Contents

1 準備 **1**
- 1.1 Caps Lock を Control に変更 . . . 1
- 1.2 init.el . . . 1
- 1.3 tpl.cpp . . . 1
- 1.4 get input . . . 2
- 1.5 alias . . . 2

2 文字列 **2**
- 2.1 マッチング . . . 2
  - 2.1.1 複数文字列マッチング (Aho-Corasick 法) . . . 2
- 2.2 Suffix Array . . . 2
- 2.3 Z-algorithm . . . 2
- 2.4 回文長 (Manacher) . . . 3

3 グラフ **3**
- 3.1 強連結成分分解 . . . 3
  - 3.1.1 関節点 . . . 3
  - 3.1.2 橋 . . . 3
  - 3.1.3 強連結成分分解 . . . 3
  - 3.1.4 無向中国人郵便配達問題 . . . 4
  - 3.1.5 全点対間最短路 (Johnson) . . . 4
  - 3.1.6 無向グラフの全域最小カット . . . 4
- 3.2 フロー . . . 5
  - 3.2.1 最大流 . . . 5
  - 3.2.2 二部マッチング . . . 5
  - 3.2.3 一般グラフの最大マッチング . . . 5
  - 3.2.4 最小費用流 . . . 6
  - 3.2.5 Gomory-Hu 木 . . . 6
- 3.3 木 . . . 7
  - 3.3.1 木の直径: double sweep . . . 7
  - 3.3.2 最小全域木 . . . 7
  - 3.3.3 最小全域有向木 . . . 7
  - 3.3.4 最小シュタイナー木 . . . 8
  - 3.3.5 木の同型性判定 . . . 8
  - 3.3.6 HL 分解 . . . 8
  - 3.3.7 重心分解 . . . 9
- 3.4 彩色数 . . . 9
  - 3.4.1 包除原理 . . . 9
  - 3.4.2 極大独立集合 . . . 10

4 数学 **10**
- 4.1 整数 . . . 10
  - 4.1.1 剰余 . . . 10
  - 4.1.2 離散対数 . . . 10
  - 4.1.3 カタラン数 . . . 11
  - 4.1.4 乱数 (xor shift) . . . 11
  - 4.1.5 確率的素数判定 (Miller-Rabin 法) . . . 11
- 4.2 多項式 . . . 11
  - 4.2.1 FFT(complex) . . . 11
  - 4.2.2 FFT(modulo) . . . 11
  - 4.2.3 積 (FMT) . . . 12
  - 4.2.4 逆元 (FMT) . . . 12
  - 4.2.5 平方根 (FMT) . . . 12
- 4.3 行列 . . . 12
  - 4.3.1 線形方程式の解 (Givens 消去法) . . . 13
- 4.4 割り当て問題 . . . 13
  - 4.4.1 ハンガリアン法 . . . 13

5 幾何 **13**
- 5.1 点 . . . 13
- 5.2 直線と線分 . . . 13
- 5.3 円 . . . 14
- 5.4 多角形 . . . 14
  - 5.4.1 凸包 . . . 15
  - 5.4.2 最近点対 . . . 15
  - 5.4.3 点-多角形包含判定 . . . 15
  - 5.4.4 凸多角形の共通部分 . . . 15
  - 5.4.5 凸多角形の直径 . . . 15
  - 5.4.6 ドロネー三角形分割 (逐次添加法) . . . 16

6 データ構造 **16**
- 6.1 Union-Find 木 . . . 16
- 6.2 Meldable Heap . . . 16
- 6.3 Binary-Indexed-Tree . . . 17
- 6.4 Segment Tree . . . 17
- 6.5 Range Tree (Simple) . . . 17
- 6.6 Sparse table . . . 18
- 6.7 RBST . . . 18
- 6.8 永続 RBST . . . 18
- 6.9 赤黒木 . . . 19
- 6.10 永続赤黒木 . . . 20
- 6.11 wavelet 行列 . . . 21
  - 6.11.1 完備辞書 . . . 21
  - 6.11.2 wavelet 行列 . . . 22

7 その他 **23**
- 7.1 ビジュアライザ . . . 23

1 準備

1.1 Caps Lock を Control に変更

2 つ

1. 変更

```
setxkbmap -option ctrl:nocaps;
```

元に戻す

```
setxkbmap -option;
```

2. 上でダメな場合

```
xmodmap -e 'remove Lock = Caps_Lock';
xmodmap -e 'add Control = Caps_Lock';
xmodmap -e 'keysym Caps_Lock = Control_L';
```

1.2 init.el

linum は emacs24 のみ

```
(keyboard-translate ?\C-h ?\C-?)
(global-linum-mode t)
(setq linum-format "%4d ")
```

1.3 tpl.cpp

```
#include <bits/stdc++.h>
using namespace std;
#define rep(i,n) repi(i,0,n)
#define repi(i,a,b) for(int i=(int)(a);i<(int)(b);++i)
#define all(u) begin(u),end(u)
#define long int64_t
#define mp make_pair
#define pb push_back

void input() {
}

void solve() {
}

int main() {
    cin.tie(0);
    ios_base::sync_with_stdio(false);
    input(); // multiple testcases?
    solve();
}
```

1.4 get input

```
wget -r http://(url of sample input)
```

1.5 alias

```
alias g++='g++ -g -O2 -std=gnu++0x -Wl,-stack_size,64000000';
alias emacs='emacs -nw';
```

2 文字列

2.1 マッチング

2.1.1 複数文字列マッチング (Aho-Corasick 法)

$O(N + M)$

```
const int C = 128;
struct pma_node {
    pma_node *next[C]; // use next[0] as failure link
    vector<int> match;
    pma_node() { fill(next, next + C, (pma_node *) NULL); }
    ~pma_node() { rep(i, C) if (next[i] != NULL) delete next[i]; }
};
pma_node *construct_pma(const vector<string>& pat) {
    pma_node *const root = new pma_node();
    root->next[0] = root;
    // construct trie
    rep(i, pat.size()) {
        const string& s = pat[i];
        pma_node *now = root;
        for (const char c : s) {
            if (now->next[int(c)] == NULL) now->next[int(c)] = new pma_node();
            now = now->next[int(c)];
        }
        now->match.pb(i);
    }
    // make failure links with BFS
    queue<pma_node *> q;
    repi(i, 1, C) {
        if (root->next[i] == NULL) root->next[i] = root;
        else {
            root->next[i]->next[0] = root;
            q.push(root->next[i]);
        }
    }
    while (not q.empty()) {
        auto now = q.front();
        q.pop();
        repi(i, 1, C) if (now->next[i] != NULL) {
            auto next = now->next[0];
            while (next->next[i] == NULL) next = next->next[0];
            now->next[i]->next[0] = next->next[i];
            vector<int> tmp;
            set_union(all(now->next[i]->match), all(next->next[i]->match), back_inserter
                (tmp));
            now->next[i]->match = tmp;
            q.push(now->next[i]);
        }
```

```
    }
    return root;
}
void match(pma_node*& now, const string s, vector<int>& ret) {
    for (const char c : s) {
        while (now->next[int(c)] == NULL) now = now->next[0];
        now = now->next[int(c)];
        for (const int e : now->match) ret[e] = true;
    }
}
```

2.2 Suffix Array

find_string() : $O(|T|\log|S|)$
S 中に T が含まれないなら-1, 含まれるならその先頭.
LCS() : $O(|S+T|)$
最長共通部分文字列. (先頭, 長さ) を返す.

```
// verified: http://www.spoj.com/problems/{SARRAY,SUBLEX}/
int n, k;
vector<int> rnk, tmp, sa, lcp;
bool compare_sa(int i, int j) {
    if (rnk[i] != rnk[j]) return rnk[i] < rnk[j];
    else {
        int ri = i+k <= n ? rnk[i+k] : -1;
        int rj = j+k <= n ? rnk[j+k] : -1;
        return ri < rj;
    }
}
void construct_sa(const string &s) {
    n = s.size();
    rnk.assign(n+1, 0);
    tmp.assign(n+1, 0);
    sa.assign(n+1, 0);
    lcp.assign(n+1, 0);
    rep(i,n+1) {
        sa[i] = i;
        rnk[i] = i < n ? s[i] : -1;
    }
    for (k = 1; k <= n; k *= 2) {
        sort(sa.begin(), sa.end(), compare_sa);
        tmp[sa[0]] = 0;
        repi(i,1,n+1) tmp[sa[i]] = tmp[sa[i-1]] + (compare_sa(sa[i-1], sa[i]) ? 1 : 0);
        rep(i,n+1) rnk[i] = tmp[i];
    }
}
void construct_lcp(const string &s) {
    rep(i,n+1) rnk[sa[i]] = i;
    int h = lcp[0] = 0;
    rep(i,n) {
        int j = sa[rnk[i] - 1];
        if (h > 0) h--;
        for (; j+h < n and i+h < n; h++) {
            if (s[j+h] != s[i+h]) break;
        }
        lcp[rnk[i] - 1] = h;
    }
}
```

2.3 Z-algorithm

s, `s[i:]` の最長共通部分列の長さ

```
vector<int> lcp0(const string& s) {
    const int n = s.length();
    vector<int> ret(n);
    ret[0] = n;
    for (int i = 1, j = 0, k; i < n; ) {
        while (i+j < n and s[i+j] == s[j]) ++j;
        ret[i] = j;
        if (j == 0) { ++i; continue; }
        for (k = 1; i+k < n and k+ret[k] < j; ++k) {
            ret[i+k] = ret[k];
        }
        i += k, j -= k;
    }
    return ret;
}
```

2.4 回文長 (Manacher)

$O(N)$
各文字を中心とした時の回文の長さ.
偶数長の回文はダミーを挟むことで求められている.

```
vector<int> manacher(const string &s) {
    int n = s.size()*2;
    vector<int> rad(n,0);
    for (int i = 0, j = 0, k; i < n; i += k, j = max(j-k, 0)) {
        while (i-j >= 0 && i+j+1 < n && s[(i-j)/2] == s[(i+j+1)/2]) ++j;
        rad[i] = j;
        for (k = 1; i-k >= 0 && rad[i]-k >= 0 && rad[i-k] != rad[i]-k; ++k)
            rad[i+k] = min(rad[i-k], rad[i]-k);
    }
    return rad;
}
```

3 グラフ

3.1 強連結成分分解

3.1.1 関節点

$O(E)$
ある関節点 u がグラフを k 個に分割するとき art には k-1 個の u が含まれる. 不要な場合は unique を忘れないこと.

```
struct articulation {
    const int n; graph G;
    int cnt;
    vector<int> num, low, is_art;
    void dfs(int v) {
        num[v] = low[v] = ++cnt;
        for (int nv : G[v]) {
            if (num[nv] == 0) {
                dfs(nv);
                low[v] = min(low[v], low[nv]);
                if ((num[v] == 1 and num[nv] != 2) or
                    (num[v] != 1 and low[nv] >= num[v])) {
                    is_art[v] = true;
                }
```

```
            } else {
                low[v] = min(low[v], num[nv]);
            }
        }
    }
    articulation(const graph& G) : n(G.size()), G(G), cnt(0), num(n), low(n), is_art(n) {
        rep(i, n) if (num[i] == 0) dfs(i);
    }
};
```

3.1.2 橋

$O(V + E)$

```
struct bridge {
    const int n; graph G;
    int cnt;
    vector<int> num, low, in;
    stack<int> stk;
    vector<pair<int,int> > brid;
    vector<vector<int> > comp;
    void dfs(int v, int p) {
        num[v] = low[v] = ++cnt;
        stk.push(v), in[v] = true;
        for (const int nv : G[v]) {
            if (num[nv] == 0) {
                dfs(nv, v);
                low[v] = min(low[v], low[nv]);
            } else if (nv != p and in[nv]) {
                low[v] = min(low[v], num[nv]);
            }
        }
        if (low[v] == num[v]) {
            if (p != n) brid.eb(min(v, p), max(v, p));
            comp.eb();
            int w; do {
                w = stk.top();
                stk.pop(), in[w] = false;
                comp.back().pb(w);
            } while (w != v);
        }
    }
    bridge(const graph& G) : n(G.size()), G(G), cnt(0), num(n), low(n), in(n) {
        rep(i, n) if (num[i] == 0) dfs(i, n);
    }
};
```

