

Contents

1	Algorithms	2
1.1	Geometry	2
1.1.1	Circle	2
1.1.2	Convex Hull	2
1.1.3	Point	2
1.1.4	Polygon	2
1.1.5	Geometry Primitives	3
1.1.6	Segment	3
1.2	Graph	3
1.2.1	Articulations and Bridges	3
1.2.2	Bellman-Ford	3
1.2.3	Bipartite Matching	4
1.2.4	Centroid Decomposition	4
1.2.5	Dijkstra	4
1.2.6	Dinic's	5
1.2.7	Edmonds-Karp	5
1.2.8	Floyd Warshall	5
1.2.9	Ford-Fulkerson	5
1.2.10	Heavy Light Decomposition	6
1.2.11	Hopcroft-Karp	6
1.2.12	Kosaraju	7
1.2.13	Kruskal	7
1.2.14	Lowest Common Ancestor (LCA)	7
1.2.15	Minimum Cost Maximum Flow	8
1.2.16	Prim	8
1.2.17	Steiner Tree	9
1.2.18	Tarjan	9
1.2.19	Topological Sort	9
1.2.20	Travelling Salesman	9
1.3	Math	10
1.3.1	Big Integer	10
1.3.2	Binary Exponentiation	12
1.3.3	Chinese Remainder Theorem	12
1.3.4	Euler Totient (ϕ)	12
1.3.5	Extended Euclidean algorithm	12
1.3.6	Fast Fourier Transform (FFT)	12
1.3.7	Gale-Shapley (Stable Marriage)	13
1.3.8	Karatsuba	13
1.3.9	Legendre's Formula	14
1.3.10	Linear Diophantine Equation	14

CONTENTS	2
1.3.11 Matrix	14
1.3.12 Miller-Rabin primality test	14
1.3.13 Modular Multiplicative Inverse	14
1.3.14 Pollard's Rho	14
1.3.15 Sieve of Eratosthenes	15
1.4 Paradigm	15
1.4.1 Edit Distance	15
1.4.2 Kadane	15
1.4.3 Longest Increasing Subsequence (LIS)	15
1.4.4 Longest Common Subsequence	15
1.4.5 Ternary Search	15
1.5 String	16
1.5.1 Booth's Algorithm	16
1.5.2 Knuth-Morris-Pratt (KMP)	16
1.5.3 Z-function	16
1.6 Structure	16
1.6.1 AVL tree	16
1.6.2 Binary Indexed Tree (BIT)	17
1.6.3 Binary Indexed Tree 2D (BIT2D)	17
1.6.4 Bitmask	17
1.6.5 Disjoint-set	17
1.6.6 Lazy Segment Tree	18
1.6.7 Mo's Algorithm	18
1.6.8 Policy Tree	18
1.6.9 Segment Tree	19
1.6.10 2D Segment Tree	19
1.6.11 Sqrt Decomposition	20
1.6.12 Trie	20
2 Misc	20
2.1 Environment	20
2.1.1 Vim Config	21
2.1.2 Template	21
3 Problems	21
3.1 A Simple Task	21
3.2 Crise Hidrica	22
3.3 Escalacao	23
3.4 Exponial	24
3.5 Trees Partition	25
3.6 XOR submatrix	25

4	Contests	26
4.1	Cadernaveis	26
4.1.1	Bale	26
4.1.2	A Lei vai a Cavalos	27
4.1.3	Xenia and Tree	28
4.1.4	Pedido de Desculpas	29
4.1.5	Easy Sudoku	30
4.1.6	Engarraamento	30
4.1.7	Gincana	31
4.1.8	Janela	31
4.1.9	Crescimento das Populacoes de bacilos	32
4.1.10	Krakovia	35
4.1.11	Pie!	37
4.1.12	Haunted Graveyard	38
4.1.13	Trash Removal	39
4.1.14	Internet Bandwidth	39
4.1.15	Manutencao	40
4.1.16	Mercado do Cairo	41
4.1.17	Nlogonian Tickets	42
4.1.18	Orkut	43
4.1.19	To Poland	44
4.1.20	Serie de Tubos	44
4.1.21	Quantas Chamadas Recursivas	45
4.1.22	Ir e Vir	46
4.1.23	Jogo da Velha	47
4.1.24	Ant's Colony	47
4.1.25	Jupter Ataca!	48
4.1.26	Emoticons	49
4.1.27	Camadas de Cebolas	50
4.1.28	Homem-Elefante-Rato	50
4.1.29	O Labirinto de Ninguem	51
4.1.30	Lobos Stark	52
4.1.31	A Caminhada da Vergonha	53
4.1.32	Bolsa de Valores (iterativo)	53
4.1.33	Bolsa de Valores (recursivo)	54
4.1.34	Colheita de Caju	54
4.1.35	Data Flow	55
4.1.36	I Love Strings! (kmp)	56
4.1.37	Query on a tree	56
4.2	Icpc La16	58
4.2.1	A. Assigning Teams	58
4.2.2	B. Back to the Future	58
4.2.3	D. Dating On-Line	59
4.2.4	F. Farm robot	59

4.2.5	G. Game of Matchings	60
4.2.6	H. Hotel Rewards	60
4.2.7	J. Just in Time	61
4.2.8	K. Kill the Werewolf	61
4.3	Icpc La17	62
4.3.1	B. Buggy ICPC	62
4.3.2	C. Complete Naebbirac's sequence	63
4.3.3	D. Dauting device	63
4.3.4	E. Enigma	64
4.3.5	F. Fundraising	64
4.3.6	G. Gates of uncertainty	65
4.3.7	H. Hard choice	66
4.3.8	I. Imperial roads	66
4.3.9	J. Jumping Frog	68
4.4	Icpc La18	68
4.4.1	A. A Symmetrical Pizza	68
4.4.2	B. Building a Field	69
4.4.3	C. Cheap Trips	69
4.4.4	E. Escape, Polygon!	70
4.4.5	F. Fantastic Beasts	70
4.4.6	H. Highway Decommission	71
4.4.7	I. Ink Colors	72
4.4.8	L. Looking for the Risk Factor	72
4.4.9	M. Mount Marathon	73
4.5	Sbc15	74
4.5.1	A. Mania de Par	74
4.5.2	B. Bolsa de Valores	74
4.5.3	C. Tri-du	75
4.5.4	D. Quebra-cabeca	75
4.5.5	E. Espiral	76
4.5.6	F. Fatorial	76
4.5.7	J. Jogo da Estrategia	77
4.5.8	K. Palindromo	77
4.6	Sbc16	77
4.6.1	A. Andando no Tempo	78
4.6.2	H. huaauhahhuahau	78
4.7	Sbc17	78
4.7.1	A. Acordes Intergalaticos	78
4.7.2	B. Brincadeira	80
4.7.3	C. Cigarras Periodicas	80
4.7.4	D. Despojados	81
4.7.5	E. Escala Musical	81
4.7.6	F. Fase	81
4.7.7	G. Ginastica	82

CONTENTS		5
	4.7.8 H. Hipercampo	82
	4.7.9 I. Imposto Real	83
	4.7.10 J. Jogo de Boca	83
	4.7.11 K. K-esimo	84
	4.7.12 L. Laboratorio de Biotecnologia	84
	4.7.13 M. Maquina de Cafe	86
4.8	Sbc18	86
	4.8.1 B. Bolinhas de Gude	86
	4.8.2 C. Cortador de Pizza	87
	4.8.3 D. Desvendando Monty Hall	87
	4.8.4 E. Enigma	88
	4.8.5 F. Festival	88
	4.8.6 G. Gasolina	89
	4.8.7 I. Interruptores	90
	4.8.8 J. Juntando Capitais	90
	4.8.9 L. Linhas de Metro	91

1 Algorithms

1.1 Geometry

1.1.1 Circle

```
struct Circle {
    Point<> c;
    double r;

    Circle(Point<> c, double r) : c(c), r(r) {}

    // Circumcircle
    Circle(Point<> a, Point<> b, Point<> c) {
        Point<> u((b - a).y, -(b - a).x);
        Point<> v((c - a).y, -(c - a).x);
        Point<> n = (c - b)*0.5;

        double t = u.cross(n) / v.cross(u);

        this->c = (a + c)*0.5 + v*t;
        this->r = dist(this->c, a);
    }

    // Minimum enclosing circle: O(n)
    Circle(vector<Point<>> p) {
        random_shuffle(all(p));
        Circle C(p[0], 0.0);

        for (int i = 0; i < p.size(); ++i) {
            if (C.contains(p[i])) continue;
            C = Circle(p[i], 0.0);

            for (int j = 0; j < i; ++j) {
                if (C.contains(p[j])) continue;
                C = Circle((p[j] + p[i])*0.5, 0.5*dist(p[j], p[i]));

                for (int k = 0; k < j; ++k) {
                    if (C.contains(p[k])) continue;
                    C = Circle(p[j], p[i], p[k]);
                }
            }
        }

        this->c = C.c;
        this->r = C.r;
    }

    bool contains(Point<double> p) {
        return (dist(c, p) <= r + EPS);
    }
};
```

1.1.2 Convex Hull

Time: $\mathcal{O}(n \log n)$

Space: $\mathcal{O}(n)$

```
bool cw(Point<> a, Point<> b, Point<> c) {
    return (b - a).cross(c - a) <= 0;
}

vector<Point<>> convex_hull(vector<Point<>> &v) {
    int k = 0;
    vector<Point<>> ans(v.size() * 2);

    sort(all(v), [](const Point<> &a, const Point<> &b) {
        return (a.x == b.x) ? (a.y < b.y) : (a.x < b.x);
    });

    for (int i = 0; i < v.size(); ++i) {
        for (; k >= 2 && cw(ans[k-2], ans[k-1], v[i]); --k);
        ans[k++] = v[i];
    }

    for (int i = v.size() - 2, t = k + 1; i >= 0; --i) {
        for (; k >= t && cw(ans[k-2], ans[k-1], v[i]); --k);
        ans[k++] = v[i];
    }

    ans.resize(k);
    return ans;
}
```

1.1.3 Point

```
template <typename T = double>
struct Point {
    T x, y;

    Point() {}
    Point(T x, T y) : x(x), y(y) {}

    Point operator+(Point p) { return Point(x+p.x, y+p.y); }
    Point operator-(Point p) { return Point(x-p.x, y-p.y); }
    Point operator*(T s) { return Point(x*s, y*s); }

    T dot(Point p) { return (x*p.x) + (y*p.y); }
    T cross(Point p) { return (x*p.y) - (y*p.x); }

    // Returns angle between this and p:
    // atan2(y, x) is in the range [-180,180]. To
    // get [0, 360], atan2(-y, -x) + 180 is used
    T angle(Point p) {
        return to_deg(atan2(-cross(p), -dot(p))) + 180.0;
    }

    // Returns cosine value between this and p.
    T cosine(Point p) {
        return (dot(p) / (sqrt(dot(*this))*sqrt(p.dot(p))));
    }
}
```

```
// Returns sine value between this and p.
T sine(Point p) {
    return (cross(p) / (sqrt(dot(*this))*sqrt(p.dot(p))));
}

// Returns whether point is inside the triangle
// abc or not.
bool inside_triangle(Point a, Point b, Point c) {
    bool c1 = (*this - b).cross(a - b) < 0;
    bool c2 = (*this - c).cross(b - c) < 0;
    bool c3 = (*this - a).cross(c - a) < 0;
    return c1 == c2 && c1 == c3;
}

// Finds orientation of ordered triplet (a,b,c).
// Colinear (0), Clockwise (1), Counterclockwise (2)
static int orientation(Point<> a, Point<> b, Point<> c) {
    T val = (b - a).cross(c - b);
    if (val == 0) return 0;
    return (val > 0) ? 1 : 2;
}
};
```

1.1.4 Polygon

```
template <typename T = double>
struct Polygon {
    vector<Point<T>> v;

    Polygon() {}
    Polygon(vector<Point<T>> v) : v(v) {}

    // Adds a vertex to the polygon.
    void add_point(Point<T> p) { v.pb(p); }

    // Returns area of polygon (only works when vertices
    // are sorted in clockwise or counterclockwise order).
    double area() {
        double ans = 0;
        for (int i = 0; i < v.size(); ++i)
            ans += v[i].cross(v[(i + 1) % v.size()]);

        return fabs(ans) / 2.0;
    }

    // Rotating Calipers
    double width() {
        vector<Point<>> h = convex_hull(v);

        int n = h.size() - 1;
        double ans = 1e14;

        h[0] = h[n];
        for (int i = 1, j = 1; i <= n; ++i) {
```

```

    while ((h[i] - h[i-1]).cross(h[j%n+1] - h[i-1]) >
           (h[i] - h[i-1]).cross(h[j] - h[i-1]))
        j = j % n + 1;

    Segment<> seg(h[i], h[i-1]);
    ans = min(ans, seg.dist(h[j]));
}

return ans;
}

// Rotating Calipers
double diameter() {
    vector<Point<>> h = convex_hull(v);

    if (h.size() == 1) return 0;
    if (h.size() == 2) return dist(h[0], h[1]);

    int n = h.size() - 1;
    double ans = -1e14;

    h[0] = h[n];
    for (int i = 1, j = 1; i <= n; ++i) {
        while ((h[i] - h[i-1]).cross(h[j%n+1] - h[i-1]) >
               (h[i] - h[i-1]).cross(h[j] - h[i-1]))
            j = j % n + 1;

        ans = max(ans, dist(h[j], h[i]));
        ans = max(ans, dist(h[j], h[i-1]));
    }

    return ans;
}

```

};

1.1.5 Geometry Primitives

```

#define to_deg(x) ((x * 180.0) / M_PI)
#define to_rad(x) ((x * M_PI) / 180.0)

double dist(Point<> a, Point<> b) {
    return hypot(a.x - b.x, a.y - b.y);
}

```

1.1.6 Segment

```

template <typename T = double>
struct Segment {
    Point<T> a, b;

    Segment(Point<T> a, Point<T> b) : a(a), b(b) {}

    // Checks if points p and q are on the same side
    // of the segment.
    bool same_side(Point<T> p, Point<T> q) {
        T cpp = (p - a).cross(b - a);
        T cpq = (q - a).cross(b - a);
        return ((cpp > 0 && cpq > 0) ||
                (cpp < 0 && cpq < 0));
    }
}

```

```

// Checks if point p is on the segment.
bool on_segment(Point<T> p) {
    return (p.x <= max(a.x, b.x) &&
            p.x >= min(a.x, b.x) &&
            p.y <= max(a.y, b.y) &&
            p.y >= min(a.y, b.y));
}

// Distance between segment and point
double dist(Point<T> p) {
    return (a - b).cross(p - b)/sqrt((b - a).dot(b - a));
}

// Checks if segment intersects with s.
bool intersect(Segment<T> s) {
    int o1 = Point<>::orientation( a, b, s.a);
    int o2 = Point<>::orientation( a, b, s.b);
    int o3 = Point<>::orientation(s.a, s.b, a);
    int o4 = Point<>::orientation(s.a, s.b, b);

    if (o1 != o2 && o3 != o4)
        return true;

    if (o1 == 0 && on_segment(s.a)) return true;
    if (o2 == 0 && on_segment(s.b)) return true;
    if (o3 == 0 && s.on_segment(a)) return true;
    if (o4 == 0 && s.on_segment(b)) return true;

    return false;
}
};

```

1.2 Graph

1.2.1 Articulations and Bridges

Time: $\mathcal{O}(V + E)$

Space: $\mathcal{O}(V + E)$

```

vector<int> graph[MAX];

struct ArticulationsBridges {
    int N;
    vector<int> vis, par, L, low;

    vector<ii> brid;
    vector<int> arti;

    ArticulationsBridges(int N) :
        N(N), vis(N), par(N), L(N), low(N)
    { init(); }

    void init() {
        fill(all(L), 0);
        fill(all(vis), 0);
        fill(all(par), -1);
    }

    void dfs(int x) {

```

```

        int child = 0;
        vis[x] = 1;

        for (auto i : graph[x]) {
            if (!vis[i]) {
                child++;
                par[i] = x;

                low[i] = L[i] = L[x] + 1;
                dfs(i);
                low[x] = min(low[x], low[i]);

                if ((par[x] == -1 && child > 1) ||
                    (par[x] != -1 && low[i] >= L[x]))
                    arti.pb(x);

                if (low[i] > L[x])
                    brid.pb(ii(x, i));
            } else if (par[x] != i)
                low[x] = min(low[x], L[i]);
        }

        void run() {

```

```

        for (int i = 0; i < N; ++i)
            if (!vis[i])
                dfs(i);

        sort(all(arti));
        arti.erase(unique(all(arti)), arti.end());
    }
};

```

1.2.2 Bellman-Ford

Time: $\mathcal{O}(V \cdot E)$

Space: $\mathcal{O}(V + E)$

```

struct BellmanFord {
    struct Edge { int u, v, w; };

    int N;
    vector<int> dist;
    vector<Edge> graph;

    BellmanFord(int N) :
        N(N), dist(N)

```

```
{ init(); }

void init() {
    fill(all(dist), inf);
}

void add_edge(int u, int v, int w) {
    graph.pb({ u, v, w });
}

int run(int s, int d) {
    dist[s] = 0;

    for (int i = 0; i < N; ++i)
        for (auto e : graph)
            if (dist[e.u] != inf &&
                dist[e.u] + e.w < dist[e.v])
                dist[e.v] = dist[e.u] + e.w;

    // Check for negative cycles, return -inf if
    // there is one
    for (auto e : graph)
        if (dist[e.u] != inf &&
            dist[e.u] + e.w < dist[e.v])
            return -inf;

    return dist[d];
}
};
```

1.2.3 Bipartite Matching

Time: $\mathcal{O}(V \cdot E)$

Space: $\mathcal{O}(V \cdot E)$

```
vector<int> graph[MAX];

struct BipartiteMatching {
    int N;
    vector<int> vis, match;

    BipartiteMatching(int N) :
        N(N), vis(N), match(N)
    { init(); }

    void init() {
        fill(all(vis), 0);
        fill(all(match), -1);
    }

    int dfs(int x) {
        if (vis[x])
            return 0;

        vis[x] = 1;
        for (auto i : graph[x])
            if (match[i] == -1 || dfs(match[i])) {
                match[i] = x;
                return 1;
            }
    }

    return 0;
}
```

```
int run() {
    int ans = 0;
    for (int i = 0; i < N; ++i)
        ans += dfs(i);

    return ans;
}
};
```

1.2.4 Centroid Decomposition

Description:

The Centroid Decomposition of a tree is a tree where: 1) its root is the centroid of the original tree, and 2) its children are the centroid of each tree resulting from the removal of the root from the original tree.

The result is a tree with $\log n$ height, where the path from a to b , in the original tree, can be decomposed into the path from a to $\text{lca}(a, b)$ and from $\text{lca}(a, b)$ to b .

This is useful because each one of the n^2 paths of the original tree is a concatenation of two paths in a set of $\mathcal{O}(n \log n)$ paths (from each node to all of its ancestors in the centroid decomposition).

Time: $\mathcal{O}(V \log V)$

Space: $\mathcal{O}(V + E)$

```
// Must be a tree
vector<int> graph[MAX];

struct CentroidDecomposition {
    vector<int> par, size, vis;

    CentroidDecomposition(int N) :
        par(N), size(N), vis(N)
    { init(); }

    void init() {
        fill(all(vis), 0);
        build(0); // 0-indexed vertices
    }

    void build(int x, int p = -1) {
        int n = dfs(x);
        int c = get_centroid(x, n);

        vis[c] = 1;
        par[c] = p;

        for (auto i : graph[c])
            if (!vis[i])
                build(i, c);
    }

    // Calculates size of every subtree.
    int dfs(int x, int p = -1) {
        size[x] = 1;
        for (auto i : graph[x])
            if (i != p && !vis[i])
                size[x] += dfs(i, x);
        return size[x];
    }
};
```

```
}

int get_centroid(int x, int n, int p = -1) {
    for (auto i : graph[x])
        if (i != p && size[i] > n / 2 && !vis[i])
            return get_centroid(i, n, x);
    return x;
}

int operator[](int i) {
    return par[i];
}
};
```

1.2.5 Dijkstra

Description:

Dijkstra's algorithm for finding the shortest paths between nodes in a graph. It works by greedily extending the shortest path at each step.

Doesn't work with negative-weighted edges, for that, Bellman-Ford algorithm must be used.

Time: $\mathcal{O}(E + V \log V)$

Space: $\mathcal{O}(V + E)$

```
vector<int> graph[MAX];

struct Dijkstra {
    int N;
    vector<int> dist, vis;

    Dijkstra(int N) :
        N(N), dist(N), vis(N)
    { init(); }

    void init() {
        fill(all(vis), 0);
        fill(all(dist), inf);
    }

    int run(int s, int d) {
        set<ii> pq;

        dist[s] = 0;
        pq.insert(ii(0, s));

        while (pq.size() != 0) {
            int u = pq.begin()->se;
            pq.erase(pq.begin());

            if (vis[u]) continue;
            vis[u] = 1;

            for (auto i : graph[u]) {
                if (!vis[i.fi] && dist[i.fi] > dist[u] + i.se) {
                    dist[i.fi] = dist[u] + i.se;
                    pq.insert(ii(dist[i.fi], i.fi));
                }
            }
        }
    }
};
```



```

    return dist[d];
}
};

```

1.2.6 Dinic's

Time: $\mathcal{O}(E \cdot V^2)$
Space: $\mathcal{O}(V + E)$

```

struct Dinic {
    struct Edge { int u, f, c, r; };

    int N;
    vector<int> depth, start;
    vector<vector<Edge>> graph;

    Dinic(int N) :
        N(N), depth(N), start(N), graph(N) {}

    void add_edge(int u, int v, int c) {
        Edge forw = { v, 0, c, (int) graph[v].size() };
        Edge back = { u, 0, 0, (int) graph[u].size() };
        graph[u].pb(forw);
        graph[v].pb(back);
    }

    bool bfs(int s, int t) {
        queue<int> Q;
        Q.push(s);

        fill(all(depth), -1);
        depth[s] = 0;

        while (!Q.empty()) {
            int v = Q.front(); Q.pop();

            for (auto i : graph[v])
                if (depth[i.u] == -1 && i.f < i.c) {
                    depth[i.u] = depth[v] + 1;
                    Q.push(i.u);
                }
        }

        return depth[t] != -1;
    }

    int dfs(int s, int t, int f) {
        if (s == t)
            return f;

        for (; start[s] < graph[s].size(); ++start[s]) {
            Edge &e = graph[s][start[s]];

            if (depth[e.u] == depth[s] + 1 && e.f < e.c) {
                int min_f = dfs(e.u, t, min(f, e.c - e.f));

                if (min_f > 0) {
                    e.f += min_f;
                    graph[e.u][e.r].f -= min_f;
                    return min_f;
                }
            }
        }
    }
}

```

```

    return 0;
}

int run(int s, int t) {
    int ans = 0;
    while (bfs(s, t)) {
        fill(all(start), 0);
        while (int flow = dfs(s, t, inf))
            ans += flow;
    }

    return ans;
}
};

```

1.2.7 Edmonds-Karp

Time: $\mathcal{O}(V \cdot E^2)$
Space: $\mathcal{O}(V^2)$

```

int rg[MAX][MAX];
int graph[MAX][MAX];

struct EdmondsKarp {
    int N;
    vector<int> par, vis;

    EdmondsKarp(int N) :
        N(N), par(N), vis(N)
    { init(); }

    void init() {
        fill(all(vis), 0);
    }

    bool bfs(int s, int t) {
        queue<int> Q;
        Q.push(s);
        vis[s] = true;

        while (!Q.empty()) {
            int u = Q.front(); Q.pop();

            if (u == t)
                return true;

            for (int i = 0; i < N; ++i)
                if (!vis[i] && rg[u][i]) {
                    vis[i] = true;
                    par[i] = u;
                    Q.push(i);
                }
        }

        return false;
    }

    int run(int s, int t) {
        int ans = 0;
        par[s] = -1;

        memcpy(rg, graph, sizeof(graph));
    }
}

```

```

while (bfs(s, t)) {
    int flow = inf;
    for (int i = t; par[i] != -1; i = par[i])
        flow = min(flow, rg[par[i]][i]);

    for (int i = t; par[i] != -1; i = par[i]) {
        rg[par[i]][i] -= flow;
        rg[i][par[i]] += flow;
    }

    ans += flow;
    init();
}

return ans;
}
};

```

1.2.8 Floyd Warshall

Time: $\mathcal{O}(V^3)$
Space: $\mathcal{O}(V^2)$

```

int dist[MAX][MAX];
int graph[MAX][MAX];

void floyd_warshall(int n) {
    mset(dist, inf);

    for (int i = 0; i < n; ++i)
        for (int j = 0; j < n; ++j)
            if (graph[i][j])
                dist[i][j] = graph[i][j];

    for (int k = 0; k < n; ++k)
        for (int i = 0; i < n; ++i)
            for (int j = 0; j < n; ++j)
                dist[i][j] = min(dist[i][j],
                    dist[i][k] + dist[k][j]);
}

```

1.2.9 Ford-Fulkerson

Time: $\mathcal{O}(E \cdot f)$
Space: $\mathcal{O}(V^2)$

```

int rg[MAX][MAX];
int graph[MAX][MAX];

struct FordFulkerson {
    int N;
    vector<int> par, vis;

    FordFulkerson(int N) :
        N(N), par(N), vis(N)
    { init(); }

    void init() { fill(all(vis), 0); }
}

```

```

bool dfs(int s, int t) {
    vis[s] = true;
    if (s == t)
        return true;

    for (int i = 0; i < N; ++i)
        if (!vis[i] && rg[s][i]) {
            par[i] = s;

            if (dfs(i, t))
                return true;
        }

    return false;
}

int run(int s, int t) {
    int ans = 0;
    par[s] = -1;

    memcpy(rg, graph, sizeof(graph));

    while (dfs(s, t)) {
        int flow = inf;
        for (int i = t; par[i] != -1; i = par[i])
            flow = min(flow, rg[par[i]][i]);

        for (int i = t; par[i] != -1; i = par[i]) {
            rg[par[i]][i] -= flow;
            rg[i][par[i]] += flow;
        }

        ans += flow;
        init();
    }

    return ans;
}
};

```

1.2.10 Heavy Light Decomposition

Description:

The Heavy Light Decomposition splits a tree into several path so that we can reach the root from any node by traversing at most $\log n$ paths. For every vertice v , an edge is called heavy if it leads to a child with the largest subtree size; all other edges are called light.

The set of disjoint paths containing heavy edges are called the “heavy chains”. These paths (both heavy and light) can be indexed by a single segment tree, allowing fast queries and updates to edges (or vertices). And also, information about the chains (such as the head), allows for fast traversals from any node to any of its ancestors.

Time:

- build: $\mathcal{O}(n \log n)$
- query: $\mathcal{O}(\log^2 n)$
- update: $\mathcal{O}(\log n)$

Space: $\mathcal{O}(n \log n)$

Caution:

- Modifications are necessary if values are associated with vertices;

```

ii edge[MAX];
vector<ii> graph[MAX];

template <typename ST>
struct HLD {
    ST &seg;
    int cnum, ptr;

    // depth, parent, value of node
    vector<int> dep, par, val;

    // head[i]: head of i-th node's chain;
    // heavy[i]: "special child" of i-th node
    // pos[i]: position of i-th node on segtree
    // bot[i]: bottommost (depth-wise) node on the i-th
    // edge (required only when edges have weights)
    vector<int> head, heavy, pos, bot;

    HLD(int n, ST &seg) :
        seg(seg),
        dep(n, 0), par(n), val(n),
        head(n), heavy(n, -1), pos(n), bot(n)
    {
        cnum = ptr = 0;

        N = n; // global N for segtree
        dfs(0);
        decompose(0);

        // (required only when edges have weights)
        for (int i = 0; i < n - 1; ++i)
            if (dep[edge[i].fi] > dep[edge[i].se])
                bot[i] = edge[i].fi;
            else
                bot[i] = edge[i].se;
    }

    int dfs(int x, int p = -1) {
        int size = 1;
        par[x] = p;

        int max_size = 0;
        for (auto i : graph[x])
            if (i.fi != p) {
                dep[i.fi] = dep[x] + 1;
                val[i.fi] = i.se;
                int isize = dfs(i.fi, x);

                size += isize;
                if (isize > max_size)
                    max_size = isize, heavy[x] = i.fi;
            }

        return size;
    }

    void decompose(int x, int h = 0) {
        head[x] = h;
        seg.update(ptr, val[x]);
        pos[x] = ptr++;
    }
}

```

```

if (heavy[x] != -1)
    decompose(heavy[x], h);

for (auto i : graph[x])
    if (i.fi != par[x] && i.fi != heavy[x])
        decompose(i.fi, i.fi);
}

// Queries max edge (or vertice) on the path
// between a and b
int query(int a, int b) {
    int ans = seg.idcnt;
    for (; head[a] != head[b]; b = par[head[b]]) {
        if (dep[head[a]] > dep[head[b]])
            swap(a, b);
        ans = seg.func(ans, seg.query(pos[head[b]], pos[b]));
    }

    if (dep[a] > dep[b])
        swap(a, b);

    // Remove "+ 1" when values are associated with vertices
    return seg.func(ans, seg.query(pos[a] + 1, pos[b]));
}

// Updates value of i-th edge (or vertice)
void update(int i, int val) {
    seg.update(pos[bot[i]], val);
}
};

```

1.2.11 Hopcroft-Karp

Time: $\mathcal{O}(E \cdot \sqrt{V})$

Space: $\mathcal{O}(V + E)$

Caution:

- Assumes 1-indexed vertices in graph.

```

vector<int> graph[MAX];

struct HopcroftKarp {
    int L, R;
    vector<int> dist;
    vector<int> matchL, matchR;

    HopcroftKarp(int L, int R) :
        L(L), R(R), dist(L),
        matchL(L), matchR(R)
    { init(); }

    void init() {
        fill(all(matchL), 0);
        fill(all(matchR), 0);
    }

    bool bfs() {
        queue<int> Q;

        for (int l = 1; l <= L; ++l)
            if (matchL[l] == 0) {

```

```

        dist[l] = 0;
        Q.push(l);
    } else
        dist[l] = inf;

    dist[0] = inf;
    while (!Q.empty()) {
        int l = Q.front(); Q.pop();

        if (dist[l] < dist[0])
            for (auto r : graph[l])
                if (dist[matchR[r]] == inf) {
                    dist[matchR[r]] = dist[l] + 1;
                    Q.push(matchR[r]);
                }
    }

    return (dist[0] != inf);
}

bool dfs(int l) {
    if (l == 0)
        return true;

    for (auto r : graph[l])
        if (dist[matchR[r]] == dist[l] + 1)
            if (dfs(matchR[r])) {
                matchR[r] = l;
                matchL[l] = r;
                return true;
            }

    dist[l] = inf;
    return false;
}

int run() {
    int ans = 0;

    while (bfs())
        for (int l = 1; l <= L; ++l)
            if (matchL[l] == 0 && dfs(l))
                ans++;

    return ans;
}
};

```

1.2.12 Kosaraju

Time: $\mathcal{O}(V + E)$
Space: $\mathcal{O}(V + E)$

```
vector<int> graph[MAX];
vector<int> transp[MAX];
```

```
struct Kosaraju {
    int N;
    stack<int> S;
    vector<int> vis;
```

```
Kosaraju(int N) :
    N(N), vis(N)
```

```

{ init(); }

void init() { fill(all(vis), 0); }

void dfs(int x) {
    vis[x] = true;

    for (auto i : transp[x])
        if (!vis[i])
            dfs(i);
}

// Fills stack with DFS starting points to find SCC.
void fill_stack(int x) {
    vis[x] = true;

    for (auto i : graph[x])
        if (!vis[i])
            fill_stack(i);

    S.push(x);
}

int run() {
    int scc = 0;

    init();
    for (int i = 0; i < N; ++i)
        if (!vis[i])
            fill_stack(i);

    // Transpose graph
    for (int i = 0; i < N; ++i)
        for (auto j : graph[i])
            transp[j].push_back(i);

    init();

    // Count SCC
    while (!S.empty()) {
        int v = S.top();
        S.pop();

        if (!vis[v]) {
            dfs(v);
            scc++;
        }
    }

    return scc;
}
};

```

1.2.13 Kruskal

Time: $\mathcal{O}(E \log V)$
Space: $\mathcal{O}(E)$

```
typedef pair<ii,int> iii;
vector<iii> edges;
```

```
struct Kruskal {
    int N;
```

```

    DisjointSet ds;

    Kruskal(int N) : N(N), ds(N) {}

    // Minimum Spanning Tree: comp = less<int>()
    // Maximum Spanning Tree: comp = greater<int>()
    int run(vector<iii> &mst, function<bool(int,int)> comp) {
        sort(all(edges), [&](const iii &a, const iii &b) {
            return comp(a.se, b.se);
        });

        int size = 0;
        for (int i = 0; i < edges.size(); i++) {
            int pu = ds.find_set(edges[i].fi.fi);
            int pv = ds.find_set(edges[i].fi.se);

            if (pu != pv) {
                mst.pb(edges[i]);
                size += edges[i].se;
                ds.union_set(pu, pv);
            }
        }

        return size;
    }
};

```

1.2.14 Lowest Common Ancestor (LCA)

Description:

The LCA between two nodes in a tree is a node that is an ancestor to both nodes with the lowest height possible.

