] = c;
    for(auto v: gi[u]) if(!vis[v]) scc(v, c);
}

void kosaraju(){
    for(int i=0;i<n;i++) vis[i] = 0;
    for(int i=0;i<n;i++) if(!vis[i]) dfs(i);
    for(int i=0;i<n;i++) vis[i] = 0;
    while(S.size()){
        int u = S.top();
        S.pop();
        if(!vis[u]) scc(u, u);
    }
}
```

2.16 Centroid

```
int sz[MAX];
bool erased[MAX];
vi grafo[MAX];

void dfs(int u, int p=-1){
    sz[u] = 1;
    for(int v: grafo[u]) if(v!=p and !erased[v]){
        dfs(v, u);
        sz[u] += sz[v];
    }
}

int centroid(int u, int p=-1, int size=-1){
    if(size==-1) size = sz[u];
    for(int v: grafo[u])
        if(v!=p and !erased[v] and sz[v]>size/2)
            return centroid(v, u, size);
    return u;
}

pii centroids(int u=1){ // idx 1
    dfs(u);
    int c1=centroid(u), c2=c1;
    for(int v: grafo[c1]) if(2*sz[v]==sz[u]) c2=v;
    return {c1, c2};
}
```

2.17 Ford

```
const int N = 2000010;

struct Ford {
    struct Edge {
        int to, f, c;
    };

    int vis[N];
    vector<int> adj[N];
    vector<Edge> edges;
    int cur = 0;

    void addEdge(int a, int b, int cap, int rcap) {
        Edge e;
        e.to = b; e.c = cap; e.f = 0;
        edges.pb(e);
        adj[a].pb(cur++);

        e = Edge();
        e.to = a; e.c = rcap; e.f = 0;
        edges.pb(e);
        adj[b].pb(cur++);
    }

    int dfs(int s, int t, int f, int tempo) {
        if(s == t)
            return f;
        vis[s] = tempo;

        for(int e : adj[s]) {
            if(vis[edges[e].to] < tempo and (edges[e
    ].c - edges[e].f) > 0) {
                if(int a = dfs(edges[e].to, t, min(f,
     edges[e].c-edges[e].f) , tempo)) {
                    edges[e].f += a;
                    edges[e^1].f -= a;
                    return a;
                }
            }
        }
        return 0;
    }

    int flow(int s, int t) {
        int mflow = 0, tempo = 1;
        while(int a = dfs(s, t, INF, tempo)) {
            mflow += a;
            tempo++;
        }
        return mflow;
    }
};
```

2.18 Dinic

```
const int N = 300;

struct Dinic {
    struct Edge{
        int from, to; ll flow, cap;
    };
    vector<Edge> edge;

    vector<int> g[N];
    int ne = 0;
    int lvl[N], vis[N], pass;
    int qu[N], px[N], qt;

    ll run(int s, int sink, ll minE) {
        if(s == sink) return minE;

        ll ans = 0;

        for(; px[s] < (int)g[s].size(); px[s]++) {
            int e = g[s][ px[s] ];
            auto &v = edge[e], &rev = edge[e^1];
            if(lvl[v.to] != lvl[s]+1 || v.flow >= v.
    cap)
                continue;           // v.cap - v.flow
     < lim
            ll tmp = run(v.to, sink,min(minE, v.cap-v
    .flow));
            v.flow += tmp, rev.flow -= tmp;
            ans += tmp, minE -= tmp;
            if(minE == 0) break;
        }
        return ans;
    }
    bool bfs(int source, int sink) {
        qt = 0;
        qu[qt++] = source;
        lvl[source] = 1;
        vis[source] = ++pass;
        for(int i = 0; i < qt; i++) {
            int u = qu[i];
            px[u] = 0;
            if(u == sink) return true;
            for(auto& ed : g[u]) {
                auto v = edge[ed];
                if(v.flow >= v.cap || vis[v.to] ==
    pass)
                    continue; // v.cap - v.flow < lim
                vis[v.to] = pass;
                lvl[v.to] = lvl[u]+1;
                qu[qt++] = v.to;
            }
        }
        return false;
    }
    ll flow(int source, int sink) {
        reset_flow();
        ll ans = 0;
        //for(lim = (1LL << 62); lim >= 1; lim /= 2)
        while(bfs(source, sink))
            ans += run(source, sink, LLINF);
        return ans;
    }
    void addEdge(int u, int v, ll c, ll rc) {
        Edge e = {u, v, 0, c};
        edge.pb(e);
        g[u].push_back(ne++);

        e = {v, u, 0, rc};
        edge.pb(e);
        g[v].push_back(ne++);
    }
    void reset_flow() {
        for(int i = 0; i < ne; i++)
            edge[i].flow = 0;
        memset(lvl, 0, sizeof(lvl));
        memset(vis, 0, sizeof(vis));
        memset(qu, 0, sizeof(qu));
        memset(px, 0, sizeof(px));
            qt = 0; pass = 0;
        }
};
```

2.19 Prim

```
// Prim Algorithm
#define MAXN 10100
#define INFINITO 999999999

int n, m;
int distancia[MAXN];
int processado[MAXN];
vector<pii> vizinhos[MAXN];

int Prim()
{
    for(int i = 2;i <= n;i++) distancia[i] = INFINITO
;
    distancia[1] = 0;

    priority_queue< pii, vector<pii>, greater<pii> >
fila;
    fila.push( pii(distancia[1], 1) );

    while(1){
        int davez = -1;

        while(!fila.empty()){
            int atual = fila.top().second;
            fila.pop();

            if(!processado[atual]){
                davez = atual;
                break;
            }
        }

        if(davez == -1)
            break;

        processado[davez] = true;

        for(int i = 0;i < (int)vizinhos[davez].size()
;i++){
            int dist  = vizinhos[davez][i].first;
            int atual = vizinhos[davez][i].second;

            if( distancia[atual] > dist && !
processado[atual])
            {
                distancia[atual] = dist;
                fila.push( pii(distancia[atual],
atual) );
            }
        }
    }

    int custo_arvore = 0;
    for(int i = 1;i <= n;i++)
        custo_arvore += distancia[i];

    return custo_arvore;
}

int main(){

    cin >> n >> m;

    for(int i = 1;i <= m;i++){

        int x, y, tempo;
        cin >> x >> y >> tempo;

        vizinhos[x].pb( pii(tempo, y) );
        vizinhos[y].pb( pii(tempo, x) );
    }

    cout << Prim() << endl;

    return 0;
}
```

3 Geometria

3.1 NumIntersectionLine

```
int main()
{
    int lim = 1e6;
    Segtree st(lim+100);
    int n, m, y, x, l, r;
    cin >> n >> m;

    int open=-1, close=INF; // open -> check -> close
    vector< pair<int, pii> > sweep;

    ll ans = 0;
    for(int i=0;i<n;i++){ // horizontal
        cin >> y >> l >> r;
        sweep.pb({l, {open, y}});
        sweep.pb({r, {close, y}});
    }
    for(int i=0;i<m;i++){ // vertical
        cin >> x >> l >> r;
        sweep.pb({x, {l, r}});
    }
    sort(sweep.begin(), sweep.end());

    // set<int> on;
    for(auto s: sweep){
        if(s.ss.ff==open){
            st.update(s.ss.ss, 1);
            // on.insert(s.ss.ss);
        }
        else if(s.ss.ff==close){
            st.update(s.ss.ss, -1);
            // on.erase(s.ss.ss);
        }
        else{
            ans += st.query(s.ss.ff, s.ss.ss);
            // auto it1 = on.lower_bound(s.ss.ff);
            // auto it2 = on.upper_bound(s.ss.ss);
            // for(auto it = it1; it!=it2; it++){
            //     intersection -> (s.ff, it);
            // }
        }
    }

    cout << ans << endl;


    return 0;
}
```

3.2 Inside-polygon

```
bool insideT(point a, point b, point c, point e){
    int x = ccw(a, b, e);
    int y = ccw(b, c, e);
    int z = ccw(c, a, e);
    // if(!x or !y or !z) return false; // bordo
    return !((x==1 or y==1 or z==1) and (x==-1 or y
==-1 or z==-1));
}

bool inside(vp &vet, point e){ // ccw
    int l=2, r=(int)vet.size()-1;
    int res=r;
    while(l<r){
        int mid = (l+r)/2;
        if(ccw(vet[0], vet[mid], e)==1)
            l=mid+1;
        else{
            r=mid;
            res=mid;
        }
    }

    return insideT(vet[0], vet[res-1], vet[res], e);
}
```

3.3 Polygon-Diameter

```
double diameter(const vector<point> &p) {
    vector<point> h = convexHull(p);
    int m = h.size();
    if (m == 1)
        return 0;
    if (m == 2)
        return dist(h[0], h[1]);
    int k = 1;
    while (area(h[m - 1], h[0], h[(k + 1) % m]) >
area(h[m - 1], h[0], h[k]))
        ++k;
    double res = 0;
    for (int i = 0, j = k; i <= k && j < m; i++) {
        res = max(res, dist(h[i], h[j]));
        while (j < m && area(h[i], h[(i + 1) % m], h
[(j + 1) % m]) > area(h[i], h[(i + 1) % m], h[j])
) {
            res = max(res, dist(h[i], h[(j + 1) % m])
);
            ++j;
        }
    }
    return res;
}
```

3.4 MinDistPair

```
ll MinDistPair(vp &vet){
    int n = vet.size();
    sort(vet.begin(), vet.end());
    set<point> s;

    ll best_dist = LLINF;
    int j=0;
    for(int i=0;i<n;i++){
        int d = ceil(sqrt(best_dist));
        while(j<n and vet[i].x-vet[j].x >= d){
            s.erase(point(vet[j].y, vet[j].x));
            j++;
        }

        auto it1 = s.lower_bound({vet[i].y - d, vet[i
].x});
        auto it2 = s.upper_bound({vet[i].y + d, vet[i
].x});

        for(auto it=it1; it!=it2; it++){
            ll dx = vet[i].x - it->y;
            ll dy = vet[i].y - it->x;
            if(best_dist > dx*dx + dy*dy){
                best_dist = dx*dx + dy*dy;
                // vet[i] e inv(it)
            }
        }

        s.insert(point(vet[i].y, vet[i].x));
    }
    return best_dist;
}
```

3.5 Intersect-polygon