3.1.3 強連結成分分解

$O(V + E)$

```
struct scc {
    const int n;
    graph G;
    int cnt;
    vector<int> num, low, in;
    stack<int> stk;
    vector<vector<int> > comp;
    void dfs(int v) {
        num[v] = low[v] = ++cnt;
        stk.push(v), in[v] = true;
        for (const int nv : G[v]) {
            if (num[nv] == 0) {
                dfs(nv);
                low[v] = min(low[v], low[nv]);
            } else if (in[nv]) {
                low[v] = min(low[v], num[nv]);
            }
        }
        if (low[v] == num[v]) {
            comp.eb();
            int w; do {
                w = stk.top();
                stk.pop(), in[w] = false;
                comp.back().pb(w);
            } while (w != v);
        }
    }
    scc(const graph& G) : n(G.size()), G(G), cnt(0), num(n), low(n), in(n) {
        rep(i, n) if (num[i] == 0) dfs(i);
    }
};
```

3.1.4 無向中国人郵便配達問題

$O(om \log n + o^2 2^o)$, $-O2$ で $o \le 18$ 程度が限界

```
long chinesePostman(const graph &g) {
    long total = 0;
    vector<int> odds;
    rep(u, g.size()) {
        for(auto &e: g[u]) total += e.w;
        if (g[u].size() % 2) odds.push_back(u);
    }
    total /= 2;
    int n = odds.size(), N = 1 << n;
    int w[n][n]; // make odd vertices graph
    rep(u,n) {
        int s = odds[u]; // dijkstra's shortest path
        vector<int> dist(g.size(), 1e9); dist[s] = 0;
        vector<int> prev(g.size(), -2);
        priority_queue<edge> Q;
        Q.push( edge(-1, s, 0) );
        while (!Q.empty()) {
            edge e = Q.top(); Q.pop();
            if (prev[e.to] != -2) continue;
            prev[e.to] = e.src;
            for(auto &f: g[e.to]) {
                if (dist[f->to] > e.w+f->w) {
                    dist[f->to] = e.w+f->w;
                    Q.push(edge(f->src, f->to, e.w+f->w));
                }
            }
        }
        rep(v,n) w[u][v] = dist[odds[v]];
    }
    long best[N]; // DP for general matching
    rep(S,N) best[S] = INF;
    best[0] = 0;

    for (int S = 0; S < N; ++S)
        for (int i = 0; i < n; ++i)
            if (!(S&(1<<i)))
                for (int j = i+1; j < n; ++j)
                    if (!(S&(1<<j)))
```

```
                        best[S|(1<<i)|(1<<j)] = min(best[S|(1<<i)|(1<<j)], best[S]+w[i][
                            j]);
    return total + best[N-1];
}
```

3.1.5 全点対間最短路 (Johnson)

$O(max(VE\log V, V^2))$

```
bool shortest_path(const graph &g, vector<vector<int> > &dist, vector<vector<int> > &
     prev) {
    int n = g.size();
    vector<int> h(n+1);
    rep(k,n) rep(i,n) for(auto &e: g[i]) {
        if (h[e.to] > h[e.from] + e->w) {
            h[e.to] = h[e.from] + e->w;
            if (k == n-1) return false; // negative cycle
        }
    }
    dist.assign(n, vector<int>(n, 1e9));
    prev.assign(n, vector<int>(n, -2));
    rep(s, n) {
        priority_queue<edge> q;
        q.push(edge(s, s, 0));
        while (!q.empty()) {
            edge e = q.top(); q.pop();
            if (prev[s][e.dst] != -2) continue;
            prev[s][e.to] = e.from;
            for(auto &f:g[e.to]) {
                if (dist[s][f.to] > e.w + f->w) {
                    dist[s][f.to] = e.w + f->w;
                    q.push(edge(f-.from, f.to, e.w + f->w));
                }
            }
        }
        rep(u, n) dist[s][u] += h[u] - h[s];
    }
}

vector<int> build_path(const vector<vector<int> >& prev, int s, int t) {
    vector<int> path;
    for (int u = t; u >= 0; u = prev[s][u])
        path.push_back(u);
    reverse(begin(path), end(path));
    return path;
}
```

3.1.6 無向グラフの全域最小カット

$O(V^3)$

```
int minimum_cut(const graph &g) {
    int n = g.size();
    vector< vector<int> > h(n, vector<int>(n)); // make adj. matrix
    rep(u,n) for(auto &e: g[u]) h[e.src][e.dst] += e.weight;
    vector<int> V(n); rep(u, n) V[u] = u;

    int cut = 1e9;
    for(int m = n; m > 1; m--) {
        vector<int> ws(m, 0);
        int u, v;
        int w;
        rep(k, m) {
            u = v; v = max_element(ws.begin(), ws.end())-ws.begin();
            w = ws[v]; ws[v] = -1;
            rep(i, m) if (ws[i] >= 0) ws[i] += h[V[v]][V[i]];
        }
        rep(i, m) {
            h[V[i]][V[u]] += h[V[i]][V[v]];
            h[V[u]][V[i]] += h[V[v]][V[i]];
        }
        V.erase(V.begin()+v);
        cut = min(cut, w);
    }
    return cut;
}
```

3.2 フロー

3.2.1 最大流

$O(EV^2)$

```
const int inf = 1e9;
struct edge {
    int to, cap, rev;
    edge(int to, int cap, int rev) : to(to), cap(cap), rev(rev) {}
};
typedef vector<vector<edge> > graph;
void add_edge(graph& G, int from, int to, int cap) {
    G[from].eb(to, cap, G[to].size());
    G[to].eb(from, 0, G[from].size() - 1);
}
struct max_flow {
    const int n; graph& G;
    vector<int> level, iter;
    void bfs(int s, int t) {
        level.assign(n, -1);
        queue<int> q;
        level[s] = 0, q.push(s);
        while (not q.empty()) {
            const int v = q.front();
            q.pop();
            if (v == t) return;
            for (const auto& e : G[v]) {
                if (e.cap > 0 and level[e.to] < 0) {
                    level[e.to] = level[v] + 1;
                    q.push(e.to);
                }
            }
        }
    }
    int dfs(int v, int t, int f) {
        if (v == t) return f;
        for (int& i = iter[v]; i < (int) G[v].size(); ++i) {
            edge& e = G[v][i];
            if (e.cap > 0 and level[v] < level[e.to]) {
                const int d = dfs(e.to, t, min(f, e.cap));
                if (d > 0) {
                    e.cap -= d, G[e.to][e.rev].cap += d;
                    return d;
                }
            }
        }
        return 0;
    }
```

```
    max_flow(graph& G) : n(G.size()), G(G) {}
    int calc(int s, int t) {
        int ret = 0, d;
        while (bfs(s, t), level[t] >= 0) {
            iter.assign(n, 0);
            while ((d = dfs(s, t, inf)) > 0) ret += d;
        }
        return ret;
    }
};
```

3.2.2 二部マッチング

$O(EV)$

```
int  V;
vector<int> G[MAX_V];
int match[MAX_V];
bool used[MAX_V];

void add_edge(int u, int v){
    G[u].push_back(v);
    G[v].push_back(u);
}

bool dfs(int v){
    used[v] = 1;
    rep(i,G[v].size()){
        int u = G[v][i], w = match[u];
        if(w < 0 || !used[w] && dfs(w)){
            match[v] = u;
            match[u] = v;
            return 1;
        }
    }
    return 0;
}

int bi_matching(){
    int res = 0;
    memset(match, -1, sizeof(match));
    rep(v,V) if(match[v] < 0){
        memset(used, 0, sizeof(used));
        if(dfs(v)) res++;
    }
    return res;
}
```

3.2.3 一般グラフの最大マッチング

$O(V^3)$

```
#define rep(i,n) repi(i,0,n)
#define repi(i,a,b) for(int i=(int)(a);i<(int)(b);++i)

#define even(x) (mu[x] == x or phi[mu[x]] != mu[x])
#define out(x) (mu[x] != x and phi[mu[x]] == mu[x] and phi[x] == x)
int maximum_matching(const vector<vector<int>>& G, vector<pair<int,int>>& ret) {
    const int n = G.size();
    vector<int> mu(n), phi(n), rho(n), done(n);
    rep(v, n) mu[v] = phi[v] = rho[v] = v;
    for (int x = -1; ; ) {
        if (x < 0) {
            for (x = 0; x < n and (done[x] or !even(x)); ++x);
            if (x == n) break;
        }
        int y = -1;
        for (int v : G[x]) if (out(v) or (even(v) and rho[v] != rho[x])) y = v;
        if (y == -1) {
            done[x] = true, x = -1;
        } else if (out(y)) {
            phi[y] = x;
        } else {
            vector<int> dx(n, -2), dy(n, -2); // x % 2 --> x >= 0
            for (int k = 0, w = x; dx[w] < 0; w = k % 2 ? mu[w] : phi[w]) dx[w] = k++;
            for (int k = 0, w = y; dy[w] < 0; w = k % 2 ? mu[w] : phi[w]) dy[w] = k++;
            bool disjoint = true;
            rep(v, n) if (dx[v] >= 0 and dy[v] > 0) disjoint = false;
            if (disjoint) {
                rep(v, n) if (dx[v] % 2) mu[phi[v]] = v, mu[v] = phi[v];
                rep(v, n) if (dy[v] % 2) mu[phi[v]] = v, mu[v] = phi[v];
                mu[x] = y; mu[y] = x; x = -1;
                rep(v, n) phi[v] = rho[v] = v, done[v] = false;
            } else {
                int r = x, d = n;
                rep(v, n) if (dx[v] >= 0 and dy[v] >= 0 and rho[v] == v and d > dx[v]) d
                    = dx[v], r = v;
                rep(v, n) if (dx[v] <= d and dx[v] % 2 and rho[phi[v]] != r) phi[phi[v]]
                    = v;
                rep(v, n) if (dy[v] <= d and dy[v] % 2 and rho[phi[v]] != r) phi[phi[v]]
                    = v;
                if (rho[x] != r) phi[x] = y;
                if (rho[y] != r) phi[y] = x;
                rep(v, n) if (dx[rho[v]] >= 0 or dy[rho[v]] >= 0) rho[v] = r;
            }
        }
    }
    ret.clear();
    rep(v, n) if (v < mu[v]) ret.emplace_back(v, mu[v]);
    return ret.size();
}
```

3.2.4 最小費用流

$O(FE \log V)$

```
const int inf = 1e9;
struct edge {
    int to, cap, cost, rev;
    edge(int to, int cap, int cost, int rev) : to(to), cap(cap), cost(cost), rev(rev) {}
};
typedef vector<vector<edge> > graph;
void add_edge(graph& G, int from, int to, int cap, int cost) {
    G[from].eb(to, cap, cost, G[to].size());
    G[to].eb(from, 0, -cost, G[from].size() - 1);
}
int min_cost_flow(graph& G, int s, int t, int f) {
    const int n = G.size();
    struct state {
        int v, d;
        state(int v, int d) : v(v), d(d) {}
        bool operator <(const state& t) const { return d > t.d; }
    };
    int ret = 0;
    vector<int> h(n, 0), dist, prev(n), prev_e(n);
    while (f > 0) {
        dist.assign(n, inf);
```

```cpp
        priority_queue<state> q;
        dist[s] = 0, q.emplace(s, 0);
        while (not q.empty()) {
            const int v = q.top().v;
            const int d = q.top().d;
            q.pop();
            if (dist[v] < d) continue;
            rep(i, G[v].size()) {
                const edge& e = G[v][i];
                if (e.cap > 0 and dist[e.to] > dist[v] + e.cost + h[v] - h[e.to]) {
                    dist[e.to] = dist[v] + e.cost + h[v] - h[e.to];
                    prev[e.to] = v, prev_e[e.to] = i;
                    q.emplace(e.to, dist[e.to]);
                }
            }
        }
        if (dist[t] == inf) return -1;
        rep(i, n) h[i] += dist[i];
        int d = f;
        for (int v = t; v != s; v = prev[v]) {
            d = min(d, G[prev[v]][prev_e[v]].cap);
        }
        f -= d, ret += d * h[t];
        for (int v = t; v != s; v = prev[v]) {
            edge& e = G[prev[v]][prev_e[v]];
            e.cap -= d, G[v][e.rev].cap += d;
        }
    }
    return ret;
}
```