The algorithm works by following the path up the tree from both nodes “simultaneously” until a common node is found. The naive approach for that would be $\mathcal{O}(n)$ in the worst case. To improve that, this implementation uses “binary lifting” which is a way of figuring out the right number of up-moves needed to find the LCA by following the binary representation of the distance to the destination (similar to the “binary search by jumping”), but, for that, a preprocessing must be done to set every parent at a 2^i distance.

Time:

- preprocess: $\mathcal{O}(V \log V)$
- query: $\mathcal{O}(\log V)$

Space: $\mathcal{O}(V + E + V \log V)$

```
#define LOG 20 // log2(MAX)
```

```
vector<ii> graph[MAX];
int par[MAX][LOG], cost[MAX][LOG];
```

```
template <class F = function<T(const T&, const &T)>>
struct LCA {
    F func;
    vector<int> h;

    LCA(int N, F &func) : h(N), func(func)
    { init(); }
```

```

void init() {
    mset(par, -1);
    mset(cost, 0);
    dfs(0); // root is 0
}

void dfs(int v, int p = -1, int c = 0) {
    par[v][0] = p;
    cost[v][0] = c;

    if (p != -1)
        h[v] = h[p] + 1;

    for (int i = 1; i < LOG; ++i)
        if (par[v][i - 1] != -1) {
            par[v][i] = par[par[v][i - 1]][i - 1];
            cost[v][i] = func(cost[v][i], func(cost[v][i - 1],
                cost[par[v][i - 1]][i - 1]));
        }

    for (auto u : graph[v])
        if (p != u.fi)
            dfs(u.fi, v, u.se);
}

int query(int a, int b) {
    int ans = 0;

    if (h[a] < h[b])
        swap(a, b);

    for (int i = LOG - 1; i >= 0; --i)
        if (par[a][i] != -1 && h[par[a][i]] >= h[b]) {
            ans = func(ans, cost[a][i]);
            a = par[a][i];
        }

    if (a == b) {
#ifdef COST
        return ans;
#else
        return a;
#endif
    }

    for (int i = LOG - 1; i >= 0; --i)
        if (par[a][i] != -1 && par[a][i] != par[b][i]) {
            ans = func(ans, func(cost[a][i], cost[b][i]));
            a = par[a][i];
            b = par[b][i];
        }

#ifdef COST
    if (a == b)
        return ans;
    else
        return func(ans, func(cost[a][0], cost[b][0]));
#else
    return par[a][0];
#endif
}
};

```

1.2.15 Minimum Cost Maximum Flow

Time: $\mathcal{O}(V^2 \cdot E)$

Space: $\mathcal{O}(V + E)$

```

struct MinCostMaxFlow {
    struct Edge { int u, v, cap, cost; };

    vector<Edge> edges;
    vector<vector<int>> adj;
    vector<int> vis, dist, par, ind;

    MinCostMaxFlow(int N) :
        vis(N), dist(N), par(N), ind(N), adj(N) {}

    void add_edge(int u, int v, int cap, int cost) {
        adj[u].pb(edges.size());
        edges.pb({ u, v, cap, cost });

        adj[v].pb(edges.size());
        edges.pb({ v, u, 0, -cost });
    }

    // Shortest Path Faster Algorithm (slower than
    // Dijkstra but works with negative edges).
    bool spfa(int s, int t) {
        fill(all(dist), inf);
        dist[s] = 0;

        queue<int> Q;
        Q.push(s);

        while (!Q.empty()) {
            int u = Q.front(); Q.pop();
            vis[u] = 0;

            for (auto i : adj[u]) {
                Edge &e = edges[i];
                int v = e.v;

                if (e.cap > 0 && dist[v] > dist[u] + e.cost) {
                    dist[v] = dist[u] + e.cost;
                    par[v] = u;
                    ind[v] = i;

                    if (!vis[v]) {
                        Q.push(v);
                        vis[v] = 1;
                    }
                }
            }
        }

        return dist[t] < inf;
    }

    // Returns pair (min_cost, max_flow).
    ii run(int s, int t) {
        int min_cost = 0;
        int max_flow = 0;

        while (spfa(s, t)) {
            int flow = inf;
            for (int i = t; i != s; i = par[i])
                flow = min(flow, edges[ind[i]].cap);

```

```

        for (int i = t; i != s; i = par[i]) {
            edges[ind[i]].cap -= flow;
            edges[ind[i]^1].cap += flow;
        }

        min_cost += flow * dist[t];
        max_flow += flow;
    }

    return ii(min_cost, max_flow);
}
};

```

1.2.16 Prim

Time: $\mathcal{O}(E \log E)$

Space: $\mathcal{O}(V + E)$

```

vector<ii> graph[MAX];

struct Prim {
    int N;
    vector<int> vis;

    Prim(int N) :
        N(N), vis(N)
    { init(); }

    void init() {
        fill(all(vis), 0);
    }

    int run() {
        vis[0] = true;

        priority_queue<ii> pq;
        for (auto i : graph[0])
            pq.push(ii(-i.se, -i.fi));

        int ans = 0;
        while (!pq.empty()) {
            ii front = pq.top(); pq.pop();
            int u = -front.se;
            int w = -front.fi;

            if (!vis[u]) {
                ans += w;
                vis[u] = true;

                for (auto i : graph[u])
                    if (!vis[i.fi])
                        pq.push(ii(-i.se, -i.fi));
            }
        }

        return ans;
    }
};

```

1.2.17 Steiner Tree

Description:

A (minimum) Steiner tree is a tree that, given a graph G and a set of terminal vertices T (inside the graph), connects all vertices in T using other vertices in G if necessary, minimizing the sum of weights of the included edges.

The algorithm works by using dynamic programming, where $dp[i][mask]$ stores the value of a Steiner tree rooted at i containing the terminal nodes represented by the bits in bitmask. The algorithm iterates over all bitmasks, and, at each iteration $mask$, every pair of complementary subsets are tested and $dp[i][mask]$ is updated with the value given by the combination of both trees. Then, still at step $mask$, $dp[j][mask]$ is updated for every j with the “extension” of i (for every i), as if the root was moving around.

The result must be retrieved from a node (root of the final Steiner tree) that contains all terminal nodes and has the smallest value.

Time: $\mathcal{O}(n^2 \cdot 2^t + n \cdot 3^t)$

Space: $\mathcal{O}(n \cdot 2^t + n^2)$

```
int dist[MAXN][MAXN];
int graph[MAXN][MAXN];
int dp[MAXN][1 << MAXT];

int steiner_tree(int n, int t) {
    floyd_warshall(n);

    mset(dp, inf);
    for (int i = 0; i < t; ++i)
        dp[i][1 << i] = 0;

    for (int mask = 1; mask < (1 << t); ++mask) {
        for (int i = 0; i < n; ++i)
            for (int ss = mask; ss > 0; ss = (ss - 1) & mask)
                dp[i][mask] = min(dp[i][mask],
                                   dp[i][ss] + dp[i][mask ^ ss]);

        for (int i = 0; i < n; ++i)
            for (int j = 0; j < n; ++j)
                dp[i][mask] = min(dp[j][mask],
                                   dp[i][mask] + dist[i][j]);
    }

    int ans = inf;
    for (int i = 0; i < n; ++i)
        ans = min(ans, dp[i][(1 << t) - 1]);
    return ans;
}
```

1.2.18 Tarjan

Time: $\mathcal{O}(V + E)$

Space: $\mathcal{O}(V + E)$

```
vector<int> scc[MAX];
vector<int> graph[MAX];
```

```
struct Tarjan {
    int N, ncomp, ind;
```

```
    stack<int> S;
    vector<int> vis, id, low;

    Tarjan(int N) :
        N(N), vis(N), id(N), low(N)
    { init(); }

    void init() {
        fill(all(id), -1);
        fill(all(vis), 0);
    }

    void dfs(int x) {
        id[x] = low[x] = ind++;
        vis[x] = 1;

        S.push(x);

        for (auto i : graph[x])
            if (id[i] == -1) {
                dfs(i);
                low[x] = min(low[x], low[i]);
            } else if (vis[i])
                low[x] = min(low[x], id[i]);

        // A SCC was found
        if (low[x] == id[x]) {
            int w;

            do {
                w = S.top(); S.pop();
                vis[w] = 0;
                scc[ncomp].pb(w);
            } while (w != x);

            ncomp++;
        }
    }

    int run() {
        init();
        ncomp = ind = 0;

        for (int i = 0; i < N; ++i)
            scc[i].clear();

        // Apply tarjan in every component
        for (int i = 0; i < N; ++i)
            if (id[i] == -1)
                dfs(i);

        return ncomp;
    }
};
```

1.2.19 Topological Sort

Time: $\mathcal{O}(V + E)$

Space: $\mathcal{O}(V + E)$

```
vector<int> graph[MAX];
```

```
struct TopologicalSort {
    int N;
    stack<int> S;
    vector<int> vis;

    TopologicalSort(int N) :
        N(N), vis(N)
    { init(); }

    void init() { fill(all(vis), 0); }

    bool dfs(int x) {
        vis[x] = 1;

        for (auto i : graph[x]) {
            if (vis[i] == 1) return true;
            if (!vis[i] && dfs(i)) return true;
        }

        vis[x] = 2;
        S.push(x);

        return false;
    }

    // Returns whether graph contains cycle
    // or not.
    bool run(vector<int> &tsort) {
        init();

        bool cycle = false;
        for (int i = 0; i < N; ++i)
            if (!vis[i])
                cycle |= dfs(i);

        if (cycle)
            return true;

        while (!S.empty()) {
            tsort.pb(S.top());
            S.pop();
        }

        return false;
    }
};
```

1.2.20 Travelling Salesman

Description:

Given a graph and an origin vertex, this algorithm return the shortest possible route that visits each vertex and returns to the origin.

The algorithm works by using dynamic programming, where $dp[i][mask]$ stores the last visited vertex i and a set of visited vertices represented by a bitmask $mask$. Given a state, the next vertex in the path is chosen by a recursive call, until the bitmask is full, in which case the weight of the edge between the last vertex and the origin is returned.

Time: $\mathcal{O}(2^n \cdot n^2)$
Space: $\mathcal{O}(2^n \cdot n)$

```
int dp[MAX][1 << MAX];
int graph[MAX][MAX];
```

```
struct TSP {
    int N;

    TSP(int N) : N(N)
    { init(); }
```

```
void init() { mset(dp, -1); }

int solve(int i, int mask) {
    if (mask == (1 << N) - 1)
        return graph[i][0];

    if (dp[i][mask] != -1)
        return dp[i][mask];

    int ans = inf;
    for (int j = 0; j < N; ++j)
        if (!(mask & (1 << j)) && (i != j))
```

```
        ans = min(ans, graph[i][j] +
            solve(j, mask | (1 << j)));

    return dp[i][mask] = ans;
}

int run(int start) {
    return solve(start, 1 << start);
}
};
```

1.3 Math

1.3.1 Big Integer

Space: $\mathcal{O}(n)$

Caution:

- Just use Python if possible.

```
const int base = 1000000000;
const int base_d = 9;
```

```
struct BigInt {
    int sign;
    vector<int> num;

    BigInt() : sign(1) {}
    BigInt(ll x) { *this = x; }
    BigInt(const string &s) { read(s); }
```

```
void operator=(const BigInt &x) {
    sign = x.sign;
    num = x.num;
}
```

```
void operator=(ll x) {
    sign = 1;
    if (x < 0) sign = -1, x = -x;
    for (; x > 0; x /= base)
        pb(x % base);
}
```

```
BigInt operator+(const BigInt &x) const {
    if (sign != x.sign) return *this - (-x);
```

```
    int carry = 0;
    BigInt res = x;

    for (int i = 0; i < max(size(), x.size()) || carry; ++i) {
        if (i == (int) res.size())
            res.push_back(0);

        res[i] += carry + (i < size() ? num[i] : 0);
        carry = res[i] >= base;
        if (carry) res[i] -= base;
    }

    return res;
```

```
    }

    BigInt operator-(const BigInt &x) const {
        if (sign != x.sign) return *this + (-x);
        if (abs() < x.abs()) return -(x - *this);

        int carry = 0;
        BigInt res = *this;

        for (int i = 0; i < x.size() || carry; ++i) {
            res[i] -= carry + (i < x.size() ? x[i] : 0);
            carry = res[i] < 0;
            if (carry) res[i] += base;
        }

        res.trim();
        return res;
    }
```

```
void operator*=(int x) {
    if (x < 0) sign = -sign, x = -x;

    int carry = 0;
    for (int i = 0; i < size() || carry; ++i) {
        if (i == size()) pb(0);
        ll cur = num[i] * (ll) x + carry;

        carry = (int) (cur / base);
        num[i] = (int) (cur % base);
    }
}
```

```
    trim();
}
```

```
BigInt operator*(int x) const {
    BigInt res = *this;
    res *= x;
    return res;
}

friend pair<BigInt, BigInt> divmod(const BigInt &a1,
    const BigInt &b1)
{
    int norm = base / (b1.back() + 1);
    BigInt a = a1.abs() * norm;
    BigInt b = b1.abs() * norm;
```

```
    BigInt q, r;
    q.resize(a.size());

    for (int i = a.size() - 1; i >= 0; i--) {
        r *= base;
        r += a[i];

        int s1 = r.size() <= b.size() ? 0 : r[b.size()];
        int s2 = r.size() <= b.size() - 1 ? 0 : r[b.size() - 1];
        int d = ((ll) base * s1 + s2) / b.back();

        r -= b * d;
        while (r < 0) r += b, --d;
        q[i] = d;
    }

    q.sign = a1.sign * b1.sign;
    r.sign = a1.sign;
    q.trim(); r.trim();

    return make_pair(q, r / norm);
}

BigInt operator/(const BigInt &x) const {
    return divmod(*this, x).fi;
}

BigInt operator%(const BigInt &x) const {
    return divmod(*this, x).se;
}

void operator/=(int x) {
    if (x < 0) sign = -sign, x = -x;

    for (int i = size() - 1, rem = 0; i >= 0; --i) {
        ll cur = num[i] + rem * (ll) base;
        num[i] = (int) (cur / x);
        rem = (int) (cur % x);
    }

    trim();
}

BigInt operator/(int x) const {
    BigInt res = *this;
    res /= x;
```

```

    return res;
}

int operator%(int x) const {
    if (x < 0) x = -x;

    int m = 0;
    for (int i = size() - 1; i >= 0; --i)
        m = (num[i] + m * (11) base) % x;

    return m * sign;
}

void operator+=(const BigInt &x) { *this = *this + x; }
void operator-=(const BigInt &x) { *this = *this - x; }
void operator*=(const BigInt &x) { *this = *this * x; }
void operator/=(const BigInt &x) { *this = *this / x; }

bool operator<(const BigInt &x) const {
    if (sign != x.sign)
        return sign < x.sign;

    if (size() != x.size())
        return size() * sign < x.size() * x.sign;

    for (int i = size() - 1; i >= 0; i--)
        if (num[i] != x[i])
            return num[i] * sign < x[i] * sign;

    return false;
}

bool operator>(const BigInt &x) const {
    return x < *this;
}

bool operator<=(const BigInt &x) const {
    return !(x < *this);
}

bool operator>=(const BigInt &x) const {
    return !(*this < x);
}

bool operator==(const BigInt &x) const {
    return !(*this < x) && !(x < *this);
}

bool operator!=(const BigInt &x) const {
    return *this < x || x < *this;
}

void trim() {
    while (!empty() && !back()) pop_back();
    if (empty()) sign = 1;
}

bool is_zero() const {
    return empty() || (size() == 1 && !num[0]);
}

BigInt operator-() const {
    BigInt res = *this;
    res.sign = -sign;
    return res;
}

BigInt abs() const {
    BigInt res = *this;

```

```

    res.sign *= res.sign;
    return res;
}

ll to_long() const {
    ll res = 0;
    for (int i = size() - 1; i >= 0; i--)
        res = res * base + num[i];
    return res * sign;
}

friend BigInt gcd(const BigInt &a, const BigInt &b) {
    return b.is_zero() ? a : gcd(b, a % b);
}

friend BigInt lcm(const BigInt &a, const BigInt &b) {
    return a / gcd(a, b) * b;
}

void read(const string &s) {
    sign = 1;
    num.clear();

    int pos = 0;
    while (pos < s.size() &&
           (s[pos] == '-' || s[pos] == '+')) {
        if (s[pos] == '-')
            sign = -sign;
        ++pos;
    }

    for (int i = s.size() - 1; i >= pos; i -= base_d) {
        int x = 0;
        for (int j = max(pos, i - base_d + 1); j <= i; j++)
            x = x * 10 + s[j] - '0';
        num.push_back(x);
    }

    trim();
}

friend istream& operator>>(istream &stream, BigInt &v) {
    string s; stream >> s;
    v.read(s);
    return stream;
}

friend ostream& operator<<(ostream &stream, const BigInt &x)
{
    if (x.sign == -1)
        stream << '-';

    stream << (x.empty() ? 0 : x.back());
    for (int i = x.size() - 2; i >= 0; --i)
        stream << setw(base_d) << setfill('0') << x.num[i];

    return stream;
}

static vector<int> convert_base(
    const vector<int> &a,
    int oldd, int newd) {
    vector<ll> p(max(oldd, newd) + 1);
    p[0] = 1;
    for (int i = 1; i < p.size(); i++)

```

```

        p[i] = p[i - 1] * 10;

    ll cur = 0;
    int curd = 0;
    vector<int> res;

    for (int i = 0; i < a.size(); i++) {
        cur += a[i] * p[curd];
        curd += oldd;

        while (curd >= newd) {
            res.pb(int(cur % p[newd]));
            cur /= p[newd];
            curd -= newd;
        }

        res.pb((int) cur);
        while (!res.empty() && !res.back())
            res.pop_back();
        return res;
    }

    BigInt operator*(const BigInt &x) const {
        vector<int> a6 = convert_base(this->num, base_d, 6);
        vector<int> b6 = convert_base(x.num, base_d, 6);

        vector<ll> a(all(a6));
        vector<ll> b(all(b6));

        while (a.size() < b.size()) a.pb(0);
        while (b.size() < a.size()) b.pb(0);
        while (a.size() & (a.size() - 1))
            a.pb(0), b.pb(0);

        vector<ll> c = karatsuba(a, b);

        BigInt res;
        int carry = 0;
        res.sign = sign * x.sign;

        for (int i = 0; i < c.size(); i++) {
            ll cur = c[i] + carry;
            res.pb((int) (cur % 1000000));
            carry = (int) (cur / 1000000);
        }

        res.num = convert_base(res.num, 6, base_d);
        res.trim();
        return res;
    }

    // Handles vector operations.
    int back() const { return num.back(); }
    bool empty() const { return num.empty(); }
    size_t size() const { return num.size(); }

    void pop_back() { num.pop_back(); }
    void resize(int x) { num.resize(x); }
    void push_back(int x) { num.push_back(x); }

    int &operator[](int i) { return num[i]; }
    int operator[](int i) const { return num[i]; }
};

```

1.3.2 Binary Exponentiation

Description:

Computes fast exponentiation by looking at the binary representation of the exponent. The usage of “bin_mul” is not necessary, but it ensures that no overflow will occur when multiplying two large integers. It is definitely required in order to work with Miller-Rabin and Pollard’s Rho implementations (when dealing with numbers that might go up to 10^{18}).

Time: $\mathcal{O}(\log n)$

Space: $\mathcal{O}(1)$

```
ll bin_mul(ll a, ll b, ll M = MOD) {
    ll x = 0;
    a %= M;
    while (b) {
        if (b & 1) x = (x + a) % M;
        b >>= 1;
        a = (a * 2) % M;
    }
    return x % M;
}

ll bin_exp(ll a, ll e, ll M = MOD) {
    ll x = 1;
    while (e) {
        if (e & 1) x = bin_mul(x, a, M);
        e >>= 1;
        a = bin_mul(a, a, M);
    }
    return x % M;
}
```

1.3.3 Chinese Remainder Theorem

Description:

Given t linear congruences in the format:

$$x \equiv a_i \pmod{m_i}$$

the Chinese Remainder Theorem (CRT) finds x that satisfies every given congruence or states that there are no solutions.

This implementation does not require all m_i to be coprime, it works with any value and returns the smallest possible x ; infinitely many solutions can be obtained by incrementing or decrementing $LCM(m_1, m_2, \dots, m_t)$ from x .

Time: $\mathcal{O}(t \log LCM(m_1, m_2, \dots, m_t))$

Space: $\mathcal{O}(t)$

Caution:

- It is very easy to get overflow, since the LCM is computed from all m_i , BigInt or Python should be considered if inputs are too large

```
ll norm(ll a, ll b) {
    a %= b;
    return (a < 0) ? a + b : a;
}
```

```
}

pair<ll, ll> crt_single(ll a, ll n, ll b, ll m) {
    ans_t e = ext_gcd(n, m);

    if ((a - b) % e.d != 0)
        return {-1, -1}; // No solution

    ll lcm = (m/e.d) * n;
    ll ans = norm(a + e.x*(b-a) / e.d % (m/e.d)*n, lcm);
    return {norm(ans, lcm), lcm};
}

ll crt(vector<ll> a, vector<ll> m) {
    ll ans = a[0];
    ll lcm = m[0];

    int t = a.size();
    for (int i = 1; i < t; ++i) {
        auto ss = crt_single(ans, lcm, a[i], m[i]);
        if (ss.fi == -1)
            return -1; // No solution

        ans = ss.fi;
        lcm = ss.se;
    }

    return ans;
}
```

1.3.4 Euler Totient (ϕ)

Time: $\mathcal{O}(\sqrt{n})$

Space: $\mathcal{O}(1)$

```
int phi(int n) {
    int result = n;

    for (int i = 2; i*i <= n; i++)
        if (n % i == 0) {
            while (n % i == 0) n /= i;
            result -= result / i;
        }

    if (n > 1)
        result -= (result / n);

    return result;
}
```

1.3.5 Extended Euclidean algorithm

Time: $\mathcal{O}(\log \min(a, b))$

Space: $\mathcal{O}(1)$

```
struct ans_t { ll x, y, d; };

ans_t ext_gcd(ll a, ll b) {
    if (a == 0) return {0, 1, b};
    ans_t e = ext_gcd(b % a, a);
```

```
    return {e.y - (b/a)*e.x, e.x, e.d};
}
```

1.3.6 Fast Fourier Transform (FFT)

Time: $\mathcal{O}(N \log N)$

Space: $\mathcal{O}(N)$

```
struct FFT {
    struct Complex {
        float r, i;

        Complex() : r(0), i(0) {}
        Complex(float r, float i) : r(r), i(i) {}

        Complex operator+(Complex b) {
            return Complex(r + b.r, i + b.i);
        }

        Complex operator-(Complex b) {
            return Complex(r - b.r, i - b.i);
        }

        Complex operator*(Complex b) {
            return Complex(r*b.r - i*b.i, r*b.i + i*b.r);
        }

        Complex operator/(Complex b) {
            float div = (b.r * b.r) + (b.i * b.i);
            return Complex((r * b.r + i * b.i) / div,
                           (i * b.r - r * b.i) / div);
        }

        static inline Complex conj(Complex a) {
            return Complex(a.r, -a.i);
        }
    };

    vector<int> rev = {0, 1};
    vector<Complex> roots = {{0, 0}, {1, 0}};

    // Initializes reversed-bit vector (rev) and
    // roots of unity vector (roots)
    void init(int nbase) {
        rev.resize(1 << nbase);
        roots.resize(1 << nbase);

        // Build rev vector
        for (int i = 0; i < (1 << nbase); ++i)
            rev[i] = (rev[i >> 1] >> 1) + \
                ((i & 1) << (nbase - 1));

        // Build roots vector
        for (int base = 1; base < nbase; ++base) {
            float angle = 2 * M_PI / (1 << (base + 1));

            for (int i = 1 << (base - 1); i < (1 << base); ++i) {
                float angle_i = angle * (2*i + 1 - (1 << base));

                roots[i << 1] = roots[i];
                roots[(i << 1) + 1] = Complex(cos(angle_i),
                                                sin(angle_i));
            }
        }
    }
};
```



```

    }
}

void fft(vector<Complex> &a) {
    int n = a.size();

    for (int i = 0; i < n; ++i)
        if (i < rev[i])
            swap(a[i], a[rev[i]]);

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

vector<int> multiply(const vector<int> &a,
                   const vector<int> &b)
{
    int nbase, need = a.size() + b.size() + 1;

    for (nbase = 0; (1 << nbase) < need; ++nbase);
    init(nbase);

    int size = 1 << nbase;
    vector<Complex> fa(size);

    for (int i = 0; i < size; ++i) {
        int x = (i < a.size() ? a[i] : 0);
        int y = (i < b.size() ? b[i] : 0);
        fa[i] = Complex(x, y);
    }

    fft(fa);

    Complex r(0, -0.25 / size);
    for (int i = 0; i <= (size >> 1); ++i) {
        int j = (size - i) & (size - 1);
        Complex 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);

    vector<int> res(need);
    for (int i = 0; i < need; ++i)
        res[i] = fa[i].r + 0.5;

    return res;
}
};

```

1.3.7 Gale-Shapley (Stable Marriage)

Description:

Two groups, each of size N are given: men and women. Each person has a list of preference ranking all N people of the opposite sex. The task is to unite both groups into stable pairs. A set of pairs is stable if there are no unassigned couple that like each other more than their assigned pair.

The algorithm's steps are: 1) allow every men to propose to their highest ranking woman; 2) the women become tentatively engaged to their top choice of men; 3) all rejected men propose to their next choice; 4) the woman replaces their current pair in case a man with a higher rank proposes to her, the replaced men are now marked as rejected; 5) repeat step 3 until all men are paired.

The result is guaranteed to return a configuration where every man gets the best possible wife, while every woman gets the worst possible husband (it benefits the group that chooses first).

Time: $\mathcal{O}(n^2)$

Space: $\mathcal{O}(n^2)$

Caution:

- Men are indexed by $[0, N)$
- The result prioritizes men, swapping the bottom half of the matrix by the top half will invert who's given preference

*// Receives matrix of preferences pref[2*N][N] and returns
// vector v where v[m] contains preference of the m-th man.*

```

vector<int> gale_shapley(const vector<vector<int>> &pref) {
    int n = pref[0].size();
    vector<int> w_part(n, -1);
    vector<int> m_part(n, -1);
    vector<int> start(n, 0);

    while (true) {
        int m;
        for (m = 0; m < n; ++m)
            if (m_part[m] == -1)
                break;

        if (m == n) break;

        for (; start[m] < n && m_part[m] == -1; ++start[m]) {
            int w = pref[m][start[m]];

            if (w_part[w - n] == -1) {
                w_part[w - n] = m;
                m_part[m] = w;
            } else {
                int m1 = w_part[w - n];
                bool pref_m = false;

                for (int j = 0; j < n; ++j)
                    if (pref[w][j] == m) {
                        pref_m = true;
                        break;
                    } else if (pref[w][j] == m1)
                        break;

                if (pref_m) {
                    w_part[w - n] = m;

```

```

                m_part[m] = w;
                m_part[m1] = -1;
            }
        }
    }

    return m_part;
}

```

1.3.8 Karatsuba

Time: $\mathcal{O}(n^{\log(3)})$

Space: $\mathcal{O}(n)$

```

vector<ll> karatsuba(const vector<ll> &a,
                   const vector<ll> &b)
{
    int n = a.size();
    vector<ll> res(n + n);

    if (n <= 32) {
        for (int i = 0; i < n; ++i)
            for (int j = 0; j < n; ++j)
                res[i + j] += a[i] * b[j];

        return res;
    }

    int k = n >> 1;
    vector<ll> a1(a.begin(), a.begin() + k);
    vector<ll> a2(a.begin() + k, a.end());
    vector<ll> b1(b.begin(), b.begin() + k);
    vector<ll> b2(b.begin() + k, b.end());

    vector<ll> a1b1 = karatsuba(a1, b1);
    vector<ll> a2b2 = karatsuba(a2, b2);

    for (int i = 0; i < k; ++i)
        a2[i] += a1[i];
    for (int i = 0; i < k; ++i)
        b2[i] += b1[i];

    vector<ll> r = karatsuba(a2, b2);
    for (int i = 0; i < a1b1.size(); ++i)
        r[i] -= a1b1[i];
    for (int i = 0; i < a2b2.size(); ++i)
        r[i] -= a2b2[i];

    for (int i = 0; i < r.size(); ++i)
        res[i + k] += r[i];
    for (int i = 0; i < a1b1.size(); ++i)
        res[i] += a1b1[i];
    for (int i = 0; i < a2b2.size(); ++i)
        res[i + n] += a2b2[i];

    return res;
}

```

1.3.9 Legendre’s Formula

Description:
Given an integer n and a prime number p , find the largest power of p (x) such that $n!$ is divisible by k^x .
Writing $n!$ as $1 \cdot 2 \cdot 3 \dots (n-1) \cdot n$ shows that every k -th element of the product is divisible by k (i.e. adds 1 to the answer), the number of such elements is $\lfloor \frac{n}{k} \rfloor$. The same is true for every k^i , thus, the answer is given by:

$$\lfloor \frac{n}{k} \rfloor + \lfloor \frac{n}{k^2} \rfloor + \dots + \lfloor \frac{n}{k^i} \rfloor$$

only approximately the first $\log_p n$ elements are greater than zero, therefore the sum is finite.

