Floyd (WA)rshall

Contents

1 Geometry

1.1 polygon isomorfism

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 4007
#define mod 998244353

struct pt
{
  int x, y;
  pt operator+(pt p) { return {x + p.x, y + p.y}; }
  pt operator-(pt p) { return {x - p.x, y - p.y}; }
  bool operator==(pt p) { return (x == p.x && y == p.y); }
  int cross(pt p) { return x * p.y - y * p.x; }
  int cross(pt a, pt b) { return (a - *this).cross(b - *this); }
  int dot(pt p) { return x * p.x + y * p.y; }
};
bool cmp_x(pt a, pt b)
{
  if (a.x != b.x)
    return a.x < b.x;
  return a.y < b.y;
}
vector<pt> convex_hull(vector<pt> pts)
{
  if (pts.size() <= 1)
    return pts;
  sort(pts.begin(), pts.end(), cmp_x);
  vector<pt> h(pts.size() + 1);
  int s = 0, t = 0;
  for (int it = 2; it--; s = --t, reverse(pts.begin(), pts.end()))
  {
    for (auto const &p : pts)
    {
      while (t >= s + 2 && h[t - 2].cross(h[t - 1], p) <= 0)
        t--;
      h[t++] = p;
    }
  }
  return {h.begin(), h.begin() + t - (t == 2 && h[0] == h[1])};
}
int max_suffix(vector<pair<long long, long double>> s, bool mi = false)
{
  s.push_back(*min_element(s.begin(), s.end()));
  s.back().first -= 1;
  s.back().second -= 1;
  int ans = 0;
  for (int i = 1; i < s.size(); i++)
  {
    int j = 0;
    while (ans + j < i and s[i + j] == s[ans + j])
      j++;
    if (s[i + j] > s[ans + j])
    {
      if (!mi or i != s.size() - 2)
        ans = i;
    }
    else if (j)
      i += j - 1;
  }
  return ans;
}
vector<pair<long long, long double>> max_cyclic_shift(vector<pair<long long,
     long double>> s)
{
  int n = s.size();
  for (int i = 0; i < n; i++)
```

```
    s.pb(s[i]);
  int id = max_suffix(s);
  vector<pair<long long, long double>> ans;
  for (int i = 0; i < n; i++)
  {
    ans.pb(s[id]);
    id = (id + 1) % n;
  }
  return ans;
}
int sqr(int x)
{
  return x * x;
}
int dd(pt a, pt b)
{
  return sqr(a.x - b.x) + sqr(a.y - b.y);
}
long long dot(pt a, pt b)
{
  return a.x * b.x + a.y * b.y;
}
vector<pair<long long, long double>> get_sides_and_dots(vector<pt> v)
{
  int n = (int)v.size();
  vector<pair<long long, long double>> ans;
  for (int i = 0; i < n; i++)
  {
    pt prv = v[i ? i - 1 : n - 1];
    pt nxt = v[i + 1 < n ? i + 1 : 0];
    long long dist = dd(v[i], v[(i + 1) % n]);
    long double angle = dot(prv - v[i], nxt - v[i]);
    ans.emplace_back(dist, angle);
  }
  return ans;
}
signed main()
{
  ios::sync_with_stdio(false);
  cin.tie(0);
  int n, m;
  cin >> n >> m;
  vector<pt> a(n);
  for (int i = 0; i < n; i++)
  {
    cin >> a[i].x >> a[i].y;
  }
  auto cha = convex_hull(a);
  auto distsA = get_sides_and_dots(cha);
  vector<pt> b(m);
  for (int i = 0; i < m; i++)
  {
    cin >> b[i].x >> b[i].y;
  }
  auto chb = convex_hull(b);
  auto distsB = get_sides_and_dots(chb);
  vector<pair<long long, long double>> aa = max_cyclic_shift(distsA);
  vector<pair<long long, long double>> bb = max_cyclic_shift(distsB);
  (aa == bb) ? cout << "YES\n" : cout << "NO\n";
  return 0;
}
// https://codeforces.com/problemset/problem/1017/E
// dados dois conjuntos de pontos
// achar o convex hull de cada conjunto
// e em seguida ver se os poligonos sao isomorfos