```
bool intersect(vector<point> A, vector<point> B) // Ordered ccw
{
    for(auto a: A)
        if(inside(B, a))
            return true;
    for(auto b: B)
        if(inside(A, b))
            return true;

    if(inside(B, center(A)))
        return true;

    return false;
}
```

3.6 Sort-by-Angle

```
int quarter(point a)
{
    if(a.x>0 and a.y>=0) return 0;
    if(a.x<=0 and a.y>0) return 1;
    if(a.x<0 and a.y<=0) return 2;
    return 3;
}

point c;
bool comp(point a, point b) //ccw
{
    a=a-c;b=b-c;
    int qa = quarter(a);
    int qb = quarter(b);
    if(qa==qb)
        return (a^b)>0;
    else
        return qa<qb;
}

c = center(A);
sort(A.begin(), A.end(), comp);
```

3.7 Convex-Hull

```
vp convex_hull(const vp &points)
{
    vp P(points);
    sort(P.begin(), P.end());
    vp L, U;
    for(auto p: P){
        while(L.size()>=2 and ccw(L[L.size()-2], L.back(), p)!=1)
            L.pop_back();
        L.push_back(p);
    }
    reverse(P.begin(), P.end());
    for(auto p: P){
        while(U.size()>=2 and ccw(U[U.size()-2], U.back(), p)!=1)
            U.pop_back();
        U.push_back(p);
    }
    L.pop_back();
    L.insert(L.end(), U.begin(), U.end()-1);
    return L;
}
```

3.8 Inter-Retangulos

```
bool doOverlap(point l1, point r1, point l2, point r2)
{
    if (l1.x>r2.x or l2.x>r1.x or l1.y<r2.y or l2.y<r1.y)
        return false;
    return true;
}
```

3.9 Half-Plane-Intersect

```
// Half plane intersect O(n3)
vp half_plane_intersect(vector<line> &v){
    vp ret;
    int n = v.size();
    for(int i=0; i<n; i++){
        for(int j=i+1; j<n; j++){
            point crs = inter(v[i], v[j]);
            if(crs.x == INF) continue;
            bool bad = 0;
            for(int k=0; k<n; k++)
                if(v[k].eval(crs) < -EPS){
                    bad = 1;
                    break;
                }

            if(!bad) ret.push_back(crs);
        }
    }
    return ret;
}
```

3.10 Tetrahedron-Distance3D

```
bool nulo(point a){
    return (eq(a.x, 0) and eq(a.y, 0) and eq(a.z, 0));
}

ld misto(point p1, point p2, point p3){
    return (p1^p2)*p3;
}

ld dist_pt_face(point p, vp v){
    assert(v.size()==3);

    point v1 = v[1]-v[0];
    point v2 = v[2]-v[0];
    point n = (v1^v2);

    for(int i=0;i<3;i++){
        point va = p-v[i];
        point vb = v[(i+1)%3]-v[i];
        point ve = vb^n;
        ld d = ve*v[i];
        //se ponto coplanar com um dos lados do prisma (va^vb eh nulo),
        //ele esta dentro do prisma (poderia desconsiderar pois distancia
        //vai ser a msm da distancia do ponto ao segmento)
        if(!nulo(va^vb) and (v[(i+2)%3]*ve>d) ^ (p*ve>d)) return LLINF;
    }

```

```cpp
    //se ponto for coplanar ao triangulo (e dentro do triangulo)
    //vai retornar zero corretamente
    return fabs(misto(p-v[0],v1,v2)/norm(n));
}

ld dist_pt_seg(point p, vp li){
    return norm((li[1]-li[0])^(p-li[0]))/norm(li[1]-li[0]);
}

ld dist_line(vp l1, vp l2){
    point n = (l1[1]-l1[0])^(l2[1]-l2[0]);
    if(nulo(n)) //retas paralelas - dist ponto a reta
        return dist_pt_seg(l2[0],l1);

    point o1o2 = l2[0]-l1[0];
    return fabs((o1o2*n)/norm(n));
}
// retas paralelas e intersecao nao nula
ld dist_seg(vp l1, vp l2){

    assert(l2.size()==2);
    assert(l1.size()==2);

    //pontos extremos do segmento
    ld ans = LLINF;
    for(int i=0;i<2;i++)
        for(int j=0;j<2;j++)
            ans = min(ans, norm(l1[i]-l2[j]));

    //verificando distancia de ponto extremo com ponto interno dos segs
    for(int t=0;t<2;t++){
        for(int i=0;i<2;i++){
            bool c=true;
            for(int k=0;k<2;k++){
                point va = l1[i]-l2[k];
                point vb = l2[!k]-l2[k];
                ld ang = atan2(norm((vb^va)), vb*va);
                if(ang>PI/2) c = false;
            }
            if(c)
                ans = min(ans,dist_pt_seg(l1[i],l2));
        }
        swap(l1,l2);
    }

    //ponto interno com ponto interno dos segmentos
    point v1 = l1[1]-l1[0], v2 = l2[1]-l2[0];
    point n = v1^v2;
    if(!nulo(n)){
        bool ok = true;
        for(int t=0;t<2;t++){
            point n2 = v2^n;
            point o1o2 = l2[0]-l1[0];
            ld escalar = (o1o2*n2)/(v1*n2);
            if(escalar<0 or escalar>1) ok = false;
            swap(l1,l2);
            swap(v1,v2);
        }
        if(ok) ans = min(ans,dist_line(l1,l2));
    }

    return ans;
}

ld ver(vector<vp> &vet){
    ld ans = LLINF;
    // vertice - face
    for(int k=0;k<2;k++)
        for(int pt=0;pt<4;pt++)
            for(int i=0;i<4;i++){
                vp v;
                for(int j=0;j<4;j++){
                    if(i!=j) v.pb(vet[!k][j]);
                }
                ans = min(ans, dist_pt_face(vet[k][pt], v));
            }

    // edge - edge
    for(int i1=0;i1<4;i1++)
        for(int j1=0;j1<i1;j1++)
            for(int i2=0;i2<4;i2++)
                for(int j2=0;j2<i2;j2++)
                    ans = min(ans, dist_seg({vet[0][i1], vet[0][j1]},
                                            {vet[1][i2], vet[1][j2]}));

    return ans;
}
```

3.11 3D

```cpp
// typedef int cod;
// bool eq(cod a, cod b){ return (a==b); }

#define vp vector<point>
typedef ld cod;
bool eq(cod a, cod b){ return fabs(a - b) <= EPS; }

struct point
{
    cod x, y, z;
    point(cod x=0, cod y=0, cod z=0): x(x), y(y), z(z){}

    point operator+(const point &o) const{
        return {x+o.x, y+o.y, z+o.z};
    }
    point operator-(const point &o) const{
        return {x-o.x, y-o.y, z-o.z};
    }
    point operator*(cod t) const{
        return {x*t, y*t, z*t};
    }
    point operator/(cod t) const{
        return {x/t, y/t, z/t};
    }
    bool operator==(const point &o) const{
        return eq(x, o.x) and eq(y, o.y) and eq(z, o.z);
    }
    cod operator*(const point &o) const{ // dot
        return x*o.x + y*o.y + z*o.z;
    }
    point operator^(const point &o) const{ // cross
        return point(y*o.z - z*o.y,
                     z*o.x - x*o.z,
                     x*o.y - y*o.x);
    }
};

ld dist(point a, point b){
    return sqrt((a-b)*(a-b));
}
bool nulo(point a){
    return (eq(a.x, 0) and eq(a.y, 0) and eq(a.z, 0));
}

ld norm(point a){ // Modulo
    return sqrt(a*a);
}
```

```
ld proj(point a, point b){ // a sobre b
    return (a*b)/norm(b);
}
ld angle(point a, point b){ // em radianos
    return acos((a*b) / norm(a) / norm(b));
}

cod triple(point a, point b, point c){
    return dot(a, b^c); // Area do paralelepipedo
}


struct plane{
    point p1, p2, p3;
    plane(point p1=0, point p2=0, point p3=0): p1(p1)
    , p2(p2), p3(p3){}

    point aux = (p1-p3)^(p2-p3);
    cod a = aux.x, b = aux.y, c = aux.z;
    cod d = -a*p1.x - b*p1.y - c*p1.z;
    // ax+by+cz+d = 0;
};

cod dist(plane pl, point p){
    return fabs(pl.a*p.x + pl.b*p.y + pl.c*p.z + pl.d
    ) / sqrt(pl.a*pl.a + pl.b*pl.b + pl.c*pl.c);
}

point rotate(point v, point k, ld theta){
    // Rotaciona o vetor v theta graus em torno do
    eixo k
    // theta *= PI/180; // graus
    return rotated = (v*cos(theta)) +
                     ((k^v)*sin(theta)) +
                     (k*(k*v))*(1-cos(theta));
}
```

3.12 Heron

```
ld heron(int a, int b, int c){
    ld s = (a+b+c)/2.0;
    return sqrtl(s*(s-a)*(s-b)*(s-c));
}

ld heron(int a, int b, int c, int d){
    ld s = (a+b+c+d)/2.0;
    return sqrtl((s-a)*(s-b)*(s-c)*(s-d));
}
```

3.13 Uniao-segmentos

```
int length_union(const vector<pii> &a){
    int n = a.size();
    vector<pair<int, bool>> x(n*2);
    for(int i = 0; i < n; i++){
        x[i*2] = {a[i].ff, false};
        x[i*2+1] = {a[i].ss, true};
    }

    sort(x.begin(), x.end());

    int result=0;
    int c=0;
    for(int i=0;i<2*n;i++){
        if(i and c and x[i].ff>x[i-1].ff)
            result += x[i].ff-x[i-1].ff;

        if(x[i].ss) c--;
        else c++;
    }
    return result;
}
```

3.14 Minkowski-Sum

```
vp mk(const vp &a,const vp &b){
    int i = 0, j = 0;
    for(int k = 0; k < (int)a.size(); k++)if(a[k]<a[i
    ])
        i = k;
    for(int k = 0; k < (int)b.size(); k++)if(b[k]<b[j
    ])
        j = k;

    vp c;
    c.reserve(a.size() + b.size());
    for(int k = 0; k < int(a.size()+b.size()); k++){
        point pt{a[i] + b[j]};
        if((int)c.size() >= 2 and !ccw(c[c.size()-2],
    c.back(), pt))
            c.pop_back();
        c.pb(pt);
        int q = i+1, w = j+1;
        if(q == int(a.size())) q = 0;
        if(w == int(b.size())) w = 0;
        if(ccw(c.back(), a[i]+b[w], a[q]+b[j]) < 0) i
    = q;
        else j = w;
    }

    if(!ccw(c[0], c[(int)c.size()-1], c[(int)c.size()
    -2]))
        c.pop_back();
    if(!ccw(c.back(), c[0], c[1])){
        c[0]=c.back();
        c.pop_back();
    }
    c.shrink_to_fit();

    return c;
}
```

3.15 Simetria-central

```
bool simetric(vector<point> &a){ //ordered
    int n = a.size();
    c = center(a);
    if(n&1) return false;
    for(int i=0;i<n/2;i++)
        if(!collinear(a[i], a[i+n/2], c))
            return false;
    return true;
}
```

3.16 Rotating-Callipers