Time: $\mathcal{O}(\log_p n)$
Space: $\mathcal{O}(1)$

```
int legendre(int n, int p) {
    int x = 0;

    while (n) {
        n /= p;
        x += n;
    }

    return x;
}
```

1.3.10 Linear Diophantine Equation

Description:
A Linear Diophantine Equation is an equation in the form

$$ax + by = c$$

A solution of this equation is a pair (x, y) that satisfies the equation. The locus of (lattice) points whose coordinates x and y satisfy the equation is a straigh line.
The equation has a solution only if $\gcd(a, b) | c$. In the case of existing a solution for the provided a, b, c , the infinite set of coordinates (x, y) can be obtained with $\text{get}(t)$ for $t = \dots, -2, -1, 0, 1, 2, \dots$

Time: $\mathcal{O}(\log \min(a, b))$
Space: $\mathcal{O}(1)$

```
struct Diophantine {
    int a, b, c, d;
    int x0, y0;

    bool has_solution;

    Diophantine(int a, int b, int c) :
        a(a), b(b), c(c)
    { init(); }

    void init() {
        ans_t e = ext_gcd(a, b); d = e.d;
        if (c % d == 0) {
            x0 = e.x * (c / d);

```

```
        y0 = e.y * (c / d);
        has_solution = true;
    } else
        has_solution = false;
}

ii get(int t) {
    if (!has_solution) return ii(inf, inf);
    return ii(x0 + t * (b / d), y0 - t * (a / d));
}
};
```

1.3.11 Matrix

Space: $\mathcal{O}(R \cdot C)$

```
template <typename T>
struct Matrix {
    int r, c;
    vector<vector<T>> m;

    Matrix(int r, int c) : r(r), c(c) {
        m = vector<vector<T>>(r, vector<T>(c, 0));
    }

    Matrix operator*(Matrix a) {
        assert(r == a.c && c == a.r);

        Matrix res(r, c);
        for (int i = 0; i < r; i++)
            for (int j = 0; j < c; j++) {
                res[i][j] = 0;

                for (int k = 0; k < c; k++)
                    res[i][j] += m[i][k] * a[k][j];
            }

        return res;
    }

    vector<T> &operator[](int i) {
        return m[i];
    }
};
```

1.3.12 Miller-Rabin primality test

Time: $\mathcal{O}(k \cdot \log^3 n)$
Space: $\mathcal{O}(1)$

```
const vector<ll> A = {
    2, 325, 9375, 28178, 450775, 9780504, 1795265022
};

bool is_prime(ll n) {
    if (n < 2 || n % 6 % 4 != 1)
        return n - 2 < 2;

    ll s = __builtin_ctzll(n - 1);
    ll d = n >> s;
```

```
for (auto a : A) {
    ll p = bin_exp(a, d, n), i = s;
    while (p != 1 && p != n - 1 && a % n && i--)
        p = bin_mul(p, p, n);
    if (p != n - 1 && i != s)
        return 0;
}

return 1;
}
```

1.3.13 Modular Multiplicative Inverse

Description:
Given integers a and m , the modular multiplicative inverse of a is an integer x such that

$$a \cdot x \equiv 1 \pmod{m}$$

also denoted as a^{-1} .

Time: $\mathcal{O}(\log m)$
Space: $\mathcal{O}(1)$

```
// Fermat's Little Theorem: Used when m is prime
int fermat(int a, int m) {
    return bin_exp(a, m - 2);
}

// Extended Euclidean Algorithm: Used when m
// and a are coprime
int extended_euclidean(int a, int m) {
    ans_t g = ext_gcd(a, m);
    return (g.x % m + m) % m;
}
```

1.3.14 Pollard’s Rho

Description:
Returns prime factorization of n .

Time: $\mathcal{O}(n^{1/4})$
Space: $\mathcal{O}(1)$

```
ll pollard(ll n) {
    auto f = [n](ll x) {
        return (bin_mul(x, x, n) + 1) % n;
    };

    if (n % 2 == 0) return 2;

    for (ll i = 2;; ++i) {
        ll x = i, y = f(x), p;
        while ((p = __gcd(n + y - x, n)) == 1)
            x = f(x), y = f(f(y));
        if (p != n) return p;
    }
}
```

```
vector<ll> factor(ll n) {
    if (n == 1) return {};
    if (is_prime(n)) // Use Miller-Rabin
        return {n};

    ll x = pollard(n);
    auto l = factor(x);
    auto r = factor(n/x);

    l.insert(l.end(), all(r));
    return l;
}
```

1.3.15 Sieve of Eratosthenes

Time: $\mathcal{O}(n \cdot \log \log n)$

Space: $\mathcal{O}(n)$

```
vector<int> sieve(int n) {
    vector<int> primes;
    vector<int> is_prime(n + 1, 1);
```

```
    for (int p = 2; p*p <= n; ++p)
        if (is_prime[p])
            for (int i = p*p; i <= n; i += p)
                is_prime[i] = false;

    for (int p = 2; p <= n; ++p)
        if (is_prime[p])
            primes.pb(p);

    return primes;
}
```

1.4 Paradigm

1.4.1 Edit Distance

Time: $\mathcal{O}(m \cdot n)$

Space: $\mathcal{O}(m \cdot n)$

```
int dp[MAX][MAX];

int edit_distance(string a, string b) {
    for (int i = 0; i <= a.size(); ++i)
        for (int j = 0; j <= b.size(); ++j)
            if (i == 0)
                dp[i][j] = j;
            else if (j == 0)
                dp[i][j] = i;
            else if (a[i-1] == b[j-1])
                dp[i][j] = dp[i-1][j-1];
            else
                dp[i][j] = 1 + min({dp[i][j-1],
                                   dp[i-1][j],
                                   dp[i-1][j-1]});

    return dp[a.size()][b.size()];
}
```

```
        s = i + 1;
    }
}

return msf;
}
```

1.4.3 Longest Increasing Subsequence (LIS)

Time: $\mathcal{O}(n^2)$

Space: $\mathcal{O}(n)$

```
int lis(vector<int> v) {
    vector<int> lis(v.size()); lis[0] = 1;

    for (int i = 1; i < v.size(); ++i) {
        lis[i] = 1;

        for (int j = 0; j < i; ++j)
            if (v[i] > v[j] && lis[i] < lis[j] + 1)
                lis[i] = lis[j] + 1;
    }

    return *max_element(all(lis));
}
```

```
        dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
    }
}

// The size is already at dp[n][m], now the common
// subsequence is retrieved:

int idx = dp[a.size()][b.size()];
string ans(idx, ' ');

int i = a.size(), j = b.size();
while (i > 0 && j > 0) {
    if (a[i - 1] == b[j - 1]) {
        ans[idx - 1] = a[i - 1];
        i--, j--, idx--;
    } else if (dp[i - 1][j] > dp[i][j - 1])
        i--;
    else
        j--;
}

return ans;
}
```

1.4.2 Kadane

Time: $\mathcal{O}(n + m)$

Space: $\mathcal{O}(n + m)$

```
int kadane(const vector<int> &v, int &start, int &end) {
    start = end = 0;
    int msf = -inf, meh = 0, s = 0;

    for (int i = 0; i < v.size(); ++i) {
        meh += v[i];

        if (msf < meh) {
            msf = meh;
            start = s, end = i;
        }

        if (meh < 0) {
            meh = 0;

```

1.4.4 Longest Common Subsequence

Time: $\mathcal{O}(n \cdot m)$

Space: $\mathcal{O}(n \cdot m)$

```
int dp[MAX][MAX];

string lcs(string a, string b) {
    for (int i = 0; i <= a.size(); ++i) {
        for (int j = 0; j <= b.size(); ++j) {
            if (i == 0 || j == 0)
                dp[i][j] = 0;
            else if (a[i - 1] == b[j - 1])
                dp[i][j] = dp[i - 1][j - 1] + 1;
            else

```

1.4.5 Ternary Search

Description:

Searches for the minimum (or maximum) value of a unimodal function $f(x)$ for x between l and r .

Time: $\mathcal{O}(\log n)$

Space: $\mathcal{O}(1)$

Caution:

- f must be a unimodal function

```
// Minimum of f: comp = less<T>()
// Maximum of f: comp = greater<T>()
template <typename T, class F = function<T(T)>>
T ternary_search(T l, T r, F f, function<bool(T,T)> comp) {
    T rt, lt;

    for (int i = 0; i < 500; ++i) {

```

```

if (fabs(r - l) < EPS)
    return (l + r) / 2;

lt = (r - l) / 3 + l;
rt = ((r - l) * 2) / 3 + l;

```

```

if (comp(f(lt), f(rt)))
    l = lt;
else
    r = rt;

```

```

}

return (l + r) / 2;
}

```

1.5 String

1.5.1 Booth's Algorithm

Description:

This algorithm finds the lexicographically minimal (or maximal) string rotation, that is, given a string s , the goal is to find a rotation of s possessing the lowest (or highest) lexicographical order among all possible rotations.

Time: $\mathcal{O}(n)$
Space: $\mathcal{O}(n)$

```

// Maximal: func = greater<char>()
// Minimal: func = less<char>()
string booth(string s, function<bool(char, char)> func) {
    string S = s + s;
    vector<int> f(S.size(), -1);

    int k = 0;
    for (int j = 1; j < S.size(); ++j) {
        char sj = S[j];
        int i = f[j - k - 1];

        while (i != -1 && sj != S[k+i+1]) {
            if (func(sj, S[k + i + 1]))
                k = j - i - 1;
            i = f[i];
        }

        if (sj != S[k+i+1]) {
            if (func(sj, S[k]))
                k = j;

            f[j-k] = -1;
        } else
            f[j-k] = i + 1;
    }

    string ans(s.size(), 0);
    for (int i = 0; i < s.size(); ++i)

```

```

    ans[i] = S[k+i];
    return ans;
}

```

1.5.2 Knuth-Morris-Pratt (KMP)

Time:

- preprocess: $\mathcal{O}(m)$
- search: $\mathcal{O}(n)$

Space: $\mathcal{O}(n + m)$

```

struct KMP {
    string patt;
    vector<int> table;

    KMP(string patt) :
        patt(patt), table(patt.size()+1)
    { preprocess(); }

    void preprocess() {
        fill(all(table), -1);

        for (int i = 0, j = -1; i < patt.size(); ++i) {
            while (j >= 0 && patt[i] != patt[j])
                j = table[j];
            table[i + 1] = ++j;
        }
    }

    vector<int> search(const string &txt) {
        vector<int> occurs;

        for (int i = 0, j = 0; i < txt.size(); ++i) {
            while (j >= 0 && txt[i] != patt[j])
                j = table[j];
            j++;
        }
    }
}

```

```

        if (j == patt.size()) {
            occurs.pb(i - j);
            j = table[j];
        }
    }

    return occurs;
}
};

```

1.5.3 Z-function

Time: $\mathcal{O}(n)$
Space: $\mathcal{O}(n)$

```

vector<int> z_function(string s) {
    int n = (int) s.length();
    vector<int> z(n);

    int l = 0, r = 0;
    for (int i = 1; i < n; ++i) {
        if (i <= r)
            z[i] = min(r - i + 1, z[i - l]);

        while (i + z[i] < n && s[z[i]] == s[i + z[i]])
            z[i]++;

        if (i + z[i] - 1 > r) {
            l = i;
            r = i + z[i] - 1;
        }
    }

    return z;
}

```

1.6 Structure

1.6.1 AVL tree

Time: $\mathcal{O}(\log n)$
Space: $\mathcal{O}(n)$

```

struct AVL {

```

```

    struct Node {
        int key, height;
        Node *left, *right;

        Node(int k) :
            key(k), height(1),

```

```

        left(nullptr), right(nullptr)
    {}

    static int get_height(Node *n) {
        return (n == nullptr) ? 0 : n->height;
    }
}

```

```

void fix_state() {
    height = max(Node::get_height(left),
        Node::get_height(right)) + 1;
}

int get_balance() {
    return Node::get_height(left) -
        Node::get_height(right);
}
};

Node *root;

AVL() : root(nullptr) {}

void insert(int key) {
    root = insert(root, key);
}

```

private:

```

Node *rotate_right(Node *n) {
    Node *aux1 = n->left;
    Node *aux2 = aux1->right;
    aux1->right = n; aux1->fix_state();
    n->left = aux2; n->fix_state();
    return aux1;
}

Node *rotate_left(Node *n) {
    Node *aux1 = n->right;
    Node *aux2 = aux1->left;
    aux1->left = n; aux1->fix_state();
    n->right = aux2; n->fix_state();
    return aux1;
}

Node *insert(Node *n, int key) {
    if (n == nullptr) {
        Node *new_node = new Node(key);
        if (root == nullptr) root = new_node;
        return new_node;
    }

    if (key < n->key)
        n->left = insert(n->left, key);
    else
        n->right = insert(n->right, key);

    int balance = n->get_balance();
    n->fix_state();

    if (balance > 1 && key < n->left->key) {
        return rotate_right(n);
    } else if (balance < -1 && key > n->right->key) {
        return rotate_left(n);
    } else if (balance > 1 && key > n->left->key) {
        n->left = rotate_left(n->left);
        return rotate_right(n);
    } else if (balance < -1 && key < n->right->key) {
        n->right = rotate_right(n->right);
        return rotate_left(n);
    }
}

```

```

    return n;
}
};

```

1.6.2 Binary Indexed Tree (BIT)

Time:

- update: $\mathcal{O}(\log n)$
- query: $\mathcal{O}(\log n)$

Space: $\mathcal{O}(n)$

```

struct BIT {
    int N;
    vector<int> tree;

    BIT(int N) : N(N), tree(N, 0) {}

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

    void update(int idx, int val) {
        for (; idx < N; idx += (idx & -idx))
            tree[idx] += val;
    }
};

```

1.6.3 Binary Indexed Tree 2D (BIT2D)

Time:

- update: $\mathcal{O}(\log^2 n)$
- query: $\mathcal{O}(\log^2 n)$

Space: $\mathcal{O}(n^2)$

```

struct BIT2D {
    int N, M;
    vector<vector<int>> tree;

    BIT2D(int N, int M) : N(N), M(M),
        tree(N, vector<int>(M, 0)) {}

    int query(int idx, int idy) {
        int sum = 0;
        for (; idx > 0; idx -= (idx & -idx))
            for (int m = idy; m > 0; m -= (m & -m))
                sum += tree[idx][m];
        return sum;
    }

    void update(int idx, int idy, int val) {
        for (; idx < N; idx += (idx & -idx))
            for (int m = idy; m < M; m += (m & -m))
                tree[idx][m] += val;
    }
}

```

```

}
};

```

1.6.4 Bitmask

Time: $\mathcal{O}(1)$

Space: $\mathcal{O}(1)$

```

struct Bitmask {
    ll state;

    Bitmask(ll state) :
        state(state) {}

    void set(int pos) {
        state |= (1 << pos);
    }

    void set_all(int n) {
        state = (1 << n) - 1;
    }

    void unset(int pos) {
        state &= ~(1 << pos);
    }

    void unset_all() {
        state = 0;
    }

    int get(int pos) {
        return state & (1 << pos);
    }

    void toggle(int pos) {
        state ^= (1 << pos);
    }

    int least_significant_one() {
        return state & (-state);
    }
};

```

1.6.5 Disjoint-set

Time:

- make_set: $\mathcal{O}(1)$
- find_set: $\mathcal{O}(a(n))$
- union_set: $\mathcal{O}(a(n))$

Space: $\mathcal{O}(n)$

```

struct DisjointSet {
    int N;
    vector<int> rnk, par;

    DisjointSet(int N) :
        N(N), rnk(N), par(N), siz(N)
    { init(); }
}

```

```

void init() {
    iota(all(par), 0);
    fill(all(rnk), 0);
    fill(all(siz), 1);
}

int find_set(int x) {
    if (par[x] != x)
        par[x] = find_set(par[x]);
    return par[x];
}

void union_set(int x, int y) {
    x = find_set(x);
    y = find_set(y);

    if (x == y) return;
    if (rnk[x] < rnk[y]) swap(x, y);
    if (rnk[x] == rnk[y]) rnk[x]++;
    par[y] = x;
    siz[x] += siz[y];
}
};

```

1.6.6 Lazy Segment Tree

Time:

- build: $\mathcal{O}(n \log n)$
- update: $\mathcal{O}(\log n)$
- query: $\mathcal{O}(\log n)$

Space: $\mathcal{O}(n)$

Caution:

- Provide value for N before any operation

```

#define left(x) (x << 1)
#define right(x) ((x << 1) + 1)

int N;

template <typename T, class F = function<T(const T&, const T&)>>
struct LazySegmentTree {
    F func;
    T id = T();
    vector<T> tree, lazy;

    LazySegmentTree(F func) :
        func(func), tree(MAX*4, 0), lazy(MAX*4, 0) {}

    void build(const vector<T> &v,
               int node = 1, int l = 0, int r = N - 1)
    {
        if (l > r) return;

        if (l == r)
            tree[node] = v[l];
        else {
            int m = (l + r) / 2;

```

```

            build(v, left(node), l, m);
            build(v, right(node), m + 1, r);
            tree[node] = func(tree[left(node)], tree[right(node)]);
        }
    }

    void push(int node, int l, int r, T val) {
        tree[node] += (r - l + 1) * val;

        if (l != r) {
            lazy[left(node)] += val;
            lazy[right(node)] += val;
        }

        lazy[node] = 0;
    }

    void update(int i, int j, T val,
               int node = 1, int l = 0, int r = N - 1)
    {
        if (lazy[node] != 0)
            push(node, l, r, lazy[node]);

        if (l > r || l > j || r < i) return;

        if (i <= l && r <= j)
            push(node, l, r, val);
        else {
            int m = (l + r) / 2;
            update(i, j, val, left(node), l, m);
            update(i, j, val, right(node), m + 1, r);
            tree[node] = func(tree[left(node)], tree[right(node)]);
        }
    }

    T query(int i, int j,
           int node = 1, int l = 0, int r = N - 1)
    {
        if (l > r || l > j || r < i) return id;

        if (lazy[node])
            push(node, l, r, lazy[node]);

        if (l >= i && r <= j)
            return tree[node];

        int m = (l + r) / 2;
        T q1 = query(i, j, left(node), l, m);
        T q2 = query(i, j, right(node), m + 1, r);
        return func(q1, q2);
    }
};

```

1.6.7 Mo's Algorithm

Time: $\mathcal{O}((n + q) \cdot \sqrt{n})$

Space: $\mathcal{O}(n + q)$

Caution:

- Remember to implement add, remove, and get_ans functions.

```
struct Query {
```

```

    int l, r, idx;
};

vector<int> mos_algorithm(vector<Query> Q) {
    int blk_size = (int) sqrt(v.size() + 0.0) + 1;

    vector<int> ans(Q.size());
    sort(all(Q), [](Query a, Query b) {
        return ii(a.l / blk_size, a.r) < ii(b.l / blk_size, b.r);
    });

    int curr_l = 0, curr_r = -1;

    for (auto q : Q) {
        while (curr_l > q.l) {
            curr_l--;
            add(curr_l);
        }
        while (curr_r < q.r) {
            curr_r++;
            add(curr_r);
        }
        while (curr_l < q.l) {
            remove(curr_l);
            curr_l++;
        }
        while (curr_r > q.r) {
            remove(curr_r);
            curr_r--;
        }

        ans[q.idx] = get_ans();
    }

    return ans;
}

```

1.6.8 Policy Tree

Description:

A set-like STL structure with order statistics.

Time:

- insert: $\mathcal{O}(\log n)$
- erase: $\mathcal{O}(\log n)$
- find_by_order: $\mathcal{O}(\log n)$
- order_of_key: $\mathcal{O}(\log n)$

Space: $\mathcal{O}(n)$

```

#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;

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

void operations() {
    ordered_set S;

```

```

S.insert(x);
S.erase(x);

// Return iordered_set iterator to the k-th largest element
// (counting from zero)
int pos = *S.find_by_order(k);

// Return the number of items strictly smaller
// than x
int ord = S.order_of_key(x)
}

```

1.6.9 Segment Tree

Time:

- update: $\mathcal{O}(\log n)$
- query: $\mathcal{O}(\log n)$

Space: $\mathcal{O}(n)$

Caution:

- Provide value for N before any operation

```

#define left(x) (x << 1)
#define right(x) ((x << 1) + 1)

int N;

template <typename T, class F = function<T(const T&, const T&)>>
struct SegmentTree {
    F func;
    T ident = T();
    vector<T> tree;

    SegmentTree(F &func) : func(func), tree(MAX*4) {}

    void build(const vector<T> &v,
               int node = 1, int l = 0, int r = N - 1)
    {
        if (l > r)
            return;

        if (l == r)
            tree[node] = v[l];
        else {
            int m = (l + r) / 2;
            build(v, left(node), l, m);
            build(v, right(node), m + 1, r);
            tree[node] = func(tree[left(node)], tree[right(node)]);
        }
    }

    void update(int i, T val,
               int node = 1, int l = 0, int r = N - 1)
    {
        if (l > r || l > i || r < i)
            return;

        if (l == r)

```

```

        tree[node] = val;
    else {
        int m = (l + r) / 2;
        update(i, val, left(node), l, m);
        update(i, val, right(node), m + 1, r);
        tree[node] = func(tree[left(node)], tree[right(node)]);
    }
}

T query(int i, int j,
        int node = 1, int l = 0, int r = N - 1)
{
    if (l > r || l > j || r < i)
        return ident;

    if (l >= i && r <= j)
        return tree[node];

    int m = (l + r) / 2;
    T q1 = query(i, j, left(node), l, m);
    T q2 = query(i, j, right(node), m + 1, r);
    return func(q1, q2);
}
};

```

1.6.10 2D Segment Tree

Time:

- build: $\mathcal{O}(n \cdot m \cdot \log n \cdot \log m)$
- update: $\mathcal{O}(\log n \cdot \log m)$
- query: $\mathcal{O}(\log n \cdot \log m)$

Space: $\mathcal{O}(16 \cdot n \cdot m)$

Caution:

- Very high constant in space complexity

```

#define left(x) (x << 1)
#define right(x) ((x << 1) + 1)

int N, M;

template<typename T, class F = function<T(const T&, const T&)>>
struct SegmentTree2D {
    using matrix<T> = vector<vector<T>>;

    F func;
    T id = T();
    matrix<T> tree;

    SegmentTree2D(F func) :
        tree(4*MAX, vector<T>(4*MAX, 0)),
        func(func) {}

    void build_row(const matrix<T> &mat,
                  int ni, int li, int ri,
                  int nj = 1, int lj = 0, int rj = M - 1)
    {
        if (lj == rj) {
            if (li == ri)
                tree[ni][nj] = mat[li][lj];

```

```

        else
            tree[ni][nj] = func(tree[left(ni)][nj], tree[right(ni)
                                ][nj]);
        } else {
            int m = (lj + rj) / 2;
            build_row(mat, ni, li, ri, left(nj), lj, m);
            build_row(mat, ni, li, ri, right(nj), m+1, rj);
            tree[ni][nj] = func(tree[ni][left(nj)], tree[ni][right(nj)
                                ]);
        }
    }

    void build(const matrix<T> &mat,
               int ni = 1, int li = 0, int ri = N - 1)
    {
        if (li != ri) {
            int m = (li + ri) / 2;
            build(mat, left(ni), li, m);
            build(mat, right(ni), m+1, ri);
        }
        build_row(mat, ni, li, ri);
    }

    T query_row(int j1, int j2, int i,
                int nj = 1, int lj = 0, int rj = M - 1)
    {
        if (lj > rj || lj > j2 || rj < j1) return id;

        if (j1 <= lj && rj <= j2)
            return tree[i][nj];

        int m = (lj + rj) / 2;
        T q1 = query_row(j1, j2, i, left(nj), lj, m);
        T q2 = query_row(j1, j2, i, right(nj), m + 1, rj);
        return func(q1, q2);
    }

    T query(int i1, int j1, int i2, int j2,
            int ni = 1, int li = 0, int ri = N - 1)
    {
        if (li > ri || li > i2 || ri < i1) return id;

        if (i1 <= li && ri <= i2)
            return query_row(j1, j2, ni);

        int m = (li + ri) / 2;
        T q1 = query(i1, j1, i2, j2, left(ni), li, m);
        T q2 = query(i1, j1, i2, j2, right(ni), m + 1, ri);
        return func(q1, q2);
    }

    void update_row(int i, int j, T val,
                   int ni, int li, int ri,
                   int nj = 1, int lj = 0, int rj = M - 1)
    {
        if (lj > rj || lj > j || rj < j) return;

        if (lj == rj) {
            if (li == ri)
                tree[ni][nj] = val;
            else
                tree[ni][nj] = func(tree[left(ni)][nj], tree[right(ni)
                                    ][nj]);
        } else {
            int m = (lj + rj) / 2;

```

```

    update_row(i, j, val, ni, li, ri, left(nj), lj, m);
    update_row(i, j, val, ni, li, ri, right(nj), m+1, rj);
    tree[ni][nj] = func(tree[ni][left(nj)], tree[ni][right(nj)]);
}
}

void update(int i, int j, T val,
            int ni = 1, int li = 0, int ri = N - 1)
{
    if (li > ri || li > i || ri < i) return;

    if (li != ri) {
        int m = (li + ri) / 2;
        update(i, j, val, left(ni), li, m);
        update(i, j, val, right(ni), m+1, ri);
    }

    update_row(i, j, val, ni, li, ri);
}
};

```

1.6.11 Sqrt Decomposition

Time:

- preprocess: $\mathcal{O}(n)$
- query: $\mathcal{O}(\sqrt{n})$
- update: $\mathcal{O}(1)$

Space: $\mathcal{O}(n)$

```

struct SqrtDecomposition {
    int blk_size;
    vector<int> v, blk;

    SqrtDecomposition(vector<int> v) :
        v(v), blk(v.size())
    { init(); }

    void init() {
        build();
    }

    void update(int idx, int val) {
        blk[idx / blk_size] += val - v[idx];
        v[idx] = val;
    }
}

```

```

int query(int l, int r) {
    int ans = 0;
    int cl = l/blk_size, cr = r/blk_size;

    if (cl == cr) {
        for (int i = l; i <= r; ++i)
            ans += v[i];
    } else {
        for (int i = l, end=(cl+1)*blk_size-1; i <= end; ++i)
            ans += v[i];
        for (int i = cl+1; i <= cr - 1; ++i)
            ans += blk[i];
        for (int i = cr*blk_size; i <= r; ++i)
            ans += v[i];
    }

    return ans;
}

void build() {
    int n = v.size();
    blk_size = (int) sqrt(n + 0.0) + 1;
    for (int i = 0; i < n; ++i)
        blk[idx / blk_size] += v[i];
}
};

```

1.6.12 Trie

Time:

- insert: $\mathcal{O}(M)$
- search: $\mathcal{O}(M)$

Space: $\mathcal{O}(\text{alph} \cdot N)$

Caution:

- If Trie stores integers, remember to check order of bits (most/least significant first).

```

template <typename T>
struct Trie {
    int states;

    vector<int> ending;
    vector<vector<int>>> trie;
}

```

```

// Number of words (N) and number of letters per word
// (M), and number of letters in alphabet (alph).
Trie(int N, int M, int alph) :
    ending(N * M),
    trie(N * M, vector<int>(alph))
{ init(); }

void init() {
    states = 0;
    for (auto &i : trie)
        fill(all(i), -1);
}

int len(T x) {
    if (constexpr(is_same_v<T,int>))
        return 32;
    return x.size();
}

int idx(T x) {
    if (constexpr(is_same_v<T,int>))
        return !(x & (1 << i));
    return x[i] - 'a';
}

void insert(T x) {
    int node = 0;

    for (int i = 0; i < len(x); ++i) {
        if (trie[node][idx(x, i)] == -1)
            trie[node][idx(x, i)] = ++states;
        node = trie[node][idx(x, i)];
    }

    ending[node] = true;
}

bool search(T x) {
    int node = 0;

    for (int i = 0; i < len(x); ++i) {
        node = trie[node][idx(x, i)];
        if (node == -1)
            return false;
    }

    return ending[node];
}
};

```

2 Misc

2.1 Environment

2.1.1 Vim Config

```
" Tabs
set expandtab
set smarttab

" Indents
set shiftwidth=2
set tabstop=2
set autoindent
set smartindent
set cindent

" Turn backup off
set nobackup
set nowb
set noswapfile
```

```
" Highlight matching brackets
set showmatch

" Display line numbers
set number
```

2.1.2 Template

```
#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
```

```
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int,int>;

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    return 0;
}
```

3 Problems

3.1 A Simple Task

```
#include <bits/stdc++.h>

#define MAX 100001
#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define sz size()
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

#define left(x) ((x << 1))
#define right(x) ((x << 1) + 1)

using namespace std;

typedef long long ll;
typedef pair<int,int> ii;

int N, q;
int tree[4 * MAX][26], lazy[4 * MAX][26];

string s;

void build(int node = 1, int a = 0, int b = N-1) {
    if (a > b)
        return;
```

```
    if (a == b) {
        tree[node][s[a] - 'a'] = 1;
        return;
    }

    build(left(node), a, (a + b) / 2);
    build(right(node), (a + b) / 2 + 1, b);

    for (int i = 0; i < 26; ++i)
        tree[node][i] = tree[left(node)][i] + tree[right(node)][i];
}

void push(int let, int node, int a, int b, int val) {
    tree[node][let] = (b - a + 1) * val;

    if (a != b) {
        lazy[left(node)][let] = val;
        lazy[right(node)][let] = val;
    }

    lazy[node][let] = -1;
}

void update(int let, int i, int j, int val, int node = 1, int a = 0, int b = N-1) {
    if (lazy[node][let] != -1)
        push(let, node, a, b, lazy[node][let]);

    if (a > b or a > j or b < i)
        return;

    if (i <= a and b <= j) {
        push(let, node, a, b, val);
        return;
    }

    update(let, i, j, val, left(node), a, (a + b) / 2);
    update(let, i, j, val, right(node), (a + b) / 2 + 1, b);
    tree[node][let] = tree[left(node)][let] + tree[right(node)][let];
}
```

```

}

int query(int let, int i, int j, int node = 1, int a = 0, int b = N-1) {
    if (a > b || a > j || b < i)
        return 0;

    if (lazy[node][let] != -1)
        push(let, node, a, b, lazy[node][let]);

    if (a >= i and b <= j)
        return tree[node][let];

    int q1 = query(let, i, j, left(node), a, (a + b) / 2);
    int q2 = query(let, i, j, right(node), (a + b) / 2 + 1, b);
    return q1 + q2;
}

void get_ans(int let, string &ans, int node = 1, int l = 0, int r = N-1) {
    if (lazy[node][let] != -1)
        push(let, node, l, r, lazy[node][let]);

    if (!tree[node][let])
        return;

    if (l == r) {
        ans[l] = let + 'a';
        return;
    }

    get_ans(let, ans, left(node), l, (l+r)/2);
    get_ans(let, ans, right(node), (l+r)/2 + 1, r);
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    cin >> N >> q;
    cin >> s;

    build();
    mset(lazy, -1);

    vector<int> cnt(26);
    for (int i = 0; i < q; ++i) {
        int a, b, k; cin >> a >> b >> k;
        a--, b--;
        for (int j = 0; j < 26; ++j) {
            cnt[j] = query(j, a, b);
            if (cnt[j]) update(j, a, b, 0);
        }

        if (k) {
            int bord = a;
            for (int j = 0; j < 26; ++j)
                if (cnt[j]) {
                    update(j, bord, bord + cnt[j] - 1, 1);
                    bord += cnt[j];
                }
        } else {
            int bord = a;
            for (int j = 25; j >= 0; --j)
                if (cnt[j]) {
                    update(j, bord, bord + cnt[j] - 1, 1);
                    bord += cnt[j];
                }
        }
    }
}

```

```

    }
}

fill(all(cnt), 0);

string ans(N, '-');
for (int j = 0; j < 26; ++j)
    get_ans(j, ans);
cout << ans << endl;
return 0;
}

```

3.2 Crise Hidrica

```

#include <bits/stdc++.h>

#define MAX 5010
#define MAXLOG 20
#define MOD 1000000007
#define EPS 1e-6
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define sz size()
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

typedef long long ll;
typedef pair<int,int> ii;

int found;
ll dp[MAX][1010];
ll v[MAX];
vector<int> graph[MAX];

int h[MAX];
int par[MAX][MAXLOG];
ll upd[MAX];

void dfs(int va, int p = -1) {
    par[va][0] = p;

    if (p + 1)
        h[va] = h[p] + 1;

    for (int i = 1; i < MAXLOG; ++i)
        if (par[va][i - 1] + 1)
            par[va][i] = par[par[va][i - 1]][i - 1];

    for (auto u : graph[va])
        if (p != u)
            dfs(u, va);
}