3.2.5 Gomory-Hu 木

$O(VMAXFLOW)$

```cpp
#define RESIDUE(s,t) (capacity[s][t]-flow[s][t])
graph cutTree(const graph &g) {
    int n = g.size();
    Matrix capacity(n, Array(n)), flow(n, Array(n));
    rep(u,n) for(auto &e: g[u]) capacity[e.from][e.to] += e.w;

    vector<int> p(n), prev;
    vector<int> w(n);
    for (int s = 1; s < n; ++s) {
        int t = p[s]; // max-flow(s, t)
        rep(i,n) rep(j,n) flow[i][j] = 0;
        int total = 0;
        while (1) {
            queue<int> Q; Q.push(s);
            prev.assign(n, -1); prev[s] = s;
            while (!Q.empty() && prev[t] < 0) {
                int u = Q.front(); Q.pop();
                for(auto &e: g[u]) if (prev[e.to] < 0 && RESIDUE(u, e.to) > 0) {
                    prev[e.to] = u;
                    Q.push(e.to);
                }
            }
            if (prev[t] < 0) goto esc;
            int inc = 1e9;
            for (int j = t; prev[j] != j; j = prev[j])
                inc = min(inc, RESIDUE(prev[j], j));
            for (int j = t; prev[j] != j; j = prev[j])
                flow[prev[j]][j] += inc, flow[j][prev[j]] -= inc;
            total += inc;
        }
```

```cpp
    esc:w[s] = total; // make tree
        rep(u, n) if (u != s && prev[u] != -1 && p[u] == t)
            p[u] = s;
        if (prev[p[t]] != -1)
            p[s] = p[t], p[t] = s, w[s] = w[t], w[t] = total;
    }
    graph T(n); // (s, p[s]) is a tree edge of weight w[s]
    rep(s, n) if (s != p[s]) {
        T[  s ].push_back( Edge(s, p[s], w[s]) );
        T[p[s]].push_back( Edge(p[s], s, w[s]) );
    }
    return T;
}

// Gomory-Hu tree を用いた最大流 O(n)
int max_flow(const graph &T, int u, int t, int p = -1, int w = 1e9) {
    if (u == t) return w;
    int d = 1e9;
    for(auto &e: T[u]) if (e.to != p)
        d = min(d, max_flow(T, e.to, t, u, min(w, e.w)));
    return d;
}
```

3.3 木

3.3.1 木の直径: double sweep

3.3.2 最小全域木

```cpp
struct uedge {
    int u, v; long w;
    uedge(int u, int v, long w) : u(u), v(v), w(w) {}
    bool operator <(const uedge& t) const { return w < t.w; }
    bool operator >(const uedge& t) const { return w > t.w; }
};
graph kruskal(const graph& G) {
    const int n = G.size();
    vector<uedge> E;
    rep(i, n) for (const auto& e : G[i]) {
        if (i < e.to) E.eb(i, e.to, e.w);
    }
    sort(all(E));
    graph T(n);
    disjoint_set uf(n);
    for (const auto& e : E) {
        if (not uf.same(e.u, e.v)) {
            T[e.u].eb(e.v, e.w);
            T[e.v].eb(e.u, e.w);
            uf.merge(e.u, e.v);
        }
    }
    return T;
}
graph prim(const vector<vector<long> >& A, int s = 0) {
    const int n = A.size();
    graph T(n);
    vector<int> done(n);
    priority_queue<uedge, vector<uedge>, greater<uedge> > q;
    q.emplace(-1, s, 0);
    while (not q.empty()) {
        const auto e = q.top(); q.pop();
        if (done[e.v]) continue;
        done[e.v] = 1;
```

```
        if (e.u >= 0) {
            T[e.u].eb(e.v, e.w);
            T[e.v].eb(e.u, e.w);
        }
        rep(i, n) if (not done[i]) {
            q.emplace(e.v, i, A[e.v][i]);
        }
    }
    return T;
}
```

3.3.3 最小全域有向木

$O(VE)$

```
void visit(Graph &h, int v, int s, int r,
           vector<int> &no, vector< vector<int> > &comp,
           vector<int> &prev, vector< vector<int> > &next, vector<int> &mcost,
           vector<int> &mark, int &cost, bool &found) {
    const int n = h.size();
    if (mark[v]) {
        vector<int> temp = no;
        found = true;
        do {
            cost += mcost[v];
            v = prev[v];
            if (v != s) {
                while (comp[v].size() > 0) {
                    no[comp[v].back()] = s;
                    comp[s].push_back(comp[v].back());
                    comp[v].pop_back();
                }
            }
        } while (v != s);
        for(auto &j: comp[s]) if (j != r) for(auto &e: h[j])
            if (no[e.from] != s) e.w -= mcost[temp[j]];
    }
    mark[v] = true;
    for(auto &i: next[v]) if (no[i] != no[v] && prev[no[i]] == v)
        if (!mark[no[i]] || i == s)
            visit(h, i, s, r, no, comp, prev, next, mcost, mark, cost, found);
}
int minimum_spanning_arborescence(const graph &g, int r) {
    const int n = g.size();
    graph h(n);
    rep(u,n) for(auto &e: g[u]) h[e.to].push_back(e);

    vector<int> no(n);
    vector< vector<int> > comp(n);
    rep(u, n) comp[u].push_back(no[u] = u);

    for (int cost = 0; ;) {
        vector<int> prev(n, -1);
        vector<int> mcost(n, INF);

        rep(j,n) if (j != r) for(auto &e: g[j])
            if (no[e.from] != no[j])
                if (e.w < mcost[no[j]])
                    mcost[no[j]] = e.w, prev[no[j]] = no[e.from];

        vector< vector<int> > next(n);
        rep(u,n) if (prev[u] >= 0)
            next[prev[u]].push_back(u);

        bool stop = true;
        vector<int> mark(n);
        rep(u,n) if (u != r && !mark[u] && !comp[u].empty()) {
            bool found = false;
            visit(h, u, u, r, no, comp, prev, next, mcost, mark, cost, found);
            if (found) stop = false;
        }
        if (stop) {
            rep(u,n) if (prev[u] >= 0) cost += mcost[u];
            return cost;
        }
    }
}
```

3.3.4 最小シュタイナー木

$O(4^{|T|}V)$
g は無向グラフの隣接行列. T は使いたい頂点の集合.

```
int minimum_steiner_tree(vi &T, vvi &g){
    int n = g.size(), t = T.size();
    if(t <= 1) return 0;
    vvi d(g);    // all-pair shortest
    rep(k,n)rep(i,n)rep(j,n)    //Warshall Floyd
        d[i][j] = min(d[i][j], d[i][k] + d[k][j]);

    int opt[1 << t][n];
    rep(S,1<<t) rep(x,n)
        opt[S][x] = INF;

    rep(p,t) rep(q,n)   // trivial case
        opt[1 << p][q] = d[T[p]][q];

    repi(S,1,1<<t){   // DP step
        if(!(S & (S-1))) continue;
        rep(p,n) rep(E,S)
            if((E | S) == S)
                opt[S][p] = min(opt[S][p], opt[E][p] + opt[S-E][p]);
        rep(p,n) rep(q,n)
            opt[S][p] = min(opt[S][p], opt[S][q] + d[p][q]);
    }

    int ans = INF;
    rep(S,1<<t) rep(q,n)
        ans = min(ans, opt[S][q] + opt[((1<<t)-1)-S][q]);
    return ans;
}
```

3.3.5 木の同型性判定

順序付き $O(n)$, 順序なし $O(n \log n)$

```
// ordered
struct node {
    vector<node*> child;
};
bool otreeIsomorphism(node *n, node *m) {
    if (n->child.size() != m->child.size()) return false;
    rep(i, n->child.size())
        if (!otreeIsomorphism(n->child[i], m->child[i])) return false;
    return true;
}
```

```cpp

// not ordered
struct node {
    vector<node *> child;
    vector<int> code;
};
void code(node *n) {
    int size = 1;
    vector< pair<vector<int>, int> > codes;
    rep(i, n->child.size()) {
        code(n->child[i]);
        codes.push_back( make_pair(n->child[i]->code, i) );
        size += codes[i].first[0];
    }
    sort(codes.rbegin(), codes.rend()); // !reverse
    n->code.push_back(size);
    for (int i = 0; i < n->child.size(); ++i) {
        swap(n->child[i], n->child[ codes[i].second ]);
        n->code.insert(n->code.end(),
                       codes[i].first.begin(), codes[i].first.end());
    }
}
bool utreeIsomorphism(node *n, node *m) {
    code(n); code(m); return n->code == m->code;
}
```

3.3.6 HL 分解

```cpp
namespace HLD {
const int N = 200010;
vector<vector<int>> chains, childs;
int V, dep[N], par[N], heavy[N], head[N], chain[N], id[N], size[N], q[N];

void calc_heavy() {
    int root = -1;
    childs.assign(V,vector<int>());
    for(int v = 0; v < V; v++) {
        size[v] = 0;
        heavy[v] = -1;
        if(par[v] < 0) root = v;
        else childs[par[v]].push_back(v);
    }
    int l = 0, r = 0;
    q[r++] = root;
    while(l < r) {
        int v = q[l++];
        for(auto &w: childs[v]) {
            if(w == par[v]) continue;
            dep[w] = dep[v]+1;
            q[r++] = w;
        }
    }
    reverse(q,q+V);
    for(int i = 1; i < V; i++) {
        int v = q[i], &u = par[v];
        size[u] += ++size[v];
        if(heavy[u] == -1 or size[v] > size[heavy[u]]) heavy[u] = v;
    }
}
void calc_chain() {
    chains.clear();
    int idx = 0;
    for (int v = 0; v < V; v++) {
        if(par[v] < 0 or heavy[par[v]] != v) {
            chains.push_back(vector<int>());
            for (int w = v; w != -1; w = heavy[w]) {
                chain[w] = idx;
                head[w] = v;
                id[w] = chains.back().size();
                chains.back().push_back(w);
            }
            idx++;
        }
    }
}
void make_par(const vector<vector<int>> &g, int root = 0) {
    memset(par,-1,sizeof(par));
    par[root] = 0;
    int l = 0, r = 0;
    q[r++] = root;
    while(l < r) {
        int v = q[l++];
        for(const int &w: g[v]) if(par[w] < 0) q[r++] = w, par[w] = v;
    }
    par[root] = -1;
}
void build(const vector<vector<int>> &g, int root = 0) {
    V = g.size();
    make_par(g,root);
    calc_heavy();
    calc_chain();
}
int lca(int u, int v) {
    while (chain[u] != chain[v]) {
        if (dep[head[u]] > dep[head[v]]) swap(u,v);
        v = par[head[v]];
    }
    return dep[u] < dep[v]? u: v;
}
}
```

3.3.7 重心分解

```cpp
const int N = 100010;
int level[N], par[N], done[N];
vector<int> bfs(int s) {
    vector<int> ret;
    queue<int> que;
    que.push(s), par[s] = -1;
    while (not que.empty()) {
        int v = que.front(); que.pop();
        ret.push_back(v);
        done[v] = true;
        for (int u : G[v]) {
            if (level[u] == 0 and not done[u]) {
                que.push(u), par[u] = v;
            }
        }
    }
    return ret;
}
int size[N], ch[N];
void update(int v) {
    size[v] = 1, ch[v] = 0;
    for (int u : G[v]) {
        if (u != par[v] and level[u] == 0) {
            size[v] += size[u];
            ch[v] = max(ch[v], size[u]);
```

```
        }
    }
}
void decompomposite() {
    auto ord = bfs(0);
    rep(i, 26) {
        fill_n(done, n, 0);
        for (int v : ord) {
            if (level[v] == 0 and not done[v]) {
                auto sub = bfs(v);
                reverse(all(sub));
                for (int u : sub) update(u);
                int whole = size[v], petal = ch[v];
                for (bool flag = true; flag; ) {
                    flag = false;
                    for (int c : G[v]) {
                        if (level[c] == 0) {
                            int tmp = max(ch[c], whole - size[c]);
                            if (petal > tmp) {
                                v = c, petal = tmp;
                                flag = true;
                                break;
                            }
                        }
                    }
                }
                // v is a centroid
                level[v] = i + 1;
            }
        }
    }
}
```