```
int N;

int sum(int i, int x){
    if(i+x>N-1) return (i+x-N);
    return i+x;
}

ld rotating_callipers(vp &vet){
    N = vet.size();
    ld ans = 0;
    // 2 triangulos (p1, p3, p4) (p1, p2, p3);
    for(int i=0;i<N;i++){ // p1
        int p2 = sum(i, 1); // p2
        int p4 = sum(i, 3); // p4
        for(int j=sum(i, 2);j!=i;j=sum(j, 1)){ // p3
            if(j==p2) p2 = sum(p2, 1);
            while(sum(p2, 1)!=j and areaT(vet[p2],
    vet[i], vet[j]) < areaT(vet[sum(p2, 1)], vet[i],
    vet[j]))
                p2 = sum(p2, 1);
            while(sum(p4, 1)!=i and areaT(vet[p4],
    vet[i], vet[j]) < areaT(vet[sum(p4, 1)], vet[i],
    vet[j]))
                p4 = sum(p4, 1);

            ans = max(ans, area(vet[i], vet[p2], vet[
    j], vet[p4]));
        }
    }

    return ans;
}
```

3.17 2D

```
#define vp vector<point>

// typedef int cod;
// bool eq(cod a, cod b){ return (a==b); }
typedef ld cod;
bool eq(cod a, cod b){ return abs(a - b) <= EPS; }

struct point{
    cod x, y;
    int id;
    point(cod x=0, cod y=0): x(x), y(y){}


    point operator+(const point &o) const{
        return {x+o.x, y+o.y};
    }
    point operator-(const point &o) const{
        return {x-o.x, y-o.y};
    }
    point operator*(cod t) const{
        return {x*t, y*t};
    }
    point operator/(cod t) const{
        return {x/t, y/t};
    }
    cod operator*(const point &o) const{ // dot
        return x * o.x + y * o.y;
    }
    cod operator^(const point &o) const{ // cross
        return x * o.y - y * o.x;
    }
    bool operator<(const point &o) const{
        if(!eq(x, o.x)) return x < o.x;
        return y < o.y;
    }
    bool operator==(const point &o) const{
        return eq(x, o.x) and eq(y, o.y);
    }

};

ld norm(point a){ // Modulo
    return sqrt(a*a);
}
bool nulo(point a){
    return (eq(a.x, 0) and eq(a.y, 0));
}

int ccw(point a, point b, point e){ //-1=dir; 0=
    collinear; 1=esq;
    cod tmp = (b-a)^(e-a); // from a to b
    return (tmp > EPS) - (tmp < -EPS);
}
point rotccw(point p, ld a){
    // a = PI*a/180; // graus
    return point((p.x*cos(a)-p.y*sin(a)), (p.y*cos(a)
    +p.x*sin(a)));
}
point rot90cw(point a) { return point(a.y, -a.x); };
point rot90ccw(point a) { return point(-a.y, a.x); };

ld proj(point a, point b){ // a sobre b
    return a*b/norm(b);
}
ld angle(point a, point b){ // em radianos
    ld ang = a*b / norm(a) / norm(b);
    return acos(max(min(ang, (ld)1), (ld)-1));
}
ld angle_vec(point v){
    // return 180/PI*atan2(v.x, v.y); // graus
    return atan2(v.x, v.y);
}
ld order_angle(point a, point b){ // from a to b ccw
    (a in front of b)
    ld aux = angle(a,b)*180/PI;
    return ((a^b)<=0 ? aux:360-aux);
}
bool angle_less(point a1, point b1, point a2, point
    b2){ // ang(a1,b1) <= ang(a2,b2)
    point p1((a1*b1), abs((a1^b1)));
    point p2((a2*b2), abs((a2^b2)));
    return (p1^p2) <= 0;
}

ld area(vp &p){ // (points sorted)
    ld ret = 0;
    for(int i=2;i<(int)p.size();i++)
        ret += (p[i]-p[0])^(p[i-1]-p[0]);
    return abs(ret/2);
}
ld areaT(point &a, point &b, point &c){
    return abs((b-a)^(c-a))/2.0;
}

point center(vp &A){
    point c = point();
    int len = A.size();
    for(int i=0;i<len;i++)
        c=c+A[i];
    return c/len;
}

point forca_mod(point p, ld m){
    ld cm = norm(p);
    if(cm<EPS) return point();
    return point(p.x*m/cm,p.y*m/cm);
}


///////////
//  Line  //
///////////

struct line{
    point p1, p2;
    cod a, b, c; // ax+by+c = 0;
    // y-y1 = ((y2-y1)/(x2-x1))(x-x1)
    line(point p1=0, point p2=0): p1(p1), p2(p2){
        a = p1.y-p2.y;
        b = p2.x-p1.x;
        c = -(a*p1.x + b*p1.y);
    }
    line(cod a=0, cod b=0, cod c=0): a(a), b(b), c(c)
    {
        // Gera os pontos p1 p2 dados os coeficientes
        // isso aqui eh um lixo mas quebra um galho
    kkkkkk
        if(b==0){
            p1 = point(1, -c/a);
            p1 = point(0, -c/a);
        }else{
            p1 = point(1, (-c-a*1)/b);
            p2 = point(0, -c/b);
        }
    }

    cod eval(point p){
        return a*p.x+b*p.y+c;
    }
    bool inside(point p){
        return eq(eval(p), 0);
    }
    point normal(){
        return point(a, b);
    }

    bool inside_seg(point p){
        return (inside(p) and
                min(p1.x, p2.x)<=p.x and p.x<=max(p1.x, p2.x) and
                min(p1.y, p2.y)<=p.y and p.y<=max(p1.y, p2.y));
    }

};

vp inter_line(line l1, line l2){
    ld det = l1.a*l2.b - l1.b*l2.a;
    if(det==0) return {};
    ld x = (l1.b*l2.c - l1.c*l2.b)/det;
    ld y = (l1.c*l2.a - l1.a*l2.c)/det;
    return {point(x, y)};
}

vp inter_seg(line l1, line l2){
    vp ans = inter_line(l1, l2);
    if(ans.empty() or !l1.inside_seg(ans[0]) or !l2.inside_seg(ans[0]))
        return {};
    return ans;
}

ld dseg(point p, point a, point b){ // point - seg
    if(((p-a)*(b-a)) < EPS) return norm(p-a);
    if(((p-b)*(a-b)) < EPS) return norm(p-b);
    return abs((p-a)^(b-a))/norm(b-a);
}

ld dline(point p, line l){ // point - line
    return abs(l.eval(p))/sqrt(l.a*l.a + l.b*l.b);
}

line mediatrix(point a, point b){
    point d = (b-a)*2;
    return line(d.x, d.y, a*a - b*b);
}

line perpendicular(line l, point p){ // passes through p
    return line(l.b, -l.a, -l.b*p.x + l.a*p.y);
}


////////////
// Circle //
////////////

struct circle{
    point c; cod r;
    circle() : c(0, 0), r(0){}
    circle(const point o) : c(o), r(0){}
    circle(const point a, const point b){
        c = (a+b)/2;
        r = norm(a-c);
    }
    circle(const point a, const point b, const point cc){
        c = inter_line(mediatrix(a, b), mediatrix(b, cc));
        r = norm(a-c);
    }
    bool inside(const point &a) const{
        return norm(a - c) <= r;
    }
    pair<point, point> getTangentPoint(point p) {
        ld d1 = norm(p-c), theta = asin(r/d1);
        point p1 = rotccw(c-p,-theta);
        point p2 = rotccw(c-p,theta);
        p1 = p1*(sqrt(d1*d1-r*r)/d1)+p;
        p2 = p2*(sqrt(d1*d1-r*r)/d1)+p;
        return {p1,p2};
    }
};

// minimum circle cover O(n) amortizado
circle min_circle_cover(vector<point> v){
    random_shuffle(v.begin(), v.end());
    circle ans;
    int n = v.size();
    for(int i=0;i<n;i++) if(!ans.inside(v[i])){
        ans = circle(v[i]);
        for(int j=0;j<i;j++) if(!ans.inside(v[j])){
            ans = circle(v[i], v[j]);
            for(int k=0;k<j;k++) if(!ans.inside(v[k])){
                ans = circle(v[i], v[j], v[k]);
            }
        }
    }
    return ans;
}


circle incircle( point p1, point p2, point p3 ){
    ld m1=norm(p2-p3);
    ld m2=norm(p1-p3);
    ld m3=norm(p1-p2);
    point c = (p1*m1+p2*m2+p3*m3)*(1/(m1+m2+m3));
    ld s = 0.5*(m1+m2+m3);
    ld r = sqrt(s*(s-m1)*(s-m2)*(s-m3))/s;
    return circle(c, r);
}

circle circumcircle(point a, point b, point c) {
    circle ans;
    point u = point((b-a).y, -(b-a).x);
    point v = point((c-a).y, -(c-a).x);
    point n = (c-b)*0.5;
    ld t = (u^n)/(v^u);
    ans.c = ((a+c)*0.5) + (v*t);
    ans.r = norm(ans.c-a);
    return ans;
}

vp inter_circle_line(circle C, line L){
    point ab = L.p2 - L.p1, p = L.p1 + ab * ((C.c-L.p1)*(ab) / (ab*ab));
    ld s = (L.p2-L.p1)^(C.c-L.p1), h2 = C.r*C.r - s*s / (ab*ab);
    if (h2 < 0) return {};
    if (h2 == 0) return {p};
    point h = (ab/norm(ab)) * sqrt(h2);
    return {p - h, p + h};
}

vp inter_circle(circle C1, circle C2){
    if(C1.c == C2.c) { assert(C1.r != C2.r); return
    {}; }
    point vec = C2.c - C1.c;
    ld d2 = vec*vec, sum = C1.r+C2.r, dif = C1.r-C2.r
    ;
    ld p = (d2 + C1.r*C1.r - C2.r*C2.r)/(d2*2), h2 =
    C1.r*C1.r - p*p*d2;
    if (sum*sum < d2 or dif*dif > d2) return {};
    point mid = C1.c + vec*p, per = point(-vec.y, vec
    .x) * sqrt(max((ld)0, h2) / d2);
    if(eq(per.x, 0) and eq(per.y, 0)) return {mid};
    return {mid + per, mid - per};
}
```

4 ED

4.1 Trie

```
class Trie {
private:
   struct Node {
      map<char, Node*> children;
      int qt = 0;
      ll size = 0;
   };

   Node* root;

   void dfs(Node* cur) {
     ll sz = 1;

     for(auto prox : cur->children) {
         dfs(prox.second);
         sz += (prox.second)->size;
     }

     cur->size = sz;
   }

    void del(Node* cur, int dep, string &s) {
        if(dep >= 32)
            return;

        Node* prox = cur->children[s[dep]];
        prox->qt--;
        del(prox, dep+1, s);

        if(prox->qt == 0)
           cur->children.erase(s[dep]);
    }

public:
   Trie() {
      root = new Node();
      root->qt = 1;
   }

   void add(string s) {
      Node* cur = root;

      for(auto c : s) {
         if(cur->children.count(c) == 0) {
            cur->children[c] = new Node();
         }
         cur->children[c]->qt++;
         cur = cur->children[c];
      }
   }

   void del(string &s) {
       Node* cur = root;
       del(cur, 0, s);
   }

   void size() {
       this->dfs(root);
   }
};
```

4.2 Prefixsum2D

```
ll find_sum(vector<vi> &mat, int x1, int y1, int x2,
    int y2){
    // superior-esq(x1,y1) (x2,y2)inferior-dir
    return mat[x2][y2]-mat[x2][y1-1]-mat[x1-1][y2]+
    mat[x1-1][y1-1];
}

int main(){

    for(int i=1;i<=n;i++)
        for(int j=1;j<=n;j++)
            mat[i][j]+=mat[i-1][j]+mat[i][j-1]-mat[i
    -1][j-1];