// podemos checar olhando para o comprimento de cada aresta e o dot product
```

1.2 LineSweep

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 200005
#define mod 1000000007
#define PI acos(-1)

const double EPS = 1e-9;

struct pt
{
  double x, y;
};
struct seg
{
  pt p, q;
  int id;
  double get_y(double x) const
  {
    if (abs(p.x - q.x) < EPS)
      return p.y;
    return p.y + (q.y - p.y) * (x - p.x) / (q.x - p.x);
  }
};
bool intersect1d(double l1, double r1, double l2, double r2)
{
  if (l1 > r1)
    swap(l1, r1);
  if (l2 > r2)
    swap(l2, r2);
  return max(l1, l2) <= min(r1, r2) + EPS;
}
int vec(const pt &a, const pt &b, const pt &c)
{
  double s = (b.x - a.x) * (c.y - a.y) - (b.y - a.y) * (c.x - a.x);
  return abs(s) < EPS ? 0 : s > 0 ? +1
                                    : -1;
}
bool intersect(const seg &a, const seg &b)
{
  return intersect1d(a.p.x, a.q.x, b.p.x, b.q.x) &&
         intersect1d(a.p.y, a.q.y, b.p.y, b.q.y) &&
         vec(a.p, a.q, b.p) * vec(a.p, a.q, b.q) <= 0 &&
         vec(b.p, b.q, a.p) * vec(b.p, b.q, a.q) <= 0;
}
bool operator<(const seg &a, const seg &b)
{
  double x = max(min(a.p.x, a.q.x), min(b.p.x, b.q.x));
  return a.get_y(x) < b.get_y(x) - EPS;
}
struct event
{
  double x;
  int tp, id;
  event() {}
  event(double x, int tp, int id) : x(x), tp(tp), id(id) {}
  bool operator<(const event &e) const
  {
    if (abs(x - e.x) > EPS)
      return x < e.x;
    return tp > e.tp;
  }
};

set<seg> s;

set<seg>::iterator prev(set<seg>::iterator it)
{
  return it == s.begin() ? s.end() : --it;
}
set<seg>::iterator next(set<seg>::iterator it)
{
  return ++it;
}
pi line_sweep(vector<seg> v)
{
  vector<event> e;
  for (int i = 0; i < v.size(); i++)
  {
    e.push_back({min(v[i].p.x, v[i].q.x), 1, i});
    e.push_back({max(v[i].p.x, v[i].q.x), 0, i});
  }
  sort(e.begin(), e.end());
  for (int i = 0; i < e.size(); i++)
  {
    int id = e[i].id;
    if (e[i].tp == 1)
    {
      auto nxt = s.lower_bound(v[id]), prv = prev(nxt);
      if (nxt != s.end() && intersect(*nxt, v[id]))
        return {(*nxt).id, id};
      if (prv != s.end() && intersect(*prv, v[id]))
        return {(*prv).id, id};
      s.insert(nxt, v[id]);
    }
    else
    {
      auto where = s.lower_bound(v[id]);
      auto nxt = next(where), prv = prev(where);
      if (nxt != s.end() && prv != s.end() && intersect(*nxt, *prv))
        return {(*prv).id, (*nxt).id};
      s.erase(where);
    }
  }
  return {-1, -1};
}
signed main()
{
  int n;
  cin >> n;
  vector<seg> v(n);
  for (int i = 0; i < n; i++)
  {
    cin >> v[i].p.x >> v[i].p.y >> v[i].q.x >> v[i].q.y;
    v[i].id = i;
  }
  pi ans = line_sweep(v);
  if (ans.fir == -1)
  {
    cout << "NO\n";
  }
  else
  {
    cout << "YES\n";
    cout << ans.fir + 1 << " " << ans.sec + 1 << endl;
  }
  return 0;
}
// https://cp-algorithms.com/geometry/intersecting_segments.html
// https://acm.timus.ru/problem.aspx?space=1&num=1469
```