```

```

void preprocess(int va) {
    memset(par, -1, sizeof par);
    dfs(va);
}

int query(int p, int q) {
    if (h[p] < h[q])
        swap(p, q);

    for (int i = MAXLOG - 1; i >= 0; --i)
        if (par[p][i] + 1 && h[par[p][i]] >= h[q]) {
            p = par[p][i];
        }

    if (p == q)
        return p;

    for (int i = MAXLOG - 1; i >= 0; --i)
        if (par[p][i] + 1 && par[p][i] != par[q][i]) {
            p = par[p][i];
            q = par[q][i];
        }

    return par[p][0];
}

ll fill(int x, int pare) {
    ll ans = 0;

    for (auto i : graph[x])
        if (i != pare)
            ans += fill(i, x);

    return v[x] = ans + upd[x];
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n, d, a, b;
    cin >> n >> d;
    for (int i = 0; i < n - 1; ++i) {
        cin >> a >> b;
        graph[a].pb(b);
        graph[b].pb(a);
    }

    int m, vv, c;
    cin >> m;
    vector<int> w(m+1), avail;
    for (int i = 0; i < m; ++i) {
        cin >> c >> vv;
        w[i+1] = vv;
        avail.pb(c);
    }

    preprocess(1);

    ll q, l;
    int x, y;
    cin >> q;
    for (int i = 0; i < q; ++i) {
        cin >> x >> y >> l;

```

```

    int lca = query(x, y);

    upd[lca] -= l;
    upd[x] += l;
    upd[y] += l;
    if (lca != 1)
        upd[par[lca][0]] -= l;
    }

    fill(1, -1);

    vector<ll> vv(m+1);
    for (int i = 0; i < m; ++i)
        vv[i+1] = v[avail[i]];

    for (int i = 1; i <= m; ++i)
        for (int j = 0; j <= d; ++j)
            if (j >= w[i])
                dp[i][j] = max(dp[i-1][j], dp[i-1][j - w[i]] + vv[i]);
            else
                dp[i][j] = dp[i-1][j];

    cout << dp[m][d] << endl;
    return 0;
}

```

3.3 Escalacao

```

#include <bits/stdc++.h>

#define MAX 1010101
#define MOD 1000000007
#define EPS 1e-6
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define sz size()
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

typedef long long ll;
typedef pair<int,int> ii;

ll v[MAX];
ll n, k, r;
vector<ll> tree[MAX];
vector<ll> res;

void merge(vector<ll> a, vector<ll> b, vector<ll> &ans) {
    int i = 0, j = 0, cnt = 0;
    for ( ; cnt < k && i < a.size() && j < b.size(); ++cnt)
        if (a[i] > b[j])
            ans.push_back(a[i++]);
        else

```

```

        ans.push_back(b[j++]);
    }
    if (cnt < k) {
        if (i < a.size())
            for (; cnt < k && i < a.size(); ++i, ++cnt)
                ans.push_back(a[i]);
        else
            for (; cnt < k && j < b.size(); ++j, ++cnt)
                ans.push_back(b[j]);
    }
}

void build(int node = 1, int a = 0, int b = n - 1) {
    if (a > b)
        return;
    if (a == b) {
        tree[node].push_back(v[a]);
        return;
    }

    build(node * 2, a, (a + b) / 2);
    build(node * 2 + 1, 1 + (a + b) / 2, b);
    merge(tree[node * 2], tree[node * 2 + 1], tree[node]);
}

void query(int i, int j, int node = 1, int a = 0, int b = n - 1) {
    if (a > b || a > j || b < i)
        return;
    if (a >= i && b <= j) {
        res.insert(res.end(), tree[node].begin(), tree[node].end());
        return;
    }

    query(i, j, node * 2, a, (a + b) / 2);
    query(i, j, 1 + node * 2, 1 + (a + b) / 2, b);
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    cin >> n >> k >> r;
    for (int i = 0; i < n; ++i)
        cin >> v[i];
    build();

    for (int i = 0; i < r; ++i) {
        int ans = 1;
        res.clear();
        ll a, b; cin >> a >> b;
        query(a - 1, b - 1);

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

        int cnt = 0;
        for (int j = res.size() - 1; j >= 0; --j) {
            if (cnt >= k || res[j] == 0) {
                if (res[j] == 0 && j == res.size() - 1)
                    ans = 0;
                break;
            }
            ans = (ans * res[j]) % MOD;
            cnt++;
        }
    }
}

```

```

        cout << ans << endl;
    }

    return 0;
}

```

3.4 Exponial

```

#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int, int>;

ll phi(ll n) {
    ll result = n;

    for (ll i = 2; i * i <= n; i++) {
        if (n % i == 0) {
            while (n % i == 0) n /= i;
            result -= result / i;
        }
    }

    if (n > 1)
        result -= result / n;

    return result;
}

ll fast_pow(ll x, ll n, ll m) {
    ll ans = 1;

    while (n) {
        if (n & 1)
            ans = (ans * x) % m;

        n >>= 1;
        x = (x * x) % m;
    }

    return ans % m;
}

ll solve(ll n, ll m) {
    if (m == 1)
        return 0;
}

```

```

if (n <= 5) {
    if (n == 4) return 262144 % m;
    else if (n == 3) return 9 % m;
    else if (n == 2) return 2 % m;
    else if (n == 1) return 1 % m;
    else return fast_pow(5, 262144, m);
}

// n^e % m == n^(phi(m) + e % phi(m)) mod m
ll p = phi(m);
ll e = p + solve(n - 1, p) % p;
return fast_pow(n, e, m);
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    ll n, m; cin >> n >> m;
    cout << solve(n, m) << endl;

    return 0;
}

```

3.5 Trees Partition

```

#include <bits/stdc++.h>

#define MAX 301010
#define MOD 1000000007
#define inf 0x3f3f3f3f

#define fi first
#define se second
#define sz size()
#define pb push_back

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()

using namespace std;

typedef long long ll;
typedef pair<int, int> ii;

ll hsh[MAX];
vector<int> t1[MAX], t2[MAX];

unordered_map<ll, int> M;
ll ans = 0, tot = 0;

ll dfs(int x, int par) {
    ll down = hsh[x];

    for (auto i : t1[x]) {
        if (i != par) {
            ll bel = dfs(i, x);
            down ^= bel;
            M[bel] = 1;
        }
    }
}

```

```

return down;
}

ll solve(int x, int par) {
    ll down = hsh[x];

    for (auto i : t2[x]) {
        if (i != par) {
            ll bel = solve(i, x);
            down ^= bel;

            if (M[bel] || M[bel ^ tot]) {
                ans++;
                M[bel] = M[bel ^ tot] = 0;
            }
        }
    }

    return down;
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n, x;
    cin >> n;
    mt19937_64 mt(time(NULL));

    for (int i = 0; i < n - 1; ++i) {
        cin >> x;
        t1[i+1].pb(x-1);
        t1[x-1].pb(i+1);
    }

    for (int i = 0; i < n - 1; ++i) {
        cin >> x;
        t2[i+1].pb(x-1);
        t2[x-1].pb(i+1);
    }

    for (int i = 0; i < n; ++i) {
        hsh[i] = mt();
        tot ^= hsh[i];
    }

    dfs(0, -1);
    solve(0, -1);

    cout << ans << '\n';
    return 0;
}

```

3.6 XOR submatrix

```

#include <bits/stdc++.h>

#define MAX 10101010
#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f

```

```

#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define sz size()
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

typedef long long ll;
typedef pair<int,int> ii;

int states = 0;
int trie[MAX][2];

void insert(int x) {
    int node = 0;

    for (int i = 30; i >= 0; --i) {
        int b = !(x & (1 << i));

        if (trie[node][b] == -1)
            trie[node][b] = ++states;
        node = trie[node][b];
    }
}

int search(int x) {
    int node = 0;
    int ans = 0;

    for (int i = 30; i >= 0; --i) {
        int b = !(x & (1 << i));

        if (trie[node][b] == -1)
            b ^= 1;

        node = trie[node][b];
        ans |= (b << i);
    }
}

```

```

}

return ans;
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    mset(trie, -1);

    int n, m; cin >> n >> m;
    vector<int> v(n+1), u(m+1);

    v[0] = u[0] = 0;
    for (int i = 1; i <= n; ++i) {
        int x; cin >> x;
        v[i] = v[i-1] ^ x;
    }

    for (int i = 1; i <= m; ++i) {
        int x; cin >> x;
        u[i] = u[i-1] ^ x;
    }

    int ans = 0;
    for (int i = 0; i <= n; ++i)
        for (int j = i; j <= n; ++j)
            if ((j - i) % 2)
                insert(v[i] ^ v[j]);
            else
                ans = max(ans, v[i] ^ v[j]);

    for (int i = 0; i <= m; ++i)
        for (int j = i; j <= m; ++j)
            if ((j - i) % 2)
                ans = max(ans, u[i] ^ u[j] ^ search(u[i] ^ u[j]));
            else
                ans = max(ans, u[i] ^ u[j]);

    cout << ans << ende;
    return 0;
}

```

4 Contests

4.1 Cadernaveis

4.1.1 Bale

```

#include <bits/stdc++.h>

#define MAX 101010
#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

```

```

#define fi first
#define se second
#define sz size()
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

```

```
using namespace std;

typedef long long ll;
typedef pair<int,int> ii;

int v[MAX];
int tree[MAX];

int query(int idx) {
    int sum = 0;
    for (; idx > 0; idx -= (idx & -idx))
        sum += tree[idx];

    return sum;
}

void update(int idx, int val) {
    for (; idx <= MAX; idx += (idx & -idx))
        tree[idx] += val;
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

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

    int ans = 0;
    for (int i = 0; i < n; ++i) {
        ans += query(n - v[i] + 1);
        update(n - v[i] + 1, 1);
    }

    cout << ans << '\n';
    return 0;
}
```

4.1.2 A Lei vai a Cavalos

```
#include <bits/stdc++.h>

#define MAX 300
#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define sz size()
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;
```

```
typedef long long ll;
typedef pair<int,int> ii;

int N;
int par[MAX];
int graph[MAX][MAX], rg[MAX][MAX];

bool cont[MAX];

bool path(int s, int t) {
    cont[s] = true;
    if (s == t)
        return true;

    for (int i = 0; i < N; ++i)
        if (!cont[i] && rg[s][i]) {
            par[i] = s;

            if (path(i, t))
                return true;
        }

    return false;
}

int ford_fulkerson(int s, int t) {
    int ans = 0;
    par[s] = -1;

    mset(cont, 0);
    memcpy(rg, graph, sizeof(graph));

    while (path(s, t)) {
        int flow = inf;

        for (int i = t; par[i] != -1; i = par[i])
            flow = min(flow, rg[par[i]][i]);

        for (int i = t; par[i] != -1; i = par[i]) {
            rg[par[i]][i] -= flow;
            rg[i][par[i]] += flow;
        }

        ans += flow;
        mset(cont, 0);
    }

    return ans;
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n, m, k, cas = 1;
    while (cin >> n >> m >> k) {
        mset(graph, 0);

        N = MAX;
        int s = 0, t = MAX-1;
        for (int i = 1; i <= n; ++i) {
            int c; cin >> c;
```

```

    graph[i + 120][t] = c;
}

for (int i = 0; i < k; ++i) {
    int x, y; cin >> x >> y;
    graph[y][x + 120] = 1;
}

for (int i = 1; i <= m; ++i)
    graph[s][i] = 1;

cout << "Instancia " << cas++ << "ende;
cout << ford_fulkerson(s, t) << "ende;
cout << "ende;
}
return 0;
}

```

4.1.3 Xenia and Tree

```

#include <bits/stdc++.h>

#define MAX 101010
#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int, int>;

#define MAXLOG 20

vector<ii> graph[MAX];

struct LCA {
    vector<int> h;
    vector<vector<int>> par, cost;

    LCA(int N) :
        h(N),
        par(N, vector<int>(MAXLOG)),
        cost(N, vector<int>(MAXLOG))
    {
        init();
    }

    void init() {
        for (auto &i : par)
            fill(all(i), -1);
        for (auto &i : cost)

```

```

            fill(all(i), 0);
            dfs(0);
        }

        inline int op(int a, int b) {
            return a + b;
        }

        void dfs(int v, int p = -1, int c = 0) {
            par[v][0] = p;
            cost[v][0] = c;

            if (p != -1)
                h[v] = h[p] + 1;

            for (int i = 1; i < MAXLOG; ++i)
                if (par[v][i - 1] != -1) {
                    par[v][i] = par[par[v][i - 1]][i - 1];
                    cost[v][i] = op(cost[v][i], op(cost[par[v][i - 1]][i - 1], cost[v][i - 1]));
                }

            for (auto u : graph[v])
                if (p != u.fi)
                    dfs(u.fi, v, u.se);
        }

        int query(int p, int q) {
            int ans = 0; // *

            if (h[p] < h[q])
                swap(p, q);

            for (int i = MAXLOG - 1; i >= 0; --i)
                if (par[p][i] != -1 && h[par[p][i]] >= h[q]) {
                    ans = op(ans, cost[p][i]);
                    p = par[p][i];
                }

            if (p == q) {
                return ans;
            }

            for (int i = MAXLOG - 1; i >= 0; --i)
                if (par[p][i] != -1 && par[p][i] != par[q][i]) {
                    ans = op(ans, op(cost[p][i], cost[q][i]));
                    p = par[p][i];
                    q = par[q][i];
                }

            if (p == q) return ans;
            else return op(ans, op(cost[p][0], cost[q][0]));
        }
    };

    struct CentroidDecomposition {
        vector<int> par, size, marked;

        CentroidDecomposition(int N) :
            par(N), size(N), marked(N)
        {
            init();
        }

        void init() {
            fill(all(marked), 0);

```



```

    build(0, -1);
}

void build(int x, int p) {
    int n = dfs(x, -1);
    int centroid = get_centroid(x, -1, n);
    marked[centroid] = 1;

    par[centroid] = p;

    for (auto i : graph[centroid]) {
        if (!marked[i.fi])
            build(i.fi, centroid);
    }
}

int dfs(int x, int p) {
    size[x] = 1;
    for (auto i : graph[x])
        if (i.fi != p && !marked[i.fi])
            size[x] += dfs(i.fi, x);

    return size[x];
}

int get_centroid(int x, int p, int n) {
    for (auto i : graph[x])
        if (i.fi != p && size[i.fi] > n / 2 && !marked[i.fi])
            return get_centroid(i.fi, x, n);
    return x;
}

int operator[](int i) {
    return par[i];
}
};

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n, m; cin >> n >> m;
    vector<int> ans(n, inf);

    for (int i = 0; i < n - 1; ++i) {
        int a, b; cin >> a >> b;
        a--, b--;
        graph[a].pb(ii(b, 1));
        graph[b].pb(ii(a, 1));
    }

    LCA lca(n);
    CentroidDecomposition cd(n);

    int p = 0;
    while (p != -1) {
        ans[p] = min(ans[p], lca.query(0, p));
        p = cd[p];
    }

    for (int i = 0; i < m; ++i) {
        int a, b; cin >> a >> b;
        b--;
        if (a == 1) {
            int p = b;

```

```

        while (p != -1) {
            ans[p] = min(ans[p], lca.query(b, p));
            p = cd[p];
        }
    } else {
        int p = b, res = inf;
        while (p != -1) {
            res = min(res, lca.query(b, p) + ans[p]);
            p = cd[p];
        }
        cout << res << endl;
    }
}

return 0;
}

```

4.1.4 Pedido de Desculpas

```

#include <bits/stdc++.h>

#define MAX 1010
#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define sz size()
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

typedef long long ll;
typedef pair<int,int> ii;

int dp[60][MAX];
int v[MAX], w[MAX];

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int c, f, cas = 1;
    while (cin >> c >> f && (c || f)) {
        memset(dp, 0, sizeof dp);
        for (int i = 1; i <= f; ++i)
            cin >> w[i] >> v[i];

        for (int i = 1; i <= f; ++i)
            for (int j = 0; j <= c; ++j)
                if (j >= w[i])
                    dp[i][j] = max(dp[i-1][j], dp[i-1][j - w[i]] + v[i]);
                else
                    dp[i][j] = dp[i-1][j];
    }
}

```

```

    cout << "Teste " << cas << ende;
    cout << dp[f][c] << ende << ende;
    cas++;
}

return 0;
}

```

4.1.5 Easy Sudoku

```

#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int, int>;

vector<vector<int>> mat, filled;
vector<int> init = {0, 0, 0, 3, 3, 3, 6, 6, 6};

int check(int r, int c) {
    int ans = 0;
    for (int i = 0; i < 9; ++i) ans |= (1 << mat[r][i]);
    for (int i = 0; i < 9; ++i) ans |= (1 << mat[i][c]);

    for (int i = init[r]; i < init[r] + 3; ++i)
        for (int j = init[c]; j < init[c] + 3; ++j)
            ans |= (1 << mat[i][j]);

    return ans;
}

bool solve(int i, int j) {
    if (filled[i][j]) {
        if (i == 8 && j == 8) return true;
        else return solve(i + (j == 8), (j + 1) % 9);
    }

    int poss = check(i, j);
    for (int k = 1; k <= 9; ++k) {
        if ((poss & (1 << k)) == 0) {
            mat[i][j] = k;
            if (i == 8 && j == 8) return true;
            if (solve(i + (j == 8), (j + 1) % 9)) return true;
            mat[i][j] = filled[i][j];
        }
    }
}

```

```

return false;
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    mat.resize(9, vector<int>(9));

    int n; cin >> n;
    for (int cas = 0; cas < n; ++cas) {
        for (auto &i : mat)
            for (auto &j : i)
                cin >> j;

        filled = mat;

        if (solve(0, 0))
            for (auto &i : mat) {
                for (auto j : i) cout << j << " ";
                cout << ende;
            }
        else
            cout << "No solution" << ende;
    }

    return 0;
}

```

4.1.6 Engarrafamento

```

#include <bits/stdc++.h>

#define MAX 200
#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define sz size()
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

typedef long long ll;
typedef pair<int, int> ii;

vector<ii> graph[MAX];

struct Dijkstra {
    int N;
    vector<int> dist, vis;

    Dijkstra(int N) :
        N(N), dist(N), vis(N)
}

```

```

{}

void init() {
    fill(all(dist), inf);
    fill(all(vis), 0);
}

int run(int s, int d) {
    set<ii> pq;

    dist[s] = 0;
    pq.insert(ii(0, s));

    while (pq.size() != 0) {
        int u = pq.begin()->se;
        pq.erase(pq.begin());

        if (vis[u]) continue;
        vis[u] = 1;

        for (auto i : graph[u]) {
            int v = i.fi;
            int wt = i.se;

            if (!vis[v] && dist[v] > dist[u] + wt) {
                dist[v] = dist[u] + wt;
                pq.insert(ii(dist[v], v));
            }
        }

        return dist[d];
    }
} dijkstra(MAX);

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n, m;
    while (cin >> n >> m && (n || m)) {
        dijkstra.init();
        for (int i = 0; i <= n; ++i)
            graph[i].clear();

        for (int i = 0; i < m; ++i) {
            int o, d, t; cin >> o >> d >> t;
            graph[o].pb(ii(d, t));
        }

        int s, t; cin >> s >> t;
        int ans = dijkstra.run(s, t);
        cout << ((ans == inf) ? -1 : ans) << ende;
    }

    return 0;
}

```

4.1.7 Gincana

```
#include <bits/stdc++.h>
```

```

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define sz size()
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

typedef long long ll;
typedef pair<int,int> ii;

int cont[1010];
vector<int> graph[1010];

void dfs(int x) {
    cont[x] = 1;
    for (auto i : graph[x])
        if (!cont[i])
            dfs(i);
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n, m, a, b;
    cin >> n >> m;
    for (int i = 0; i < m; ++i) {
        cin >> a >> b;
        a--, b--;
        graph[a].pb(b);
        graph[b].pb(a);
    }

    int ans = 0;
    for (int i = 0; i < n; ++i) {
        if (!cont[i]) {
            dfs(i);
            ans++;
        }
    }

    cout << ans << '\n';
    return 0;
}

```

4.1.8 Janela

```
#include <bits/stdc++.h>
```

```

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f

```

```

#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int,int>;

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n; n = 3;
    vector<ii> v;
    v.pb(ii(0, -1));
    v.pb(ii(600, 2));
    for (int i = 0; i < n; ++i) {
        int x; cin >> x;
        v.pb(ii(x, 0));
        v.pb(ii(x+200, 1));
    }

    int curr = 0, ans = 0;
    bool count = false;
    sort(all(v));
    for (int i = 0; i < v.size(); ++i) {
        if (v[i].se == 0)
            curr++;
        else if (v[i].se == 1)
            curr--;

        if (count) {
            ans += v[i].fi - v[i-1].fi;
            count = false;
        }

        if (curr == 0)
            count = true;
    }

    cout << ans * 100 << ende;
    return 0;
}

```

4.1.9 Crescimento das Populacoes de bacilos

```

#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first

```

```

#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int,int>;

const int base = 1000000000;
const int base_d = 9;

struct BigInt {
    int sign;
    vector<int> num;

    BigInt() : sign(1) {}
    BigInt(ll x) { *this = x; }
    BigInt(const string &s) { read(s); }

    void operator=(const BigInt &x) {
        sign = x.sign;
        num = x.num;
    }

    void operator=(ll x) {
        sign = 1;
        if (x < 0) sign = -1, x = -x;
        for (; x > 0; x /= base)
            pb(x % base);
    }

    BigInt operator+(const BigInt &x) const {
        if (sign != x.sign) return *this - (-x);

        int carry = 0;
        BigInt res = x;

        for (int i = 0; i < max(size(), x.size()) || carry; ++i) {
            if (i == (int) res.size())
                res.push_back(0);

            res[i] += carry + (i < size() ? num[i] : 0);
            carry = res[i] >= base;
            if (carry) res[i] -= base;
        }

        return res;
    }

    BigInt operator-(const BigInt &x) const {
        if (sign != x.sign) return *this + (-x);
        if (abs() < x.abs()) return -(x - *this);

        int carry = 0;
        BigInt res = *this;

        for (int i = 0; i < x.size() || carry; ++i) {
            res[i] -= carry + (i < x.size() ? x[i] : 0);
            carry = res[i] < 0;

```

```

    if (carry) res[i] += base;
}

res.trim();
return res;
}

void operator*=(int x) {
    if (x < 0) sign = -sign, x = -x;

    int carry = 0;
    for (int i = 0; i < size() || carry; ++i) {
        if (i == size()) pb(0);
        ll cur = num[i] * (ll) x + carry;

        carry = (int) (cur / base);
        num[i] = (int) (cur % base);
    }

    trim();
}

BigInt operator*(int x) const {
    BigInt res = *this;
    res *= x;
    return res;
}

friend pair<BigInt, BigInt> divmod(const BigInt &a1,
    const BigInt &b1)
{
    int norm = base / (b1.back() + 1);
    BigInt a = a1.abs() * norm;
    BigInt b = b1.abs() * norm;
    BigInt q, r;
    q.resize(a.size());

    for (int i = a.size() - 1; i >= 0; i--) {
        r *= base;
        r += a[i];

        int s1 = r.size() <= b.size() ? 0 : r[b.size()];
        int s2 = r.size() <= b.size() - 1 ? 0 : r[b.size() - 1];
        int d = ((ll) base * s1 + s2) / b.back();

        r -= b * d;
        while (r < 0) r += b, --d;
        q[i] = d;
    }

    q.sign = a1.sign * b1.sign;
    r.sign = a1.sign;
    q.trim(); r.trim();

    return make_pair(q, r / norm);
}

BigInt operator/(const BigInt &x) const {
    return divmod(*this, x).fi;
}

BigInt operator%(const BigInt &x) const {
    return divmod(*this, x).se;
}

```

```

void operator/=(int x) {
    if (x < 0) sign = -sign, x = -x;

    for (int i = size() - 1, rem = 0; i >= 0; --i) {
        ll cur = num[i] + rem * (ll) base;
        num[i] = (int) (cur / x);
        rem = (int) (cur % x);
    }

    trim();
}

BigInt operator/(int x) const {
    BigInt res = *this;
    res /= x;
    return res;
}

int operator%(int x) const {
    if (x < 0) x = -x;

    int m = 0;
    for (int i = size() - 1; i >= 0; --i)
        m = (num[i] + m * (ll) base) % x;

    return m * sign;
}

void operator+=(const BigInt &x) {
    *this = *this + x;
}

void operator-=(const BigInt &x) {
    *this = *this - x;
}

void operator*=(const BigInt &x) {
    *this = *this * x;
}

void operator/=(const BigInt &x) {
    *this = *this / x;
}

bool operator<(const BigInt &x) const {
    if (sign != x.sign)
        return sign < x.sign;

    if (size() != x.size())
        return size() * sign < x.size() * x.sign;

    for (int i = size() - 1; i >= 0; i--)
        if (num[i] != x[i])
            return num[i] * sign < x[i] * sign;

    return false;
}

bool operator>(const BigInt &x) const {
    return x < *this;
}

bool operator<=(const BigInt &x) const {
    return !(x < *this);
}

bool operator>=(const BigInt &x) const {
    return !(*this < x);
}

bool operator==(const BigInt &x) const {

```

```

    return !(*this < x) && !(x < *this);
}
bool operator!=(const BigInt &x) const {
    return *this < x || x < *this;
}

void trim() {
    while (!empty() && !back()) pop_back();
    if (empty()) sign = 1;
}

bool is_zero() const {
    return empty() || (size() == 1 && !num[0]);
}

BigInt operator-() const {
    BigInt res = *this;
    res.sign = -sign;
    return res;
}

BigInt abs() const {
    BigInt res = *this;
    res.sign *= res.sign;
    return res;
}

ll to_long() const {
    ll res = 0;
    for (int i = size() - 1; i >= 0; i--)
        res = res * base + num[i];
    return res * sign;
}

friend BigInt gcd(const BigInt &a, const BigInt &b) {
    return b.is_zero() ? a : gcd(b, a % b);
}

friend BigInt lcm(const BigInt &a, const BigInt &b) {
    return a / gcd(a, b) * b;
}

void read(const string &s) {
    sign = 1;
    num.clear();

    int pos = 0;
    while (pos < s.size() &&
           (s[pos] == '-' || s[pos] == '+')) {
        if (s[pos] == '-')
            sign = -sign;
        ++pos;
    }

    for (int i = s.size() - 1; i >= pos; i -= base_d) {
        int x = 0;
        for (int j = max(pos, i - base_d + 1); j <= i; j++)
            x = x * 10 + s[j] - '0';
        num.push_back(x);
    }

    trim();
}

friend istream& operator>>(istream &stream, BigInt &v) {

```

```

    string s; stream >> s;
    v.read(s);
    return stream;
}

friend ostream& operator<<(ostream &stream, const BigInt &x) {
    if (x.sign == -1)
        stream << '-';

    stream << (x.empty() ? 0 : x.back());
    for (int i = x.size() - 2; i >= 0; --i)
        stream << setw(base_d) << setfill('0') << x.num[i];

    return stream;
}

static vector<int> convert_base(const vector<int> &a,
                                int oldd, int newd)
{
    vector<ll> p(max(oldd, newd) + 1);
    p[0] = 1;
    for (int i = 1; i < p.size(); i++)
        p[i] = p[i - 1] * 10;

    ll cur = 0;
    int curd = 0;
    vector<int> res;

    for (int i = 0; i < a.size(); i++) {
        cur += a[i] * p[curd];
        curd += oldd;

        while (curd >= newd) {
            res.pb(int(cur % p[newd]));
            cur /= p[newd];
            curd -= newd;
        }

        res.pb((int) cur);
        while (!res.empty() && !res.back())
            res.pop_back();
        return res;
    }

    static vector<ll> karatsuba(const vector<ll> &a,
                                const vector<ll> &b)
    {
        int n = a.size();
        vector<ll> res(n + n);

        if (n <= 32) {
            for (int i = 0; i < n; i++)
                for (int j = 0; j < n; j++)
                    res[i + j] += a[i] * b[j];
            return res;
        }

        int k = n >> 1;
        vector<ll> a1(a.begin(), a.begin() + k);
        vector<ll> a2(a.begin() + k, a.end());
        vector<ll> b1(b.begin(), b.begin() + k);
        vector<ll> b2(b.begin() + k, b.end());

        vector<ll> a1b1 = karatsuba(a1, b1);

```

```

vector<ll> a2b2 = karatsuba(a2, b2);

for (int i = 0; i < k; i++)
    a2[i] += a1[i];
for (int i = 0; i < k; i++)
    b2[i] += b1[i];

vector<ll> r = karatsuba(a2, b2);
for (int i = 0; i < a1b1.size(); i++) r[i] -= a1b1[i];
for (int i = 0; i < a2b2.size(); i++) r[i] -= a2b2[i];

for (int i = 0; i < r.size(); i++) res[i + k] += r[i];
for (int i = 0; i < a1b1.size(); i++) res[i] += a1b1[i];
for (int i = 0; i < a2b2.size(); i++) res[i + n] += a2b2[i];

return res;
}

BigInt operator*(const BigInt &x) const {
    vector<int> a6 = convert_base(this->num, base_d, 6);
    vector<int> b6 = convert_base(x.num, base_d, 6);

    vector<ll> a(all(a6));
    vector<ll> b(all(b6));

    while (a.size() < b.size()) a.pb(0);
    while (b.size() < a.size()) b.pb(0);
    while (a.size() & (a.size() - 1))
        a.pb(0), b.pb(0);

    vector<ll> c = karatsuba(a, b);

    BigInt res;
    int carry = 0;
    res.sign = sign * x.sign;

    for (int i = 0; i < c.size(); i++) {
        ll cur = c[i] + carry;
        res.pb((int) (cur % 1000000));
        carry = (int) (cur / 1000000);
    }

    res.num = convert_base(res.num, 6, base_d);
    res.trim();
    return res;
}

// Handles vector operations.
int back() const { return num.back(); }
bool empty() const { return num.empty(); }
size_t size() const { return num.size(); }

void pop_back() { num.pop_back(); }
void resize(int x) { num.resize(x); }
void push_back(int x) { num.push_back(x); }

int &operator[](int i) { return num[i]; }
int operator[](int i) const { return num[i]; }
};

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    vector<int> fib(1501);

```

```

fib[0] = 0;
fib[1] = 1;
for (int i = 2; i <= 1500; ++i)
    fib[i] = (fib[i-1] + fib[i-2]) % 1000;

int t; cin >> t;
for (int cas = 1; cas <= t; ++cas) {
    BigInt x; cin >> x;
    x = x % BigInt(1500);
    cout << setw(3) << setfill('0') << fib[x.to_long()]<< endl;
}

return 0;
}

```

4.1.10 Krakovia

```

#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int,int>;

const int base = 1000000000;
const int base_d = 9;

struct BigInt {
    int sign = 1;
    vector<int> num;

    BigInt() {}
    BigInt(ll x) { *this = x; }
    BigInt(const string &x) { read(x); }

    void operator=(ll x) {
        sign = 1;
        if (x < 0) sign = -1, x = -x;
        for (; x > 0; x /= base)
            num.pb(x % base);
    }

    BigInt operator+(const BigInt &x) const {
        if (sign != x.sign) return *this - (-x);

        BigInt ans = x;
        int carry = 0;
        for (int i = 0; i < max(size(), x.size()) || carry; ++i) {

```

```

    if (i == ans.size()) ans.push_back(0);

    if (i < size()) ans[i] += carry + num[i];
    else ans[i] += carry;

    carry = ans[i] >= base;
    if (carry) ans[i] -= base;
}

return ans;
}

BigInt operator-(const BigInt& x) const {
    if (sign != x.sign)
        return *this + (-x);
    if (abs() < x.abs())
        return -(x - *this);

    BigInt ans = *this;
    int carry = 0;
    for (int i = 0; i < x.size() || carry; ++i) {
        if (i < x.size()) ans[i] -= carry + x[i];
        else ans[i] -= carry;

        carry = ans[i] < 0;
        if (carry) ans[i] += base;
    }

    ans.trim();
    return ans;
}

void operator+=(const BigInt &v) {
    *this = *this + v;
}

void operator-=(const BigInt &v) {
    *this = *this - v;
}

void operator*=(int v) {
    if (v < 0) sign = -sign, v = -v;
    for (int i = 0, carry = 0; i < (int) size() || carry; ++i) {
        if (i == (int) size())
            num.push_back(0);

        ll cur = num[i] * (ll) v + carry;
        carry = (int) (cur / base);
        num[i] = (int) (cur % base);
    }

    trim();
}

BigInt operator*(int v) const {
    BigInt res = *this;
    res *= v;
    return res;
}

// Returns pair = (x / y, x % y).
friend pair<BigInt, BigInt> divmod(const BigInt &x,
    const BigInt &y) {
    int norm = base / (y.back() + 1);

    BigInt a = x.abs() * norm;

```

```

    BigInt b = y.abs() * norm;
    BigInt q, r;
    q.num.resize(a.size());

    for (int i = a.size() - 1; i >= 0; i--) {
        r *= base;
        r += a[i];
        int s1 = r.size() <= b.size() ? 0 : r.num[b.size()];
        int s2 = r.size() <= b.size() - 1 ? 0 : r.num[b.size() - 1];

        int d = ((ll) base * s1 + s2) / b.back();
        r -= b * d;
        while (r < 0) r += b, --d;
        q.num[i] = d;
    }

    q.sign = x.sign * y.sign;
    r.sign = x.sign;
    q.trim(); r.trim();
    return make_pair(q, r / norm);
}

BigInt operator/(const BigInt &x) const {
    return divmod(*this, x).fi;
}

void operator/=(int v) {
    if (v < 0) sign = -sign, v = -v;

    for (int i = (int) size() - 1, rem = 0; i >= 0; --i) {
        ll cur = num[i] + rem * (ll) base;
        num[i] = (int) (cur / v);
        rem = (int) (cur % v);
    }

    trim();
}

BigInt operator/(int x) const {
    BigInt res = *this;
    res /= x;
    return res;
}

// Removes leading zeros.
void trim() {
    while (!num.empty() && num.back() == 0)
        num.pop_back();

    if (num.empty())
        sign = 1;
}

bool operator<(const BigInt &x) const {
    if (sign != x.sign)
        return sign < x.sign;

    if (size() != x.size())
        return (size() * sign) < (x.size() * x.sign);

    for (int i = size() - 1; i >= 0; i--)
        if (num[i] != x[i])
            return (num[i] * sign) < (x[i] * x.sign);

    return false;
}

```



```

}

bool operator==(const BigInt &x) const { return !(*this < x) && !(x < *this); }
bool operator>(const BigInt &x) const { return (x < *this); }
bool operator<=(const BigInt &x) const { return !(x < *this); }
bool operator>=(const BigInt &x) const { return !(*this < x); }
bool operator!=(const BigInt &x) const { return !(*this == x); }

// Handles -x (change of sign).
BigInt operator-() const {
    BigInt ans = *this;
    ans.sign = -sign;
    return ans;
}

// Returs absolute value.
BigInt abs() const {
    BigInt ans = *this;
    ans.sign *= ans.sign;
    return ans;
}

// Transforms string into BigInt.
void read(const string &s) {
    sign = 1;
    num.clear();

    int pos = 0;
    while (pos < (int) s.size() &&
           (s[pos] == '-' || s[pos] == '+'))
    {
        if (s[pos] == '-')
            sign = -sign;
        ++pos;
    }

    for (int i = s.size() - 1; i >= pos; i -= base_d) {
        int x = 0;
        for (int j = max(pos, i - base_d + 1); j <= i; j++)
            x = x * 10 + s[j] - '0';
        num.push_back(x);
    }

    trim();
}

friend istream& operator>>(istream &stream, BigInt &v) {
    string s; stream >> s;
    v.read(s);
    return stream;
}

friend ostream& operator<<(ostream &stream,
    const BigInt &x) {
    if (x.sign == -1)
        stream << '-';

    stream << (x.empty() ? 0 : x.back());
    for (int i = x.size() - 2; i >= 0; --i)
        stream << setw(base_d) << setfill('0') << x.num[i];

    return stream;
}

// Handles vector operations.