}
```

4.3 Delta-Encoding

```
// Delta encoding

for(int i=0;i<q;i++){
    int l,r,x;
    cin >> l >> r >> x;
    delta[l] += x;
    delta[r+1] -= x;
}

int atual = 0;

for(int i=0;i<n;i++){
    atual += delta[i];
    v[i] += atual;
}
```

4.4 BIT-2D

```
// BIT 2D

int bit[MAX][MAX];

int sum(int x, int y)
{
    int resp=0;

    for(int i=x;i>0;i-=i&-i)
        for(int j=y;j>0;j-=j&-j)
            resp+=bit[i][j];

    return resp;
}

void update(int x, int y, int delta)
{
    for(int i=x;i<MAX;i+=i&-i)
        for(int j=y;j<MAX;j+=j&-j)
            bit[i][j]+=delta;
}

int query(int x1, y1, x2, y2)
{
    return sum(x2,y2) - sum(x2,y1) - sum(x1,y2) + sum
    (x1,y1);
}
```

4.5 BIT-bigger-k

```cpp
struct node{
    int pos, l, r, val;
    bool operator<(node &o){
        if(val==o.val) return l>o.l;
        return val>o.val;
    }
};

struct FT {
    vector<int> bit;  // indexado em 0
    int n;

    FT(int n) {
        this->n = n+1;
        bit.assign(n+1, 0);
    }

    int sum(int idx) {
        int ret = 0;
        for (; idx > 0; idx -= idx & -idx)
            ret += bit[idx];
        return ret;
    }

    int sum(int l, int r) {
        return sum(r) - sum(l - 1);
    }

    void add(int idx, int delta) {
        for (; idx < n; idx += idx & -idx)
            bit[idx] += delta;
    }
};

vi solveQuery(vi arr, vi ql, vi qr, vi qk){
    // indexing [l, r] in 1
    int n = arr.size();
    int q = qk.size();
    node a[n+q];

    for(int i=0;i<n;i++){
        a[i].val = arr[i];
        a[i].pos = a[i].l = 0;
        a[i].r = i+1;
    }

    for(int i=n;i<n+q;i++){
        a[i].pos = i+1-n;
        a[i].val = qk[i-n];
        a[i].l = ql[i-n];
        a[i].r = qr[i-n];
    }
    sort(a, a+n+q);

    FT ft(n);
    vi ans(q+1, 0);

    for(int i=0;i<n+q;i++){
        if(a[i].pos != 0)
            ans[a[i].pos] = ft.sum(a[i].l, a[i].r);
        else
            ft.add(a[i].r, 1);
    }
    return ans;
}
int main()
{
    vi arr = { 7, 3, 9, 13, 5, 4 };

    vi QueryL = { 1, 2 };
    vi QueryR = { 4, 6 };

    vi QueryK = { 6, 8 };

    solveQuery(arr, QueryL, QueryR, QueryK);

    return 0;
}
```

4.6 BIT

```cpp
struct FT {
    vi bit;  // indexado em 1
    int n;

    FT(int n) {
        this->n = n+1;
        bit.assign(n+2, 0);
    }

    int sum(int idx) {
        int ret = 0;
        for(++idx; idx > 0; idx -= idx & -idx)
            ret += bit[idx];
        return ret;
    }

    int sum(int l, int r) { // [l, r]
        return sum(r) - sum(l - 1);
    }

    void add(int idx, int delta) {
        for(++idx; idx < n; idx += idx & -idx)
            bit[idx] += delta;
    }
};
```

4.7 Minqueue

```cpp
struct MinQ {
    stack<pair<ll,ll>> in;
    stack<pair<ll,ll>> out;

    void add(ll val) {
        ll minimum = in.empty() ? val : min(val, in.
top().ss);
        in.push(mp(val, minimum));
    }

    ll pop() {
        if(out.empty()) {
            while(!in.empty()) {
                ll val = in.top().ff;
                in.pop();
                ll minimum = out.empty() ? val : min(
val, out.top().ss);
                out.push({val, minimum});
            }
        }
        ll res = out.top().ff;
        out.pop();
        return res;
    }

    ll minn() {
        ll minimum = LLINF;
        if(in.empty() || out.empty())
            minimum = in.empty() ? (ll)out.top().ss :
 (ll)in.top().ss;
        else
            minimum = min((ll)in.top().ss, (ll)out.
top().ss);

```

```
        return minimum;
    }

    ll size() {
        return in.size() + out.size();
    }
};
```

4.8 BIT-kth

```
struct FT {
    vector<int> bit;  // indexado em 1
    int n;

    FT(int n) {
        this->n = n + 1;
        bit.assign(n + 1, 0);
    }

    int kth(int x){
        int resp = 0;
        x--;
        for(int i=26;i>=0;i--){
            if(resp + (1<<i) >= n) continue;
            if(bit[resp + (1<<i)] <= x){
                x -= bit[resp + (1<<i)];
                resp += (1<<i);
            }
        }
        return resp + 1;
    }

    void upd(int pos, int val){
        for(int i = pos; i < n; i += (i&-i))
            bit[i] += val;
    }
};
```

4.9 Mergesorttree

```
struct ST { // indexado em 0, O(n * log^2(n) )
    int size;
    vector<vl> v;

    vl f(vl a, vl& b) {
        vl res = a;
        for(auto val : b) {
            res.pb(val);
        }
        sort(all(res));
        return res;
    }

    void init(int n) {
        size = 1;
        while(size < n) size *= 2;
        v.assign(2*size, vl());
    }

    void build(vector<ll>& a, int x, int lx, int rx)
    {
        if(rx-lx == 1) {
            if(lx < (int)a.size()) {
                v[x].pb(a[lx]);
            }
            return;
        }
        int m = (lx+rx)/2;
        build(a, 2*x +1, lx, m);
        build(a, 2*x +2, m, rx);
        v[x] = f(v[2*x +1], v[2*x + 2]);
    }

    void build(vector<ll>& a) {
        init(a.size());
        build(a, 0, 0, size);
    }

    ll greaterequal(int l, int r, int k, int x, int
    lx, int rx) {
        if(r <= lx or l >= rx) return 0;
        if(l <= lx && rx <= r) {
            auto it = lower_bound(all(v[x]), k);
            return (v[x].end() - it);
        }
        int m = (lx + rx)/2;
        ll s1 = greaterequal(l, r, k, 2*x +1, lx, m);
        ll s2 = greaterequal(l, r, k, 2*x +2, m, rx);

        return s1 +s2;
    }

    ll greaterequal(int l, int r, int k) {
        return greaterequal(l, r+1, k, 0, 0, size);
    }
};
```

4.10 Sparse-Table

```
int logv[MAX+1];
void make_log() {
    logv[1] = 0; // pre-computar tabela de log
    for (int i = 2; i <= MAX; i++)
        logv[i] = logv[i/2] + 1;
}
struct Sparse {
    int n;
    vector<vi> st;

    Sparse(vi& v) {
        n = v.size();
        int k = logv[n];
        st.assign(n+1, vi(k+1, 0));

        forn(i, n) {
            st[i][0] = v[i];
        }

        for(int j = 1; j <= k; j++) {
            for(int i = 0; i + (1 << j) <= n; i++) {
                st[i][j] = f(st[i][j-1], st[i + (1 <<
 (j-1))][j-1]);
            }
        }
    }

    int f(int a, int b) {
        return min(a, b);
    }

    int query(int l, int r) {
        int k = logv[r-l+1];
        return f(st[l][k], st[r - (1 << k) + 1][k]);
    }
};
```

4.11 Union-Find

```
struct DSU {
    int n;
    vi parent, size;

    DSU(int n) {
        this->n = n;
        parent.assign(n+1, 0);
        size.assign(n+1, 1);

        for(int i=0;i<=n;i++)
            parent[i] = i;
    }

    int find(int v) {
        if(v==parent[v])
            return v;
        return parent[v]=find(parent[v]);
    }

    void join(int a, int b) {
        a = find(a);
        b = find(b);
        if(a!=b) {
            if(size[a]<size[b])
                swap(a, b);

            parent[b]=a;
            size[a]+=size[b];
        }
    }
};
```

4.12 CHT

```
const ll is_query = -LLINF;
struct Line{
    ll m, b;
    mutable function<const Line*()> succ;
    bool operator<(const Line& rhs) const{
        if(rhs.b != is_query) return m < rhs.m;
        const Line* s = succ();
        if(!s) return 0;
        ll x = rhs.m;
        return b - s->b < (s->m - m) * x;
    }
};
struct Cht : public multiset<Line>{ // maintain max m*x+b
    bool bad(iterator y){
        auto z = next(y);
        if(y == begin()){
            if(z == end()) return 0;
            return y->m == z->m && y->b <= z->b;
        }
        auto x = prev(y);
        if(z == end()) return y->m == x->m && y->b <= x->b;
        return (ld)(x->b - y->b)*(z->m - y->m) >= (ld)(y->b - z->b)*(y->m - x->m);
    }
    void insert_line(ll m, ll b){ // min -> insert (-m,-b) -> -eval()
        auto y = insert({ m, b });
        y->succ = [=]{ return next(y) == end() ? 0 : &*next(y); };
        if(bad(y)){ erase(y); return; }
        while(next(y) != end() && bad(next(y))) erase(next(y));
        while(y != begin() && bad(prev(y))) erase(prev(y));
    }
    ll eval(ll x){
        auto l = *lower_bound((Line) { x, is_query });
        return l.m * x + l.b;
    }
};
```

4.13 Mo

```
const int BLK = 600; // tamanho do bloco, algo entre 500 e 700 eh nice

struct Query {
    int l, r, idx;
    Query(int l, int r, int idx) {
        this->l = l;
        this->r = r;
        this->idx = idx;
    }
    bool operator<(Query other) const {
        if(l/BLK != other.l/BLK)
            return l/BLK < other.l/BLK;
        return (l/BLK & 1) ? r < other.r : r > other.r;
    }
};

inline void add() {}
inline void remove() {} // implementar operacoes de acordo com o problema

vector<int> mo(vector<Query>& queries) {
    vector<int> res(queries.size());
    sort(queries.begin(), queries.end());
    resposta = 0;

    int l = 0, r = -1;
    for(Query q : queries) {
        while(l > q.l) {
                l--;
                add(l);
        }
        while(r < q.r) {
                r++;
                add(r);
        }
        while(l < q.l) {
                remove(l);
                l++;
        }
        while(r > q.r) {
                remove(r);
                r--;
        }
        res[q.idx] = resposta; // adicionar resposta de acordo com o problema
     }
     return res; // ordernar o vetor pelo indice e responder queries na ordem
}
```

5 DP

5.1 Mochila