1.3 convex hull point location

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 5005
#define mod 998244353

struct pt
{
  int x, y;
  pt operator+(pt p) { return {x + p.x, y + p.y}; }
  pt operator-(pt p) { return {x - p.x, y - p.y}; }
  bool operator==(pt p) { return (x == p.x && y == p.y); }
  int cross(pt p) { return x * p.y - y * p.x; }
  int cross(pt a, pt b) { return (a - *this).cross(b - *this); }
  int dot(pt p) { return x * p.x + y * p.y; }
};
bool cmp_x(pt a, pt b)
{
  if (a.x != b.x)
    return a.x < b.x;
  return a.y < b.y;
}
// acha o convex hull
vector<pt> convex_hull(vector<pt> pts)
{
  if (pts.size() <= 1)
    return pts;
  sort(pts.begin(), pts.end(), cmp_x);
  vector<pt> h(pts.size() + 1);
  int s = 0, t = 0;
  for (int it = 2; it--; s = --t, reverse(pts.begin(), pts.end()))
  {
    for (auto const &p : pts)
    {
      while (t >= s + 2 && h[t - 2].cross(h[t - 1], p) <= 0)
        t--;
      h[t++] = p;
    }
  }
  return {h.begin(), h.begin() + t - (t == 2 && h[0] == h[1])};
}
int sgn(int x)
{
  return (x > 0) - (x < 0);
}
int side_of(pt s, pt e, pt p)
{
  return sgn(s.cross(e, p));
}
bool on_segment(pt s, pt e, pt p)
{
  return p.cross(s, e) == 0 && (s - p).dot(e - p) <= 0;
}
// retorna se o ponto p esta dentro ou nao do convex hull l
// caso strict = true, entao considera true se tiver na borda
// caso strict = false, entao considera false se tiver na borda
bool is_hull(vector<pt> &l, pt p, bool strict = true)
{
  int a = 1, b = l.size() - 1, r = !strict;
  if (l.size() < 3)
    return r && on_segment(l[0], l.back(), p);
  if (side_of(l[0], l[a], l[b]) > 0)
    swap(a, b);
  if (side_of(l[0], l[a], p) >= r || side_of(l[0], l[b], p) <= -r)
    return false;
  while (abs(a - b) > 1)
  {
    int c = (a + b) / 2;
    (side_of(l[0], l[c], p) > 0 ? b : a) = c;
  }
  return sgn(l[a].cross(l[b], p)) < r;
}
signed main()
{
  int n;
  cin >> n;
  vector<pt> v(n);
  for (int i = 0; i < n; i++)
  {
    cin >> v[i].x >> v[i].y;
  }
  vector<pt> ans = convex_hull(v);
  for (int i = 0; i < n; i++)
  {
    if (!is_hull(ans, v[i]))
      cout << i + 1 << " ";
  }
  cout << endl;
}
// h da subregional - https://codeforces.com/gym/104555/problem/G
```

1.4 polygons distance

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

#define int long long int
#define endl ’\n’
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 5005
#define mod 998244353

struct pt
{
  double x, y;
  pt operator+(const pt p) const { return pt(x + p.x, y + p.y); }
  pt operator-(const pt p) const { return pt(x - p.x, y - p.y); }
  pt operator*(const double c) const { return pt(x * c, y * c); }
  pt operator/(const double c) const { return pt(x / c, y / c); }
  double operator*(const pt p) const { return x * p.x + y * p.y; }
  double operator^(const pt p) const { return x * p.y - y * p.x; }
};
struct line
{
  pt p, q;
  line() {}
  line(pt p_, pt q_) : p(p_), q(q_) {}
};
struct building
{
  // 0 - circulo, 1 - quadrado, 2 - triangulo
  int type, r;
  vector<pt> v;
  building()
  {
    v.clear();
    r = 0;
    type = 0;
  }
  void find_vertices()
  {
    pt a, b;
    double vx = v[1].x - v[0].x;
    double vy = v[1].y - v[0].y;
    a.x = (v[0].x + v[1].x) / 2 + (-vy) / 2;
    a.y = (v[0].y + v[1].y) / 2 + (vx) / 2;
    b.x = (v[0].x + v[1].x) / 2 - (-vy) / 2;
    b.y = (v[0].y + v[1].y) / 2 - (vx) / 2;
    v.pb(a);
    swap(v[1], v[2]);
    v.pb(b);
  }
};

int c, q, t, n;
vector<building> v;
vector<int> vc, vq, vt;
```

```
double sarea(pt p, pt q, pt r)
{
  return ((q - p) ^ (r - q)) / 2;
}
double dist(pt p, pt q)
{
  return hypot(p.y - q.y, p.x - q.x);
}
double disttoline(pt p, line r)
{
  return 2 * abs(sarea(p, r.p, r.q)) / dist(r.p, r.q);
}
double disttoseg(pt p, line r)
{
  if ((r.q - r.p) * (p - r.p) < 0)