3.4 彩色数

3.4.1 包除原理

$O(2^V V)$

N[i] := i と隣接する頂点の集合 (i も含む)

```
const int MAX_V=16;
const int mod = 10009;
int N[MAX_V], I[1<<MAX_V], V;
inline int mpow(int a, int k){ return k==0? 1: k%2? a*mpow(a,k-1)%mod: mpow(a*a%mod,k
    /2);}

bool can(int k){
    int res = 0;
    rep(S, 1<<V){
        if(__builtin_popcountll(S)%2) res -= mpow(I[S], k);
        else res += mpow(I[S],k);
    }
    return (res%mod+mod)%mod;
}

int color_number(){
    memset(I, 0, sizeof(I));
    I[0] = 1;
    repi(S,1,1<<V){
        int v = 0;
        while(!(S&(1<<v))) v++;
        I[S] = I[S-(1<<v)] + I[S&(~N[v])];
    }
    int lb = 0, ub = V, mid;
    while(ub-lb>1){
        mid = (lb+ub)/2;
        if(can(mid)) ub = mid;
        else lb = mid;
    }
    return ub;
}
```

3.4.2 極大独立集合

```
typedef vector<vector<int>> graph;
class maximal_indsets {
    const int n;
    const graph& G;
    vector<vector<int>> ret;
    vector<int> cur, exists, deg, block;
    void erase(int v) {
        if (exists[v]) {
            exists[v] = false;
            for (int nv : G[v]) --deg[nv];
        }
    }
    void restore(int v) {
        exists[v] = true;
        for (int nv : G[v]) ++deg[nv];
    }
    void select(int v) {
        cur.push_back(v);
        ++block[v], erase(v);
        for (int nv : G[v]) ++block[nv], erase(nv);
    }
    void unselect(int v) {
        cur.pop_back();
        --block[v], restore(v);
        for (int nv : G[v]) {
            if (--block[nv] == 0) restore(nv);
        }
    }
    void dfs() {
        int mn = n, v = -1;
        rep(u, n) if (exists[u]) {
            if (deg[u] < mn) mn = deg[u], v = u;
        }
        if (v == -1) {
            ret.push_back(cur);
        } else {
            select(v), dfs(), unselect(v);
            for (int nv : G[v]) {
                if (exists[nv]) select(nv), dfs(), unselect(nv);
            }
        }
    }
public:
    maximal_indsets(const graph& G) : n(G.size()), G(G), exists(n, true), deg(n), block(
        n) {
        rep(v, n) deg[v] = G[v].size();
        dfs();
    }
    const vector<vector<int>>& get() const { return ret; }
};
```

4 数学

4.1 整数

4.1.1 剰余

```
// (x, y) s.t. a x + b y = gcd(a, b)
long extgcd(long a, long b, long& x, long& y) {
    long g = a; x = 1, y = 0;
    if (b != 0) g = extgcd(b, a % b, y, x), y -= (a / b) * x;
    return g;
}
// inv[1] = 1; repi(i,2,n) inv[i] = inv[p%i] * (p - p/i) % p;
long mod_inv(long a, long m) {
    long x, y;
    if (extgcd(a, m, x, y) != 1) return 0;
    return (x % m + m) % m;
}
// a mod p where n! = a p^e in O(log_p n)
long mod_fact(long n, long p, long& e) {
    const int P = 1000010;
    static long fac[P] = {1};
    for (static int once = 1; once; --once) {
        repi(i,1,P) fac[i] = fac[i-1] * i % p;
    }
    e = 0;
    if (n == 0) return 1;
    long ret = mod_fact(n/p, p, e);
    e += n/p;
    return ret * (n/p%2 ? p - fac[n%p] : fac[n%p]) % p;
}
long mod_binom(long n, long k, long p) {
    if (k < 0 or n < k) return 0;
    long e1, e2, e3;
    long a1 = mod_fact(n, p, e1);
    long a2 = mod_fact(k, p, e2);
    long a3 = mod_fact(n - k, p, e3);
    if (e1 > e2 + e3) return 0;
    return a1 * mod_inv(a2 * a3 % p, p) % p;
}
long mod_pow(long a, long b, long m) {
    long ret = 1;
    do {
        if (b & 1) ret = ret * a % m;
        a = a * a % m;
    } while (b >>= 1);
    return ret;
}
inline long mod_mul(long a, long b, long m) {
    long ret = a * b - m * long(roundl((long double)(a) * b / m));
    return ret < 0 ? ret + m : ret;
}
```

4.1.2 離散対数

```
long discrete_log(long a, long m) {
    if (a == 0) return -1;
    long b = sqrt(m)+1, t = 1;
    unordered_map<long,long> mem;
    rep(i, b) {
        mem[t] = i;
        t = t * a % m;
        if (t == 1) return i+1;
    }
    long u = t;
    for (int i = b; i < m; i += b) {
        if (mem.find(mod_inv(u, m)) != mem.end()) {
            return mem[mod_inv(u, m)] + i;
        }
        u = u * t % m;
    }
    return -1;
}
```

4.1.3 カタラン数

() を正しく並べる方法, 二分木, 格子状の経路の数え上げ, 平面グラフの交差などに使われる. $C_{16} = 35357670$ が限界？

$$C_n = \frac{1}{n+1}\binom{2n}{n} = \binom{2n}{n} - \binom{2n}{n-1} \approx \frac{4^n}{n^{3/2}\sqrt{\pi}}$$

4.1.4 乱数 (xor shift)

周期は $2^{128} - 1$

```
unsigned xorshift() {
    static unsigned x = 123456789;
    static unsigned y = 362436069;
    static unsigned z = 521288629;
    static unsigned w = 88675123;
    unsigned t;
    t = x ^^cb^^86 (x << 11);
    x = y; y = z; z = w;
    return w = (w ^^cb^^86 (w >> 19)) ^^cb^^86 (t ^^cb^^86 (t >> 8));
}
```

4.1.5 確率的素数判定 (Miller-Rabin 法)

$O(k \log^3 n)$
合成数を素数と判定する確率は最大で 4^{-k}

```
bool suspect(long a, int s, long d, long n) {
    long x = mod_pow(a, d, n); // use mod_powl instead for large n
    if (x == 1) return true;
    for (int r = 0; r < s; ++r) {
        if (x == n - 1) return true;
        x = x * x % n; // use mod_mul instead for large n
    }
    return false;
}
// {2,7,61,-1}                 is for n < 4759123141 (= 2^32)
// {2,3,5,7,11,13,17,19,23,-1} is for n < 10^16 (at least)
bool is_prime(long n) {
    if (n <= 1 || (n > 2 && n % 2 == 0)) return false;
    int test[] = {2,3,5,7,11,13,17,19,23,-1};
    long d = n - 1, s = 0;
    while (d % 2 == 0) ++s, d /= 2;
    for (int i = 0; test[i] < n && test[i] != -1; ++i)
        if (!suspect(test[i], s, d, n)) return false;
    return true;
```

```
}
```

4.2 多項式

FFT は基本定数重めなので TLE に注意する.

4.2.1 FFT(complex)

$O(N \log N)$
複素数を用いた FFT. 変換する vector のサイズは 2 の冪乗にすること.

```
typedef complex<double> cd;
vector<cd> fft(vector<cd> f, bool inv){
    int n, N = f.size();
    for(n=0;;n++) if(N == (1<<n)) break;
    rep(m,N){
        int m2 = 0;
        rep(i,n) if(m&(1<<i)) m2 |= (1<<(n-1-i));
        if(m < m2) swap(f[m], f[m2]);
    }

    for(int t=1;t<N;t*=2){
        double theta = acos(-1.0) / t;
        cd w(cos(theta), sin(theta));
        if(inv) w = cd(cos(theta), -sin(theta));
        for(int i=0;i<N;i+=2*t){
            cd power(1.0, 0.0);
            rep(j,t){
                cd tmp1 = f[i+j] + f[i+t+j] * power;
                cd tmp2 = f[i+j] - f[i+t+j] * power;
                f[i+j] = tmp1;
                f[i+t+j] = tmp2;
                power = power * w;
            }
        }
    }
    if(inv) rep(i,N) f[i] /= N;
    return f;
}
```

4.2.2 FFT(modulo)

$O(N \log N)$
剰余環を用いた FFT(FMT). 変換する vector のサイズは 2 の冪乗にすること. mod は $a*2^e+1$ の形.

```
#include "number_theory.cpp"

const int mod = 7*17*(1<<23)+1;
vector<int> fmt(vector<int> f, bool inv){
    int e, N = f.size();
    // assert((N&(N-1))==0 and "f.size() must be power of 2");
    for(e=0;;e++) if(N == (1<<e)) break;
    rep(m,N){
        int m2 = 0;
        rep(i,e) if(m&(1<<i)) m2 |= (1<<(e-1-i));
        if(m < m2) swap(f[m], f[m2]);
    }
    for(int t=1; t<N; t*=2){
        int r = pow_mod(3,(mod-1)/(t*2),mod);
        if(inv) r = mod_inverse(r,mod);
        for(int i=0; i<N; i+=2*t){
            int power = 1;
            rep(j,t){
                int x = f[i+j], y = 1LL*f[i+t+j]*power%mod;
                f[i+j] = (x+y)%mod;
                f[i+t+j] = (x-y+mod)%mod;
                power = 1LL*power*r%mod;
            }
        }
    }
    if(inv) for(int i=0,ni=mod_inv(N,mod);i<N;i++) f[i] = 1LL*f[i]*ni%mod;
    return f;
}
```

4.2.3 積 (FMT)

$O(N \log N)$. `fmt()` が必要.

```
vector<int> poly_mul(vector<int> f, vector<int> g){
    int N = max(f.size(),g.size())*2;
    f.resize(N); g.resize(N);
    f = fmt(f,0); g = fmt(g,0);
    rep(i,N) f[i] = 1LL*f[i]*g[i]%mod;
    f = fmt(f,1);
    return f;
}
```

4.2.4 逆元 (FMT)

$O(N \log N)$. `extgcd()`, `mod_inverse()`, `poly_mul()`, `fmt()` が必要.

```
vector<int> poly_inv(const vector<int> &f){
    int N = f.size();
    vector<int> r(1,mod_inv(f[0],mod));
    for(int k = 2; k <= N; k <<= 1){
        vector<int> nr = poly_mul(poly_mul(r,r), vector<int>(f.begin(),f.begin()+k));
        nr.resize(k);
        rep(i,k/2) {
            nr[i] = (2*r[i]-nr[i]+mod)%mod;
            nr[i+k/2] = (mod-nr[i+k/2])%mod;
        }
        r = nr;
    }
    return r;
}
```

4.2.5 平方根 (FMT)

$O(NlogN)$. `extgcd()`, `mod_inverse()`, `poly_inv()`, `poly_mul()`, `fmt()` が必要.

```
const int inv2 = (mod+1)/2;
vector<int> poly_sqrt(const vector<int> &f) {
    int N = f.size();
    vector<int> s(1,1); // s[0] = sqrt(f[0])
    for(int k = 2; k <= N; k <<= 1) {
        s.resize(k);
        vector<int> ns = poly_mul(poly_inv(s), vector<int>(f.begin(),f.begin()+k));
        ns.resize(k);
```

```
        rep(i,k) s[i] = 1LL*(s[i]+ns[i])*inv2%mod;
    }
    return s;
}
```