```
int val[MAXN], peso[MAXN], dp[MAXN][MAXS];

int knapsack(int n, int m){ // n Objetos | Peso max
    for(int i=0;i<=n;i++){
       for(int j=0;j<=m;j++){
            if(i==0 or j==0)
                dp[i][j] = 0;
            else if(peso[i-1]<=j)
                dp[i][j] = max(val[i-1]+dp[i-1][j-peso[i-1]], dp[i-1][j]);
            else
                dp[i][j] = dp[i-1][j];
       }
```

```cpp
    }
    return dp[n][m];
}


// space optimized

int val[MAX], wt[MAX], dp[MAX];
int knapsack(int n, int W){
    for(int i=0; i < n; i++)
        for(int j=W; j>=wt[i]; j--)
            dp[j] = max(dp[j], val[i] + dp[j-wt[i]]);
    return dp[W];
}
```

5.2 Dp-digitos

```cpp
// dp de quantidade de numeros <= r com ate qt digitos diferentes de 0
ll dp(int idx, string& r, bool menor, int qt, vector<vector<vi>>& tab) {
    if(qt > 3) return 0;
    if(idx >= r.size()) {
        return 1;
    }
    if(tab[idx][menor][qt] != -1)
        return tab[idx][menor][qt];

    ll res = 0;
    for(int i = 0; i <= 9; i++) {
        if(menor or i <= r[idx]-'0') {
            res += dp(idx+1, r, menor or i < (r[idx]-'0') , qt+(i>0), tab);
        }
    }

    return tab[idx][menor][qt] = res;
}
```

5.3 Unbounded-Knapsack

```cpp
int w, n;
int c[MAX], v[MAX];

int unbounded_knapsack(){
    int dp[w+1];
    memset(dp, 0, sizeof dp);

    for(int i=0;i<=w;i++)
        for(int j=0;j<n;j++)
            if(c[j] <= i)
                dp[i] = max(dp[i], dp[i-c[j]] + v[j]);

    return dp[w];
}
```

5.4 Largest-KSubmatrix

```cpp
int n, m;
int a[MAX][MAX];
// Largest K such that exists a block K*K with equal numbers
int largestKSubmatrix(){
    int dp[n][m];
    memset(dp, 0, sizeof(dp));

    int result = 0;
    for(int i = 0 ; i < n ; i++){
        for(int j = 0 ; j < m ; j++){
            if(!i or !j)
                dp[i][j] = 1;
            else if(a[i][j] == a[i-1][j] and
                    a[i][j] == a[i][j-1] and
                    a[i][j] == a[i-1][j-1])
                dp[i][j] = min(min(dp[i-1][j], dp[i][j-1]),
                               dp[i-1][j-1]) + 1;
            else dp[i][j] = 1;

            result = max(result, dp[i][j]);
        }
    }

    return result;
}
```

5.5 LIS

```cpp
multiset<int> S;
for(int i=0;i<n;i++){
    auto it = S.upper_bound(vet[i]); // low for inc
    if(it != S.end())
        S.erase(it);
    S.insert(vet[i]);
}
// size of the lis
int ans = S.size();

/////////////////////////// see that later
// https://codeforces.com/blog/entry/13225?#comment-180208

vi LIS(const vi &elements){
    auto compare = [&](int x, int y) {
        return elements[x] < elements[y];
    };
    set< int, decltype(compare) > S(compare);

    vi previous( elements.size(), -1 );
    for(int i=0; i<int( elements.size() ); ++i){
        auto it = S.insert(i).first;
        if(it != S.begin())
            previous[i] = *prev(it);
        if(*it == i and next(it) != S.end())
            S.erase(next(it));
    }

    vi answer;
    answer.push_back( *S.rbegin() );
    while ( previous[answer.back()] != -1 )
        answer.push_back( previous[answer.back()] );
    reverse( answer.begin(), answer.end() );
    return answer;
}
```

5.6 Partition-Problem

```cpp
// Partition Problem DP O(n2)
bool findPartition(vi &arr){
    int sum = 0;
    int n = arr.size();

    for(int i=0;i<n;i++)
        sum += arr[i];

    if(sum&1) return false;

    bool part[sum/2+1][n+1];

    for(int i=0;i<=n;i++)
        part[0][i] = true;

    for(int i=1;i<=sum/2;i++)
```

```
        part[i][0] = false;

    for(int i=1;i<=sum/2;i++){
        for(int j=1;j<=n;j++){
            part[i][j] = part[i][j-1];
            if(i >= arr[j-1])
                part[i][j] |= part[i - arr[j-1]][j-1];
        }
    }
    return part[sum / 2][n];
}
```

6 Math

6.1 Totient

```
// phi(p^k) = (p^(k-1))*(p-1) com p primo
// O(sqrt(m))
ll phi(ll m){
    ll res = m;
    for(ll d=2;d*d<=m;d++){
        if(m % d == 0){
            res = (res/d)*(d-1);
            while(m%d == 0)
                m /= d;
        }
    }
    if(m > 1) {
        res /= m;
        res *= (m-1);
    }
    return res;
}

// modificacao do crivo, O(n*log(log(n)))
vl phi_to_n(ll n){
    vector<bool> isprime(n+1, true);
    vl tot(n+1);
    tot[0] = 0; tot[1] = 1;
    for(ll i=1;i<=n; i++){
        tot[i] = i;
    }

    for(ll p=2;p<=n;p++){
        if(isprime[p]){
            tot[p] = p-1;
            for(ll i=p+p;i<=n;i+=p){
                isprime[i] = false;
                tot[i] = (tot[i]/p)*(p-1);
            }
        }
    }
    return tot;
}
```

6.2 Double-gcd

```
ld gcdf(ld a, ld b){
    if(a<b) return gcdf(b, a);

    if(fabs(b)<EPS)
        return a;
    else
        return (gcdf(b, a - floor(a/b)*b));
}
```

6.3 Linear-Diophantine-Equation

```
// Linear Diophantine Equation
int gcd(int a, int b, int &x, int &y)
{
    if (a == 0)
    {
        x = 0; y = 1;
        return b;
    }
    int x1, y1;
    int d = gcd(b%a, a, x1, y1);
    x = y1 - (b / a) * x1;
    y = x1;
    return d;
}

bool find_any_solution(int a, int b, int c, int &x0,
    int &y0, int &g)
{
    g = gcd(abs(a), abs(b), x0, y0);
    if (c % g)
        return false;

    x0 *= c / g;
    y0 *= c / g;
    if (a < 0) x0 = -x0;
    if (b < 0) y0 = -y0;
    return true;
}

//  All solutions
//  x = x0 + k*b/g
//  y = y0 - k*a/g
```

6.4 Factorization-sqrt

```
// Factorization of a number in sqrt(n)

vi fact(ll n){
    vector<int> div;
    for(ll i=2;i*i<=n;i++)
        if(n%i==0){
            div.pb(i);
            while(n%i==0)
                n/=i;
        }
    if(n!=1) div.pb(n);
    return div;
}
```

6.5 FFT-simple

```
struct num{
    ld a {0.0}, b {0.0};
    num(){}
    num(ld na) : a{na}{}
    num(ld na, ld nb) : a{na}, b{nb} {}
    const num operator+(const num &c) const{
        return num(a + c.a, b + c.b);
    }
    const num operator-(const num &c) const{
        return num(a - c.a, b - c.b);
    }
    const num operator*(const num &c) const{
        return num(a*c.a - b*c.b, a*c.b + b*c.a);
    }
    const num operator/(const int &c) const{
         return num(a/c, b/c);
    }
};

void fft(vector<num> &a, bool invert){
    int n = a.size();
    for(int i=1,j=0;i<n;i++){
        int bit = n>>1;
        for(; j&bit; bit>>=1)
            j^=bit;
        j^=bit;
        if(i<j)
            swap(a[i], a[j]);
    }
    for(int len = 2; len <= n; len <<= 1){
        ld ang = 2 * PI / len * (invert ? -1 : 1);
        num wlen(cos(ang), sin(ang));
        for(int i=0;i<n;i+=len){
            num w(1);
            for (int j=0;j<len/2;j++){
                num u = a[i+j], v = a[i+j+len/2] * w;
                a[i+j] = u + v;
                a[i+j+len/2] = u - v;
                w = w * wlen;
            }
        }
    }
    if(invert)
      for(num &x: a)
          x = x/n;

}

vl multiply(vi const& a, vi const& b){
    vector<num> fa(a.begin(), a.end());
    vector<num> fb(b.begin(), b.end());
    int n = 1;
    while(n < int(a.size() + b.size()) )
        n <<= 1;
    fa.resize(n);
    fb.resize(n);
    fft(fa, false);
    fft(fb, false);
    for(int i=0;i<n;i++)
        fa[i] = fa[i]*fb[i];
    fft(fa, true);
    vl result(n);
    for(int i=0;i<n;i++)
        result[i] = round(fa[i].a);
    while(result.back()==0) result.pop_back();
    return result;
}
```

6.6 Mulmod

```
ll mulmod(ll a, ll b) {
    if(a == 0) {
        return 0LL;
    }
    if(a%2 == 0) {
        ll val = mulmod(a/2, b);
        return (val + val) % MOD;
    }
    else {
        ll val = mulmod((a-1)/2, b);
        val = (val + val) % MOD;
        return (val + b) % MOD;
    }
}
```

6.7 Lagrange-interpolation

```
// Lagrange's interpolation (n+1 points)
ld interpolate(vii d, ld x){
    ld y = 0;
    int n = d.size();
    for(int i=0;i<n;i++){
        ld yi = d[i].ss;
        for(int j=0;j<n;j++)
            if(j!=i)
                yi = yi*(x - d[j].ff)/(ld)(d[i].ff - d
    [j].ff);

        y += yi;
    }
    return y;
}

ld inv_interpolate(vii d, ld y){
    ld x = 0;
    int n = d.size();
    for(int i=0;i<n;i++){
        ld xi = d[i].ff;
        for(int j=0;j<n;j++)
            if(j!=i)
                xi = xi*(y - d[j].ss)/(ld)(d[i].ss -
    d[j].ss);

        x += xi;
    }
    return x;
}
```

6.8 Crt

```
tuple<ll, ll, ll> ext_gcd(ll a, ll b) {
    if (!a) return {b, 0, 1};
    auto [g, x, y] = ext_gcd(b%a, a);
    return {g, y - b/a*x, x};
}

struct crt {
    ll a, m;

    crt() : a(0), m(1) {}
    crt(ll a_, ll m_) : a(a_), m(m_) {}
    crt operator * (crt C) {
        auto [g, x, y] = ext_gcd(m, C.m);
        if ((a - C.a) % g) a = -1;
        if (a == -1 or C.a == -1) return crt(-1, 0);
        ll lcm = m/g*C.m;
        ll ans = a + (x*(C.a-a)/g % (C.m/g))*m;
        return crt((ans % lcm + lcm) % lcm, lcm);
    }
};
```

6.9 Miller-Habin