```

```

int back() const { return num.back(); }
bool empty() const { return num.empty(); }
size_t size() const { return num.size(); }
void push_back(int x) { num.push_back(x); }

int &operator[](int i) { return num[i]; }
int operator[](int i) const { return num[i]; }
};

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n, f, cas = 1;
    while (cin >> n >> f && (n || f)) {
        vector<BigInt> v(n);
        for (int i = 0; i < n; ++i)
            cin >> v[i];

        BigInt sum = accumulate(all(v), BigInt("0"));
        cout << "Bill #" << cas << " costs " << sum << ": each friend should pay " << sum / f <<
            endl;
        cout << endl;
        cas++;
    }

    return 0;
}

```

4.1.11 Pie!

```

#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define sz size()
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

typedef long long ll;
typedef pair<int,int> ii;

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout << setprecision(9) << fixed;

    int t; cin >> t;
    while (t--) {
        int n, f; cin >> n >> f;
        vector<double> v(n);
    }
}

```

```

for (auto &i : v)
    cin >> i;

double l = 0.0, r = (double) llinf;
for (int i = 0; i < 300; ++i) {
    double m = (l + r) / 2.0;
    int p = 0;

    for (int j = 0; j < n; ++j)
        p += (int) floor((v[j] * v[j] * M_PI) / m);

    if (p >= f + 1)
        l = m;
    else
        r = m;
}

cout << l << endl;
}

return 0;
}

```

4.1.12 Haunted Graveyard

```

#include <bits/stdc++.h>

#define MAX 40
#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define endl '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int, int>;

int dx[] = {-1, 1, 0, 0};
int dy[] = {0, 0, 1, -1};

int mat[MAX][MAX];

struct BellmanFord {
    struct Edge { int u, v, w; };

    int N;
    vector<int> dist;
    vector<Edge> graph;

    BellmanFord(int N) :
        N(N), dist(N)
    { init(); }

    void init() {
        fill(all(dist), inf);
    }

    void add_edge(int u, int v, int w) {
        graph.pb({ u, v, w });
    }

    int run(int s, int d) {
        dist[s] = 0;

        for (int i = 0; i < N; ++i)
            for (auto e : graph)
                if (dist[e.u] != inf &&
                    dist[e.u] + e.w < dist[e.v])
                    dist[e.v] = dist[e.u] + e.w;

        for (auto e : graph)
            if (dist[e.u] != inf &&
                dist[e.u] + e.w < dist[e.v])
                return -inf;

        return dist[d];
    }
};

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int r, c;
    while (cin >> c >> r && (c || r)) {
        mset(mat, 0);
        BellmanFord bell(r*c);

        int g; cin >> g;
        for (int i = 0; i < g; ++i) {
            int x, y; cin >> x >> y;
            mat[y][x] = 1;
        }

        int e; cin >> e;
        for (int i = 0; i < e; ++i) {
            int x1, y1; cin >> x1 >> y1;
            int x2, y2; cin >> x2 >> y2;
            int t; cin >> t;

            bell.add_edge(y1 * c + x1, y2 * c + x2, t);
            mat[y1][x1] = 1;
        }

        mat[r-1][c-1] = 1;
        for (int i = 0; i < r; ++i)
            for (int j = 0; j < c; ++j)
                if (!mat[i][j])
                    for (int k = 0; k < 4; ++k) {
                        int di = i + dx[k];
                        int dj = j + dy[k];

                        if (di >= 0 && di < r && dj >= 0 && dj < c)
                            bell.add_edge(i*c + j, di*c + dj, 1);
                    }

        int ans = bell.run(0, (r-1)*c + (c-1));
    }
}

```

```

    if (ans == -inf) cout << "Never" << ende;
    else if (ans == inf) cout << "Impossible" << ende;
    else cout << ans << ende;
}

return 0;
}

```

4.1.13 Trash Removal

```

#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int, int>;

template <typename T = double>
struct Point {
    T x, y;

    Point() {}
    Point(T x, T y) : x(x), y(y) {}

    Point operator+(Point p) { return Point(x+p.x, y+p.y); }
    Point operator-(Point p) { return Point(x-p.x, y-p.y); }
    Point operator*(T s) { return Point(x*s, y*s); }

    T dot(Point p) { return (x*p.x) + (y*p.y); }
    T cross(Point p) { return (x*p.y) - (y*p.x); }
};

bool cw(Point<> a, Point<> b, Point<> c) {
    return (b - a).cross(c - a) <= 0;
}

template <typename T = double>
struct Segment {
    Point<T> a, b;

    Segment(Point<T> a, Point<T> b) : a(a), b(b) {}

    double dist(Point<T> p) {
        return (a - b).cross(p - b)/sqrt((b - a).dot(b - a));
    }
};

```

```

vector<Point<>> convex_hull(vector<Point<>> &v) {
    int k = 0;
    vector<Point<>> ans(v.size() * 2);

    sort(all(v), [](const Point<> &a, const Point<> &b) {
        return (a.x == b.x) ? (a.y < b.y) : (a.x < b.x);
    });

    for (int i = 0; i < v.size(); ++i) {
        for (; k >= 2 && cw(ans[k-2], ans[k-1], v[i]); --k);
        ans[k++] = v[i];
    }

    for (int i = v.size() - 2, t = k + 1; i >= 0; --i) {
        for (; k >= t && cw(ans[k-2], ans[k-1], v[i]); --k);
        ans[k++] = v[i];
    }

    ans.resize(k);
    return ans;
}

double width(vector<Point<>> &v) {
    vector<Point<>> h = convex_hull(v);

    int n = h.size() - 1;
    double ans = 1e14;

    h[0] = h[n];
    for (int i = 1, j = 1; i <= n; ++i) {
        while ((h[i] - h[i-1]).cross(h[j%n+1] - h[i-1]) >
            (h[i] - h[i-1]).cross(h[j] - h[i-1]))
            j = j % n + 1;

        Segment<> seg(h[i], h[i-1]);
        ans = min(ans, seg.dist(h[j]));
    }

    return ceil(ans*100)/100;
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout << setprecision(2) << fixed;

    int n, cas = 1;
    while (cin >> n && n) {
        vector<Point<>> v(n);
        for (int i = 0; i < n; ++i)
            cin >> v[i].x >> v[i].y;

        cout << "Case " << cas << ": " << width(v) << ende;
        cas++;
    }

    return 0;
}

```

4.1.14 Internet Bandwidth

```

#include <bits/stdc++.h>

```

```

#define MAX 1000
#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define sz size()
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

typedef long long ll;
typedef pair<int,int> ii;

int N;
int par[MAX];
int graph[MAX][MAX], rg[MAX][MAX];

bool cont[MAX];

bool path(int s, int t) {
    queue<int> Q;
    Q.push(s);
    cont[s] = true;

    while (!Q.empty()) {
        int u = Q.front(); Q.pop();

        if (u == t)
            return true;

        for (int i = 0; i < N; ++i)
            if (!cont[i] && rg[u][i]) {
                cont[i] = true;
                par[i] = u;
                Q.push(i);
            }
    }

    return false;
}

int edmonds_karp(int s, int t) {
    int ans = 0;
    par[s] = -1;

    mset(cont, 0);
    memcpy(rg, graph, sizeof(graph));

    while (path(s, t)) {
        int flow = inf;

        for (int i = t; par[i] != -1; i = par[i])
            flow = min(flow, rg[par[i]][i]);

        for (int i = t; par[i] != -1; i = par[i]) {

```

```

            rg[par[i]][i] -= flow;
            rg[i][par[i]] += flow;
        }

        ans += flow;
        mset(cont, 0);
    }

    return ans;
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n, cas = 1;
    while (cin >> n && n) {
        N = n;
        int s, t, c; cin >> s >> t >> c;

        mset(graph, 0);
        for (int i = 0; i < c; ++i) {
            int o, d, ca; cin >> o >> d >> ca;
            o--, d--;
            graph[o][d] += ca;
            graph[d][o] += ca;
        }

        cout << "Network " << cas++ << ende;
        cout << "The bandwidth is " << edmonds_karp(s-1, t-1) << "." << ende;
        cout << ende;
    }

    return 0;
}

```

4.1.15 Manutencao

```

#include <bits/stdc++.h>

#define MAX 500
#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

typedef long long ll;
typedef pair<int,int> ii;

vector<int> graph[MAX];

```

```

int visited[MAX], parent[MAX];

int L[MAX];
int low[MAX];
vector<int> articulations;

void dfs(int x) {
    int child = 0;
    visited[x] = 1;

    for (auto i : graph[x]) {
        if (!visited[i]) {
            child++;
            parent[i] = x;

            low[i] = L[i] = L[x] + 1;
            dfs(i);

            low[x] = min(low[x], low[i]);

            if ((parent[x] == -1 && child > 1) ||
                (parent[x] != -1 && low[i] >= L[x]))
                articulations.pb(x);

        } else if (parent[x] != i)
            low[x] = min(low[x], L[i]);
    }
}

void tarjan(int n) {
    mset(L, 0);
    mset(visited, 0);
    mset(parent, -1);
    articulations.clear();

    dfs(n);

    sort(all(articulations));
    articulations.erase(unique(all(articulations)), articulations.end());
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n, m;
    int cas = 1;
    while (cin >> n >> m && (n || m)) {
        for (int i = 0; i < n; ++i)
            graph[i].clear();

        for (int i = 0; i < m; ++i) {
            int x, y; cin >> x >> y;
            x--, y--;
            graph[x].pb(y);
            graph[y].pb(x);
        }

        tarjan(0);
        cout << "Teste " << cas << ende;
        if (!articulations.size())
            cout << "nenhum" << ende;
        else {
            for (auto i : articulations)

```

```

        cout << i + 1 << " ";
        cout << ende;
    }

    cas++;
    cout << ende;
}

return 0;
}

```

4.1.16 Mercado do Cairo

```

#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int,int>;

struct BIT2D {
    int N, M;
    vector<vector<int>> tree;

    BIT2D(int N, int M) :
        N(N), M(M), tree(N, vector<int>(M))
    { init(); }

    void init() {
        for (auto &i : tree)
            fill(all(i), 0);
    }

    int query(int idx, int idy) {
        int sum = 0;
        for (; idx > 0; idx -= (idx & -idx))
            for (int m = idy; m > 0; m -= (m & -m))
                sum = max(sum, tree[idx][m]);
        return sum;
    }

    void update(int idx, int idy, int val) {
        for (; idx < N; idx += (idx & -idx))
            for (int m = idy; m < M; m += (m & -m))
                tree[idx][m] = max(tree[idx][m], val);
    }
};

```

```
int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int t; cin >> t;
    for (int cas = 1; cas <= t; ++cas) {
        BIT2D bit(1001, 1001);

        int n; cin >> n;
        vector<int> x(n), y(n);
        for (int i = 0; i < n; ++i)
            cin >> x[i] >> y[i];

        for (int i = 0; i < n; ++i)
            bit.update(x[i], y[i], 1 + bit.query(x[i], y[i]));
        cout << bit.query(1000, 1000) << endl;
    }

    return 0;
}
```

4.1.17 Nlogonian Tickets

```
#include <bits/stdc++.h>

#define MAX 101010
#define MAXLOG 20
#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define sz size()
#define pb push_back
#define endl '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

typedef long long ll;
typedef pair<int, ll> ii;

vector<ii> graph[MAX];

ll h[MAX];
ll par[MAX][MAXLOG], cost[MAX][MAXLOG];

void dfs(int v, int p = -1, ll c = 0) {
    par[v][0] = p;
    cost[v][0] = c;

    if (p + 1)
        h[v] = h[p] + 1;

    for (int i = 1; i < MAXLOG; ++i)
        if (par[v][i - 1] + 1) {
            par[v][i] = par[par[v][i - 1]][i - 1];

```

```
            cost[v][i] = max(cost[v][i], max(cost[par[v][i - 1]][i - 1], cost[v][i - 1]));
        }

    for (auto u : graph[v])
        if (p != u.fi)
            dfs(u.fi, v, u.se);
}

void preprocess(int v) {
    memset(par, -1, sizeof par);
    memset(cost, 0, sizeof cost);
    dfs(v);
}

ll query(int p, int q) {
    ll ans = 0;

    if (h[p] < h[q])
        swap(p, q);

    for (int i = MAXLOG - 1; i >= 0; --i)
        if (par[p][i] + 1 && h[par[p][i]] >= h[q]) {
            ans = max(ans, cost[p][i]);
            p = par[p][i];
        }

    if (p == q)
        return ans;

    for (int i = MAXLOG - 1; i >= 0; --i)
        if (par[p][i] + 1 && par[p][i] != par[q][i]) {
            ans = max(ans, max(cost[p][i], cost[q][i]));
            p = par[p][i];
            q = par[q][i];
        }

    if (p == q) return ans;
    else return max(ans, max(cost[p][0], cost[q][0]));
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n, a, b;
    ll t;

    while (cin >> n && n) {
        for (int i = 0; i <= n; ++i)
            graph[i].clear();

        for (int i = 0; i < n - 1; ++i) {
            cin >> a >> b >> t;
            a--, b--;
            graph[a].pb(ii(b, t));
            graph[b].pb(ii(a, t));
        }

        int q;
        preprocess(0);

        cin >> q;
        for (int i = 0; i < q; ++i) {

```

```

    cin >> a >> b;
    a--, b--;
    cout << query(a, b) << '\n';
}
cout << '\n';
}

return 0;
}

```

4.1.18 Orkut

```

#include <bits/stdc++.h>

#define MAX 1000
#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int, int>;

vector<int> graph[MAX];

struct TopologicalSort {
    int N;
    stack<int> S;
    vector<int> cont;

    TopologicalSort(int N) :
        N(N), cont(N)
    {}

    void init() {
        fill(all(cont), 0);
    }

    bool dfs(int x) {
        cont[x] = 1;

        for (auto i : graph[x]) {
            if (cont[i] == 1)
                return true;
            if (!cont[i] && dfs(i))
                return true;
        }

        cont[x] = 2;
        S.push(x);
    }
}

```

```

        return false;
    }

    bool run(vector<int> &tsort) {
        init();

        bool cycle = false;
        for (int i = 0; i < N; ++i) {
            if (!cont[i])
                cycle |= dfs(i);
        }

        if (cycle)
            return true;

        while (!S.empty()) {
            tsort.pb(S.top());
            S.pop();
        }

        return false;
    }
};

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n;
    for (int cas = 1; cin >> n && n; ++cas) {
        for (int i = 0; i < MAX; ++i)
            graph[i].clear();

        vector<string> v(n);
        map<string, int> M;
        for (int i = 0; i < n; ++i) {
            cin >> v[i];
            M[v[i]] = i;
        }

        for (int i = 0; i < n; ++i) {
            string a; cin >> a;
            int m; cin >> m;
            for (int j = 0; j < m; ++j) {
                string b; cin >> b;
                graph[M[b]].pb(M[a]);
            }
        }

        TopologicalSort ts(n);
        vector<int> ans;
        bool cycle = ts.run(ans);

        cout << "Teste " << cas << ende;
        if (cycle)
            cout << "impossivel" << ende;
        else {
            for (auto i : ans)
                cout << v[i] << " ";
            cout << ende;
        }
        cout << ende;
    }

    return 0;
}

```

4.1.19 To Poland

```
#include <bits/stdc++.h>

#define MAX 101010
#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define sz size()
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

typedef long long ll;
typedef pair<int,int> ii;

ll N, tree[4 * MAX], v[MAX];

void build_tree(int node = 1, int a = 0, int b = N) {
    if (a > b)
        return;

    if (a == b) {
        tree[node] = v[a];
        return;
    }

    build_tree(node * 2, a, (a + b) / 2);
    build_tree(node * 2 + 1, 1 + (a + b) / 2, b);

    tree[node] = max(tree[node * 2], tree[node * 2 + 1]);
}

void update_tree(int idx, ll val, int node = 1, int a = 0, int b = N) {
    if (a > b || a > idx || b < idx)
        return;

    if (a == b) {
        tree[node] = val;
        return;
    }

    update_tree(idx, val, node * 2, a, (a + b) / 2);
    update_tree(idx, val, node * 2 + 1, 1 + (a + b) / 2, b);

    tree[node] = max(tree[node * 2], tree[node * 2 + 1]);
}

int query_tree(int i, int j, int node = 1, int a = 0, int b = N) {
```

```
    if (a > b || a > j || b < i)
        return 0;

    if (a >= i && b <= j)
        return tree[node];

    int res1 = query_tree(i, j, node * 2, a, (a + b) / 2);
    int res2 = query_tree(i, j, node * 2 + 1, 1 + (a + b) / 2, b);

    return max(res1, res2);
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    ll t, n, m;
    cin >> t;
    for (int cas = 0; cas < t; ++cas) {
        cout << "Testcase " << cas << ":\n";
        cin >> n >> m;
        N = n;

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

        ll q, a, b;
        string op;
        cin >> q;
        for (int i = 0; i < q; ++i) {
            cin >> op;
            if (op[0] == 'A') {
                cin >> a;
                m += a;
            } else if (op[0] == 'B') {
                cin >> a >> b;
                update_tree(a, b);
            } else {
                cin >> a >> b;
                cout << abs(m - query_tree(a, b)) << '\n';
            }
        }
        cout << '\n';
    }

    return 0;
}
```

4.1.20 Serie de Tubos

```
#include <bits/stdc++.h>

#define MAX 1010
#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
```



```

#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

typedef long long ll;
typedef pair<int,int> ii;

vector<int> graph[MAX];
int visited[MAX], parent[MAX];

int L[MAX];
int low[MAX];
vector<ii> bridges;

void dfs(int x) {
    int child = 0;
    visited[x] = 1;

    for (auto i : graph[x]) {
        if (!visited[i]) {
            child++;
            parent[i] = x;

            low[i] = L[i] = L[x] + 1;
            dfs(i);

            low[x] = min(low[x], low[i]);

            if (low[i] > L[x])
                bridges.pb(ii(x, i));
        } else if (parent[x] != i)
            low[x] = min(low[x], L[i]);
    }
}

void tarjan(int n) {
    mset(visited, 0);
    mset(parent, -1);
    mset(L, 0);
    bridges.clear();
    dfs(n);
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n, m;
    while (cin >> n >> m && (n || m)) {
        for (int i = 0; i < n; ++i)
            graph[i].clear();

        for (int i = 0; i < m; ++i) {
            int x, y; cin >> x >> y;
            x--, y--;
            graph[x].pb(y);
            graph[y].pb(x);
        }
    }
}

```

```

    tarjan(0);
    cout << (bridges.size() ? "N" : "S") << ende;
}

return 0;
}

```

4.1.21 Quantas Chamadas Recursivas

```

#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

typedef unsigned long long ll;
typedef pair<int,int> ii;

struct matrix {
    int N;
    ll b;
    vector<vector<ll>> m;

    matrix(int N, ll b) : N(N), b(b) {
        m = vector<vector<ll>>(N, vector<ll>(N, 0));
    }

    matrix operator*(matrix a) {
        matrix res(N, b);
        for (int i = 0; i < N; i++)
            for (int j = 0; j < N; j++) {
                res[i][j] = 0;

                for (int k = 0; k < N; k++)
                    res[i][j] = ((m[i][k] * a[k][j]) % b) + res[i][j] % b;
            }

        return res;
    }

    void to_identity() {
        for (auto &i : m)
            fill(all(i), 0);
        for (int i = 0; i < N; ++i)
            m[i][i] = 1;
    }

    vector<ll> &operator[](int i) {
        return m[i];
    }
}

```

```

};

ll fast_pow(matrix in, ll n, ll b) {
    matrix ans(2, b);
    ans.to_identity();

    while (n) {
        if (n & 1)
            ans = ans * in;

        n >>= 1;
        in = in * in;
    }

    return ans[0][0];
}

ll solve(ll n, ll b) {
    matrix in(2, b);

    in[0][0] = 1;
    in[0][1] = 1;

    in[1][0] = 1;
    in[1][1] = 0;

    return fast_pow(in, n, b);
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    ll n, b;
    int cas = 1;
    while (cin >> n >> b && (n || b)) {
        ll ans = solve(n, b) % b;
        ans = (ans + ans) % b;
        ans = (ans - 1 + b) % b;

        cout << "Case " << cas++ << ":" << " " << n << " " << b << " " << ans << endl;
    }

    return 0;
}

```

4.1.22 Ir e Vir

```

#include <bits/stdc++.h>

#define MAX 2010
#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()

```

```

#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

typedef long long ll;
typedef pair<int,int> ii;

stack<int> S;
vector<int> graph[MAX];

int ncomp, ind;
int visited[MAX];

int id[MAX];

int low[MAX];

vector<int> scc[MAX];

void dfs(int x) {
    id[x] = low[x] = ind++;
    visited[x] = 1;

    S.push(x);

    for (auto i : graph[x])
        if (id[i] == -1) {
            dfs(i);

            low[x] = min(low[x], low[i]);

        } else if (visited[i])
            low[x] = min(low[x], id[i]);

    if (low[x] == id[x]) {
        int w;

        do {
            w = S.top(); S.pop();
            visited[w] = 0;
            scc[ncomp].pb(w);
        } while (w != x);

        ncomp++;
    }
}

int tarjan(int n) {
    mset(id, -1);
    ncomp = ind = 0;

    for (int i = 0; i < n; ++i)
        scc[i].clear();

    for (int i = 0; i < n; ++i)
        if (id[i] == -1)
            dfs(i);

    return ncomp;
}

int main() {
    ios::sync_with_stdio(0);

```

```

cin.tie(0);

int n, m;
while (cin >> n >> m && (n || m)) {
    for (int i = 0; i < n; ++i)
        graph[i].clear();

    for (int i = 0; i < m; ++i) {
        int v, w, p; cin >> v >> w >> p;
        v--, w--;
        if (p == 1)
            graph[v].pb(w);
        else {
            graph[v].pb(w);
            graph[w].pb(v);
        }
    }

    int ans = tarjan(n);
    cout << (ans == 1) << ende;
}

return 0;
}

```

4.1.23 Jogo da Velha

```

#include <bits/stdc++.h>

#define MAX 10101
#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int, int>;

int dp[MAX];
//int foi[MAX];

int solve(int n) {
    if (n < 5)
        return 0;
    if (dp[n] != -1)
        return dp[n];

    vector<int> foi(MAX, 0);
    for (int j = 2; j < n - 2; ++j)
        foi[solve(j) ^ solve(n - 1 - j)] = 1;
}

```

```

for (int j = 0; j < MAX; ++j)
    if (!foi[j]) {
        dp[n] = j;
        break;
    }

return dp[n];
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n;
    mset(dp, -1);
    while (cin >> n && n) {
        string s; cin >> s;

        bool done = false;
        for (int i = 1; i < n; ++i)
            if (s[i] == 'X' && s[i-1] == 'X') done = true;
        for (int i = 2; i < n; ++i)
            if (s[i] == 'X' && s[i-2] == 'X') done = true;

        if (done) {
            cout << 'S' << ende;
            continue;
        }

        int ans = 0;
        int last = -3;
        for (int i = 0; i < n; ++i) {
            if (s[i] == 'X') {
                ans ^= solve(i-last-1);
                last = i;
            }
        }

        ans ^= solve(n-last+1);
        cout << (ans ? 'S' : 'N') << endl;
    }

    return 0;
}

```

4.1.24 Ant's Colony

```

#include <bits/stdc++.h>

#define MAX 101010
#define MAXLOG 20
#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define sz size()
#define pb push_back
#define ende '\n'

```

```

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

typedef long long ll;
typedef pair<ll,ll> ii;

vector<ii> graph[MAX];

ll h[MAX];
ll par[MAX][MAXLOG], cost[MAX][MAXLOG];

void dfs(int v, int p = -1, ll c = 0) {
    par[v][0] = p;
    cost[v][0] = c;

    if (p + 1)
        h[v] = h[p] + 1;

    for (int i = 1; i < MAXLOG; ++i)
        if (par[v][i - 1] + 1) {
            par[v][i] = par[par[v][i - 1]][i - 1];
            cost[v][i] += cost[v][i - 1] + cost[par[v][i - 1]][i - 1];
        }

    for (auto u : graph[v])
        if (p != u.fi)
            dfs(u.fi, v, u.se);
}

void preprocess(int v) {
    memset(par, -1, sizeof par);
    memset(cost, 0, sizeof cost);
    dfs(v);
}

ll query(int p, int q) {
    ll ans = 0;

    if (h[p] < h[q])
        swap(p, q);

    for (int i = MAXLOG - 1; i >= 0; --i)
        if (par[p][i] + 1 && h[par[p][i]] >= h[q]) {
            ans += cost[p][i];
            p = par[p][i];
        }

    if (p == q)
        return ans;

    for (int i = MAXLOG - 1; i >= 0; --i)
        if (par[p][i] + 1 && par[p][i] != par[q][i]) {
            ans += cost[p][i] + cost[q][i];
            p = par[p][i];
            q = par[q][i];
        }

    if (p == q) return ans;
    else return ans + cost[p][0] + cost[q][0];
}

```

```

int main() {
    ll li;
    int ai, n;
    int q, s, t;

    while (scanf("%d", &n) && n) {
        for (int i = 0; i < n + 1; ++i)
            graph[i].clear();

        for (int i = 1; i <= n - 1; ++i) {
            scanf("%d %lld", &ai, &li);
            graph[ai].pb(ii(i, li));
        }

        preprocess(0);

        scanf("%d", &q);
        for (int i = 0; i < q; ++i) {
            scanf("%d %d", &s, &t);
            if (i) printf(" ");
            printf("%lld", query(s, t));
        }
        printf("\n");
    }

    return 0;
}

```

4.1.25 Jupyter Ataca!

```

#include <bits/stdc++.h>

#define MAX 101010
#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define sz size()
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

typedef long long ll;
typedef pair<int,int> ii;

ll tree[MAX];
ll b, p, l, n;

ll query(ll idx) {
    ll sum = 0;
    for (; idx > 0; idx -= (idx & -idx))
        sum = (sum + tree[idx] + p) % p;
}

```

```

    return sum % p;
}

void update(ll idx, ll val) {
    for (; idx <= MAX; idx += (idx & -idx))
        tree[idx] = (tree[idx] + val + p) % p;
}

ll power(ll x, ll y) {
    ll ans = 1;

    while (y) {
        if (y & 1)
            ans = (ans * x) % p;

        y >>= 1;
        x = (x * x) % p;
    }

    return ans % p;
}

ll mod_inverse(ll a) {
    return power(a, p - 2);
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    while (cin >> b >> p >> l >> n && (b || p || l || n)) {
        mset(tree, 0);

        vector<ll> powB(l + 1); powB[0] = 1;
        for (int i = 1; i <= l; ++i)
            powB[i] = (powB[i-1] * (b % p)) % p;

        for (int i = 0; i < n; ++i) {
            string op; cin >> op;

            if (op[0] == 'E') {
                ll pos, val; cin >> pos >> val;

                // Remove current value from bit
                update(pos, -((query(pos) - query(pos - 1) + p) % p));

                // Insert new value into bit (val*b^(l-pos))
                update(pos, ((val % p) * powB[l - pos]) % p);
            } else {
                ll L, R; cin >> L >> R;

                // Get range sum
                ll sum = (query(R) - query(L - 1) + p) % p;

                // Result is (sum / b^(l-R)) % p
                cout << (sum * mod_inverse(powB[l - R])) % p << " ";
            }
        }

        cout << "- " << " ";
    }
}