4.3 行列

```
typedef double number;
typedef vector<number> vec;
typedef vector<vec> mat;
vec mul(const mat& A, const vec& x) {
    const int n = A.size();
    vec b(n);
    rep(i, n) rep(j, A[0].size()) {
        b[i] = A[i][j] * x[j];
    }
    return b;
}
mat mul(const mat& A, const mat& B) {
    const int n = A.size();
    const int o = A[0].size();
    const int m = B[0].size();
    mat C(n, vec(m));
    rep(i, n) rep(k, o) rep(j, m) {
        C[i][j] += A[i][k] * B[k][j];
    }
    return C;
}
mat pow(mat A, long m) {
    const int n = A.size();
    mat B(n, vec(n));
    rep(i, n) B[i][i] = 1;
    do {
        if (m & 1) B = mul(B, A);
        A = mul(A, A);
    } while (m >>= 1);
    return B;
}
const number eps = 1e-4;
// determinant; O(n^3)
number det(mat A) {
    int n = A.size();
    number D = 1;
    rep(i,n){
        int pivot = i;
        repi(j,i+1,n)
            if (abs(A[j][i]) > abs(A[pivot][i])) pivot = j;
        swap(A[pivot], A[i]);
        D *= A[i][i] * (i != pivot ? -1 : 1);
        if (abs(A[i][i]) < eps) break;
        repi(j,i+1,n)
            for(int k=n-1;k>=i;--k)
                A[j][k] -= A[i][k] * A[j][i] / A[i][i];
    }
    return D;
}
// rank; O(n^3)
int rank(mat A) {
    int n = A.size(), m = A[0].size(), r = 0;
    for(int i = 0; i < m and r < n; i++){
        int pivot = r;
        repi(j,r+1,n)
            if (abs(A[j][i]) > abs(A[pivot][i])) pivot = j;
        swap(A[pivot], A[r]);
        if (abs(A[r][i]) < eps) continue;
        for(int k=m-1;k>=i;--k)
            A[r][k] /= A[r][i];
        repi(j,r+1,n) repi(k,i,m)
            A[j][k] -= A[r][k] * A[j][i];
        ++r;
    }
    return r;
}
```

4.3.1 線形方程式の解 (Givens 消去法)

$O(N^3)$

```
typedef double number;
typedef vector<vector<number> > matrix;
inline double my_hypot(double x, double y) { return sqrt(x * x + y * y); }
inline void givens_rotate(number& x, number& y, number c, number s) {
    number u = c * x + s * y, v = -s * x + c * y;
    x = u, y = v;
}
vector<number> givens(matrix A, vector<number> b) {
    const int n = b.size();
    rep(i, n) repi(j, i + 1, n) {
        const number r = my_hypot(A[i][i], A[j][i]);
        const number c = A[i][i] / r, s = A[j][i] / r;
        givens_rotate(b[i], b[j], c, s);
        repi(k, i, n) givens_rotate(A[i][k], A[j][k], c, s);
    }
    for (int i = n - 1; i >= 0; --i) {
        repi(j, i + 1, n) b[i] -= A[i][j] * b[j];
        b[i] /= A[i][i];
    }
    return b;
}
```

4.4 割り当て問題

4.4.1 ハンガリアン法

$O(N^2)$

```
int hungarian(const vector<vector<int>> &a) {
    int n = a.size(), p, q;
    vector<int> fx(n, inf), fy(n, 0), x(n, -1), y(n, -1);
    rep(i,n) rep(j,n) fx[i] = max(fx[i], a[i][j]);
    for (int i = 0; i < n; ) {
        vector<int> t(n, -1), s(n+1, i);
        for (p = q = 0; p <= q && x[i] < 0; ++p)
            for (int k = s[p], j = 0; j < n && x[i] < 0; ++j)
                if (fx[k] + fy[j] == a[k][j] && t[j] < 0) {
                    s[++q] = y[j], t[j] = k;
                    if (s[q] < 0)
                        for (p = j; p >= 0; j = p)
                            y[j] = k = t[j], p = x[k], x[k] = j;
                }
        if (x[i] < 0) {
            int d = inf;
            rep(k,q+1) rep(j,n) if (t[j] < 0) d = min(d, fx[s[k]] + fy[j] - a[s[k]][j]);
            rep(j,n) fy[j] += (t[j] < 0 ? 0 : d);
            rep(k,q+1) fx[s[k]] -= d;
        } else i++;
```

```
    }
    int ret = 0;
    rep(i,n) ret += a[i][x[i]];
    return ret;
}
```

5 幾何

```
// constants and eps-considered operators
const double eps = 1e-8; // choose carefully!
const double pi = acos(-1.0);

inline bool lt(double a, double b) { return a < b - eps; }
inline bool gt(double a, double b) { return lt(b, a); }
inline bool le(double a, double b) { return !lt(b, a); }
inline bool ge(double a, double b) { return !lt(a, b); }
inline bool ne(double a, double b) { return lt(a, b) or lt(b, a); }
inline bool eq(double a, double b) { return !ne(a, b); }
```

5.1 点

```
typedef complex<double> point;
inline double dot  (point a, point b) { return real(conj(a) * b); }
inline double cross(point a, point b) { return imag(conj(a) * b); }
/*
 *  Here is what ccw(a, b, c) returns:
 *
 *          1
 *  ------------------
 *    2 |a   0   b| -2
 *  ------------------
 *         -1
 *
 *  Note: we can implement intersectPS(p, s) as !ccw(s.a, s.b, p).
 */
int ccw(point a, point b, point c) {
    b -= a, c -= a;
    if (cross(b, c) > eps)    return +1;
    if (cross(b, c) < eps)    return -1;
    if (dot(b, c) < eps)      return +2; // c -- a -- b
    if (lt(norm(b), norm(c))) return -2; // a -- b -- c
    return 0;
}
```

5.2 直線と線分

```
struct line {
    point a, b;
    line(point a, point b) : a(a), b(b) {}
};

bool intersectLS(const line& l, const line& s) {
    return ccw(l.a, l.b, s.a) * ccw(l.a, l.b, s.b) <= 0;
}
bool intersectSS(const line& s, const line& t) {
    return intersectLS(s, t) and intersectLS(t, s);
}
bool intersectLL(const line& l, const line& m) {
    return ne(cross(l.b - l.a, m.b - m.a), 0.0)  // not parallel
        or eq(cross(l.b - l.a, m.a - l.a), 0.0); // overlap
}
point crosspointLL(const line& l, const line& m) {
    double A = cross(l.b - l.a, m.b - m.a);
    double B = cross(l.b - l.a, m.a - l.a);
    if (eq(A, 0.0) and eq(B, 0.0)) return m.a; // overlap
    assert(ne(A, 0.0));                        // not parallel
    return m.a - B / A * (m.b - m.a);
}
point proj(const line& l, point p) {
    double t = dot(l.b - l.a, p - l.a) / norm(l.b - l.a);
    return l.a + t * (l.b - l.a);
}
point reflection(const line& l, point p) { return 2.0 * proj(l, p) - p; }

double distanceLP(const line& l, point p) { return abs(proj(l, p) - p); }
double distanceLL(const line& l, const line& m) {
    return intersectLL(l, m) ? 0.0 : distanceLP(l, m.a);
}
double distanceLS(const line& l, const line& s) {
    return intersectLS(l, s) ? 0.0 : min(distanceLP(l, s.a), distanceLP(l, s.b));
}
double distancePS(point p, const line& s) {
    point h = proj(s, p);
    return ccw(s.a, s.b, h) ? min(abs(s.a - p), abs(s.b - p)) : abs(h - p);
}
double distanceSS(const line& s, const line& t) {
    if (intersectSS(s, t)) return 0.0;
    return min(min(distancePS(s.a, t), distancePS(s.b, t)),
               min(distancePS(t.a, s), distancePS(t.b, s)));
}
```

5.3 円

```
struct circle {
    point o; double r;
    circle(point o, double r) : o(o), r(r) {}
};

bool intersectCL(const circle& c, const line& l) {
    return le(norm(proj(l, c.o) - c.o), c.r * c.r);
}
int intersectCS(const circle& c, const line& s) {
    if (not intersectCL(c, s)) return 0;
    double a = abs(s.a - c.o);
    double b = abs(s.b - c.o);
    if (lt(a, c.r) and lt(b, c.r)) return 0;
    if (lt(a, c.r) or lt(b, c.r)) return 1;
    return ccw(s.a, s.b, proj(s, c.o)) ? 0 : 2;
}
bool intersectCC(const circle& c, const circle& d) {
    double dist = abs(d.o - c.o);
    return le(abs(c.r - d.r), dist) and le(dist, c.r + d.r);
}
line crosspointCL(const circle& c, const line& l) {
    point h = proj(l, c.o);
    double a = sqrt(c.r * c.r - norm(h - c.o));
    point d = a * (l.b - l.a) / abs(l.b - l.a);
    return line(h - d, h + d);
}
line crosspointCC(const circle& c, const circle& d) {
```

```
    double dist = abs(d.o - c.o), th = arg(d.o - c.o);
    double ph = acos((c.r * c.r + dist * dist - d.r * d.r) / (2.0 * c.r * dist));
    return line(c.o + polar(c.r, th - ph), c.o + polar(c.r, th + ph));
}

line tangent(const circle& c, double th) {
    point h = c.o + polar(c.r, th);
    point d = polar(c.r, th) * point(0, 1);
    return line(h - d, h + d);
}
vector<line> common_tangents(const circle& c, const circle& d) {
    vector<line> ret;
    double dist = abs(d.o - c.o), th = arg(d.o - c.o);
    if (abs(c.r - d.r) < dist) { // outer
        double ph = acos((c.r - d.r) / dist);
        ret.pb(tangent(c, th - ph));
        ret.pb(tangent(c, th + ph));
    }
    if (abs(c.r + d.r) < dist) { // inner
        double ph = acos((c.r + d.r) / dist);
        ret.pb(tangent(c, th - ph));
        ret.pb(tangent(c, th + ph));
    }
    return ret;
}
pair<circle, circle> tangent_circles(const line& l, const line& m, double r) {
    double th = arg(m.b - m.a) - arg(l.b - l.a);
    double ph = (arg(m.b - m.a) + arg(l.b - l.a)) / 2.0;
    point p = crosspointLL(l, m);
    point d = polar(r / sin(th / 2.0), ph);
    return mp(circle(p - d, r), circle(p + d, r));
}
line bisector(point a, point b);
circle circum_circle(point a, point b, point c) {
    point o = crosspointLL(bisector(a, b), bisector(a, c));
    return circle(o, abs(a - o));
}
```

5.4 多角形

```
typedef vector<point> polygon;

double area(const polygon& g) {
    double ret = 0.0;
    int j = g.size() - 1;
    rep(i, g.size()) {
        ret += cross(g[j], g[i]), j = i;
    }
    return ret / 2.0;
}
point centroid(const polygon& g) {
    if (g.size() == 1) return g[0];
    if (g.size() == 2) return (g[0] + g[1]) / 2.0;
    point ret = 0.0;
    int j = g.size() - 1;
    rep(i, g.size()) {
        ret += cross(g[j], g[i]) * (g[j] + g[i]), j = i;
    }
    return ret / area(g) / 6.0;
}
line bisector(point a, point b) {
    point m = (a + b) / 2.0;
    return line(m, m + (b - a) * point(0, 1));
}
polygon convex_cut(const polygon& g, const line& l) {
    polygon ret;
    int j = g.size() - 1;
    rep(i, g.size()) {
        if (ccw(l.a, l.b, g[j]) != -1) ret.pb(g[j]);
        if (intersectLS(l, line(g[j], g[i]))) ret.pb(crosspointLL(l, line(g[j], g[i])));
        j = i;
    }
    return ret;
}
polygon voronoi_cell(polygon g, const vector<point>& v, int k) {
    rep(i, v.size()) if (i != k) {
        g = convex_cut(g, bisector(v[i], v[k]));
    }
    return g;
}
```

5.4.1 凸包

```
namespace std {
    bool operator <(const point& a, const point& b) {
        return ne(real(a), real(b)) ? lt(real(a), real(b)) : lt(imag(a), imag(b));
    }
}

polygon convex_hull(vector<point> v) {
    const int n = v.size();
    sort(all(v));
    polygon ret(2 * n);
    int k = 0;
    for (int i = 0; i < n; ret[k++] = v[i++]) {
        while (k >= 2 and ccw(ret[k - 2], ret[k - 1], v[i]) <= 0) --k;
    }
    for (int i = n - 2, t = k + 1; i >= 0; ret[k++] = v[i--]) {
        while (k >= t and ccw(ret[k - 2], ret[k - 1], v[i]) <= 0) --k;
    }
    ret.resize(k - 1);
    return ret;
}
```