```
ll mul(ll a, ll b, ll m) {
    return (a*b-ll(a*(long double)b/m+0.5)*m+m)%m;
}

ll expo(ll a, ll b, ll m) {
    if (!b) return 1;
    ll ans = expo(mul(a, a, m), b/2, m);
    return b%2 ? mul(a, ans, m) : ans;
}

bool prime(ll n) {
    if (n < 2) return 0;
    if (n <= 3) return 1;
    if (n % 2 == 0) return 0;

    ll d = n - 1;
    int r = 0;
    while (d % 2 == 0) {
        r++;
        d /= 2;
    }

    // com esses primos, o teste funciona garantido
    para n <= 2^64
```

```cpp
    // funciona para n <= 3*10^24 com os primos ate 41
    for (int i : {2, 325, 9375, 28178, 450775, 9780504, 1795265022}) {
        if (i >= n) break;
        ll x = expo(i, d, n);
        if (x == 1 or x == n - 1) continue;

        bool deu = 1;
        for (int j = 0; j < r - 1; j++) {
            x = mul(x, x, n);
            if (x == n - 1) {
                deu = 0;
                break;
            }
        }
        if (deu) return 0;
    }
    return 1;
}
```

6.10 Inverso-Mult

```cpp
// gcd(a, m) = 1 para existir solucao
// ax + my = 1, ou a*x = 1 (mod m)
ll inv(ll a, ll m) { // com gcd
    ll x, y;
    gcd(a, m, x, y);
    return (((x % m) +m) %m);
}

ll inv(ll a, ll phim) { // com phi(m), se m for primo entao phi(m) = p-1
    ll e = phim-1;
    return fexp(a, e);
}
```

6.11 Pollard-Rho

```cpp
mt19937 rng((int) chrono::steady_clock::now().time_since_epoch().count());

ll uniform(ll l, ll r){
    uniform_int_distribution<ll> uid(l, r);
    return uid(rng);
}

ll mul(ll a, ll b, ll m) {
    ll ret = a*b - ll(a*(long double)b/m+0.5)*m;
    return ret < 0 ? ret+m : ret;
}

ll expo(ll a, ll b, ll m) {
    if (!b) return 1;
    ll ans = expo(mul(a, a, m), b/2, m);
    return b%2 ? mul(a, ans, m) : ans;
}

bool prime(ll n) {
    if (n < 2) return 0;
    if (n <= 3) return 1;
    if (n % 2 == 0) return 0;

    ll d = n - 1;
    int r = 0;
    while (d % 2 == 0) {
        r++;
        d /= 2;
    }

    for (int i : {2, 325, 9375, 28178, 450775, 9780504, 795265022}) {
        if (i >= n) break;
        ll x = expo(i, d, n);
        if (x == 1 or x == n - 1) continue;

        bool deu = 1;
        for (int j = 0; j < r - 1; j++) {
            x = mul(x, x, n);
            if (x == n - 1) {
                deu = 0;
                break;
            }
        }
        if (deu) return 0;
    }
    return 1;
}

ll rho(ll n) {
    if (n == 1 or prime(n)) return n;
    if (n % 2 == 0) return 2;

    while (1) {
        ll x = 2, y = 2, ciclo = 2, i = 0, d = 1;
        ll c = uniform(1, n-1);

        while (d == 1) {
            if (++i == ciclo) ciclo *= 2, y = x;
            x = (mul(x, x, n) + c) % n;

            if (x == y) break;

            d = __gcd(abs(x-y), n);
        }

        if (x != y) return d;
    }
}

void fact(ll n, vector<ll>& v) {
    if (n == 1) return;
    if (prime(n)) v.pb(n);
    else {
        ll d = rho(n);
        fact(d, v);
        fact(n / d, v);
    }
}
```

6.12 Verif-primo

```cpp
// Prime verification sqrt(N)

bool prime(ll x){
    if(x==2) return true;
    else if(x==1 or x%2==0) return false;
    for(ll i=3;i*i<=x;i+=2)
        if(x%i==0)
            return false;
    return true;
}
```

6.13 Mobius

```cpp
vi mobius(int n) {
    // g(n) = sum{f(d)} => f(n) = sum{mu(d)*g(n/d)}
    vi mu(n+1);
    mu[1] = 1; mu[0] = 0;
    for(int i = 1; i <= n; i++)
        for(int j = i + i; j <= n; j += i)
            mu[j] -= mu[i];

    return mu;
}
```

6.14 Exponenciacao-matriz

```cpp
struct Matrix {
    vector<vl> m;
    int r, c;

    Matrix(vector<vl> mat) {
        m = mat;
        r = mat.size();
        c = mat[0].size();
    }

    Matrix(int row, int col, bool ident=false) {
        r = row; c = col;
        m = vector<vl>(r, vl(c, 0));
        if(ident) {
            for(int i = 0; i < min(r, c); i++) {
                m[i][i] = 1;
            }
        }
    }

    Matrix operator*(const Matrix &o) const {
        assert(c == o.r); // garantir que da pra multiplicar
        vector<vl> res(r, vl(o.c, 0));

        for(int i = 0; i < r; i++) {
            for(int j = 0; j < o.c; j++) {
                for(int k = 0; k < c; k++) {
                    res[i][j] = (res[i][j] + m[i][k]*o.m[k][j]) % MOD;
                }
            }
        }

        return Matrix(res);
    }
};

Matrix fexp(Matrix b, int e, int n) {
    if(e == 0) return Matrix(n, n, true); // identidade
    Matrix res = fexp(b, e/2, n);
    res = (res * res);
    if(e%2) res = (res * b);

    return res;
}
```

6.15 Crivo

```cpp
// Sieve of Eratosthenes

vector<bool> primos(n+1, true);

primos[0]=primos[1]=false;

for(int i=2;i<=n;i++)
    if(primos[i])
        for(int j=i+i; j<=n; j+=i)
            primos[j]=false;
```

6.16 Bigmod

```cpp
ll mod(string a, ll p) {
    ll res = 0, b = 1;
    reverse(all(a));

    for(auto c : a) {
        ll tmp = (((ll)c-'0')*b) % p;
        res = (res + tmp) % p;

        b = (b * 10) % p;
    }

    return res;
}
```

6.17 Simpson's-formula

```cpp
inline ld simpson(ld fl, ld fr, ld fmid, ld l, ld r){
    return (fl+fr+4*fmid)*(r-l)/6;
}

ld rsimpson(ld slr, ld fl, ld fr, ld fmid, ld l, ld r)
{
    ld mid = (l+r)/2;
    ld fml = f((l+mid)/2), fmr = f((mid+r)/2);
    ld slm = simpson(fl,fmid,fml,l,mid);
    ld smr = simpson(fmid,fr,fmr,mid,r);
    if(fabsl(slr-slm-smr) < EPS) return slm+smr; // aprox. good enough
    return rsimpson(slm,fl,fmid,fml,l,mid)+rsimpson(smr,fmid,fr,fmr,mid,r);
}

ld integrate(ld l, ld r)
{
    ld mid = (l+r)/2;
    ld fl = f(l), fr = f(r);
    ld fmid = f(mid);
    return rsimpson(simpson(fl,fr,fmid,l,r),fl,fr,fmid,l,r);
}
```

6.18 FFT-tourist

```cpp
struct num{
    ld x, y;
    num() { x = y = 0; }
    num(ld x, ld y) : x(x), y(y) {}
};

inline num operator+(num a, num b) { return num(a.x + b.x, a.y + b.y); }
inline num operator-(num a, num b) { return num(a.x - b.x, a.y - b.y); }
inline num operator*(num a, num b) { return num(a.x * b.x - a.y * b.y, a.x * b.y + a.y * b.x); }
inline num conj(num a) { return num(a.x, -a.y); }

int base = 1;
vector<num> roots = {{0, 0}, {1, 0}};
vi rev = {0, 1};

void ensure_base(int nbase){
    if(nbase <= base)
        return;

    rev.resize(1 << nbase);
    for(int i = 0; i < (1 << nbase); i++)
        rev[i] = (rev[i >> 1] >> 1) + ((i & 1) << (nbase - 1));

    roots.resize(1 << nbase);

    while(base < nbase){
        ld angle = 2*PI / (1 << (base + 1));
        for(int i = 1 << (base - 1); i < (1 << base); i++){
            roots[i << 1] = roots[i];
            ld angle_i = angle * (2 * i + 1 - (1 << base));
            roots[(i << 1) + 1] = num(cos(angle_i), sin(angle_i));
        }
        base++;
    }
}

void fft(vector<num> &a, int n = -1){
    if(n == -1)
        n = a.size();

    assert((n & (n-1)) == 0);
    int zeros = __builtin_ctz(n);
    ensure_base(zeros);
    int shift = base - zeros;
    for(int i = 0; i < n; i++)
        if(i < (rev[i] >> shift))
            swap(a[i], a[rev[i] >> shift]);

    for(int k = 1; k < n; k <<= 1)
        for(int i = 0; i < n; i += 2 * k)
            for(int j = 0; j < k; j++){
                num z = a[i+j+k] * roots[j+k];
                a[i+j+k] = a[i+j] - z;
                a[i+j] = a[i+j] + z;
            }
}

vector<num> fa, fb;
vi multiply(vi &a, vi &b){
    int need = a.size() + b.size() - 1;
    int nbase = 0;
    while((1 << nbase) < need) nbase++;
    ensure_base(nbase);
    int sz = 1 << nbase;
    if(sz > (int) fa.size())
        fa.resize(sz);

    for(int i = 0; i < sz; i++){
        int x = (i < (int) a.size() ? a[i] : 0);
        int y = (i < (int) b.size() ? b[i] : 0);
        fa[i] = num(x, y);
    }
    fft(fa, sz);
    num r(0, -0.25 / sz);
    for(int i = 0; i <= (sz >> 1); i++){
        int j = (sz - i) & (sz - 1);
        num z = (fa[j] * fa[j] - conj(fa[i] * fa[i])) * r;
        if(i != j) {
            fa[j] = (fa[i] * fa[i] - conj(fa[j] * fa[j])) * r;
        }
        fa[i] = z;
    }
    fft(fa, sz);
    vi res(need);
    for(int i = 0; i < need; i++)
        res[i] = fa[i].x + 0.5;

    return res;
}


vi multiply_mod(vi &a, vi &b, int m, int eq = 0){
    int need = a.size() + b.size() - 1;
    int nbase = 0;
    while((1 << nbase) < need) nbase++;
    ensure_base(nbase);
    int sz = 1 << nbase;
    if(sz > (int) fa.size())
        fa.resize(sz);