```

```

    return 0;
}

```

4.1.26 Emoticons

```

#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define sz size()
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

typedef long long ll;
typedef pair<int,int> ii;

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n, m;
    while (cin >> n >> m && (n || m)) {
        vector<string> emo(n);
        for (auto &i : emo) cin >> i;
        cin.ignore();

        int ans = 0;
        for (int i = 0; i < m; ++i) {
            string s;
            getline(cin, s);

            for (int j = 0; j < s.sz; ++j) {
                for (int k = 0; k < n; ++k) {
                    if (s[j] == emo[k].back()) {
                        for (int l = j, ll = emo[k].sz - 1; ll >= 0 && l >= 0; ll--, l--) {
                            if (s[l] != emo[k][ll])
                                break;

                            if (ll == 0) {
                                s[j] = ' ';
                                ans++;
                            }
                        }
                    }
                }
            }
        }

        cout << ans << " ";
    }
}

```

```
    return 0;
}
```

4.1.27 Camadas de Cebolas

```
#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define sz size()
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

typedef long long ll;
typedef pair<int,int> ii;
typedef pair<double,double> dd;

double cross(dd a, dd b, dd c) {
    return (b.fi - a.fi) * (c.se - a.se) - (b.se - a.se) * (c.fi - a.fi);
}

int convex_hull(vector<dd> &v) {
    int k = 0;
    vector<int> ans(v.sz * 2);

    sort(v.begin(), v.end(), [](const dd &a, const dd &b) -> bool {
        return (a.fi == b.fi) ? (a.se < b.se) : (a.fi < b.fi);
    });

    for (int i = 0; i < v.sz; ++i) {
        while (k >= 2 && cross(v[ans[k - 2]], v[ans[k - 1]], v[i]) < 0) k--;
        ans[k++] = i;
    }

    for (int i = v.sz - 2, t = k + 1; i >= 0; --i) {
        while (k >= t && cross(v[ans[k - 2]], v[ans[k - 1]], v[i]) < 0) k--;
        ans[k++] = i;
    }

    ans.resize(k);
    sort(rall(ans));
    ans.erase(unique(all(ans)), ans.end());

    for (auto i : ans)
        v.erase(v.begin() + i);

    return k - 1;
}
```

```
int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n;
    while (cin >> n && n) {
        vector<dd> v;
        double x, y;

        for (int i = 0; i < n; ++i) {
            cin >> x >> y;
            v.pb(dd(x, y));
        }

        int ans = 0;
        while (v.sz) {
            convex_hull(v);
            ans++;
        }

        if (ans % 2) cout << "Take this onion to the lab!\n";
        else cout << "Do not take this onion to the lab!\n";
    }

    return 0;
}
```

4.1.28 Homem-Elefante-Rato

```
#include <bits/stdc++.h>

#define MAX 101010
#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define sz size()
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

typedef long long ll;
typedef pair<int,int> ii;

typedef struct elem {
    int h, e, r;

    elem() : h(0), e(0), r(0) {}
    elem(int h, int e, int r) : h(h), e(e), r(r) {}

    elem operator+(const elem &x) {
        return elem(h + x.h, e + x.e, r + x.r);
    }
}
```

```

}

void change(int n) {
    int a = h, b = e, c = r;
    if (n % 3 == 1) {
        e = a;
        r = b;
        h = c;
    } else if (n % 3 == 2) {
        e = c;
        r = a;
        h = b;
    }
}
} elem;

int N;
elem v[MAX];
elem tree[MAX * 4];
int lazy[MAX * 4];

#define left(x) ((x << 1))
#define right(x) ((x << 1) + 1)

void build(int node = 1, int a = 0, int b = N - 1) {
    if (a > b)
        return;

    if (a == b) {
        tree[node] = v[a];
        return;
    }

    build(left(node), a, (a + b) / 2);
    build(right(node), (a + b) / 2 + 1, b);
    tree[node] = tree[node * 2] + tree[node * 2 + 1];
}

void push(int node, int a, int b, int val) {
    tree[node].change(val);

    if (a != b) {
        lazy[left(node)] += val;
        lazy[right(node)] += val;
    }

    lazy[node] = 0;
}

void update(int i, int j, int node = 1, int a = 0, int b = N - 1) {
    if (lazy[node] != 0)
        push(node, a, b, lazy[node]);

    if (a > b or a > j or b < i)
        return;

    if (a >= i and b <= j) {
        push(node, a, b, 1);
        return;
    }

    update(i, j, left(node), a, (a + b) / 2);

```

```

    update(i, j, right(node), (a + b) / 2 + 1, b);
    tree[node] = tree[node * 2] + tree[node * 2 + 1];
}

elem query(int i, int j, int node = 1, int a = 0, int b = N - 1) {
    if (a > b || a > j || b < i)
        return elem();

    if (lazy[node])
        push(node, a, b, lazy[node]);

    if (a >= i and b <= j)
        return tree[node];

    elem q1 = query(i, j, left(node), a, (a + b) / 2);
    elem q2 = query(i, j, right(node), (a + b) / 2 + 1, b);
    return q1 + q2;
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int m;
    while (cin >> N >> m) {
        for (int i = 0; i < MAX * 4; ++i) {
            tree[i] = elem();
            lazy[i] = 0;
        }

        for (int i = 0; i < N; ++i) {
            v[i].h = 1;
            v[i].e = v[i].r = 0;
        }

        build();

        for (int i = 0; i < m; ++i) {
            char op;
            int a, b; cin >> op >> a >> b;

            if (op == 'C') {
                elem x = query(a - 1, b - 1);
                cout << x.h << " " << x.e << " " << x.r << ende;
            } else
                update(a - 1, b - 1);
        }

        cout << ende;
    }

    return 0;
}

```

4.1.29 O Labirinto de Ninguem

```

#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007

```

```
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))
```

```
using namespace std;
```

```
typedef long long ll;
typedef pair<int,int> ii;
typedef pair<ii,ii> elem;
```

```
int dx[] = {1, -1, 0, 0};
int dy[] = {0, 0, 1, -1};
int cont[110][110][1 << 8];
```

```
int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
```

```
    ii arya;
    string x;
    vector<string> mat;
```

```
    for (int i = 0; cin >> x; ++i) {
        mat.pb(x);
        for (int j = 0; j < x.size(); ++j)
            if (x[j] == '@')
                arya = {i, j};
    }
```

```
    int n = mat.size(), m = mat[0].size();
```

```
    mset(cont, 0);
    queue<elem> Q;
    Q.push({arya, {0, 0}});
    cont[arya.fi][arya.se][0] = 1;
```

```
    while (!Q.empty()) {
        elem curr = Q.front(); Q.pop();
```

```
        if (mat[curr.fi.fi][curr.fi.se] == '*')
            return cout << curr.se.se << ende, 0;
```

```
        for (int i = 0; i < 4; ++i) {
            int x = curr.fi.fi + dx[i];
            int y = curr.fi.se + dy[i];
```

```
            if (x >= 0 && x < n && y >= 0 && y < m &&
                !cont[x][y][curr.se.fi] && mat[x][y] != '#') {
                int msk = curr.se.fi;
```

```
                if (mat[x][y] >= 'A' && mat[x][y] <= 'G') {
                    if (msk & (1 << (mat[x][y] - 'A'))) {
                        cont[x][y][msk] = 1;
                        Q.push({x, y}, {msk, curr.se.se + 1});
                    }
                } else {
```

```
                    if (mat[x][y] >= 'a' && mat[x][y] <= 'g')
```

```
                        msk |= (1 << (mat[x][y] - 'a'));

                        cont[x][y][msk] = 1;
                        Q.push({x, y}, {msk, curr.se.se + 1});
                    }
                }
            }

            cout << "--" << ende;
            return 0;
        }
    }
```

4.1.30 Lobos Stark

```
#include <bits/stdc++.h>
```

```
#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f
```

```
#define fi first
#define se second
#define pb push_back
#define ende '\n'
```

```
#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))
```

```
using namespace std;
```

```
using ll = long long;
using ii = pair<int,int>;
```

```
vector<int> gale_shapley(const vector<vector<int>> &pref) {
    int n = pref[0].size();
    vector<int> w_part(n, -1);
    vector<int> m_part(n, -1);
    vector<int> start(n, 0);
```

```
    while (true) {
        int m;
        for (m = 0; m < n; ++m)
            if (m_part[m] == -1)
                break;
```

```
    if (m == n) break;
```

```
    for (; start[m] < n && m_part[m] == -1; ++start[m]) {
        int w = pref[m][start[m]];
```

```
        if (w_part[w - n] == -1) {
            w_part[w - n] = m;
            m_part[m] = w;
        } else {
            int m1 = w_part[w - n];
            bool pref_m = false;
```

```
            for (int j = 0; j < n; ++j)
                if (pref[w][j] == m) {
```



```

        pref_m = true;
        break;
    } else if (pref[w][j] == m1)
        break;

    if (pref_m) {
        w_part[w - n] = m;
        m_part[m] = w;
        m_part[m1] = -1;
    }
}
}
}

return m_part;
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n; cin >> n;
    vector<vector<int>>> pref(2*n, vector<int>(n));

    int k1 = 0, k2 = n;
    map<string,int> M;
    vector<string> names(2*n);
    for (int i = 0; i < n; ++i) {
        string s; cin >> s;
        if (M.find(s) == M.end()) {
            names[k1] = s;
            M[s] = k1++;
        }

        for (int j = 0; j < n; ++j) {
            string t; cin >> t;
            if (M.find(t) == M.end()) {
                names[k2] = t;
                M[t] = k2++;
            }

            pref[M[s]][j] = M[t];
        }
    }

    for (int i = 0; i < n; ++i) {
        string s; cin >> s;
        if (M.find(s) == M.end()) {
            names[k2] = s;
            M[s] = k2++;
        }

        for (int j = 0; j < n; ++j) {
            string t; cin >> t;
            if (M.find(t) == M.end()) {
                names[k1] = t;
                M[t] = k1++;
            }

            pref[M[s]][j] = M[t];
        }
    }

    vector<int> ans = gale_shapley(pref);
    for (int i = 0; i < n; ++i)

```

```

        cout << names[i] << " " << names[ans[i]] << ende;
        return 0;
    }
}

```

4.1.31 A Caminhada da Vergonha

```

#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int,int>;
using dd = pair<double,double>;

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout << setprecision(2) << fixed;

    int n;
    double l = 0.0, r;
    cin >> n >> r;

    vector<dd> v(n);
    for (auto &i : v) cin >> i.fi >> i.se;

    double grtl, grtr;
    for (int i = 0; i < 100; ++i) {
        double lt = (r - l) / 3.0 + l;
        double rt = ((r - l) * 2.0) / 3.0 + l;

        grtl = 0.0; for (auto j : v) grtl = max(grtl, hypot(j.fi - lt, j.se));
        grtr = 0.0; for (auto j : v) grtr = max(grtr, hypot(j.fi - rt, j.se));

        if (grtl > grtr) l = lt;
        else r = rt;
    }

    cout << l << " " << grtl << ende;
    return 0;
}

```

4.1.32 Bolsa de Valores (iterativo)

```

#include <bits/stdc++.h>

```

```

#define MAX 201010
#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define sz size()
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

typedef long long ll;
typedef pair<int,int> ii;

int n, c;
int v[MAX];
int dp[MAX][2];

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

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

    for (int i = n-1; i >= 0; --i) {
        dp[i][0] = max(dp[i+1][1] - c - v[i], dp[i+1][0]);
        dp[i][1] = max(dp[i+1][0] + v[i], dp[i+1][1]);
    }

    cout << dp[0][0] << ende;
    return 0;
}

```

4.1.33 Bolsa de Valores (recursivo)

```

#include <bits/stdc++.h>

#define MAX 201010
#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define sz size()
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

```

```

using namespace std;

typedef long long ll;
typedef pair<int,int> ii;

int n, c;
int v[MAX];
int dp[MAX][2];

int solve(int i, bool hold) {
    if (i == n)
        return 0;

    if (dp[i][hold] != inf)
        return dp[i][hold];

    if (!hold)
        return dp[i][hold] = max(solve(i + 1, true) - c - v[i],
                                   solve(i + 1, false));

    else
        return dp[i][hold] = max(solve(i + 1, false) + v[i],
                                   solve(i + 1, true));
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    mset(dp, 0x3f);

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

    cout << solve(0, false) << ende;
    return 0;
}

```

4.1.34 Colheita de Caju

```

#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int,int>;

```

```
int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int l, c, m, n; cin >> l >> c >> m >> n;
    vector<vector<int>> mat(l, vector<int>(c));
    for (auto &i : mat) for (auto &j : i) cin >> j;

    vector<vector<int>> sum(l+1, vector<int>(c+1, 0));
    for (int i = 1; i <= l; ++i)
        for (int j = 1; j <= c; ++j)
            if (j == 1) sum[i][j] = mat[i-1][j-1];
            else sum[i][j] = sum[i][j-1] + mat[i-1][j-1];

    for (int i = 2; i <= l; ++i)
        for (int j = 1; j <= c; ++j)
            sum[i][j] += sum[i-1][j];

    int ans = 0;
    for (int i = m; i <= l; ++i)
        for (int j = n; j <= c; ++j)
            ans = max(ans, sum[i][j] - sum[i-m][j] - sum[i][j-n] + sum[i-m][j-n]);

    cout << ans << endl;
    return 0;
}
```

4.1.35 Data Flow

```
#include <bits/stdc++.h>

#define MAXN 110
#define MAXM 20010
#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int, int>;

struct MinCostMaxFlow {
    struct Edge {
        int u, v, cap, cost;
    };

    int N;
    vector<Edge> edges;
    vector<vector<int>> adj;
    vector<int> vis, dist, par, ind;
```

```
MinCostMaxFlow(int N) :
    N(N), vis(N), dist(N), par(N), ind(N), adj(N) {}

void add_edge(int u, int v, int cap, int cost) {
    adj[u].pb(edges.size());
    edges.pb({ u, v, cap, cost });

    adj[v].pb(edges.size());
    edges.pb({ v, u, 0, -cost });
}

bool spfa(int s, int t) {
    fill(all(dist), inf);
    dist[s] = 0;

    queue<int> Q;
    Q.push(s);

    while (!Q.empty()) {
        int u = Q.front(); Q.pop();
        vis[u] = 0;

        for (auto i : adj[u]) {
            Edge &e = edges[i];
            int v = e.v;

            if (e.cap > 0 && dist[v] > dist[u] + e.cost) {
                dist[v] = dist[u] + e.cost;
                par[v] = u;
                ind[v] = i;

                if (!vis[v]) {
                    Q.push(v);
                    vis[v] = 1;
                }
            }
        }
    }

    return dist[t] < inf;
}

pair<ll, int> run(int s, int t) {
    ll mincost = 0;
    int maxflow = 0;

    while (spfa(s, t)) {
        ll flow = inf;
        for (int i = t; i != s; i = par[i])
            flow = min(flow, (ll) edges[ind[i]].cap);

        for (int i = t; i != s; i = par[i]) {
            edges[ind[i]].cap -= flow;
            edges[ind[i]^1].cap += flow;
        }

        mincost += flow * dist[t];
        maxflow += flow;
    }

    return make_pair(mincost, maxflow);
};
```

```
int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n, m;
    while (cin >> n >> m) {
        MinCostMaxFlow mcmf(110);

        vector<pair<ii,int>> aux(m);
        for (int i = 0; i < m; ++i)
            cin >> aux[i].fi.fi >> aux[i].fi.se >> aux[i].se;

        int d, k; cin >> d >> k;
        mcmf.add_edge(0, 1, d, 0);
        for (int i = 0; i < m; ++i) {
            mcmf.add_edge(aux[i].fi.fi, aux[i].fi.se, k, aux[i].se);
            mcmf.add_edge(aux[i].fi.se, aux[i].fi.fi, k, aux[i].se);
        }

        pair<ll,int> ans = mcmf.run(0, n);

        if (ans.se != d) cout << "Impossible." << ende;
        else cout << ans.fi << ende;
    }

    return 0;
}
```

4.1.36 I Love Strings! (kmp)

```
#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int,int>;

struct KMP {
    string patt;
    vector<int> table;

    KMP(string patt) :
        patt(patt), table(patt.size()+1)
    { preprocess(); }

    void preprocess() {
        fill(all(table), -1);
```

```
        for (int i = 0, j = -1; i < patt.size(); ++i) {
            while (j >= 0 && patt[i] != patt[j])
                j = table[j];
            table[i + 1] = ++j;
        }

        bool search(const string &txt) {
            bool found = false;

            for (int i = 0, j = 0; i < txt.size(); ++i) {
                while (j >= 0 && txt[i] != patt[j])
                    j = table[j];
                j++;

                if (j == patt.size()) {
                    found = true;
                    j = table[j];
                }
            }

            return found;
        }
    };

    int main() {
        ios::sync_with_stdio(0);
        cin.tie(0);

        int t; cin >> t;
        for (int cas = 1; cas <= t; ++cas) {
            string s; cin >> s;
            int q; cin >> q;
            for (int i = 0; i < q; ++i) {
                string t; cin >> t;
                KMP kmp(t);
                cout << ((kmp.search(s)) ? "y" : "n") << ende;
            }

            return 0;
        }
    }
```

4.1.37 Query on a tree

```
#include <bits/stdc++.h>

#define MAX 10101
#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))
```

```

using namespace std;

using ll = long long;
using ii = pair<int,int>;

#define left(x) (x << 1)
#define right(x) ((x << 1) + 1)

int N;

template <typename T>
struct SegmentTree {
    using func = function<T(T,T)>;

    func op;
    T ident = T();
    vector<T> tree;

    SegmentTree(func op) :
        op(op), tree(MAX*4) {}

    void build(const vector<T> &v,
               int node = 1, int l = 0, int r = N - 1)
    {
        if (l > r)
            return;

        if (l == r)
            tree[node] = v[l];
        else {
            int m = (l + r) / 2;
            build(v, left(node), l, m);
            build(v, right(node), m + 1, r);
            tree[node] = op(tree[left(node)], tree[right(node)]);
        }
    }

    void update(int i, T val,
               int node = 1, int l = 0, int r = N - 1)
    {
        if (l > r || l > i || r < i)
            return;

        if (l == r)
            tree[node] = val;
        else {
            int m = (l + r) / 2;
            update(i, val, left(node), l, m);
            update(i, val, right(node), m + 1, r);
            tree[node] = op(tree[left(node)], tree[right(node)]);
        }
    }

    T query(int i, int j,
            int node = 1, int l = 0, int r = N - 1)
    {
        if (l > r || l > j || r < i)
            return ident;

        if (l >= i && r <= j)
            return tree[node];

        int m = (l + r) / 2;
        T q1 = query(i, j, left(node), l, m);
        T q2 = query(i, j, right(node), m + 1, r);

```

```

        return op(q1, q2);
    }
};

ii edge[MAX];
vector<ii> graph[MAX];

template <typename ST>
struct HLD {
    ST &seg;
    int cnum, ptr;

    vector<int> dep, par, val;
    vector<int> head, heavy, pos, bot;

    HLD(int n, ST &seg) :
        seg(seg),
        dep(n, 0), par(n), val(n),
        head(n), heavy(n, -1), pos(n), bot(n)
    {
        cnum = ptr = 0;

        N = n;
        dfs(0);
        decompose(0);

        for (int i = 0; i < n - 1; ++i)
            if (dep[edge[i].fi] > dep[edge[i].se])
                bot[i] = edge[i].fi;
            else
                bot[i] = edge[i].se;
    }

    int dfs(int x, int p = -1) {
        int size = 1;
        par[x] = p;

        int max_size = 0;
        for (auto i : graph[x])
            if (i.fi != p) {
                dep[i.fi] = dep[x] + 1;
                val[i.fi] = i.se;
                int isize = dfs(i.fi, x);

                size += isize;
                if (isize > max_size)
                    max_size = isize, heavy[x] = i.fi;
            }

        return size;
    }

    void decompose(int x, int h = 0) {
        head[x] = h;
        seg.update(ptr, val[x]);
        pos[x] = ptr++;

        if (heavy[x] != -1)
            decompose(heavy[x], h);

        for (auto i : graph[x])
            if (i.fi != par[x] && i.fi != heavy[x])
                decompose(i.fi, i.fi);
    }
}

```

```

int query(int a, int b) {
    int ans = seg.ident;
    for (; head[a] != head[b]; b = par[head[b]]) {
        if (dep[head[a]] > dep[head[b]])
            swap(a, b);
        ans = seg.op(ans, seg.query(pos[head[b]], pos[b]));
    }

    if (dep[a] > dep[b])
        swap(a, b);

    return seg.op(ans, seg.query(pos[a] + 1, pos[b]));
}

void update(int i, int val) {
    seg.update(pos[bot[i]], val);
}
};

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int t; cin >> t;
    while (t--) {
        int n; cin >> n;
        for (int i = 0; i < n; ++i)
            graph[i].clear();

```

```

for (int i = 0; i < n - 1; ++i) {
    int a, b, c; cin >> a >> b >> c;
    a--, b--;
    graph[a].pb(ii(b, c));
    graph[b].pb(ii(a, c));
    edge[i] = ii(a, b);
}

SegmentTree<int> seg([])(int a, int b) { return max(a, b); });
HLD<SegmentTree<int>> hld(n, seg);

string op;
while (cin >> op && op != "DONE") {
    if (op == "QUERY") {
        int a, b; cin >> a >> b; a--, b--;
        cout << hld.query(a, b) << ende;
    } else {
        int idx, w; cin >> idx >> w; idx--;
        hld.update(idx, w);
    }
}

return 0;
}

```

4.2 Icpcl La16

4.2.1 A. Assigning Teams

```

#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int, int>;

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    vector<int> v(4);
    for (auto &i : v) cin >> i;
    sort(all(v));

```

```

    cout << abs((v[0] + v[3]) - (v[1] + v[2])) << ende;
    return 0;
}

```

4.2.2 B. Back to the Future

```

#include <bits/stdc++.h>

#define MAX 101010
#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int, int>;

```

```

set<int> graph[MAX];

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n, m, a, b; cin >> n >> m >> a >> b;
    for (int i = 0; i < m; ++i) {
        int x, y; cin >> x >> y;
        graph[x].insert(y);
        graph[y].insert(x);
    }

    set<ii> S;
    for (int i = 1; i <= n; ++i)
        S.insert(ii(graph[i].size(), i));

    bool ended = false;
    while (!ended) {
        ended = true;
        while (S.size() && S.begin()->fi < a) {
            ii u = *(S.begin()); S.erase(S.begin());

            for (auto i : graph[u.se]) {
                S.erase(ii(graph[i].size(), i));
                graph[i].erase(u.se);
                S.insert(ii(graph[i].size(), i));
            }

            graph[u.se].clear();
            ended = false;
        }

        while (S.size() && S.size() - prev(S.end())->fi - 1 < b) {
            ii u = *(prev(S.end())); S.erase(prev(S.end()));

            for (auto i : graph[u.se]) {
                S.erase(ii(graph[i].size(), i));
                graph[i].erase(u.se);
                S.insert(ii(graph[i].size(), i));
            }

            graph[u.se].clear();
            ended = false;
        }
    }

    cout << S.size() << ende;
    return 0;
}

```

4.2.3 D. Dating On-Line

```

#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first

```

```

#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

#define to_rad(x) (x * M_PI) / 180.0

using namespace std;

using ll = long long;
using ii = pair<int,int>;
using dd = pair<double,double>;

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout << setprecision(3) << fixed;

    int n; cin >> n;
    vector<int> v(n);
    for (auto &i : v) cin >> i;
    sort(all(v));

    vector<int> u(n);
    int a = 0, b = n - 1;
    for (int i = 0; i < n; ++i) {
        if (i % 2) u[b--] = v[i];
        else u[a++] = v[i];
    }

    vector<dd> points(n);
    for (int i = 0; i < n; ++i) {
        points[i].fi = u[i] * cos(to_rad((360.0 / n) * (i + 1)));
        points[i].se = u[i] * sin(to_rad((360.0 / n) * (i + 1)));
    }

    double sum = 0.0;
    for (int i = 0; i < n; ++i)
        sum += points[i].fi * points[(i+1) % n].se - points[i].se * points[(i+1) % n].fi;
    cout << sum / 2.0 << ende;
    return 0;
}

```

4.2.4 F. Farm robot

```

#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()

```

```
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int,int>;

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n, c, s; cin >> n >> c >> s; s--;
    int curr = 0, ans = curr == s;
    for (int i = 0; i < c; ++i) {
        int x; cin >> x;
        curr = ((curr + x) + n) % n;
        if (curr == s)
            ans++;
    }

    cout << ans << endl;
    return 0;
}
```

4.2.5 G. Game of Matchings

```
#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int,int>;

struct KMP {
    vector<int> patt;
    vector<int> table;

    KMP(vector<int> patt) :
        patt(patt), table(patt.size()+1)
    { preprocess(); }

    void preprocess() {
        fill(all(table), -1);

        for (int i = 0, j = -1; i < patt.size(); ++i) {
            while (j >= 0 && patt[i] != patt[j] && (patt[j] || patt[i] <= j))
                j = table[j];
            table[i + 1] = ++j;
        }
    }
};

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    string s; cin >> s;
    int n; cin >> n;
    vector<int> v(n);
    for (auto &i : v) {
        cin >> i; --i;
    }

    vector<int> last(30, -1);
    vector<int> A(s.size()), B(n);
    for (int i = 0; i < s.size(); ++i) {
        if (last[s[i] - 'a'] == -1) A[i] = 0;
        else A[i] = i - last[s[i] - 'a'];
        last[s[i] - 'a'] = i;
    }

    fill(all(last), -1);
    for (int i = 0; i < v.size(); ++i) {
        if (last[v[i]] == -1) B[i] = 0;
        else B[i] = i - last[v[i]];
        last[v[i]] = i;
    }

    KMP kmp(B);
    cout << kmp.search(A) << endl;
    return 0;
}
```

```
    }
}

int search(const vector<int> &txt) {
    int ans = 0;
    for (int i = 0, j = 0; i < txt.size(); ++i) {
        while (j >= 0 && txt[i] != patt[j] && (patt[j] || txt[i] <= j))
            j = table[j];
        ++j;
        if (j == patt.size()) {
            ans++;
            j = table[j];
        }
    }

    return ans;
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    string s; cin >> s;
    int n; cin >> n;
    vector<int> v(n);
    for (auto &i : v) {
        cin >> i; --i;
    }

    vector<int> last(30, -1);
    vector<int> A(s.size()), B(n);
    for (int i = 0; i < s.size(); ++i) {
        if (last[s[i] - 'a'] == -1) A[i] = 0;
        else A[i] = i - last[s[i] - 'a'];
        last[s[i] - 'a'] = i;
    }

    fill(all(last), -1);
    for (int i = 0; i < v.size(); ++i) {
        if (last[v[i]] == -1) B[i] = 0;
        else B[i] = i - last[v[i]];
        last[v[i]] = i;
    }

    KMP kmp(B);
    cout << kmp.search(A) << endl;
    return 0;
}
```

4.2.6 H. Hotel Rewards

```
#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
```



```

#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int,int>;

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n, k; cin >> n >> k;

    int ans = 0;
    multiset<int> S;
    for (int i = 0; i < n; ++i) {
        int x; cin >> x;
        ans += x;
        S.insert(x);
        while (S.size() > (i + 1) / (k + 1))
            S.erase(S.begin());
    }

    for (auto i : S) ans -= i;
    cout << ans << ende;
    return 0;
}

```

4.2.7 J. Just in Time

```

#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int,int>;

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n; cin >> n;
    if (n == 2) return cout << 2 << ende, 0;
    for (int i = n; i >= 0; --i) {

```

```

        if (i % 2 == 0) continue;
        bool done = true;
        for (int j = 3; j * j <= n; j += 2)
            if (i % j == 0) {
                done = false;
                break;
            }

        if (done)
            return cout << i << ende, 0;
    }

    return 0;
}

```

4.2.8 K. Kill the Werewolf

```

#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int,int>;

struct Dinic {
    struct Edge { int u, f, c, r; };

    int N;
    vector<int> depth, start;
    vector<vector<Edge>> graph;

    Dinic(int N) :
        N(N), depth(N), start(N), graph(N) {}

    void add_edge(int s, int t, int c) {
        Edge forw = { t, 0, c, (int) graph[t].size() };
        Edge back = { s, 0, 0, (int) graph[s].size() };

        graph[s].pb(forw);
        graph[t].pb(back);
    }

    bool bfs(int s, int t) {
        queue<int> Q;
        Q.push(s);

        fill(all(depth), -1);
        depth[s] = 0;

```

```

while (!Q.empty()) {
    int v = Q.front(); Q.pop();

    for (auto i : graph[v])
        if (depth[i.u] == -1 && i.f < i.c) {
            depth[i.u] = depth[v] + 1;
            Q.push(i.u);
        }

    return depth[t] != -1;
}

int dfs(int s, int t, int f) {
    if (s == t)
        return f;

    for ( ; start[s] < graph[s].size(); ++start[s]) {
        Edge &e = graph[s][start[s]];

        if (depth[e.u] == depth[s] + 1 && e.f < e.c) {
            int min_f = dfs(e.u, t, min(f, e.c - e.f));

            if (min_f > 0) {
                e.f += min_f;
                graph[e.u][e.r].f -= min_f;
                return min_f;
            }
        }
    }

    return 0;
}

int run(int s, int t) {
    int ans = 0;
    while (bfs(s, t)) {
        fill(all(start), 0);

        while (int flow = dfs(s, t, inf))
            ans += flow;
    }

    return ans;
}

```

```

};

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n; cin >> n;
    vector<ii> v(n+1);
    for (int i = 1; i <= n; ++i)
        cin >> v[i].fi >> v[i].se;

    int ans = 0;
    for (int i = 1; i <= n; ++i) {
        int votes = 0, pass = 0;
        int s = 0, t = 2*n + 1;
        Dinic dinic(t + 1);

        for (int j = 1; j <= n; ++j) {
            if (i == j)
                continue;

            if (v[j].fi == i || v[j].se == i) {
                votes++;
                continue;
            }

            dinic.add_edge(s, j, 1);
            dinic.add_edge(j, v[j].fi + n, 1);
            dinic.add_edge(j, v[j].se + n, 1);
            pass++;
        }

        for (int j = 1; j <= n; ++j)
            if (v[i].fi == j || v[i].se == j)
                dinic.add_edge(j + n, t, votes - 2);
            else
                dinic.add_edge(j + n, t, votes - 1);

        if (dinic.run(s, t) < pass)
            ans++;
    }

    cout << ans << endl;
    return 0;
}

```

4.3 IcpC La17

4.3.1 B. Buggy ICPC

```

#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second

```

```

#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int, int>;

```

```

bool is_vowel(char x) {
    return x == 'a' || x == 'e' || x == 'i' || x == 'o' || x == 'u';
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int num_vow = 0;
    string s; cin >> s;
    for (auto i : s) num_vow += is_vowel(i);
    if (num_vow && !is_vowel(s[0]))
        return cout << 0 << ende, 0;
    if (num_vow == 0)
        return cout << 1 << ende, 0;

    int i = 0, n = s.size() - 1;
    bool inverted = false;

    auto invert = [&]() {
        swap(i, n);
        inverted = !inverted;
    };

    auto trim_tail = [&]() {
        if (inverted) n++;
        else n--;
    };

    int ans = 0;
    while (abs(n - i)) {
        if (is_vowel(s[i]) && is_vowel(s[n])) {
            invert();
            trim_tail();
            num_vow--;
        } else if (!is_vowel(s[i]) && is_vowel(s[n])) {
            break;
        } else if (is_vowel(s[i]) && !is_vowel(s[n])) {
            if (num_vow == 1) ans++;
            trim_tail();
        } else if (!is_vowel(s[i]) && !is_vowel(s[n])) {
            trim_tail();
        }
    }

    if (i == n && is_vowel(s[i])) ans++;
    cout << ans << ende;
    return 0;
}