5.4.2 最近点対

だいたい $O(n \log n)$, 最悪縦 1 列に並んでる場合 $O(n^2)$

```
pair<point,point> closest_pair(vector<point> p) {
    int n = p.size(), s = 0, t = 1, m = 2, S[n];
    S[0] = 0, S[1] = 1;
    sort(all(p)); // "p < q" <=> "p.x < q.x"
    double d = norm(p[s]-p[t]);
    for (int i = 2; i < n; S[m++] = i++) rep(j, m) {
            if (norm(p[S[j]]-p[i])<d) d = norm(p[s = S[j]]-p[t = i]);
            if (real(p[S[j]]) < real(p[i]) - d) S[j--] = S[--m];
        }
    return make_pair(p[s], p[t]);
}
```

5.4.3 点-多角形包含判定

$O(n)$

```
enum { OUT, ON, IN };
int contains(const polygon& P, const point& p) {
    bool in = false;
    for (int i = 0; i < (int)P.size(); ++i) {
        point a = P[i] - p, b = P[(i+1)%P.size()] - p;
        if (imag(a) > imag(b)) swap(a, b);
        if (imag(a) <= 0 && 0 < imag(b) && cross(a, b) < 0) in = !in;
        if (cross(a, b) == 0 && dot(a, b) <= 0) return ON;
    }
    return in ? IN : OUT;
}
```

5.4.4 凸多角形の共通部分

$O(n+m)$

```
bool intersect_1pt(const point& a, const point& b,
                   const point& c, const point& d, point &r) {
    number D =  cross(b - a, d - c);
    if (eq(D,0)) return false;
    number t =  cross(c - a, d - c) / D;
    number s = -cross(a - c, b - a) / D;
    r = a + t * (b - a);
    return ge(t, 0) && le(t, 1) && ge(s, 0) && le(s, 1);
}
polygon convex_intersect(const polygon &P, const polygon &Q) {
    const int n = P.size(), m = Q.size();
    int a = 0, b = 0, aa = 0, ba = 0;
    enum { Pin, Qin, Unknown } in = Unknown;
    polygon R;
    do {
        int a1 = (a+n-1) % n, b1 = (b+m-1) % m;
        number C = cross(P[a] - P[a1], Q[b] - Q[b1]);
        number A = cross(P[a1] - Q[b], P[a] - Q[b]);
        number B = cross(Q[b1] - P[a], Q[b] - P[a]);
        point r;
        if (intersect_1pt(P[a1], P[a], Q[b1], Q[b], r)) {
            if (in == Unknown) aa = ba = 0;
            R.push_back( r );
            in = B > 0 ? Pin : A > 0 ? Qin : in;
        }
        if (C == 0 && B == 0 && A == 0) {
            if (in == Pin) { b = (b + 1) % m; ++ba; }
            else           { a = (a + 1) % m; ++aa; }
        } else if (C >= 0) {
            if (A > 0) { if (in == Pin) R.push_back(P[a]); a = (a+1)%n; ++aa; }
            else       { if (in == Qin) R.push_back(Q[b]); b = (b+1)%m; ++ba; }
        } else {
            if (B > 0) { if (in == Qin) R.push_back(Q[b]); b = (b+1)%m; ++ba; }
            else       { if (in == Pin) R.push_back(P[a]); a = (a+1)%n; ++aa; }
        }
    } while ( (aa < n || ba < m) && aa < 2*n && ba < 2*m );
    if (in == Unknown) {
        if (convex_contains(Q, P[0])) return P;
        if (convex_contains(P, Q[0])) return Q;
    }
    return R;
}
```

5.4.5 凸多角形の直径

$O(n)$

```
inline double diff(const vector<point> &P, const int &i) { return (P[(i+1)%P.size()] - P
    [i]);}
number convex_diameter(const polygon &pt) {
    const int n = pt.size();
    int is = 0, js = 0;
    for (int i = 1; i < n; ++i) {
        if (imag(pt[i]) > imag(pt[is])) is = i;
        if (imag(pt[i]) < imag(pt[js])) js = i;
    }
    number maxd = norm(pt[is]-pt[js]);

    int i, maxi, j, maxj;
    i = maxi = is;
    j = maxj = js;
    do {
        if (cross(diff(pt,i), diff(pt,j)) >= 0) j = (j+1) % n;
        else i = (i+1) % n;
        if (norm(pt[i]-pt[j]) > maxd) {
            maxd = norm(pt[i]-pt[j]);
            maxi = i; maxj = j;
        }
    } while (i != is || j != js);
    return maxd; /* farthest pair is (maxi, maxj). */
}
```

5.4.6 ドロネー三角形分割 (逐次添加法)

$O(n^2)$

```
bool incircle(point a, point b, point c, point p) {
    a -= p; b -= p; c -= p;
    return norm(a) * cross(b, c)
         + norm(b) * cross(c, a)
         + norm(c) * cross(a, b) >= 0;
    // < : inside, = cocircular, > outside
}
#define SET_TRIANGLE(i, j, r) \
    E[i].insert(j); em[i][j] = r; \
    E[j].insert(r); em[j][r] = i; \
    E[r].insert(i); em[r][i] = j; \
    S.push(pair<int,int>(i, j));
#define REMOVE_EDGE(i, j) \
    E[i].erase(j); em[i][j] = -1; \
    E[j].erase(i); em[j][i] = -1;
#define DECOMPOSE_ON(i,j,k,r) { \
        int m = em[j][i]; REMOVE_EDGE(j,i); \
        SET_TRIANGLE(i,m,r); SET_TRIANGLE(m,j,r); \
        SET_TRIANGLE(j,k,r); SET_TRIANGLE(k,i,r); }
#define DECOMPOSE_IN(i,j,k,r) { \
        SET_TRIANGLE(i,j,r); SET_TRIANGLE(j,k,r); \
        SET_TRIANGLE(k,i,r); }
#define FLIP_EDGE(i,j) { \
        int k = em[j][i]; REMOVE_EDGE(i,j); \
        SET_TRIANGLE(i,k,r); SET_TRIANGLE(k,j,r); }
#define IS_LEGAL(i, j) \
    (em[i][j] < 0 || em[j][i] < 0 || \
     !incircle(P[i],P[j],P[em[i][j]],P[em[j][i]]))
double Delaunay(vector<point> P) {
    const int n = P.size();
    P.push_back( point(-inf,-inf) );
    P.push_back( point(+inf,-inf) );
    P.push_back( point(  0 ,+inf) );
    int em[n+3][n+3]; memset(em, -1, sizeof(em));
    set<int> E[n+3];
```

```
    stack< pair<int,int> > S;
    SET_TRIANGLE(n+0, n+1, n+2);
    for (int r = 0; r < n; ++r) {
        int i = n, j = n+1, k;
        while (1) {
            k = em[i][j];
            if      (ccw(P[i], P[em[i][j]], P[r]) == +1) j = k;
            else if (ccw(P[j], P[em[i][j]], P[r]) == -1) i = k;
            else break;
        }
        if      (ccw(P[i], P[j], P[r]) != +1) { DECOMPOSE_ON(i,j,k,r); }
        else if (ccw(P[j], P[k], P[r]) != +1) { DECOMPOSE_ON(j,k,i,r); }
        else if (ccw(P[k], P[i], P[r]) != +1) { DECOMPOSE_ON(k,i,j,r); }
        else                                  { DECOMPOSE_IN(i,j,k,r); }
        while (!S.empty()) {
            int u = S.top().first, v = S.top().second; S.pop();
            if (!IS_LEGAL(u, v)) FLIP_EDGE(u, v);
        }
    }
    double minarg = 1e5;
    for (int a = 0; a < n; ++a) {
        for(auto &b: E[a]) {
            int c = em[a][b];
            if (b < n && c < n) {
                point p = P[a] - P[b], q = P[c] - P[b];
                minarg = min(minarg, acos(dot(p,q)/abs(p)/abs(q)));
            }
        }
    }
    return minarg;
}
```

6 データ構造

6.1 Union-Find 木

```
class disjoint_set {
    vector<int> p;
public:
    disjoint_set(int n) : p(n, -1) {}
    int root(int i) { return p[i] >= 0 ? p[i] = root(p[i]) : i; }
    bool same(int i, int j) { return root(i) == root(j); }
    int size(int i) { return -p[root(i)]; }
    void merge(int i, int j) {
        i = root(i), j = root(j);
        if (i == j) return;
        if (p[i] > p[j]) swap(i, j);
        p[i] += p[j], p[j] = i;
    }
};
```

6.2 Meldable Heap

```
template <class T>
class meldable_heap {
    struct node {
        node *l = NULL, *r = NULL;
        T val;
        node(const T& val) : val(val) {}
        ~node() { delete l, delete r; }
    };
    node *meld(node *a, node *b) {
        if (!a) return b;
        if (!b) return a;
        if (a->val > b->val) swap(a, b);
        a->r = meld(a->r, b);
        swap(a->l, a->r);
        return a;
    }
    node *root = NULL;
    meldable_heap(node *root) : root(root) {}
public:
    meldable_heap() {}
    bool empty() const { return !root; }
    const T& top() const { return root->val; }
    void meld(const meldable_heap<T>&& t) { root = meld(root, t.root); }
    void push(const T& val) { root = meld(root, new node(val)); }
    void pop() {
        node *t = root;
        root = meld(t->l, t->r);
        t.l = t.r = NULL;
        delete t;
    }
};
```

6.3 Binary-Indexed-Tree

0-indexed

```
template<class T> struct bit {
    int n;
    vector<T> dat;

    bit(int n) : n(n) { dat.assign(n,0); }
    // sum [0,i)
    T sum(int i){
        int ret = 0;
        for(--i; i>=0; i=(i&(i+1))-1) ret += bit[i];
        return ret;
    }
    // sum [i,j)
    T sum(int i, int j){ return sum(j) - sum(i);}
    // add x to i
    void add(int i, T x){ for(; i < n; i|=i+1) bit[i] += x;}
};
```

6.4 Segment Tree

区間 add と RMQ ができる.

```
template<class T> struct segtree {
    int N;
    vector<T> dat, sum;
    segtree(int n) {
        N = 1;
        while(N < n) N <<= 1;
        dat.assign(2*N-1,0);
        sum.assign(2*N-1,0);
    }
    void add(int a, int b, T x) { add(a,b,x,0,0,N);}
    T add(int a, int b, T x, int k, int l, int r) {
        if(b <= l or r <= a) return dat[k];
```

```
        if(a <= l and r <= b) {
            sum[k] += x;
            return dat[k] += x;
        }
        int m = (l+r)/2;
        return dat[k] = min(add(a,b,x,2*k+1,l,m),add(a,b,x,2*k+2,m,r))+sum[k];
    }
    T minimum(int a, int b) { return minimum(a,b,0,0,N);}
    T minimum(int a, int b, int k, int l, int r) {
        if(b <= l or r <= a) return 1e9;
        if(a <= l and r <= b) return dat[k];
        int m = (l+r)/2;
        return min(minimum(a,b,2*k+1,l,m),minimum(a,b,2*k+2,m,r))+sum[k];
    }
};
```