    for(int i=0;i<(int)a.size();i++){
        int x = (a[i] % m + m) % m;
        fa[i] = num(x & ((1 << 15) - 1), x >> 15);
    }
    fill(fa.begin() + a.size(), fa.begin() + sz, num {0, 0});
    fft(fa, sz);
    if(sz > (int) fb.size())
        fb.resize(sz);
    if(eq)
        copy(fa.begin(), fa.begin() + sz, fb.begin());
    else{
        for(int i = 0; i < (int) b.size(); i++){
            int x = (b[i] % m + m) % m;
            fb[i] = num(x & ((1 << 15) - 1), x >> 15);
        }
        fill(fb.begin() + b.size(), fb.begin() + sz, num {0, 0});
        fft(fb, sz);
    }
    ld ratio = 0.25 / sz;
    num r2(0, -1);
    num r3(ratio, 0);
    num r4(0, -ratio);
    num r5(0, 1);
    for(int i=0;i<=(sz >> 1);i++) {
        int j = (sz - i) & (sz - 1);
        num a1 = (fa[i] + conj(fa[j]));
        num a2 = (fa[i] - conj(fa[j])) * r2;
        num b1 = (fb[i] + conj(fb[j])) * r3;
        num b2 = (fb[i] - conj(fb[j])) * r4;
        if(i != j){
            num c1 = (fa[j] + conj(fa[i]));
            num c2 = (fa[j] - conj(fa[i])) * r2;
            num d1 = (fb[j] + conj(fb[i])) * r3;
            num d2 = (fb[j] - conj(fb[i])) * r4;
            fa[i] = c1 * d1 + c2 * d2 * r5;
            fb[i] = c1 * d2 + c2 * d1;
        }
        fa[j] = a1 * b1 + a2 * b2 * r5;
        fb[j] = a1 * b2 + a2 * b1;
    }
    fft(fa, sz);
    fft(fb, sz);
    vi res(need);
    for(int i=0;i<need;i++){
        ll aa = fa[i].x + 0.5;
        ll bb = fb[i].x + 0.5;
        ll cc = fa[i].y + 0.5;
        res[i] = (aa + ((bb % m) << 15) + ((cc % m) << 30)) % m;
    }
    return res;
}




int main()
{sws;

    //FFT
    vi fx{1, 2, 3}; // 1+2x+3x^2
    vi gx{4, 5}; // 4+5x
    vi res;

    res = multiply(fx,gx); //4 + 13x + 22x^2 + 15x^3

    return 0;

}
```

6.19 Modular-Exponentiaion

```
// Modular exponentiaion - (b^e)%mod in O(log e)
ll fexp(ll b, ll e, ll mod){
    ll res = 1;
    b%=mod;
    while(e){
        if(e&1LL)
            res=(res*b)%mod;
        e=e>>1LL;
        b=(b*b)%mod;
    }
    return res;
}
```

6.20 Next-Permutation

```
vector<int> a = {1, 2, 3};
int n = a.size();
do{
    display(a, n);// 1,2,3; 1,3,2; 2,1,3; 3,1,2;
    2,3,1; 3,2,1;
}while(next_permutation(a.begin(), a.begin() + n));
```

6.21 Raiz-primitiva

```
ll fexp(ll b, ll e, ll mod) {
    if(e == 0) return 1LL;
    ll res = fexp(b, e/2LL, mod);
    res = (res*res)%mod;
    if(e%2LL)
        res = (res*b)%mod;

    return res%mod;
}

vl fatorar(ll n) { // fatora em primos
    vl fat;
    for(int i = 2; i*i <= n; i++) {
        if(n%i == 0) {
            fat.pb(i);
            while(n%i == 0)
                n /= i;
        }
    }
    return fat;
}

// O(log(n) ^ 2)
bool raiz_prim(ll a, ll mod, ll phi, vl fat) {
    if(__gcd(a, mod) != 1 or fexp(a, phi/2, mod) ==
    1) // phi de euler sempre eh PAR
        return false;

    for(auto f : fat) {
        if(fexp(a, phi/f, mod) == 1)
            return false;
    }

    return true;
}

// mods com raizes primitivas: 2, 4, p^k, 2*p^k, p eh
     primo impar, k inteiro --- O(n log^2(n))
ll achar_raiz(ll mod, ll phi) {
    if(mod == 2) return 1;
    vl fat, elementos;
    fat = fatorar(phi);

    for(ll i = 2; i <= mod-1; i++) {
        if(raiz_prim(i, mod, phi, fat))
            return i;
    }

    return -1; // retorna -1 se nao existe
}

vl todas_raizes(ll mod, ll phi, ll raiz) {
    vl raizes;
    if(raiz == -1) return raizes;
    ll r = raiz;
    for(ll i = 1; i <= phi-1; i++) {
        if(__gcd(i, phi) == 1) {
            raizes.pb(r);
        }
        r = (r * raiz) % mod;
    }

    return raizes;
}
```

6.22 Kamenetsky

```
// Number of digits in n! O(1)

#define Pi 3.14159265358979311599796346854
#define Eul 2.71828182845904509079559829842

long long findDigits(int n)
{
    double x;

    if (n < 0)
        return 0;
    if (n == 1)
        return 1;

    x = ((n * log10(n / euler) + log10(2 * Pi * n)
    /2.0));

    return floor(x) + 1;
}
```

7 Misc

7.1 Ordered-Set

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

#include <ext/pb_ds/detail/standard_policies.hpp>

using namespace __gnu_pbds; // or pb_ds;

template<typename T, typename B = null_type>
using ordered_set = tree<T, B, less<T>, rb_tree_tag,
    tree_order_statistics_node_update>;
// find_by_order / order_of_key
```

7.2 Bitwise

```
// Bitwise
    #pragma GCC target("popcnt")
    unsigned char a = 5, b = 9; // a = (00000101), b
    = (00001001)

    AND -          a&b   // The result is 00000001
    (1)
    OR -           a|b   // The result is 00001101
    (13)
    XOR -          a^b   // The result is 00001100
    (12)
```

```cpp
    NOT -          ~a    // The result is 11111010 (250)
    Left shift -   b<<1  // The result is 00010010 (18)
    Right shift -  b>>1  // The result is 00000100 (4)

    // Exchange two int variables

        a^=b;
        b^=a;
        a^=b;

    // Even or Odd

        (x & 1)? printf("Odd"): printf("Even");

    // Turn on the j-th bit

        int S = 34; //(100010)
        int j = 3;

        S = S | (1<<j);

    // Turn off the j-th bit

        int S = 42; //(101010)
        int j = 1;

        S &= ~(1<<j)

        S == 40 //(101000)

    // Check the j-th element

        int S = 42; //(101010)
        int j = 3;

        T = S & (1<<j); // T = 0

    // Least significant bit (lsb)

        int lsb(int x){ return x&-x; }

    // Exchange o j-th element

        S ^= (1<<j)

    // Position of the first bit on

        T = (S & (-S))
        T -> 4 bit ligado //(1000)

    // Most significant digit of N

        double K = log10(N);
        K = K - floor(K);
        int X = pow(10, K);

    // Number of digits in N

        X =floor(log10(N)) + 1;

    // Power of two

        bool isPowerOfTwo(int x){ return x && (!(x&(x-1))); }

    // Turn off the first bit 1
        m = m & (m-1);

    // Built-in functions

        // Number of bits 1
        __builtin_popcount()
        __builtin_popcountll()

        // Number of leading zeros
        __builtin_clz()
        __builtin_clzll()

        // Number of trailing zeros
        __builtin_ctz()
        __builtin_ctzll()

    // floor(log2(x))

        int flog2(int x){ return 32-1-__builtin_clz(x); }

        int flog2ll(ll x){ return 64-1-__builtin_clzll(x); }
```

7.3 All-Subsets

```cpp
vi a, subset;
vector<vi> subsets;

// Iterative
void search(int k){
    if(k==(int)a.size())
        subsets.pb(subset);
    else{
        search(k+1);
        subset.pb(a[k]);
        search(k+1);
        subset.pop_back();
    }
}
search(0);

// Binary
for(int b=0;b<(1<<n);b++){
    vi subset;
    for(int i=0;i<n;i++)
        if(b&(1<<i)) subset.pb(a[i]);
    subsets.pb(subset);
}
```

7.4 Trie-bits

```cpp
struct Trie{

    int trie[MAX][10];
    bool finish[MAX];
    int nxt = 1, len = 0;

    void add(string &s){
        int node = 0;
        for(auto c: s){
            if(trie[node][c-'0'] == 0){
                node = trie[node][c-'0'] = nxt;
                nxt++;
            }else
                node = trie[node][c-'0'];
        }
        if(!finish[node]){
            finish[node] = true;
            len++;
        }
    }

    bool find(string &s, bool remove){
        int idx = 0;
        for(auto c: s)
```

```
            if(trie[idx][c-'0'] == 0)
                return false;
            else
                idx = trie[idx][c-'0'];
        if(remove and finish[idx]){
            finish[idx]=false;
            len--;
        }
        return finish[idx];
    }

    bool find(string &s){
        return find(s, 0);
    }

    void del(string &s){
        find(s, 1);
    }

    string best_xor(string s){
        int idx = 0;
        string ans;
        for(auto c: s){
            char other='1'; if(c=='1') other='0';

            if(trie[idx][other-'0'] != 0){
                idx = trie[idx][other-'0'];
                if(other=='1') ans.pb('1');
                else ans.pb('0');
            }else{
                idx = trie[idx][c-'0'];
                if(c=='1') ans.pb('1');
                else ans.pb('0');
            }
        }

        return ans;
    }

};

string sbits(ll n){
    string ans;
    for(int i=0;i<64;i++)
        ans.pb(!!(n & 1LL<<i)+'0');
    return ans;
}
```

7.5 Template

```
#include <bits/stdc++.h>
#define ff first
#define ss second
#define ll long long
#define ld long double
#define pb push_back
#define eb emplace_back
#define mp make_pair
#define mt make_tuple
#define pii pair<int, int>
#define vi vector<int>
#define vl vector<ll>
#define vii vector<pii>
#define sws ios_base::sync_with_stdio(false);cin.tie(
    NULL);cout.tie(NULL);
#define endl '\n'
#define teto(a, b) ((a+b-1)/(b))
#define all(x) x.begin(), x.end()
#define forn(i, n) for(int i = 0; i < (int)n; i++)
#define forne(i, a, b) for(int i = a; i <= b; i++)
#define dbg(msg, var) cerr << msg << " " << var <<
    endl;

using namespace std;

const int MAX = 200010;
const int MOD = 1000000007;
const int INF = 1e8;
const ll LLINF = 0x3f3f3f3f3f3f3f3f;
const ld EPS = 1e-7;

// End Template //
```

7.6 Rand

```
mt19937 rng(chrono::steady_clock::now().
    time_since_epoch().count());
uniform_int_distribution<int> distribution(1,n);

num = distribution(rng); // num no range [1, n]
shuffle(vec.begin(), vec.end(), rng); // shuffle
```

7.7 Safe-Map

```
struct custom_hash {
    static uint64_t splitmix64(uint64_t x) {
        // http://xorshift.di.unimi.it/splitmix64.c
        x += 0x9e3779b97f4a7c15;
        x = (x ^ (x >> 30)) * 0xbf58476d1ce4e5b9;
        x = (x ^ (x >> 27)) * 0x94d049bb133111eb;
        return x ^ (x >> 31);
    }

    size_t operator()(uint64_t x) const {
        static const uint64_t FIXED_RANDOM = chrono::
    steady_clock::now().time_since_epoch().count();
        return splitmix64(x + FIXED_RANDOM);
    }
};

unordered_map<long long, int, custom_hash> safe_map;
```