    return dist(r.p, p);
  if ((r.p - r.q) * (p - r.q) < 0)
    return dist(r.q, p);
  return disttoline(p, r);
}
double dist_circ_seg(pt p1, pt p2, pt p, int r)
{
  double dist = disttoseg(p, line(p2, p1));
  dist -= r;
  return dist;
}
double dist_seg_seg(line a, line b)
{
  double ret = DBL_MAX;
  ret = min(ret, disttoseg(a.p, b));
  ret = min(ret, disttoseg(a.q, b));
  ret = min(ret, disttoseg(b.p, a));
  ret = min(ret, disttoseg(b.q, a));
  return ret;
}
double dist_square_tri(int i, int j)
{
  double ans = DBL_MAX;
  for (int x = 0; x < 4; x++)
  {
    int y = (x + 1) % 4;
    for (int x2 = 0; x2 < 3; x2++)
    {
      int y2 = (x2 + 1) % 3;
      ans = min(ans, dist_seg_seg(line(v[i].v[x], v[i].v[y]), line(v[j].v[x2], v
           [j].v[y2])));
    }
  }
  return ans;
}
double dist_square_circ(int i, int j)
{
  double ans = DBL_MAX;
  for (int x = 0; x < 4; x++)
  {
    int y = (x + 1) % 4;
    ans = min(ans, dist_circ_seg(v[i].v[x], v[i].v[y], v[j].v[0], v[j].r));
  }
  return ans;
}
double dist_tri_circ(int i, int j)
{
  double ans = DBL_MAX;
  for (int x = 0; x < 3; x++)
  {
    int y = (x + 1) % 3;
    ans = min(ans, dist_circ_seg(v[i].v[x], v[i].v[y], v[j].v[0], v[j].r));
  }
  return ans;
}
double dist_circ_circ(int i, int j)
{
  double dist = (v[i].v[0].x - v[j].v[0].x) * (v[i].v[0].x - v[j].v[0].x);
  dist += (v[i].v[0].y - v[j].v[0].y) * (v[i].v[0].y - v[j].v[0].y);
  dist = sqrtl(dist);
  dist -= (v[i].r + v[j].r);
  return (dist < 0) ? 0 : dist;
}
```

```
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> c >> q >> t;
  int n = c + q + t;
  int id = 0;
  for (int i = 0; i < c; i++)
  {
    int x, y, r;
    cin >> x >> y >> r;
    building b;
    b.v.pb({x, y});
    b.r = r;
    b.type = 0;
    v.pb(b);
    vc.pb(id);
    id++;
  }
  for (int i = 0; i < q; i++)
  {
    building b;
    b.type = 1;
    for (int j = 0; j < 2; j++)
    {
      int x, y;
      cin >> x >> y;
      b.v.pb({x, y});
    }
    b.find_vertices();
    v.pb(b);
    vq.pb(id);
    id++;
  }
  for (int i = 0; i < t; i++)
  {
    building b;
    b.type = 2;
    for (int j = 0; j < 3; j++)
    {
      int x, y;
      cin >> x >> y;
      b.v.pb({x, y});
    }
    v.pb(b);
    vt.pb(id);
    id++;
  }
  vector<vector<pair<double, int>>> adj(n + 2);
  double ans = DBL_MAX;
  for (auto const &i : vq)
  {
    for (auto const &j : vt)
    {
      ans = min(ans, dist_square_tri(i, j));
    }
  }
  for (auto const &i : vq)
  {
    for (auto const &j : vc)
    {
      double curr = dist_square_circ(i, j);
      adj[i].pb({curr, j});
      adj[j].pb({curr, i});
    }
  }
  for (auto const &i : vt)
  {
    for (auto const &j : vc)
    {
      double curr = dist_tri_circ(i, j);
      adj[i].pb({curr, j});
      adj[j].pb({curr, i});
    }
  }
  for (auto const &i : vc)
  {
    for (auto const &j : vc)
    {
      double curr = dist_circ_circ(i, j);
      adj[i].pb({curr, j});
      adj[j].pb({curr, i});
    }
  }
  int src = n, sink = n + 1;
  for (auto const &i : vt)
  {
    adj[src].pb({0, i});
  }
  for (auto const &i : vq)
  {
    adj[i].pb({0, sink});
  }
  vector<double> dist(n + 2, 1e18);
  vector<bool> vis(n + 2, 0);
  dist[src] = 0;
  priority_queue<pair<double, int>, vector<pair<double, int>>, greater<pair<
       double, int>>> pq;
  pq.push({dist[src], src});
  while (!pq.empty())
  {
    int x = pq.top().sec;
    pq.pop();
    if (vis[x])
      continue;
    vis[x] = 1;
    for (auto [d, y] : adj[x])
    {
      if (dist[y] > dist[x] + d)
      {
        dist[y] = dist[x] + d;
        pq.push({dist[y], y});
      }
    }
  }
  ans = min(ans, dist[sink]);
  cout << fixed << setprecision(15) << ans << endl;
  return 0;
}
// solution for: https://codeforces.com/gym/104603/problem/I
```