6.5 Range Tree (Simple)

```
vector<pair<int,int> > ps;
struct node {
    int a, b;
    node *l, *r;
    vector<int> ys, bit;
    node(int a, int b) : a(a), b(b), l(NULL), r(NULL) {}
};
inline int leftmost(node *v) { return ps[v->a].first; }
inline int rightmost(node *v) { return ps[v->b - 1].first; }
node *construct(int a, int b) {
    if (a == b) return NULL;
    node *ret = new node(a, b);
    if (a == b-1) {
        ret->ys.push_back(ps[a].second);
        ret->bit.push_back(0);
    } else {
        int m = (a+b)/2;
        if ((ret->l = construct(a, m))) {
            ret->ys.insert(ret->ys.end(), all(ret->l->ys));
        }
        if ((ret->r = construct(m, b))) {
            ret->ys.insert(ret->ys.end(), all(ret->r->ys));
        }
        sort(all(ret->ys));
        ret->bit.resize(b-a);
    }
    return ret;
}
void insert(node *v, int x, int y) {
    if (!v) return;
    if (make_pair(x, y) < ps[v->a]) return;
    if (make_pair(x, y) > ps[v->b - 1]) return;
    int k = lower_bound(all(v->ys), y) - v->ys.begin();
    for (; k < (int)v->bit.size(); k |= k+1) {
        ++v->bit[k];
    }
    insert(v->l, x, y);
    insert(v->r, x, y);
}
int query(node *v, int x, int y) {
    if (!v or leftmost(v) > x) return 0;
    if (rightmost(v) <= x) {
        int ret = 0, k = upper_bound(all(v->ys), y) - v->ys.begin();
        for (; k; k &= k - 1) {
            ret += v->bit[k-1];
        }
        return ret;
    }
    return query(v->l, x, y) + query(v->r, x, y);
}
```

6.6 Sparse table

```
const int N = 200010;
const int K = 18;
int st[K][N];
void construct(int *a, int n) {
    copy_n(a, n, st[0]);
    repi(k, 1, K) {
        for (int i = 0; i+(1<<k) <= n; ++i) {
            st[k][i] = min(st[k-1][i], st[k-1][i+(1<<(k-1))]);
        }
    }
}
int query(int a, int b) {
    int k = 31-__builtin_clz(b-a);
    return min(st[k][a], st[k][b-(1<<k)]);
}
```

6.7 RBST

```
struct node {
    long val, sum;
    size_t size = 1;
    node *left = NULL, *right = NULL;
    node(long val) : val(val), sum(val) {}
    ~node() { delete left, delete right; }
};
inline long sum(node *u) { return u ? u->sum : 0; }
inline size_t size(node *u) { return u ? u->size : 0; }
inline node *pull(node *u) {
    u->sum = u->val + sum(u->left) + sum(u->right);
    u->size = 1 + size(u->left) + size(u->right);
    return u;
}
node *merge(node *u, node *v) {
    if (!u) return v;
    if (!v) return u;
    if (rand() * long(size(u) + size(v)) < long(size(u)) * RAND_MAX) {
        u->right = merge(u->right, v);
        return pull(u);
    } else {
        v->left = merge(u, v->left);
        return pull(v);
    }
}
pair<node*,node*> split(node *u, size_t k) {
    if (!u or k == 0) return {NULL, u};
    if (k == size(u)) return {u, NULL};
    if (size(u->left) >= k) {
        auto p = split(u->left, k);
        u->left = p.second;
        return {p.first, pull(u)};
    } else {
        auto p = split(u->right, k - size(u->left) - 1);
        u->right = p.first;
        return {pull(u), p.second};
```

```
    }
}
template <class ForwardIterator>
node *construct_from(ForwardIterator first, ForwardIterator last) {
    if (first == last) return NULL;
    auto mid = next(first, (last - first) / 2);
    node *u = new node(*mid);
    u->left  = construct_from(first, mid);
    u->right = construct_from(next(mid), last);
    return pull(u);
}
```

6.8 永続 RBST

```
template <class T, size_t N>
struct mempool {
    static T buf[N], *head;
    static size_t cnt() { return head - buf; }
    static void clear() { head = buf; }
    void *operator new(size_t _ __attribute__((unused))) { return head++; }
    void operator delete(void *_ __attribute__((unused))) {}
};
template <class T, size_t N> T  mempool<T, N>::buf[N];
template <class T, size_t N> T *mempool<T, N>::head = mempool<T, N>::buf;

struct node;
long sum(node *u);
size_t size(node *u);
struct node : mempool<node, M> {
    const long val = 0, sum = 0, lazy = 0;
    const size_t size = 1;
    node *const left = NULL, *const right = NULL;
    node() {}
    node(long val) : val(val), sum(val) {}
    node(long val, long lazy, node *left, node *right)
        : val(val),
          sum(val + ::sum(left) + ::sum(right)),
          lazy(lazy),
          size(1 + ::size(left) + ::size(right)),
          left(left),
          right(right) {}
};
inline long sum(node *u) { return u ? u->sum + u->lazy * u->size : 0; }
inline size_t size(node *u) { return u ? u->size : 0; }
inline node *add(node *u, long x) { return u ? new node(u->val, u->lazy + x, u->left, u
    ->right) : NULL; }
node *merge(node *u, node *v) {
    if (!u) return v;
    if (!v) return u;
    if (rand() * long(size(u) + size(v)) < long(size(u)) * RAND_MAX) {
        return new node(u->val + u->lazy, 0, add(u->left, u->lazy), merge(add(u->right,
             u->lazy), v));
    } else {
        return new node(v->val + v->lazy, 0, merge(u, add(v->left, v->lazy)), add(v->
             right, v->lazy));
    }
}
pair<node *, node *> split(node *u, size_t k) {
    if (!u or k == 0) return {NULL, u};
    if (k == size(u)) return {u, NULL};
    if (size(u->left) >= k) {
        auto p = split(add(u->left, u->lazy), k);
        return {p.first, new node(u->val + u->lazy, 0, p.second, add(u->right, u->lazy
             ))};
    } else {
        auto p = split(add(u->right, u->lazy), k - size(u->left) - 1);
        return {new node(u->val + u->lazy, 0, add(u->left, u->lazy), p.first), p.second
            };
    }
}
template <class OutputIterator>
OutputIterator dump(OutputIterator it, const node *u, long lazy = 0) {
    if (!u) return it;
    lazy += u->lazy;
    it = dump(it, u->left, lazy);
    *it++ = u->val + lazy;
    return dump(it, u->right, lazy);
}
template <class ForwardIterator>
node *construct_from(ForwardIterator first, ForwardIterator last) {
    if (first == last) return NULL;
    auto mid = next(first, (last - first) / 2);
    return new node(*mid, 0, construct_from(first, mid), construct_from(next(mid), last
        ));
}
```

6.9 赤黒木

```
template<class T> class rbtree {
    enum COL { BLACK, RED,};
    struct node {
        T val, lazy, min_val;
        int color, rnk, size;
        node *left, *right;
        // if !left then this node is leaf
        node(){}
        node(T v) : val(v), min_val(v), color(BLACK), rnk(0), size(1) {
            lazy = 0;
            left = right = NULL;
        }
        node(node *l, node *r, int c) : color(c) {
            lazy = 0;
            left = l;
            right = r;
            update();
        }
        void update() {
            eval();
            if(left) {
                rnk = max(left->rnk+(left->color==BLACK),
                          right->rnk+(right->color==BLACK));
                size = left->size+right->size;
                left->eval(); right->eval();
                min_val = min(left->min_val, right->min_val);
            }
        }
        void eval() {
            min_val += lazy;
            if(!left) val += lazy;
            else {
                left->lazy += lazy;
                right->lazy += lazy;
            }
            lazy = 0;
        }
    };

    node *new_node(T v) { return new node(v);}
```

```
    node *new_node(node *l, node *r, int c) { return new node(l,r,c);}
    node *rotate(node *v, int d) {
        node *w = d? v->right: v->left;
        if(d) {
            v->right = w->left;
            w->left = v;
            v->right->update();
        }
        else {
            v->left = w->right;
            w->right = v;
            v->left->update();
        }
        v->update(); w->update();
        v->color = RED;
        w->color = BLACK;
        return w;
    }
    node *merge_sub(node *u, node *v) {
        u->eval(); v->eval();
        if(u->rnk < v->rnk) {
            node *w = merge_sub(u,v->left);
            v->left = w;
            v->update();
            if(v->color == BLACK and w->color == RED and w->left->color == RED) {
                if(v->right->color == BLACK)  return rotate(v,0);
                else {
                    v->color = RED;
                    v->left->color = v->right->color = BLACK;
                    return v;
                }
            }
            else return v;
        }
        else if(u->rnk > v->rnk) {
            node *w = merge_sub(u->right,v);
            u->right = w;
            u->update();
            if(u->color == BLACK and w->color == RED and w->right->color == RED) {
                if(u->left->color == BLACK) return rotate(u,1);
                else {
                    u->color = RED;
                    u->left->color = u->right->color = BLACK;
                    return u;
                }
            }
            else return u;
        }
        else return new_node(u,v,RED);
    }
    node *insert(node *v, int k) {
        auto p = split(root,k);
        return root = merge(merge(p.first,v),p.second);
    }
    void add(node *v, int res, T val) {
        if(res < 1) return;
        v->eval();
        if(v->size == res) {
            v->lazy += val;
            return;
        }
        add(v->left, min(v->left->size, res), val);
        add(v->right, res-v->left->size, val);
        v->update();
    }
    T get(node *v, int k) {
        v->eval();
        if(!v->left) return v->val;
        if(v->left->size > k) return get(v->left, k);
        return get(v->right, k-v->left->size);
    }
    T minimum(node *v, int l, int r) {
        if(r-l < 1) return inf;
        v->eval();
        if(v->size == r-l) return v->min_val;
        return min(minimum(v->left, l, min(r, v->left->size)),
                   minimum(v->right, l-min(l, v->left->size), r-v->left->size));
    }
    T inf;
public:

    node *root;
    rbtree() {
        inf = (((1LL<<(sizeof(T)*8-2))-1)<<1)+1;
        root = NULL;
    }
    void clear() { delete root; root = NULL;}
    node *build(const vector<T> &vs) {
        if(!vs.size()) return root = NULL;
        if((int)vs.size() == 1) return root = new_node(vs[0]);
        int m = vs.size()/2;
        return root = merge(build(vector<T>(begin(vs),begin(vs)+m)),
                            build(vector<T>(begin(vs)+m,end(vs))));
    }
    int size() { return root? root->size: 0;}
    node *push_back(T val) { return root = merge(root,new_node(val));}
    node *push_front(T val) { return root = merge(new_node(val),root);}
    node *merge(node *u, node *v) {
        if(!u) return v;
        if(!v) return u;
        u = merge_sub(u,v);
        u->color = BLACK;
        return u;
    }
    pair<node*,node*> split(node *v, int k) {
        if(!k) return pair<node*,node*>(NULL,v);
        if(k == v->size) return pair<node*,node*>(v,NULL);
        v->eval();
        if(k < v->left->size) {
            auto p = split(v->left,k);
            return pair<node*,node*>(p.first,merge(p.second,v->right));
        }
        else if(k > v->left->size) {
            auto p = split(v->right,k-v->left->size);
            return pair<node*,node*>(merge(v->left,p.first),p.second);
        }
        else return pair<node*,node*>(v->left,v->right);
    }

    node *insert(int k, T val) { return insert(new_node(val),k);}
    node *erase(int k) {
        auto p = split(root,k+1);
        return root = merge(split(p.first,k).first, p.second);
    }
    void add(int l, int r, T val) { add(root, r, val); add(root, l, -val);}
    T get(int k) { return get(root, k);}
    T minimum(int l, int r) { return minimum(root, l, r);}
    T operator[](const int &i) { return get(i);}
};
```