8 Strings

8.1 Trie

```
struct Trie{

    int trie[MAX][26];
    bool finish[MAX];
    int nxt = 1, len = 0;

    void add(string &s){
        int node = 0;
        for(auto c: s){
            if(trie[node][c-'a'] == 0){
                node = trie[node][c-'a'] = nxt;
                nxt++;
            }else
                node = trie[node][c-'a'];
        }
        if(!finish[node]){
            finish[node] = true;
            len++;
        }
    }

    bool find(string &s, bool remove){
        int idx = 0;
        for(auto c: s)
            if(trie[idx][c-'a'] == 0)
                return false;
            else
                idx = trie[idx][c-'a'];
        if(remove and finish[idx]){
            finish[idx]=false;
            len--;
        }
        return finish[idx];
    }

    bool find(string &s){
        return find(s, 0);
    }

    void del(string &s){
        find(s, 1);
    }

};
```

8.2 KMP

```
vi pi(const string &s){
    int n=s.size();
    vi p(n);
    for(int i=1, j=0; i<n; i++){
        while(j>0 and s[i]!=s[j]) j=p[j-1];
        if(s[j]==s[i]) j++;
        p[i]=j;
    }
    return p;
}

vi kmp(const string &t, const string &s){
    vi p = pi(s+'$'), match;
    int n=t.size(), m=s.size();
    for(int i=0, j=0; i<n; i++){
        while(j>0 and t[i]!=s[j]) j=p[j-1];
        if(t[i]==s[j]) j++;
        if(j==m) match.pb(i-j+1);
    }
    return match;
}
```

8.3 Suffix-array

```
vi suffix_array(string s){
    s.pb('$');
    int n = s.size();

    vi p(n), c(n);
    vector< pair<char, int> > a(n);
    for(int i=0;i<n;i++) a[i] = {s[i], i};
    sort(a.begin(), a.end());

    for(int i=0;i<n;i++) p[i] = a[i].ss;
    c[p[0]]=0;
    for(int i=1;i<n;i++)
        c[p[i]] = c[p[i-1]] + (a[i].ff!=a[i-1].ff);

    int k=0;
    while((1<<k) < n){
        vector< pair<pii, int> > a(n);
        for(int i=0;i<n;i++)
            a[i] = {{c[i], c[(i+(1<<k))%n]}, i};
        sort(a.begin(), a.end());

        for(int i=0;i<n;i++) p[i] = a[i].ss;
        c[p[0]]=0;
        for(int i=1;i<n;i++)
            c[p[i]] = c[p[i-1]] + (a[i].ff!=a[i-1].ff
    );
        k++;
    }
    return p;
}
```

8.4 LCS

```
string LCSubStr(string X, string Y)
{
    int m = X.size();
    int n = Y.size();

    int result = 0, end;
    int len[2][n];
    int currRow = 0;

    for(int i=0;i<=m;i++){
        for(int j=0;j<=n;j++){
            if(i==0 || j==0)
                len[currRow][j] = 0;
            else if(X[i-1] == Y[j-1]){
                len[currRow][j] = len[1-currRow][j-1]
 + 1;
                if(len[currRow][j] > result){
                    result = len[currRow][j];
                    end = i - 1;
                }
            }
            else
                len[currRow][j] = 0;
        }

        currRow = 1 - currRow;
    }

    if(result==0)
        return string();

    return X.substr(end - result + 1, result);
}
```

8.5 LCSubseq

```
// Longest Common Subsequence
string lcs(string x, string y){
    int n = x.size(), m = y.size();
    vector<vi> dp(n+1, vi(m+1, 0));

    for(int i=0;i<=n;i++){
        for(int j=0;j<=m;j++){
            if(!i or !j)
                dp[i][j]=0;
            else if(x[i-1] == y[j-1])
                dp[i][j]=dp[i-1][j-1]+1;
            else
                dp[i][j]=max(dp[i-1][j], dp[i][j-1]);
        }
    }

    // int len = dp[n][m];
    string ans="";

    // recover string
    int i = n-1, j = m-1;
    while(i>=0 and j>=0){
        if(x[i] == y[j]){
            ans.pb(x[i]);
            i--; j--;
        }else if(dp[i][j+1]>dp[i+1][j])
            i--;
        else
            j--;
    }

    reverse(ans.begin(), ans.end());
```

```cpp

    return ans;
}
```

8.6 Edit-distance

```cpp
int edit_distance(int a, int b, string& s, string& t) {
    // indexado em 0, transforma s em t
    if(a == -1) return b+1;
    if(b == -1) return a+1;
    if(tab[a][b] != -1) return tab[a][b];

    int ins = INF, del = INF, mod = INF;
    ins = edit_distance(a-1, b, s, t) + 1;
    del = edit_distance(a, b-1, s, t) + 1;
    mod = edit_distance(a-1, b-1, s, t) + (s[a] != t[b]);

    return tab[a][b] = min(ins, min(del, mod));
}
```

8.7 Pal-int

```cpp
bool ehpalindromo(ll n) {
    if(n<0)
        return false;

    int divisor = 1;
    while(n/divisor >= 10)
        divisor *= 10;

    while(n != 0) {
        int leading = n / divisor;
        int trailing = n % 10;

        if(leading != trailing)
            return false;

        n = (n % divisor)/10;

        divisor = divisor/100;
    }

    return true;
}
```

8.8 Z-Func

```cpp
vi z_algo(const string &s)
{   // returns vector for each idx where a prefix of size i starts.
    int n = s.size();
    int L = 0, R = 0;
    vi z(n, 0);
    for(int i = 1; i < n; i++){
        if(i <= R)
            z[i] = min(z[i-L], R - i + 1);
        while(z[i]+i < n and s[ z[i]+i ] == s[ z[i] ])
            z[i]++;
        if(i+z[i]-1 > R){
            L = i;
            R = i + z[i] - 1;
        }
    }
    return z;
}
```

8.9 Hash

```cpp
struct Hash {
    vector<unordered_set<ll>> h;
    vector<ll> mods = {

    1000000009,1000000021,1000000033,1000000087,1000000093,

    1000000123,1000000181,1000000207,1000000223,1000000241,
    };
    ll p = 31;
    int num;

    Hash(int qt) {
        srand(time(0));
        num = qt;
        h.assign(num, unordered_set<ll>());
        random_shuffle(all(mods));
    }

    ll compute_hash(string const& s, ll p, ll m) {
        ll res = 0, p_pow = 1;

        for(char c : s) {
            res = ( res + (c-'a'+1) * p_pow) % m;
            p_pow = (p_pow * p) % m;
        }
        return res;
    }

    void add(string const& s) {
        forn(i, num) {
            ll value = compute_hash(s, p, mods[i]);
            h[i].insert(value);
        }
    }

    bool query(string const& s) {
        forn(i, num) {
            ll val = compute_hash(s, p, mods[i]);
            if(!h[i].count(val))
                return false;
        }
        return true;
    }
};
```

8.10 Manacher

```cpp
// O(n), d1 -> palindromo impar, d2 -> palindromo par (centro da direita)
void manacher(string &s, vi &d1, vi &d2) {
    int n = s.size();
    for(int i = 0, l = 0, r = -1; i < n; i++) {
        int k = (i > r) ? 1 : min(d1[l + r - i], r - i + 1);
        while(0 <= i - k && i + k < n && s[i - k] == s[i + k]) {
            k++;
        }
        d1[i] = k--;
        if(i + k > r) {
            l = i - k;
            r = i + k;
        }
    }

    for(int i = 0, l = 0, r = -1; i < n; i++) {
        int k = (i > r) ? 0 : min(d2[l + r - i + 1], r - i + 1);
        while(0 <= i - k - 1 && i + k < n && s[i - k - 1] == s[i + k]) {
            k++;
        }
        d2[i] = k--;
        if(i + k > r) {
            l = i - k - 1;
            r = i + k ;
        }
    }
}
```

8.11 Suffix-array-radix

```
void radix_sort(vector<pii>& rnk, vi& ind) {
    auto counting_sort = [](vector<pii>& rnk, vi& ind) {
        int n = ind.size(), maxx = -1;
        for(auto p : rnk) maxx = max(maxx, p.ff);

        vi cnt(maxx+1, 0), pos(maxx+1), ind_new(n);
        for(auto p : rnk) cnt[p.ff]++;
        pos[0] = 0;

        for(int i = 1; i <= maxx; i++) {
            pos[i] = pos[i-1] + cnt[i-1];
        }

        for(auto idx : ind) {
            int val = rnk[idx].ff;
            ind_new[pos[val]] = idx;
            pos[val]++;
        }

        swap(ind, ind_new);
    };

    for(int i = 0; i < (int)rnk.size(); i++) swap(rnk[i].ff, rnk[i].ss);
    counting_sort(rnk, ind);
    for(int i = 0; i < (int)rnk.size(); i++) swap(rnk[i].ff, rnk[i].ss);
    counting_sort(rnk, ind);
}

vi suffix_array(string s) {
    int n = s.size();
    vector<pii> rnk(n, mp(0, 0));
    vi ind(n);
    forn(i, n) {
        rnk[i].ff = (s[i] == '$') ? 0 : s[i]-'a'+1; // manter '$' como 0
        ind[i] = i;
    }

    for(int k = 1; k <= n; k = (k << 1)) {
        for(int i = 0; i < n; i++) {
            if(ind[i]+k >= n) {
                rnk[ind[i]].ss = 0;
            }
            else {
                rnk[ind[i]].ss = rnk[ind[i]+k].ff;
            }
        }
        radix_sort(rnk, ind); // sort(all(rnk), cmp) pra n*log(n), cmp com rnk[i] < rnk[j]

        vector<pii> tmp = rnk;
        tmp[ind[0]] = mp(1, 0); // rnk.ff comecar em 1 pois '$' eh o 0
        for(int i = 1; i < n; i++) {
            tmp[ind[i]].ff = tmp[ind[i-1]].ff;
            if(rnk[ind[i]] != rnk[ind[i-1]]) {
                tmp[ind[i]].ff++;
            }
        }
        swap(rnk, tmp);
    }
    return ind;
}

vi lcp_array(string s, vi sarray) {
    vi inv(s.size());
    for(int i = 0; i < (int)s.size(); i++) {
        inv[sarray[i]] = i;
    }
    vi lcp(s.size());
    int k = 0;
    for(int i = 0; i < (int)s.size()-1; i++) {
        int pi = inv[i];
        if(pi-1 < 0) continue;
        int j = sarray[pi-1];

        while(s[i+k] == s[j+k]) k++;
        lcp[pi] = k;
        k = max(k-1, 0);
    }

    return vi(lcp.begin()+1, lcp.end()); // LCP(i, j) = min(lcp[i], ..., lcp[j-1])
}
```