1.5 smallest enclosing circle

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
//#define pi pair<double, double>
#define double long double
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 500001
#define mod 1000000007

struct pt
{
  double x, y;
  pt operator+(pt p) { return {x + p.x, y + p.y}; } // soma de pontos
  pt operator-(pt p) { return {x - p.x, y - p.y}; } // subtracao de pontos
  pt operator*(double d) { return {x * d, y * d}; } // multiplicacao por um
       double
  pt operator/(double d) { return {x / d, y / d}; } // divisao por um double
};
struct circle
```

```
{
  pt c;
  double r;
};

bool inside(circle c, pt p)
{
  double dist = (c.c.x - p.x) * (c.c.x - p.x) + (c.c.y - p.y) * (c.c.y - p.y);
  return dist <= c.r;
}
circle get_circle(pt a, pt b)
{
  pt c = {(a.x + b.x) / 2.0, (a.y + b.y) / 2.0};
  double dist = sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y));
  dist /= 2.0;
  dist *= dist;
  return {c, dist};
}
pt get_center(pt b, pt c)
{
  double bb = b.x * b.x + b.y * b.y;
  double cc = c.x * c.x + c.y * c.y;
  double dd = b.x * c.y - b.y * c.x;
  return {(c.y * bb - b.y * cc) / (2 * dd), (b.x * cc - c.x * bb) / (2 * dd)};
}
circle get_circle(pt a, pt b, pt c)
{
  b = b - a;
  c = c - a;
  pt p = get_center(b, c);
  p = p + a;
  double dist = (a.x - p.x) * (a.x - p.x) + (a.y - p.y) * (a.y - p.y);
  return {p, dist};
}
circle solve2(vector<pt> &v)
{
  if (v.empty())
    return {{0, 0}, 0};
  if (v.size() == 1)
    return {v[0], 0};
  if (v.size() == 2)
    return get_circle(v[0], v[1]);
  for (int i = 0; i < 3; i++)
  {
    for (int j = i + 1; j < 3; j++)
    {
      circle c = get_circle(v[i], v[j]);
      bool ok = 1;
      for (auto const &k : v)
        ok &= inside(c, k);
      if (ok)
        return c;
    }
  }
  return get_circle(v[0], v[1], v[2]);
}
circle solve(vector<pt> &v, vector<pt> r, int n)
{
  if (n == 0 || r.size() == 3)
    return solve2(r);
  int idx = rand() % n;
  pt p = v[idx];
  swap(v[idx], v[n - 1]);
  circle c = solve(v, r, n - 1);
  if (inside(c, p))
    return c;
  r.pb(p);
  return solve(v, r, n - 1);
}
circle welzl(vector<pt> v)
{
  random_shuffle(v.begin(), v.end());
  return solve(v, {}, v.size());
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  srand(time(NULL));
  int n;
  cin >> n;
  vector<pt> v(n);
  for (int i = 0; i < n; i++)
    cin >> v[i].x >> v[i].y;
  circle ans = welzl(v);
  cout << fixed << setprecision(3) << ans.c.x << " " << ans.c.y << endl;
  cout << fixed << setprecision(3) << sqrt(ans.r) << endl;
  return 0;
}
// acmicpc.net/problem/2626
// achar uma circuferencia
// minimizando o raio
// que cobre todos os pontos dela
// ai oq tem q printar eh o centro dessa circuferencia e o raio
// Minimum enclosing circle
// Welzl's algorithm
// complexidade O(n)
```

1.6 kd tree

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 300005
#define mod 998244353
#define inf LLONG_MAX

struct pt
{
  int x, y, id;
  pt() {}
  pt(int xx, int yy) { x = xx, y = yy; }
  pt operator-(pt p) const { return pt(x - p.x, y - p.y); }
  bool operator<(pt p) const { return x < p.x; }
  int dist() const { return x * x + y * y; }
};
bool on_x(const pt &a, const pt &b) { return a.x < b.x; }
bool on_y(const pt &a, const pt &b) { return a.y < b.y; }
struct node
{
  pt pp;
  int id;
  int x0 = inf, x1 = -inf, y0 = inf, y1 = -inf;
  node *first = 0, *second = 0;
  int distance(const pt &p)
  {
    int x = (p.x < x0 ? x0 : p.x > x1 ? x1
                                      : p.x);
    int y = (p.y < y0 ? y0 : p.y > y1 ? y1
                                      : p.y);
    return (pt(x, y) - p).dist();
  }
  node(vector<pt> &&vp) : pp(vp[0])
  {
    for (pt p : vp)
    {
      x0 = min(x0, p.x);
      x1 = max(x1, p.x);
      y0 = min(y0, p.y);
      y1 = max(y1, p.y);
    }
    if (vp.size() > 1)
    {
      sort(vp.begin(), vp.end(), x1 - x0 >= y1 - y0 ? on_x : on_y);
      int half = vp.size() / 2;
      first = new node({vp.begin(), vp.begin() + half});
      second = new node({vp.begin() + half, vp.end()});
    }
  }
};
struct kd_tree
{
  node *root;
  kd_tree(const vector<pt> &vp) : root(new node({vp.begin(), vp.end()})) {}
  pi search(node *n, const pt &p)
  {
    if (!n->first)
    {
      if (n->pp.x == p.x && n->pp.y == p.y)
        return make_pair(inf, n->pp.id); // distancia infinita pra pontos iguais
      return make_pair((p - n->pp).dist(), n->pp.id);
    }
    node *f = n->first, *s = n->second;
    int bfirst = f->distance(p), bsec = s->distance(p);
    if (bfirst > bsec)
      swap(bsec, bfirst), swap(f, s);
    auto best = search(f, p);
    if (bsec < best.first || (!f->first))
      best = min(best, search(s, p));
    return best;
  }
  pi nearest(const pt &p)
  {
    return search(root, p);
  }
};
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n;
  cin >> n;
  vector<pt> v(n);
  for (int i = 0; i < n; i++)
  {
    cin >> v[i].x >> v[i].y;
    v[i].id = i;
  }
  kd_tree t(v);
  pii ans = {inf, {inf, inf}};
  for (int i = 0; i < n; i++)
  {
    pi curr = t.nearest(v[i]);
    ans = min(ans, {curr.fir, {i, curr.sec}});
  }
  cout << fixed << setprecision(6) << ans.sec.fir << " " << ans.sec.sec << " "
       << sqrt(ans.fir) << endl;
  return 0;
}
// closest pair of points com kdtree
// da pra ser adaptado pro 3d tbm
// quando um ponto (x, y) pode aparecer em mais de um indice, tratar antes
// fonte: https://github.com/kth-competitive-programming/kactl/blob/main/kactl.
    pdf

// testei em:
// https://codeforces.com/contest/429/problem/D
// https://www.spoj.com/problems/CLOPPAIR/
// https://vjudge.net/problem/UVA-10245
// https://codeforces.com/gym/104020/problem/L (3D)
```