6.10 永続赤黒木

```
//const int MAX = 15000000, BOUND = 14000000;
template<class T> class prbtree {
public:
    enum COL { BLACK, RED,};
    struct node {
        T val;
        int color;
        int rnk, size;
        node *left, *right;

        node(){}
        node(T v) : val(v), color(BLACK), rnk(0), size(1) {
            left = right = NULL;
        }
        node(node *l, node *r, int c) : color(c) {
            left = l;
            right = r;
            rnk = max((l? l->rnk+(l->color==BLACK): 0),
                      (r? r->rnk+(r->color==BLACK): 0));
            size = !l and !r? 1: !l? r->size: !r? r->size: l->size+r->size;
        }
    };

    node *root;
    //        node nodes[MAX];
    //        int called;

    prbtree() {
        root = NULL;
        // called = 0;
    }

    prbtree(T val) {
        root = new_node(val);
        // called = 0;
    }

    // node *new_node(T v) { return &(nodes[called++] = node(v));}
    // node *new_node(node *l, node *r, int c) { return &(nodes[called++] = node(l,r,c
        ));}
    node *new_node(T v) { return new node(v);}
    node *new_node(node *l, node *r, int c) { return new node(l,r,c);}

    node *merge_sub(node *u, node *v) {
        if(u->rnk < v->rnk) {
            node *w = merge_sub(u,v->left);
            if(v->color == BLACK and w->color == RED and w->left->color == RED){
                if(v->right->color == BLACK)  return new_node(w->left,new_node(w->right,
                    v->right,RED),BLACK);
                else return new_node(new_node(w->left,w->right,BLACK),new_node(v->right
                    ->left,v->right->right,BLACK),RED);
            }
            else return new_node(w,v->right,v->color);
        }
        else if(u->rnk > v->rnk) {
            node *w = merge_sub(u->right,v);
            if(u->color == BLACK and w->color == RED and w->right->color == RED){
                if(u->left->color == BLACK)  return new_node(new_node(u->left,w->left,
                    RED),w->right,BLACK);
                else return new_node(new_node(u->left->left,u->left->right,BLACK),
                    new_node(w->left,w->right,BLACK),RED);
            }
            else return new_node(u->left,w,u->color);
        }
        else return new_node(u,v,RED);
    }

    node *merge(node *u, node *v) {
        if(!u) return v;
        if(!v) return u;
        u = merge_sub(u,v);
        if(u->color == RED) return new_node(u->left,u->right,BLACK);
        return u;
    }

    pair<node*,node*> split(node *v, int k) {
        if(!k) return pair<node*,node*>(NULL,v);
        if(k == v->size) return pair<node*,node*>(v,NULL);
        if(k < v->left->size) {
            auto p = split(v->left,k);
            return pair<node*,node*>(p.first,merge(p.second,v->right));
        }
        else if(k > v->left->size) {
            auto p = split(v->right,k-v->left->size);
            return pair<node*,node*>(merge(v->left,p.first),p.second);
        }
        else return pair<node*,node*>(v->left,v->right);
    }

    node *build(const vector<T> &vs) {
        if(!vs.size()) return NULL;
        if((int)vs.size() == 1) return new_node(vs[0]);
        int m = vs.size()/2;
        return merge(build(vector<T>(begin(vs),begin(vs)+m)), build(vector<T>(begin(vs)+
            m,end(vs))));
    }

    int size() { return root->size;}

    void get(vector<T> &vs) { get(root,vs);}
    void get(node *v, vector<T> &vs) {
        if(!v->left and !v->right) vs.push_back(v->val);
        else {
            if(v->left) get(v->left,vs);
            if(v->right) get(v->right,vs);
        }
    }

    node *push_back(T val) {
        node *v = new_node(val);
        return root = merge(root,v);
    }

    // insert leaf at k
    node *insert(int k, T val) {
        return insert(new_node(val), k);
    }

    // insert tree v at k
    node *insert(node *v, int k) {
        auto p = split(root,k);
        return root = merge(merge(p.first,v),p.second);
    }

    // copy [l,r)
    node *copy(int l, int r) {
        return split(split(root, l).second, r-l).first;
    }
    // copy and insert [l,r) at k
    node *copy_paste(int l, int r, int k) {
        return insert(copy(l,r),k);
    }
};
```

6.11 wavelet 行列

N := 列の長さ
M := 最大値

6.11.1 完備辞書

function	計算量
count	$O(1)$
select	$O(\log N)$

```
template<int N> class FID {
    static const int bucket = 512, block = 16;
    static char popcount[];
    int n, B[N/bucket+10];
    unsigned short bs[N/block+10], b[N/block+10];

public:
    FID(){}
    FID(int n, bool s[]) : n(n) {
        if(!popcount[1]) for (int i = 0; i < (1<<block); i++) popcount[i] =
             __builtin_popcount(i);

        bs[0] = B[0] = b[0] = 0;
        for (int i = 0; i < n; i++) {
            if(i%block == 0) {
                bs[i/block+1] = 0;
                if(i%bucket == 0) {
                    B[i/bucket+1] = B[i/bucket];
                    b[i/block+1] = b[i/block] = 0;
                }
                else b[i/block+1] = b[i/block];
            }
            bs[i/block]   |= short(s[i])<<(i%block);
            b[i/block+1]  += s[i];
            B[i/bucket+1] += s[i];
        }
        if(n%bucket == 0) b[n/block] = 0;
    }

    // number of val in [0,r), O(1)
    int count(bool val, int r) { return val? B[r/bucket]+b[r/block]+popcount[bs[r/block
         ]&((1<<(r%block))-1)]: r-count(1,r); }
    // number of val in [l,r), O(1)
    int count(bool val, int l, int r) { return count(val,r)-count(val,l); }
    // position of ith in val, 0-indexed, O(log n)
    int select(bool val, int i) {
        if(i < 0 or count(val,n) <= i) return -1;
        i++;
        int lb = 0, ub = n, md;
        while(ub-lb>1) {
            md = (lb+ub)>>1;
            if(count(val,md) >= i) ub = md;
            else lb = md;
        }
        return ub-1;
    }
    int select(bool val, int i, int l) { return select(val,i+count(val,l)); }
    bool operator[](int i) { return bs[i/block]>>(i%block)&1; }
};
template<int N> char FID<N>::popcount[1<<FID<N>::block];
```

6.11.2 wavelet 行列

function	計算量	FID::count	FID::select
count	$O(\log M)$	o	
select	$O(\log N \log M)$	o	o
get	$O(\log M)$	o	
maximum	$O(\log M)$ or $O(k \log M)$	o	
kth_number	$O(\log M)$	o	
freq	$O(\log M)$	o	
freq_list	$O(k \log M)$	o	
get_rect	$O(k \log N \log M)$	o	o

```
template<class T, int N, int D> class wavelet {
    int n, zs[D];
    FID<N> dat[D];

    void max_dfs(int d, int l, int r, int &k, T val, vector<T> &vs) {
        if(l >= r or !k) return;
        if(d == D) {
            while(l++ < r and k > 0) vs.push_back(val), k--;
            return;
        }
        int lc = dat[d].count(1,l), rc = dat[d].count(1,r);
        // if min, change this order
        max_dfs(d+1, lc+zs[d], rc+zs[d], k, 1ULL<<(D-d-1)|val,vs);
        max_dfs(d+1, l-lc, r-rc, k, val, vs);
    }

    T max_dfs(int d, int l, int r, T val, T a, T b) {
        if(r-l <= 0 or val >= b) return -1;
        if(d == D) return val>=a? val: -1;
        int lc = dat[d].count(1,l), rc = dat[d].count(1,r);
        T ret = max_dfs(d+1, lc+zs[d], rc+zs[d], 1ULL<<(D-d-1)|val, a, b);
        if(~ret) return ret;
        return max_dfs(d+1, l-lc, r-rc, val, a, b);
    }

    int freq_dfs(int d, int l, int r, T val, T a, T b) {
        if(l == r) return 0;
        if(d == D) return (a <= val and val < b)? r-l: 0;
        T nv = 1ULL<<(D-d-1)|val, nnv = ((1ULL<<(D-d-1))-1)|nv;
        if(nnv < a or b <= val) return 0;
        if(a <= val and nnv < b) return r-l;
        int lc = dat[d].count(1,l), rc = dat[d].count(1,r);
        return freq_dfs(d+1,l-lc,r-rc,val,a,b)+
               freq_dfs(d+1,lc+zs[d],rc+zs[d],nv,a,b);
    }

    void list_dfs(int d, int l, int r, T val, T a, T b, vector<pair<T,int>> &vs) {
        if(val >= b or r-l <= 0) return;
        if(d == D) {
            if(a <= val) vs.push_back(make_pair(val,r-l));
            return;
        }
        T nv = val|(1LL<<(D-d-1)), nnv = nv|(((1LL<<(D-d-1))-1));
        if(nnv < a) return;
```

```cpp
            int lc = dat[d].count(1,l), rc = dat[d].count(1,r);
            list_dfs(d+1,l-lc,r-rc,val,a,b,vs);
            list_dfs(d+1,lc+zs[d],rc+zs[d],nv,a,b,vs);
        }
public:
    wavelet(int n, T seq[]) : n(n) {
        T f[N], l[N], r[N];
        bool b[N];
        memcpy(f, seq, sizeof(T)*n);
        for (int d = 0; d < D; d++) {
            int lh = 0, rh = 0;
            for (int i = 0; i < n; i++) {
                bool k = (f[i]>>(D-d-1))&1;
                if(k) r[rh++] = f[i];
                else l[lh++] = f[i];
                b[i] = k;
            }
            dat[d] = FID<N>(n,b);
            zs[d] = lh;
            swap(l,f);
            memcpy(f+lh, r, rh*sizeof(T));
        }
    }

    T get(int i) {
        T ret = 0;
        bool b;
        for (int d = 0; d < D; d++) {
            ret <<= 1;
            b = dat[d][i];
            ret |= b;
            i = dat[d].count(b,i)+b*zs[d];
        }
        return ret;
    }
    T operator[](int i) { return get(i); }

    int count(T val, int l, int r) {
        for (int d = 0; d < D; d++) {
            bool b = (val>>(D-d-1))&1;
            l = dat[d].count(b,l)+b*zs[d];
            r = dat[d].count(b,r)+b*zs[d];
        }
        return r-l;
    }
    int count(T val, int r) { return count(val,0,r); }

    int select(T val, int k) {
        int ls[D], rs[D], l = 0, r = n;
        for (int d = 0; d < D; d++) {
            ls[d] = l; rs[d] = r;
            bool b = val>>(D-d-1)&1;
            l = dat[d].count(b,l)+b*zs[d];
            r = dat[d].count(b,r)+b*zs[d];
        }
        for (int d = D-1; d >= 0; d--) {
            bool b = val>>(D-d-1)&1;
            k = dat[d].select(b,k,ls[d]);
            if(k >= rs[d] or k < 0) return -1;
            k -= ls[d];
        }
        return k;
    }
    int select(T val, int k, int l) { return select(val,k+count(val,l)); }

    vector<T> maximum(int l, int r, int k) {
        if (r-l < k) k = r-l;
        if(k < 0) return {};
        vector<T> ret;
        max_dfs(0,l,r,k,0,ret);
        return ret;
    }

    T maximum(int l, int r, T a, T b) { return max_dfs(0,l,r,0,a,b); }

    // k is 0-indexed
    T kth_number(int l, int r, int k) {
        if(r-l <= k or k < 0) return -1;
        T ret = 0;
        for (int d = 0; d < D; d++) {
            int lc = dat[d].count(1,l), rc = dat[d].count(1,r);
            if(rc-lc > k) {
                l = lc+zs[d];
                r = rc+zs[d];
                ret |= 1ULL<<(D-d-1);
            }
            else {
                k -= rc-lc;
                l -= lc;
                r -= rc;
            }
        }
        return ret;
    }

    vector<pair<T,int>> freq_list(int l, int r, T a, T b) {
        vector<pair<T,int>> ret;
        list_dfs(0,l,r,0,a,b,ret);
        return ret;
    }

    vector<pair<int,T>> get_rect(int l, int r, T a, T b) {
        vector<pair<T,int>> res = freq_list(l,r,a,b);
        vector<pair<int,T>> ret;
        for(auto &e: res)
            for (int i = 0; i < e.second; i++)
                ret.push_back(make_pair(select(e.first,i,l), e.first));
        return ret;
    }
    // number of elements in [l,r) in [a,b), O(D)
    int freq(int l, int r, T a, T b) { return freq_dfs(0,l,r,0,a,b); }
};
```

7 その他

7.1 ビジュアライザ

```html
<script>
function line(x,y,a,b){c.b();c.moveTo(x,y);c.lineTo(a,b);c.s();}
function circle(x,y,r){c.b();c.arc(x,y,r,0,7,0);c.s();}
window.onload=function(){d=document;d.i=d.getElementById;
c=d.i('c').getContext('2d');c.b=c.beginPath;c.s=c.stroke;
d.i('s').src='data.js?';};
</script>
<body><canvas id="c" width="500" height="500" style="border:1px solid #000;"></canvas>
<script id="s"></script></body>
```