```

4.3.2 C. Complete Naebbirac's sequence

```

#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back

```

```

#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int,int>;

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n, k; cin >> k >> n;
    vector<int> cnt(k + 1);
    for (int i = 0; i < n; ++i) {
        int x; cin >> x; cnt[x]++;
    }

    set<int> S;
    vector<int> v;

    for (int i = 1; i <= k; ++i) S.insert(cnt[i]);
    for (auto i : S) v.pb(i);

    auto find_cnt = [&](int x) {
        for (int i = 1; i <= k; ++i)
            if (cnt[i] == x)
                return i;

        assert(false);
    };

    int flo = n / k;
    int cei = (n - 1) / k + 1;

    if (n % k == 0 && (v.size() == 3 && v[0] == v[1] - 1 && v[1] == v[2] - 1))
        cout << "-" << find_cnt(v[2]) << " " << "+" << find_cnt(v[0]) << ende;
    else if (flo * k + 1 == n && (v.size() == 2 && v[0] == v[1] - 1))
        cout << "-" << find_cnt(v[1]) << ende;
    else if (cei * k - 1 == n && (v.size() == 2 && v[0] == v[1] - 1))
        cout << "+" << find_cnt(v[0]) << ende;
    else
        cout << "*" << ende;

    return 0;
}

```

4.3.3 D. Daunting device

```

#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back

```

```

#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int,int>;
using iii = pair<ii,int>;

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    ll l, c, n; cin >> l >> c >> n;
    set<iii> S;
    S.insert(iii(ii(0, l - 1), 1));
    vector<ll> num(c+1, 0);
    num[1] = l;

    for (int i = 0; i < n; ++i) {
        ll p, x, a, b; cin >> p >> x >> a >> b;
        int m1 = (a + num[p] * num[p]) % l;
        int m2 = (a + (num[p] + b) * (num[p] + b)) % l;

        if (m1 > m2) swap(m1, m2);
        auto curr = prev(S.lower_bound(iii(ii(m1 + 1, -inf), -inf)));

        vector<iii> ins, del;
        ins.pb(iii(ii(m1, m2), x));
        for ( ; curr != S.end() && curr->fi.fi <= m2; ++curr) {
            num[curr->se] -= curr->fi.se - curr->fi.fi + 1;

            // m1 intersects with section
            if (m1 >= curr->fi.fi && m1 <= curr->fi.se) {
                ins.pb(iii(ii(curr->fi.fi, m1 - 1), curr->se));
                num[curr->se] += (m1 - 1) - curr->fi.fi + 1;
            }

            // m2 intersects with section
            if (m2 >= curr->fi.fi && m2 <= curr->fi.se) {
                ins.pb(iii(ii(m2 + 1, curr->fi.se), curr->se));
                num[curr->se] += curr->fi.se - (m2 + 1) + 1;
            }

            num[x] += min(m2, curr->fi.se) - max(m1, curr->fi.fi) + 1;
            del.pb(*curr);
        }

        for (auto j : del) S.erase(j);
        for (auto j : ins) S.insert(j);
    }

    ll ans = 0;
    for (int i = 1; i <= c; ++i)
        ans = max(ans, num[i]);
    cout << ans << ende;
    return 0;
}

```

4.3.4 E. Enigma

```

#include <bits/stdc++.h>

#define MAX 1010
#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int,int>;

int n;
string s;
int dp[MAX][MAX];

bool solve(int i, int r) {
    if (i == s.size())
        return (r % n == 0);

    if (dp[i][r] != -1)
        return dp[i][r];

    if (s[i] == '?') {
        for (int j = (i == 0); j <= 9; ++j)
            if (solve(i + 1, (r * 10 + j) % n)) {
                s[i] = j + '0';
                return dp[i][r] = true;
            }
    } else {
        if (solve(i + 1, (r * 10 + (s[i] - '0')) % n))
            return dp[i][r] = true;
    }

    return dp[i][r] = false;
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    cin >> s >> n;
    mset(dp, -1);

    cout << (!solve(0, 0) ? "*" : s) << ende;
    return 0;
}

```

4.3.5 F. Fundraising

```
#include <bits/stdc++.h>

#define MAX 101010
#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<ll, ll>;
using iii = pair<ii, ll>;

struct BIT {
    int N;
    vector<ll> tree;

    BIT(int N) :
        N(N), tree(N)
    {}

    void init() {
        fill(all(tree), 0LL);
    }

    ll query(int idx) {
        ll sum = 0LL;
        for (; idx > 0; idx -= (idx & -idx))
            sum = max(sum, tree[idx]);
        return sum;
    }

    void update(int idx, ll val) {
        for (; idx < N; idx += (idx & -idx))
            tree[idx] = max(tree[idx], val);
    }
};

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    BIT bit(MAX);
    bit.init();

    map<ii, ll> M;
    int n; cin >> n;
    for (int i = 0; i < n; ++i) {
        ll a, b, c; cin >> a >> b >> c;
        M[ii(a, b)] += c;
    }

    vector<iii> v;
    for (auto i : M)
        v.pb(iii(ii(i.fi.fi, i.fi.se), i.se));

    n = v.size();

    sort(all(v), [](const iii &a, const iii &b) {
        if (a.fi.fi != b.fi.fi)
            return a.fi.fi < b.fi.fi;
        else if (a.fi.se != b.fi.se)
            return a.fi.se > b.fi.se;
        return a.se > b.se;
    });

    set<ll> S;
    map<ll, ll> compress;
    for (auto i : v)
        S.insert(i.fi.se);

    ll k = 1LL;
    for (auto i : S)
        compress[i] = k++;
    for (auto &i : v)
        i.fi.se = compress[i.fi.se];

    ll ans = 0, last = v[0].fi.fi;
    for (int i = 0; ; ) {
        for (; i < n && v[i].fi.fi == last; ++i) {
            ll bst = max(bit.query(v[i].fi.se - 1) + v[i].se,
                bit.query(v[i].fi.se));

            bit.update(v[i].fi.se, bst);
            ans = max(ans, bst);
        }

        if (i >= n) break;
        last = v[i].fi.fi;
    }

    cout << ans << ende;
    return 0;
}
```

```
n = v.size();

sort(all(v), [](const iii &a, const iii &b) {
    if (a.fi.fi != b.fi.fi)
        return a.fi.fi < b.fi.fi;
    else if (a.fi.se != b.fi.se)
        return a.fi.se > b.fi.se;
    return a.se > b.se;
});

set<ll> S;
map<ll, ll> compress;
for (auto i : v)
    S.insert(i.fi.se);

ll k = 1LL;
for (auto i : S)
    compress[i] = k++;
for (auto &i : v)
    i.fi.se = compress[i.fi.se];

ll ans = 0, last = v[0].fi.fi;
for (int i = 0; ; ) {
    for (; i < n && v[i].fi.fi == last; ++i) {
        ll bst = max(bit.query(v[i].fi.se - 1) + v[i].se,
            bit.query(v[i].fi.se));

        bit.update(v[i].fi.se, bst);
        ans = max(ans, bst);
    }

    if (i >= n) break;
    last = v[i].fi.fi;
}

cout << ans << ende;
return 0;
}
```

4.3.6 G. Gates of uncertainty

```
#include <bits/stdc++.h>

#define MAX 101010
#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int, int>;
```

```

map<int,ii> forw;
map<ii,int> backw;
int L[MAX], R[MAX], val[MAX];

struct comb {
    ll pos[4];

    comb() { mset(pos, 0); }

    static comb base() {
        comb res;
        res.pos[0] = res.pos[3] = 1;
        return res;
    }
};

comb dfs(int x) {
    comb res;
    comb l = (L[x] ? (dfs(L[x])) : (comb::base()));
    comb r = (R[x] ? (dfs(R[x])) : (comb::base()));

    for (int i = 0; i < 4; ++i)
        for (int j = 0; j < 4; ++j) {
            ii nand = ii(!(forw[i].fi && forw[j].fi), !(forw[i].se && forw[j].se));
            res.pos[backw[nand]] = (res.pos[backw[nand]] +
                (l.pos[i] % MOD) * (r.pos[j] % MOD)) % MOD;
        }

    if (val[x] == 0) {
        res.pos[0] = (res.pos[0] + res.pos[2]) % MOD, res.pos[2] = 0;
        res.pos[1] = (res.pos[1] + res.pos[3]) % MOD, res.pos[3] = 0;
    } else if (val[x] == 1) {
        res.pos[2] = (res.pos[2] + res.pos[0]) % MOD, res.pos[0] = 0;
        res.pos[3] = (res.pos[3] + res.pos[1]) % MOD, res.pos[1] = 0;
    }

    return res;
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n; cin >> n;
    for (int i = 1; i <= n; ++i)
        cin >> L[i] >> R[i] >> val[i];

    forw[0] = ii(0, 0); backw[ii(0, 0)] = 0;
    forw[1] = ii(0, 1); backw[ii(0, 1)] = 1;
    forw[2] = ii(1, 0); backw[ii(1, 0)] = 2;
    forw[3] = ii(1, 1); backw[ii(1, 1)] = 3;

    comb ans = dfs(1);
    cout << (ans.pos[1] + ans.pos[2]) % MOD << ende;
    return 0;
}

```

4.3.7 H. Hard choice

```
#include <bits/stdc++.h>
```

```

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int,int>;

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int a, b, c; cin >> a >> b >> c;
    int d, e, f; cin >> d >> e >> f;

    cout << max(0, d - a) + max(0, e - b) + max(0, f - c) << ende;
    return 0;
}

```

4.3.8 I. Imperial roads

```

#include <bits/stdc++.h>

#define MAX 201010
#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int,int>;
using iii = pair<ii,int>;

map<ii, bool> mst;
vector<ii> graph[MAX];
vector<iii> edges;

int pare[MAX];
int size[MAX];

```

```

void make_set(int x) {
    pare[x] = x;
    size[x] = 1;
}

int find_set(int x) {
    if (pare[x] != x)
        pare[x] = find_set(pare[x]);
    return pare[x];
}

void union_set(int x, int y) {
    x = find_set(x);
    y = find_set(y);

    if (x == y)
        return;

    if (size[x] > size[y])
        swap(x, y);

    pare[x] = y;
    size[y] += size[x];
}

int kruskal() {
    sort(all(edges), [&](const iii &a, const iii &b) {
        return a.se < b.se;
    });

    int ans = 0;
    for (int i = 0; i < MAX; i++)
        make_set(i);

    for (int i = 0; i < edges.size(); i++) {
        int pu = find_set(edges[i].fi.fi);
        int pv = find_set(edges[i].fi.se);

        if (pu != pv) {
            mst[edges[i].fi] = true;

            graph[edges[i].fi.fi].pb(ii(edges[i].fi.se, edges[i].se));
            graph[edges[i].fi.se].pb(ii(edges[i].fi.fi, edges[i].se));

            ans += edges[i].se;
            union_set(pu, pv);
        }
    }

    return ans;
}

#define MAXLOG 20

int h[MAX];
int par[MAX][MAXLOG];
int cost[MAX][MAXLOG];

void dfs(int v, int p = -1, int c = 0) {
    par[v][0] = p;
    cost[v][0] = c;

    if (p != -1)
        h[v] = h[p] + 1;

```

```

    for (int i = 1; i < MAXLOG; ++i)
        if (par[v][i - 1] != -1) {
            par[v][i] = par[par[v][i - 1]][i - 1];
            cost[v][i] = max(cost[v][i], max(cost[par[v][i - 1]][i - 1], cost[v][i - 1]));
        }

    for (auto u : graph[v])
        if (p != u.fi)
            dfs(u.fi, v, u.se);
}

void preprocess(int v) {
    memset(par, -1, sizeof par);
    memset(cost, 0, sizeof cost);
    dfs(v);
}

int query(int p, int q) {
    int ans = 0;

    if (h[p] < h[q])
        swap(p, q);

    for (int i = MAXLOG - 1; i >= 0; --i)
        if (par[p][i] != -1 && h[par[p][i]] >= h[q]) {
            ans = max(ans, cost[p][i]);
            p = par[p][i];
        }

    if (p == q)
        return ans;

    for (int i = MAXLOG - 1; i >= 0; --i)
        if (par[p][i] != -1 && par[p][i] != par[q][i]) {
            ans = max(ans, max(cost[p][i], cost[q][i]));
            p = par[p][i];
            q = par[q][i];
        }

    if (p == q) return ans;
    else return max(ans, max(cost[p][0], cost[q][0]));
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n, r; cin >> n >> r;
    map<ii,int> M;

    for (int i = 0; i < r; ++i) {
        int a, b, c; cin >> a >> b >> c;
        a--, b--;
        M[ii(a, b)] = c;
        M[ii(b, a)] = c;
        edges.pb(iii(ii(a, b), c));
        edges.pb(iii(ii(b, a), c));
    }

    int vmst = kruskal();
    preprocess(0);

    int q; cin >> q;
    for (int i = 0; i < q; ++i) {
        int a, b; cin >> a >> b;

```

```

    a--, b--;

    if (mst[ii(a, b)] || mst[ii(b, a)])
        cout << vmst << ende;
    else
        cout << vmst - query(a, b) + M[ii(a, b)] << ende;
}

return 0;
}

```

4.3.9 J. Jumping Frog

```

#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int, int>;

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    string s; cin >> s;

```

```

int r = 0, n = s.size();
for (int i = 0; i < n; ++i)
    r += (s[i] == 'R');

if (r == n)
    return cout << n - 1 << ende, 0;

vector<int> divs;
for (int i = 2; i*i <= n; ++i)
    if (n % i == 0) {
        divs.pb(i);
        if (i != n / i)
            divs.pb(n / i);
    }

vector<int> nums(n, 0);
for (int i = 2; i < n; ++i)
    nums[__gcd(i, n)]++;

int ans = 0;
for (auto i : divs) {
    bool poss = false;
    for (int j = 0; j < i; ++j) {
        bool onlyr = true;
        for (int k = j; k < n; k += i)
            onlyr &= (s[k] == 'R');

        if (onlyr) {
            poss = true;
            break;
        }
    }

    if (poss)
        ans += nums[i];
}

cout << ans << ende;
return 0;
}

```

4.4 Icpclal18

4.4.1 A. A Symmetrical Pizza

```

#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()

```

```

#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int, int>;

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    string s; cin >> s;
    s.erase(s.begin() + s.size() - 3);
    stringstream x(s);

    int r; x >> r;
    cout << 36000 / __gcd(36000, r) << ende;
}

```



```
    return 0;
}
```

4.4.2 B. Building a Field

```
#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int, int>;

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n; cin >> n;
    vector<int> v(n), pref(n);
    for (auto &i : v) cin >> i;

    pref[0] = v[0];
    for (int i = 1; i < n; ++i)
        pref[i] = pref[i-1] + v[i];

    if (pref.back() % 2)
        return cout << "N" << ende, 0;

    int beg = 0, ans = 0;
    for (int i = 0; i < n; ++i) {
        while (beg < i && pref[i] - pref[beg] > pref.back() / 2)
            beg++;

        if (pref[i] - pref[beg] == (pref.back() / 2))
            ans++;
    }

    if (ans >= 2)
        cout << "Y" << ende;
    else
        cout << "N" << ende;

    return 0;
}
```

4.4.3 C. Cheap Trips

```
#include <bits/stdc++.h>

#define MAX 10101
#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int, int>;

int n;
int d[MAX], c[MAX];
int dp[MAX][6][123];

int fix(int x) {
    return min(x, 121);
}

int solve(int i, int state, int mi) {
    if (i == n)
        return 0;

    if (dp[i][state][mi] != -1)
        return dp[i][state][mi];

    if (state == 0 || mi >= 120)
        return dp[i][state][mi] = solve(i + 1, 1, fix(d[i])) + c[i];

    if (state == 1)
        return dp[i][state][mi] =
            min(solve(i + 1, state + 1, fix(mi + d[i])) + (c[i] / 2),
                solve(i + 1, 1, fix(d[i])) + c[i]);

    else
        return dp[i][state][mi] =
            min(solve(i + 1, (state + 1) % 6, fix(mi + d[i])) + (c[i] / 4),
                solve(i + 1, 1, fix(d[i])) + c[i]);
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    cin >> n;
    for (int i = 0; i < n; ++i) {
        cin >> d[i] >> c[i];
        c[i] *= 100;
    }

    mset(dp, -1);
    int ans = solve(0, 0, 0);
    cout << (ans / 100) << "." << setfill('0') << setw(2) << (ans % 100) << ende;
    return 0;
}
```

4.4.4 E. Escape, Polygon!

```
#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int,int>;

struct Point {
    double x, y;
    Point() {}
    Point(double x, double y) : x(x), y(y) {}
    Point operator-(Point a) { return Point(x - a.x, y - a.y); }
};

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    ll n; cin >> n;
    vector<Point> v(n);
    for (auto &i : v)
        cin >> i.x >> i.y;

    ll ans = (n * (n - 1LL) * (n - 2LL)) / 6LL;
    for (int i = 0; i < n; ++i) {
        ll l = i + 1, r = i + n - 1;

        for (int j = 0; j < 20; ++j) {
            ll m = (l + r) / 2;

            Point a = v[(i+1)%n] - v[i];
            Point b = v[(m+1)%n] - v[m%n];

            double ang = (atan2(-(a.x*b.y - a.y*b.x), -(a.x*b.x + a.y*b.y)) * 180.0) / M_PI + 180.0;
            if (ang > 180.0)
                r = m;
            else
                l = m;
        }

        ans -= ((l - i) * (l - i - 1LL)) / 2LL;
    }

    cout << ans << ende;
    return 0;
}
```

}

4.4.5 F. Fantastic Beasts

```
#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<ll,ll>;

struct ans_t { ll x, y, d; };

ans_t ext_gcd(ll a, ll b) {
    if (a == 0) return {0, 1, b};
    ans_t e = ext_gcd(b % a, a);
    return {e.y - (b/a) * e.x, e.x, e.d};
}

ll norm(ll a, ll b) {
    a %= b;
    return (a < 0) ? a + b : a;
}

pair<ll,ll> crt_single(ll a, ll n, ll b, ll m) {
    ans_t e = ext_gcd(n, m);

    if ((a - b) % e.d != 0)
        return {-1,-1};

    ll lcm = (m/e.d) * n;
    ll ans = norm(a + e.x*(b-a) / e.d % (m/e.d)*n, lcm);
    return {norm(ans, lcm), lcm};
}

ll crt(vector<ll> a, vector<ll> m) {
    ll ans = a[0];
    ll lcm = m[0];

    int t = a.size();
    for (int i = 1; i < t; ++i) {
        auto ss = crt_single(ans, lcm, a[i], m[i]);
        if (ss.fi == -1)
            return -1;

        ans = ss.fi;
        lcm = ss.se;
    }
}
```

```

    return ans;
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int b, z; cin >> b >> z;
    vector<vector<int>> mat(b, vector<int>(z+1));
    vector<vector<int>> time(z+1, vector<int>(501));
    vector<vector<ll>> a(z + 1, vector<ll>(b, -1));
    vector<vector<ll>> m(z + 1, vector<ll>(b, -1));

    for (auto &i : mat)
        for (auto &j : i) cin >> j;

    for (int i = 0; i < b; ++i) {
        int curr = mat[i][0];
        time[curr][0] |= (1 << i);

        for (int t = 1; t <= 500; ++t) {
            curr = mat[i][curr];
            time[curr][t] |= (1 << i);
        }
    }

    for (int i = 1; i <= z; ++i) {
        for (int t = 0; t <= 500; ++t) {
            if (time[i][t] == (1 << b) - 1)
                return cout << i << " " << t << ende, 0;

            for (int j = 0; j < b; ++j)
                if (time[i][t] & (1 << j)) {
                    if (a[i][j] == -1)
                        a[i][j] = t;
                    else if (m[i][j] == -1)
                        m[i][j] = t - a[i][j];
                }
        }
    }

    ll zoo = 0;
    ll ans = llinf*2;
    for (int i = 1; i <= z; ++i) {
        bool poss = true;
        for (int j = 0; j < b; ++j)
            if (a[i][j] == -1 || m[i][j] == -1) {
                poss = false;
                break;
            }

        if (!poss)
            continue;

        ll res = crt(a[i], m[i]);
        if (res == -1)
            continue;
        else {
            if (ans > res && res > 0)
                zoo = i, ans = res;
        }
    }

    if (ans == llinf*2) cout << "*" << ende;
}

```

```

    else cout << zoo << " " << ans << ende;
    return 0;
}

```

4.4.6 H. Highway Decommission

```

#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<ll, ll>;
using iii = pair<ll, ii>;

ii operator+(const ii &a, const ii &b) {
    return ii(a.fi + b.fi, a.se + b.se);
}

struct Dijkstra {
    int N;
    vector<ll> dist;
    vector<int> cont;
    vector<vector<iii>> graph;

    Dijkstra(int N) :
        N(N), dist(N, llinf), cont(N, 0), graph(N)
    {}

    ll run(int s) {
        set<iii> pq;

        ll ans = 0;
        dist[s] = 0;
        pq.insert(iii(0, ii(s, 0)));

        while (pq.size() != 0) {
            iii top = *(pq.begin());
            pq.erase(pq.begin());

            int u = top.se.fi;
            if (cont[u])
                continue;

            cont[u] = 1;
            ans += top.se.se;

            for (auto i : graph[u]) {
                int v = i.fi;

```

```

        ll wt = i.se.fi;

        if (!cont[v] && dist[v] >= top.fi + wt) {
            dist[v] = top.fi + wt;
            pq.insert(III(dist[v], ii(v, i.se.se)));
        }
    }
}

return ans;
}
};

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n, m; cin >> n >> m;
    Dijkstra d(n+2);
    map<II,II> M;
    for (int i = 0; i < m; ++i) {
        int a, b; ll l, c;
        cin >> a >> b >> l >> c;
        if (M.find(ii(a, b)) != M.end()) {
            M[ii(a, b)] = min(M[ii(a, b)], ii(l, c));
            M[ii(b, a)] = M[ii(a, b)];
        } else {
            M[ii(a, b)] = {l, c};
            M[ii(b, a)] = {l, c};
        }
    }

    for (auto i : M)
        d.graph[i.fi.fi].pb({i.fi.se, i.se});

    ll ans = d.run(1);
    cout << ans << ende;
    return 0;
}

```

4.4.7 I. Ink Colors

```

#include <bits/stdc++.h>

#define MAX 101010
#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;

```

```

using II = pair<int,int>;

int par[MAX], deg[MAX], lev[MAX];
vector<int> graph[MAX];

void dfs(int x, int l) {
    lev[x] = l;
    for (auto i : graph[x])
        dfs(i, l + 1);
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n; cin >> n;
    for (int i = 2; i <= n; ++i) {
        int x; cin >> par[i];
        deg[par[i]]++;
        graph[par[i]].pb(i);
    }

    dfs(1, 0);
    set<II> S;
    for (int i = 2; i <= n; ++i)
        S.insert(ii(-lev[i], i));

    int ans = 0;
    while (S.size() > 0) {
        II curr = *S.begin();
        S.erase(S.begin());

        if (deg[curr.se] >= 2 && deg[par[curr.se]] >= 3 && par[curr.se] != 1 &&
            S.find(ii(-lev[par[curr.se]], par[curr.se])) != S.end()) {
            S.erase(ii(-lev[par[curr.se]], par[curr.se]));
            deg[par[curr.se]]--;
            ans++;
        }
    }

    cout << ans << ende;
    return 0;
}

```

4.4.8 L. Looking for the Risk Factor

```

#include <bits/stdc++.h>

#define MAX 101010
#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

```

```

using namespace std;

using ll = long long;
using ii = pair<int,int>;
using iii = pair<ii,int>;

vector<int> sieve(int n) {
    vector<int> primes;
    vector<bool> is_prime(n+1, true);

    for (int p = 2; p*p <= n; ++p)
        if (is_prime[p])
            for (int i = p*p; i <= n; i += p)
                is_prime[i] = false;

    for (int p = 2; p <= n; ++p)
        if (is_prime[p])
            primes.pb(p);

    return primes;
}

int N;
int v[MAX];
int tree[4 * MAX];

#define left(x) (x << 1)
#define right(x) ((x << 1) + 1)

void build(int node = 1, int a = 0, int b = N - 1) {
    if (a > b)
        return;

    if (a == b) {
        tree[node] = v[a];
        return;
    }

    build(left(node), a, (a + b) / 2);
    build(right(node), 1 + (a + b) / 2, b);
    tree[node] = tree[node * 2] + tree[node * 2 + 1];
}

void update(int idx, int val, int node = 1, int a = 0, int b = N - 1) {
    if (a > b || a > idx || b < idx)
        return;

    if (a == b) {
        tree[node] += val;
        return;
    }

    update(idx, val, left(node), a, (a + b) / 2);
    update(idx, val, right(node), 1 + (a + b) / 2, b);
    tree[node] = tree[node * 2] + tree[node * 2 + 1];
}

int query(int i, int j, int node = 1, int a = 0, int b = N - 1) {
    if (a > b || a > j || b < i)
        return 0;

    if (i <= a && b <= j)
        return tree[node];

```

```

    int q1 = query(i, j, left(node), a, (a + b) / 2);
    int q2 = query(i, j, right(node), 1 + (a + b) / 2, b);
    return q1 + q2;
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    vector<int> primes = sieve(100000);

    int q; cin >> q;
    vector<iii> que(q);
    for (int i = 0; i < q; ++i) {
        int n, k; cin >> n >> k;
        que[i] = {ii(k, n), i};
    }

    N = 100001;
    fill(v, v+N, 1);
    build();

    vector<int> ans(q);
    int curr = primes.size() - 1;
    sort(rall(que));
    for (auto i : que) {
        while (curr >= 0 && primes[curr] > i.fi.fi) {
            for (int j = primes[curr]; j <= N; j += primes[curr])
                if (query(j, j) != 0)
                    update(j, -1);

            curr--;
        }

        ans[i.se] = query(2, i.fi.se);
    }

    for (int i = 0; i < q; ++i)
        cout << ans[i] << " ";
    return 0;
}

```

4.4.9 M. Mount Marathon

```

#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

```

```
using ll = long long;
using ii = pair<int,int>;

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n; cin >> n;
    int past = 0, ans = 0;
    for (int i = 0; i < n; ++i) {
```

```
    int x; cin >> x;
    if (x > past)
        ans++;
    past = x;
}

cout << ans << endl;
return 0;
}
```

4.5 Sbc15

4.5.1 A. Mania de Par

```
#include <bits/stdc++.h>

#define MAX 10101
#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int,int>;

vector<ii> graph[MAX];
vector<ii> gg[MAX];

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int c, v; cin >> c >> v;
    for (int i = 0; i < v; ++i) {
        int a, b, w; cin >> a >> b >> w;
        graph[a].pb({b, w});
        graph[b].pb({a, w});
    }

    for (int i = 1; i <= c; ++i)
        for (auto j : graph[i])
            for (auto k : graph[j.fi])
                gg[i].pb({k.fi, j.se + k.se});

    vector<int> vis(c + 1, 0);
    vector<int> dist(c + 1, inf);

    set<ii> pq;
```

```
pq.insert({0, 1});
dist[1] = 0;
while (!pq.empty()) {
    int u = pq.begin()->se;
    pq.erase(pq.begin());

    if (vis[u]) continue;
    vis[u] = 1;

    for (auto i : gg[u])
        if (!vis[i.fi] && dist[i.fi] > dist[u] + i.se) {
            dist[i.fi] = dist[u] + i.se;
            pq.insert({dist[i.fi], i.fi});
        }
}

if (dist[c] == inf) dist[c] = -1;
cout << dist[c] << endl;
return 0;
}
```

4.5.2 B. Bolsa de Valores

```
#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int,int>;

int main() {
```

```

ios::sync_with_stdio(0);
cin.tie(0);

int n, c; cin >> n >> c;
vector<int> v(n);
vector<vector<int>> dp(n, vector<int>(2));

for (auto &i : v) cin >> i;

dp[n-1] = {0, v[n-1]};
for (int i = n - 2; i >= 0; --i) {
    dp[i][1] = max(dp[i + 1][0] + v[i], dp[i + 1][1]);
    dp[i][0] = max(dp[i + 1][0], dp[i + 1][1] - (v[i] + c));
}

cout << max(0, dp[0][0]) << ende;
return 0;
}

```

4.5.3 C. Tri-du

```

#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int, int>;

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int a, b; cin >> a >> b;
    cout << max(a, b) << ende;

    return 0;
}

```

4.5.4 D. Quebra-cabeca

```

#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f

```

```

#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int, int>;

int c[101], r[101];
string mat[101][101];

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n, m; cin >> n >> m;

    int k = 0;
    map<string, int> M;
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < m; ++j) {
            cin >> mat[i][j];
            M[mat[i][j]] = k++;
        }
        cin >> c[i];
    }

    for (int j = 0; j < m; ++j)
        cin >> r[j];

    map<string, int> ans;
    for (int i = 0; i < M.size(); ++i) {
        string found = "nono";

        for (int j = 0; j < n; ++j) {
            map<string, int> S;
            for (int k = 0; k < m; ++k)
                S[mat[j][k]]++;

            if (S.find("aaa") != S.end())
                S.erase("aaa");

            if (S.size() == 1) {
                found = S.begin()->fi;
                ans[S.begin()->fi] = c[j] / S.begin()->se;
                break;
            }
        }

        if (found == "nono") {
            for (int j = 0; j < m; ++j) {
                map<string, int> S;
                for (int k = 0; k < n; ++k)
                    S[mat[k][j]]++;

                if (S.find("aaa") != S.end())
                    S.erase("aaa");
            }
        }
    }
}

```

```

        if (S.size() == 1) {
            found = S.begin()->fi;
            ans[S.begin()->fi] = r[j] / S.begin()->se;
            break;
        }
    }
}

for (int j = 0; j < n; ++j)
    for (int k = 0; k < m; ++k)
        if (mat[j][k] == found) {
            r[k] -= ans[found];
            c[j] -= ans[found];
            mat[j][k] = "aaa";
        }
}

for (auto i : ans)
    cout << i.fi << " " << i.se << ende;
return 0;
}

```

4.5.5 E. Espiral

```

#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int, int>;

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    ll n, b; cin >> n >> b;
    if (b <= n)
        return cout << 1 << " " << b, 0;

    auto f = [&](ll x) {
        ll sum = ((n - 1LL) * n);
        return sum - (x * (x+1LL));
    };

    ll k = n - 1;
    for (ll bb = n / 2; bb >= 1; bb /= 2)
        while (k - bb >= 0 && n + f(k - bb) <= b)

```

```

        k -= bb;

    ll base = n + f(k);
    if (base + k <= b)
        base += k;

    ll kk = 0;
    for (ll bb = n / 2; bb >= 1; bb /= 2)
        while (base + kk + bb < )

    cout << f(6) << ende;
    cout << k << ende;
    cout << base << ende;
    return 0;
}

```

4.5.6 F. Fatorial

```

#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int, int>;

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n; cin >> n;
    vector<int> fat(10);
    fat[0] = 1;
    for (int i = 1; i < 10; ++i)
        fat[i] = fat[i-1] * i;

    int ans = 0;
    for (int i = 9; i >= 1; --i)
        while (n - fat[i] >= 0) {
            n -= fat[i];
            ans++;
        }

    cout << ans << ende;
    return 0;
}

```


4.5.7 J. Jogo da Estrategia

```
#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int,int>;

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int j, r; cin >> j >> r;
    vector<int> p(j, 0);
    for (int i = 0; i < j*r; ++i) {
        int x; cin >> x;
        p[i%j] += x;
    }

    int ans = 0;
    for (int i = 0; i < j; ++i)
        if (p[ans] <= p[i])
            ans = i;

    cout << ans + 1 << ende;
    return 0;
}
```

4.5.8 K. Palindromo

```
#include <bits/stdc++.h>

#define MAX 2010
#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
```

```
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int,int>;

int n;
string s;
int v[MAX];
ii dp[MAX][MAX];

ii operator+(const ii &a, const ii &b) {
    return {a.fi + b.fi, a.se + b.se};
}

ii solve(int l, int r) {
    if (l > r) return ii(0, 0);
    if (l == r) return ii(v[l], 1);

    if (dp[l][r].fi != -1)
        return dp[l][r];

    ii op = {-inf, -inf};
    if (s[l] == s[r])
        op = solve(l + 1, r - 1) + ii(v[l] + v[r], 2);

    return dp[l][r] = max({
        op,
        solve(l + 1, r - 1),
        solve(l + 1, r),
        solve(l, r - 1)
    });
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    cin >> s >> n;
    for (int i = 0; i < n; ++i) {
        int x; cin >> x;
        v[x - 1] = 1;
    }

    for (int i = 0; i < MAX; ++i)
        for (int j = 0; j < MAX; ++j)
            dp[i][j] = {-1, -1};

    cout << solve(0, s.size() - 1).se << ende;
    return 0;
}
```

4.6 Sbc16

4.6.1 A. Andando no Tempo

```
#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int,int>;

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    vector<int> v(3);
    for (auto &i: v) cin >> i;

    bool poss = false;
    for (int i = 1; i < 8; ++i) {
        for (int j = 0; j < 8; ++j) {
            int sum = 0;
            for (int k = 0; k < 3; ++k) {
                if (i & (1 << k)) {
                    if (j & (1 << k)) sum += v[k];
                    else sum -= v[k];
                }
            }

            if (sum == 0) poss = true;
        }
    }

    if (poss) cout << "S" << ende;
    else cout << "N" << ende;
}
```

```
return 0;
}
```

4.6.2 H. huaauhahhuahau

```
#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int,int>;

bool vowel(char x) {
    return (x == 'a') || (x == 'e') || (x == 'i') || (x == 'o') || (x == 'u');
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    string s; cin >> s;
    string t, w;
    for (auto i : s) if (vowel(i)) t.pb(i);
    w = t;
    reverse(all(t));
    if (w == t) cout << "S" << ende;
    else cout << "N" << ende;

    return 0;
}
```

4.7 Sbc17

4.7.1 A. Acordes Intergalaticos

```
#include <bits/stdc++.h>

#define MAX 101010
#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f
```

```
#define fi first
#define se second
#define sz size()
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))
```

```

using namespace std;

typedef long long ll;
typedef pair<int,int> ii;

typedef struct elem {
    int freq[9];

    elem() {
        for (int i = 0; i < 9; ++i) freq[i] = 0;
    }

    elem(int x) {
        for (int i = 0; i < 9; ++i) freq[i] = 0;
        freq[x] = 1;
    }

    elem operator+(const elem &a) {
        elem e;
        for (int i = 0; i < 9; ++i)
            e.freq[i] = a.freq[i] + freq[i];
        return e;
    }

    void change(int x) {
        vector<int> aux(9, 0);
        for (int i = 0; i < 9; ++i)
            aux[(i + x) % 9] += freq[i];

        for (int i = 0; i < 9; ++i)
            freq[i] = aux[i];
    }
} elem;

int N;
elem v[MAX];
elem tree[4 * MAX];
int lazy[4 * MAX];

#define left(x) (x << 1)
#define right(x) ((x << 1) + 1)

void build(int node = 1, int a = 0, int b = N - 1) {
    if (a > b)
        return;

    if (a == b) {
        tree[node] = v[a];
        return;
    }

    build(left(node), a, (a + b) / 2);
    build(right(node), (a + b) / 2 + 1, b);
    tree[node] = tree[node * 2] + tree[node * 2 + 1];
}

void push(int node, int a, int b, int val) {
    tree[node].change(val);

    if (a != b) {
        lazy[left(node)] += val;
        lazy[right(node)] += val;
    }
}

```

```

}

lazy[node] = 0;
}

void update(int i, int j, int val, int node = 1, int a = 0, int b = N - 1) {
    if (lazy[node] != 0)
        push(node, a, b, lazy[node]);

    if (a > b || a > j || b < i)
        return;

    if (i <= a && b <= j) {
        push(node, a, b, val);
        return;
    }

    update(i, j, val, left(node), a, (a + b) / 2);
    update(i, j, val, right(node), (a + b) / 2 + 1, b);
    tree[node] = tree[node * 2] + tree[node * 2 + 1];
}

elem query(int i, int j, int node = 1, int a = 0, int b = N - 1) {
    if (a > b || a > j || b < i)
        return elem();

    if (lazy[node])
        push(node, a, b, lazy[node]);

    if (a >= i && b <= j)
        return tree[node];

    elem q1 = query(i, j, left(node), a, (a + b) / 2);
    elem q2 = query(i, j, right(node), (a + b) / 2 + 1, b);
    return q1 + q2;
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n, q; cin >> n >> q;
    for (int i = 0; i < n; ++i)
        v[i] = elem(1);

    N = n;
    build();

    for (int i = 0; i < q; ++i) {
        int a, b; cin >> a >> b;
        elem e = query(a, b);

        int grt = 0;
        for (int i = 1; i < 9; ++i)
            if (e.freq[i] >= e.freq[grt])
                grt = i;

        update(a, b, grt);
    }

    for (int i = 0; i < n; ++i) {
        elem e = query(i, i);
    }
}

```

```

    int grt = 0;
    for (int i = 0; i < 9; ++i)
        if (e.freq[i] >= e.freq[grt])
            grt = i;

    cout << grt << endl;
}

return 0;
}

```

4.7.2 B. Brincadeira

```

#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define sz size()
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

typedef long long ll;
typedef pair<int,int> ii;

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    ll n, t, a0, x, y;
    cin >> n >> t >> a0 >> x >> y;

    vector<ll> tor(t);
    for (auto &i : tor) cin >> i;

    auto next = [&]() {
        ll xor = 0;
        for (auto i : tor)
            xor ^= !(a0 & (1 << i));
        return a0 = ((a0 >> 1) | (xor << (n - 1)));
    };

    vector<ll> idx(x + 1, 0);
    idx[0] = 1;

    ll acc = a0 % x;
    for (ll i = 2; ; ++i) {
        if (idx[acc] && i - idx[acc] >= y)
            return cout << idx[acc] - 1 << " " << i - 2 << endl, 0;
    }
}

```

```

    else if (!idx[acc])
        idx[acc] = i;

    acc = (acc + next()) % x;
}

return 0;
}

```

4.7.3 C. Cigarras Periodicas

```

#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define sz size()
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

typedef long long ll;
typedef pair<int,int> ii;

ll lcm(ll a, ll b) {
    return (a * b) / __gcd(a, b);
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    ll n, l; cin >> n >> l;
    ll acc = 1;
    for (int i = 0; i < n; ++i) {
        ll x; cin >> x;
        acc = lcm(acc, x);
    }

    ll ans = -1;
    ll grt = -1;
    for (int i = 1; i <= l; ++i) {
        ll lc = lcm(acc, i);
        if (lc > grt && lc <= l) {
            grt = lc;
            ans = i;
        }
    }

    cout << ans << endl;
    return 0;
}

```

4.7.4 D. Despojados

```
#include <bits/stdc++.h>

#define MAX 0
#define MOD 1000000007
#define EPS 1e-6
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define sz size()
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

typedef long long ll;
typedef pair<int,int> ii;

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    ll n; cin >> n;
    vector<ll> f;
    vector<ll> fat;

    ll acc = 1;
    fat.pb(1);
    for (ll i = 1; i <= 40; ++i) {
        acc *= i;
        fat.pb(acc);
    }

    if (n % 2 == 0) f.pb(2);
    while (n % 2 == 0) n /= 2;

    for (ll i = 3; i*i <= n; i+=2) {
        if (n % i == 0) f.pb(i);
        while (n % i == 0) n /= i;
    }

    if (n > 2) f.pb(n);

    ll ans = 0;
    for (ll i = 2; i <= f.sz; ++i)
        ans += fat[f.sz] / (fat[f.sz - i] * fat[i]);
    cout << ans << ende;

    return 0;
}
```

4.7.5 E. Escala Musical

```
#include <bits/stdc++.h>
```

```
#define MAX 0
#define MOD 1000000007
#define EPS 1e-6
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define sz size()
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

typedef long long ll;
typedef pair<int,int> ii;

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    vector<string> notas = {"do", "do#", "re", "re#", "mi", "fa", "fa#", "sol", "sol#", "la", "la#",
        "", "si"};
    vector<int> count(notas.sz);
    vector<int> diff = {2, 2, 1, 2, 2, 2, 1};
    vector<vector<int>> v(notas.sz, vector<int>(notas.sz, 0));

    for (int i = 0; i < v.sz; ++i) {
        int beg = i;
        for (int j = 0; j < diff.sz; ++j) {
            v[i][beg] = 1;
            beg = (beg + diff[j]) % 12;
        }
    }

    int n; cin >> n;
    for (int i = 0; i < n; ++i) {
        int x; cin >> x; x--;
        count[x % 12]++;
    }

    for (int i = 0; i < 12; ++i) {
        bool poss = true;
        for (int j = 0; j < 12; ++j)
            if (count[j] and !v[i][j])
                poss = false;
        if (poss)
            return cout << notas[i] << ende, 0;
    }
    cout << "desafinado" << ende;

    return 0;
}
```

4.7.6 F. Fase

```
#include <bits/stdc++.h>
```

```

#define MAX 0
#define MOD 1000000007
#define EPS 1e-6
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define sz size()
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

typedef long long ll;
typedef pair<int,int> ii;

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n, k; cin >> n >> k;
    vector<int> v(n);
    for (auto &i : v) cin >> i;
    sort(rall(v));

    int ans = k;
    while (v[ans] == v[ans-1] and ans < n) ans++;
    cout << ans << ende;

    return 0;
}

```

4.7.7 G. Ginastica

```

#include <bits/stdc++.h>

#define MAX 100001
#define MOD 1000000007
#define EPS 1e-6
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define sz size()
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

typedef long long ll;
typedef pair<int,int> ii;

```

```

ll dp[51][MAX];

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int t, m, n; cin >> t >> m >> n;

    for (int i = m; i <= n; ++i)
        dp[1][i] = 1;

    for (int i = 2; i <= t; ++i)
        for (int j = m; j <= n; ++j)
            if (j == m) dp[i][j] = dp[i-1][j+1];
            else if (j == n) dp[i][j] = dp[i-1][j-1];
            else dp[i][j] = (dp[i-1][j-1] + dp[i-1][j+1]) % MOD;

    ll ans = 0;
    for (int i = m; i <= n; ++i)
        ans = (ans + dp[t][i]) % MOD;

    cout << ans << ende;
    return 0;
}

```

4.7.8 H. Hipercampo

```

#include <bits/stdc++.h>

#define MAX 200
#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define sz size()
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

typedef long long ll;
typedef pair<int,int> ii;
typedef pair<double,double> dd;

int dp[MAX][MAX];

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n; cin >> n;
    double xa, xb; cin >> xa >> xb;

    vector<dd> pp;
}

```

```

pp.pb(dd(-1, -1));
for (int i = 0; i < n; ++i) {
    double x, y; cin >> x >> y;
    pp.pb(dd(asin(y / hypot(xa - x, y)), asin(y / hypot(xb - x, y))));
}

sort(all(pp));
for (int i = n + 1; i >= 1; --i)
    for (int j = 0; j <= n; ++j)
        if (j == 0 || (pp[i].fi > pp[j].fi && pp[i].se > pp[j].se))
            dp[i][j] = max(dp[i+1][i] + 1, dp[i+1][j]);
        else
            dp[i][j] = dp[i+1][j];

cout << dp[1][0] << ende;
return 0;
}

```

4.7.9 I. Imposto Real

```

#include <bits/stdc++.h>

#define MAX 10101
#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define sz size()
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

typedef long long ll;
typedef pair<int,int> ii;

int C;
int ans = 0;
int E[MAX];
bool cont[MAX];
vector<ii> graph[MAX];

int dfs(int x) {
    cont[x] = true;

    for (auto i : graph[x]) {
        if (!cont[i.fi]) {
            E[x] += dfs(i.fi);
            ans += 2 * i.se * ((E[i.fi] - 1) / C + 1);
        }
    }

    return E[x];
}

```

```

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

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

    for (int i = 0; i < n - 1; ++i) {
        int a, b, c; cin >> a >> b >> c;
        a--, b--;
        graph[a].pb(ii(b, c));
        graph[b].pb(ii(a, c));
    }

    dfs(0);
    cout << ans << ende;

    return 0;
}

```

4.7.10 J. Jogo de Boca

```

#include <bits/stdc++.h>

#define MAX 0
#define MOD 1000000007
#define EPS 1e-6
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define sz size()
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

typedef long long ll;
typedef pair<int,int> ii;

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    ll num = 0;
    string s; cin >> s;
    for (auto i : s) num += i - '0';
    cout << num % 3 << ende;

    return 0;
}

```

4.7.11 K. K-esimo

```
#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 10000
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define sz size()
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

typedef long long ll;
typedef pair<int,int> ii;

#define K 2

template <typename T>
struct matrix {
    T m[K][K];

    matrix operator*(matrix a) {
        matrix aux;

        for (int i = 0; i < K; i++)
            for (int j = 0; j < K; j++) {
                ll sum = 0;

                for (int k = 0; k < K; k++)
                    sum += (m[i][k] * a[k][j]) % MOD;

                aux[i][j] = sum % MOD;
            }

        return aux;
    }

    T *operator[](int i) {
        return m[i];
    }

    void clear() {
        mset(m, 0);
    }
};

matrix<ll> matrix_pow(matrix<ll> in, ll n) {
    matrix<ll> ans, b = in;

    ans.clear();
    for (int i = 0; i < K; ++i)
        ans[i][i] = 1;

    while (n) {
        if (n & 1)
```

```
        ans = ans * b;

        n >>= 1;
        b = b * b;
    }

    return ans;
}

matrix<ll> solve(ll x, ll y, ll n) {
    matrix<ll> in;

    in[0][0] = x % MOD;
    in[0][1] = y % MOD;
    in[1][0] = 1;
    in[1][1] = 0;

    return matrix_pow(in, n);
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    ll a, b, n, k; cin >> a >> b >> n >> k;

    matrix<ll> ans = solve(2 * a, (b - a * a) + MOD, n - 1);
    ll res = (ans[0][0] * 2 * a + ans[0][1] * 2) % MOD;

    if (a * a > b || (a * a < b && n % 2 == 0))
        res = ((res - 1) + MOD) % MOD;

    for (int i = 0; i < k-1; ++i)
        res /= 10;

    cout << res % 10 << ende;
    return 0;
}
```

4.7.12 L. Laboratorio de Biotecnologia

```
#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

typedef long long ll;
typedef pair<int,int> ii;
```



```

struct comp {
    float r, i;

    comp() : r(0), i(0) {}
    comp(float r, float i) : r(r), i(i) {}

    comp operator+(comp b) {
        return comp(r + b.r, i + b.i);
    }

    comp operator-(comp b) {
        return comp(r - b.r, i - b.i);
    }

    comp operator*(comp b) {
        return comp(r * b.r - i * b.i, r * b.i + i * b.r);
    }

    comp operator/(comp b) {
        float div = (b.r * b.r) + (b.i * b.i);
        return comp((r * b.r + i * b.i) / div, (i * b.r - r * b.i) / div);
    }
};

inline comp conj(comp a) {
    return comp(a.r, -a.i);
}

vector<int> rev = {0, 1};
vector<comp> roots = {{0, 0}, {1, 0}};

void init(int nbase) {
    rev.resize(1 << nbase);
    roots.resize(1 << nbase);

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

    for (int base = 1; base < nbase; ++base) {
        float angle = 2 * M_PI / (1 << (base + 1));

        for (int i = 1 << (base - 1); i < (1 << base); ++i) {
            float angle_i = angle * (2 * i + 1 - (1 << base));

            roots[i << 1] = roots[i];
            roots[(i << 1) + 1] = comp(cos(angle_i), sin(angle_i));
        }
    }
}

void fft(vector<comp> &a) {
    int n = a.size();

    for (int i = 0; i < n; ++i)
        if (i < rev[i])
            swap(a[i], a[rev[i]]);

    for (int s = 1; s < n; s <= 1) {
        for (int k = 0; k < n; k += (s << 1)) {
            for (int j = 0; j < s; ++j) {
                comp z = a[k + j + s] * roots[j + s];

                a[k + j + s] = a[k + j] - z;
                a[k + j] = a[k + j] + z;
            }
        }
    }
}

```

```

    }
}
}

vector<int> multiply(vector<int> &a, vector<int> &b) {
    int nbase, need = a.size() + b.size() + 1;

    for (nbase = 0; (1 << nbase) < need; ++nbase);
    init(nbase);

    int size = 1 << nbase;
    vector<comp> fa(size);

    for (int i = 0; i < size; ++i) {
        int x = (i < a.size() ? a[i] : 0);
        int y = (i < b.size() ? b[i] : 0);
        fa[i] = comp(x, y);
    }

    fft(fa);

    comp r(0, -0.25 / size);
    for (int i = 0; i <= (size >> 1); ++i) {
        int j = (size - i) & (size - 1);
        comp 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);

    vector<int> res(need);
    for (int i = 0; i < need; ++i)
        res[i] = fa[i].r + 0.5;

    return res;
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    string s; cin >> s;
    int n = s.size();

    int acc = 0, sum = s[0] - 'a' + 1;
    for (int i = 1; i < n; ++i)
        sum += s[i] - 'a' + 1;

    vector<int> a(sum + 1), b(sum + 1);
    a[0] = b[sum] = 1;
    for (int i = 0; i < n; ++i) {
        acc += s[i] - 'a' + 1;
        a[acc] = b[sum - acc] = 1;
    }

    vector<int> c = multiply(a, b);
    int ans = 0;
    for (int i = sum + 1; i <= 2 * sum; ++i)
        if (c[i]) ans++;

    cout << ans << endl;
}

```

```
    return 0;
}
```

4.7.13 M. Maquina de Cafe

```
#include <bits/stdc++.h>

#define MAX 0
#define MOD 1000000007
#define EPS 1e-6
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define sz size()
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
```

```
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

typedef long long ll;
typedef pair<int,int> ii;

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int a, b, c; cin >> a >> b >> c;

    int x1 = a*0 + b*2 + c*4;
    int x2 = a*2 + b*0 + c*2;
    int x3 = a*4 + b*2 + c*0;
    cout << min(x1, min(x2, x3)) << ende;

    return 0;
}
```

4.8 Sbc18

4.8.1 B. Bolinhas de Gude

```
#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int,int>;

int foi[500];
int dp[102][102];

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int N = 100;
    for (int i = 0; i < N; ++i) {
        dp[i][i] = 499;
        dp[0][i] = 499;
        dp[i][0] = 499;
    }
```

```
    }

    for (int i = 1; i <= N; ++i) {
        for (int j = 1; j <= N; ++j) {
            if (i != j) {
                mset(foi, 0);
                for (int k = 1; k <= i; ++k)
                    foi[dp[i-k][j]] = 1;
                for (int k = 1; k <= j; ++k)
                    foi[dp[i][j-k]] = 1;
                for (int k = 1; k <= min(i, j); ++k)
                    foi[dp[i-k][j-k]] = 1;

                for (int k = 0; k <= 500; ++k)
                    if (!foi[k]) {
                        dp[i][j] = k;
                        break;
                    }
            }
        }
    }

    int n; cin >> n;
    int ans = 0;
    for (int i = 0; i < n; ++i) {
        int a, b; cin >> a >> b;
        if (a == b)
            return cout << 'Y' << ende, 0;
        ans ^= dp[a][b];
    }

    cout << (ans ? 'Y' : 'N') << ende;
    return 0;
}
```

4.8.2 C. Cortador de Pizza

```
#include <bits/stdc++.h>

#define MAX 101010
#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int, int>;

int tree[MAX];

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

void update(int idx, int val) {
    for (; idx < MAX; idx += (idx & -idx))
        tree[idx] += val;
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int x, y; cin >> x >> y;
    int h, v; cin >> h >> v;
    vector<ii> H(h), V(v);

    for (int i = 0; i < h; ++i) {
        int x1, x2; cin >> x1 >> x2;
        H[i] = ii(x1, x2);
    }

    for (int i = 0; i < v; ++i) {
        int y1, y2; cin >> y1 >> y2;
        V[i] = ii(y1, y2);
    }

    sort(all(H));
    for (int i = 0; i < h; ++i)
        H[i].fi = i + 1;
    sort(all(H), [](ii a, ii b) {
        return a.se < b.se;
    });

    sort(all(V));
```

```
    for (int i = 0; i < v; ++i)
        V[i].fi = i + 1;
    sort(all(V), [](ii a, ii b) {
        return a.se < b.se;
    });

    ll ans = (h + 1LL) * (v + 1LL);
    for (int i = h - 1; i >= 0; --i) {
        ans += query(H[i].fi);
        update(H[i].fi, 1);
    }

    mset(tree, 0);
    for (int i = v - 1; i >= 0; --i) {
        ans += query(V[i].fi);
        update(V[i].fi, 1);
    }

    cout << ans << ende;
    return 0;
}
```

4.8.3 D. Desvendando Monty Hall

```
#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int, int>;

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n; cin >> n;
    int ans = 0;
    for (int i = 0; i < n; ++i) {
        int x; cin >> x;
        ans += (x != 1);
    }

    cout << ans << ende;
    return 0;
}
```

4.8.4 E. Enigma

```
#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int, int>;

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    string s, t; cin >> s >> t;
    int ans = 0;
    for (int i = 0; i <= s.size() - t.size(); ++i) {
        bool poss = true;
        for (int j = 0; j < t.size(); ++j)
            if (s[i+j] == t[j]) {
                poss = false;
                break;
            }

        if (poss)
            ans++;
    }

    cout << ans << ende;
    return 0;
}
```

4.8.5 F. Festival

```
#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
```

```
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int, int>;

struct Show { int i, f, o, s, nxt; };

int N, S;
vector<Show> v;
int dp[1010][3010];

int solve(int i, int mask) {
    if (i == N) {
        if (mask == (1 << S) - 1)
            return 0;
        return -inf;
    }

    if (dp[i][mask] != -1)
        return dp[i][mask];

    int op1 = solve(v[i].nxt, mask | (1 << v[i].s)) + v[i].o;
    int op2 = solve(i + 1, mask);
    return dp[i][mask] = max(op1, op2);
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n; cin >> n;
    S = n;
    for (int i = 0; i < n; ++i) {
        int m; cin >> m;
        for (int j = 0; j < m; ++j) {
            int ii, ff, oo; cin >> ii >> ff >> oo;
            v.pb({ii, ff, oo, i, -1});
        }
    }

    sort(all(v), [](Show a, Show b) {
        return a.i < b.i;
    });

    N = v.size();
    for (int i = 0; i < N; ++i) {
        for (int j = i + 1; j < N; ++j)
            if (v[j].i >= v[i].f) {
                v[i].nxt = j;
                break;
            }

        if (v[i].nxt == -1)
            v[i].nxt = N;
    }

    mset(dp, -1);
    cout << max(-1, solve(0, 0)) << ende;
    return 0;
}
```

4.8.6 G. Gasolina

```
#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int,int>;
using iii = pair<ii,int>;

struct Dinic {
    struct Edge { int u, f, c, r; };

    int N;
    vector<int> depth, start;
    vector<vector<Edge>> graph;

    Dinic(int N) :
        N(N), depth(N), start(N), graph(N) {}

    void add_edge(int u, int v, int c) {
        Edge forw = { v, 0, c, (int) graph[v].size() };
        Edge back = { u, 0, 0, (int) graph[u].size() };
        graph[u].pb(forw);
        graph[v].pb(back);
    }

    bool bfs(int s, int t) {
        queue<int> Q;
        Q.push(s);

        fill(all(depth), -1);
        depth[s] = 0;

        while (!Q.empty()) {
            int v = Q.front(); Q.pop();

            for (auto i : graph[v])
                if (depth[i.u] == -1 && i.f < i.c) {
                    depth[i.u] = depth[v] + 1;
                    Q.push(i.u);
                }
        }

        return depth[t] != -1;
    }

    int dfs(int s, int t, int f) {
        if (s == t) return f;

```

```
        for ( ; start[s] < graph[s].size(); ++start[s]) {
            Edge &e = graph[s][start[s]];

            if (depth[e.u] == depth[s] + 1 && e.f < e.c) {
                int min_f = dfs(e.u, t, min(f, e.c - e.f));

                if (min_f > 0) {
                    e.f += min_f;
                    graph[e.u][e.r].f -= min_f;
                    return min_f;
                }
            }
        }

        return 0;
    }

    int run(int s, int t) {
        int ans = 0;
        while (bfs(s, t)) {
            fill(all(start), 0);
            while (int flow = dfs(s, t, inf))
                ans += flow;
        }
        return ans;
    }
};

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int p, r, c; cin >> p >> r >> c;
    vector<int> vp(p);
    vector<int> vr(r);

    int sum = 0;
    for (auto &i : vp) {
        cin >> i;
        sum += i;
    }

    for (auto &i : vr) cin >> i;

    vector<iii> v(c);
    for (int i = 0; i < c; ++i) {
        int a, b, x; cin >> a >> b >> x;
        v[i] = {{a, b}, x};
    }

    int L = 0, R = 1001000;
    for (int i = 0; i < 30; ++i) {
        int m = (L + R) / 2;

        Dinic dinic(r + p + 5);
        for (int j = 0; j < r; ++j)
            dinic.add_edge(0, j + 1, vr[j]);
        for (int j = 0; j < p; ++j)
            dinic.add_edge(j + 1 + r, r + p + 4, vp[j]);

        for (int j = 0; j < c; ++j)
            if (v[j].se <= m)
                dinic.add_edge(v[j].fi.se, v[j].fi.fi + r, inf);

        if (dinic.run(0, r + p + 4) < sum)

```

```

    L = m;
    else
        R = m;
    }

    cout << ((R > 1000100) ? -1 : R) << ende;
    return 0;
}

```

4.8.7 I. Interruptores

```

#include <bits/stdc++.h>

#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int,int>;

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n, m; cin >> n >> m;
    int l; cin >> l;
    int acc = 0;

    vector<int> lamp(m);
    for (int i = 0; i < l; ++i) {
        int x; cin >> x; x--;
        lamp[x] = 1;
        acc++;
    }

    vector<vector<int>> inte(n);
    for (int i = 0; i < n; ++i) {
        int k; cin >> k;
        for (int j = 0; j < k; ++j) {
            int x; cin >> x; x--;
            inte[i].pb(x);
        }
    }

    if (acc == 0)
        return cout << 0 << ende, 0;

    int ans = 0;
    for (int i = 0; i < 2; ++i) {
        for (int j = 0; j < n; ++j) {

```

```

            ans++;
            for (auto k : inte[j])
                if (lamp[k]) { acc--; lamp[k] = 0; }
                else { acc++; lamp[k] = 1; }

            if (acc == 0)
                return cout << ans << ende, 0;
        }
    }

    cout << -1 << ende;
    return 0;
}

```

4.8.8 J. Juntando Capitais

```

#include <bits/stdc++.h>

#define MAX 105
#define MAXT 11
#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define ende '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int,int>;
using dd = pair<double,double>;

double dist[MAX][MAX];
double dp[MAX][1 << MAXT];

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout << setprecision(5) << fixed;

    int n, k; cin >> n >> k;
    vector<dd> v(n);
    for (int i = 0; i < n; ++i)
        cin >> v[i].fi >> v[i].se;

    for (int i = 0; i < n; ++i)
        for (int j = 0; j < n; ++j)
            dist[i][j] = hypot(v[i].fi - v[j].fi, v[i].se - v[j].se);

    fill(dp[0], dp[0] + (MAX * (1 << MAXT)), 1e9);
    for (int i = 0; i < k; ++i)
        dp[i][1 << i] = 0;

    for (int mask = 1; mask < (1 << k); ++mask) {

```

```

    for (int i = 0; i < n; ++i)
        for (int ss = mask; ss > 0; ss = (ss - 1) & mask)
            dp[i][mask] = min(dp[i][mask], dp[i][ss] + dp[i][mask ^ ss]);

    for (int i = 0; i < n; ++i)
        for (int j = k; j < n; ++j)
            dp[j][mask] = min(dp[j][mask], dp[i][mask] + dist[i][j]);
}

double ans = 1e9;
for (int i = 0; i < n; ++i)
    ans = min(ans, dp[i][(1 << k) - 1]);
cout << ans << endl;
return 0;
}

```

4.8.9 L. Linhas de Metro

```

#include <bits/stdc++.h>

#define MAX 101010
#define MLOG 17
#define EPS 1e-6
#define MOD 1000000007
#define inf 0x3f3f3f3f
#define llinf 0x3f3f3f3f3f3f3f3f

#define fi first
#define se second
#define pb push_back
#define endl '\n'

#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define mset(x, y) memset(&x, (y), sizeof(x))

using namespace std;

using ll = long long;
using ii = pair<int, int>;

int h[MAX];
int par[MAX][MLOG];
vector<int> graph[MAX];

void dfs(int x, int p) {
    par[x][0] = p;

    if (p != -1)
        h[x] = h[p] + 1;

    for (int i = 1; i < MLOG; ++i)
        if (par[x][i-1] != -1)
            par[x][i] = par[par[x][i-1]][i-1];

    for (auto i : graph[x])
        if (i != p)
            dfs(i, x);
}

```

```

int lca(int a, int b) {
    if (h[a] < h[b])
        swap(a, b);

    for (int i = MLOG - 1; i >= 0; --i)
        if (par[a][i] != -1 && h[par[a][i]] >= h[b])
            a = par[a][i];

    if (a == b)
        return a;

    for (int i = MLOG - 1; i >= 0; --i)
        if (par[a][i] != -1 && par[a][i] != par[b][i]) {
            a = par[a][i];
            b = par[b][i];
        }

    return par[a][0];
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n, q; cin >> n >> q;
    for (int i = 0; i < n - 1; ++i) {
        int a, b; cin >> a >> b; a--, b--;
        graph[a].pb(b);
        graph[b].pb(a);
    }

    mset(h, 0);
    mset(par, -1);
    dfs(0, -1);

    for (int i = 0; i < q; ++i) {
        int a, b, c, d;
        cin >> a >> b >> c >> d; a--, b--, c--, d--;
        vector<int> v = { lca(a, c), lca(a, d), lca(b, c), lca(b, d) };

        sort(all(v), [&](int a, int b) {
            return h[a] > h[b];
        });

        if (v[0] == v[1]) {
            bool t1 = ((lca(a, v[0]) == v[0] || lca(b, v[0]) == v[0]) && h[lca(a, b)] <= h[v[0]]);
            bool t2 = ((lca(c, v[0]) == v[0] || lca(d, v[0]) == v[0]) && h[lca(c, d)] <= h[v[0]]);
            cout << (t1 && t2) << endl;
        } else {
            int aux = lca(v[0], v[1]);

            if (aux == v[0] || aux == v[1])
                cout << abs(h[v[0]] - h[v[1]]) + 1 << endl;
            else
                cout << (h[v[0]] - h[aux] + 1) + (h[v[1]] - h[aux] + 1) - 1 << endl;
        }
    }

    return 0;
}

```