1.7 ConvexHull

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 500001
#define mod 1000000007
#define PI acos(-1)

namespace p
{
  struct pt
  {
    double x, y;
    pt operator+(pt p) { return {x + p.x, y + p.y}; } // soma de pontos
    pt operator-(pt p) { return {x - p.x, y - p.y}; } // subtracao de pontos
    pt operator*(double d) { return {x * d, y * d}; } // multiplicacao por um
        double
    pt operator/(double d) { return {x / d, y / d}; } // divisao por um double
  };
  double dot(pt v, pt w) // produto escalar (dot product)
  {
    return v.x * w.x + v.y * w.y;
  }
  bool is_perp(pt v, pt w) // retorna se dois vetores sao perpendiculares (
       angulo 90 graus)
  {
    return dot(v, w) == 0;
  }
  double cross(pt v, pt w) // produto vetorial (cross product)
  {
    return v.x * w.y - v.y * w.x;
  }
  double dist(pt a, pt b) // distancia entre 2 pontos
  {
    pt c = a - b;
    return sqrt(c.x * c.x + c.y * c.y);
  }
  double dist2(pt a, pt b) // retorna o quadrado da distancia entre dois pontos
  {
    pt c = a - b;
    return c.x * c.x + c.y * c.y;
  }
  bool is_colinear(pt a, pt b, pt c) // retorna se os pontos a, b e c sao
       colineares
  {
    return cross(b - a, c - a) == 0;
  }
  bool ccw(pt a, pt b, pt c) // retorna se os pontos a,b e c estao no sentido
       anti horario
  {
    return cross(b - a, c - b) > 0;
  }
  bool cw(pt a, pt b, pt c) // retorna se os pontos a,b e c estao no sentido
       horario
  {
    return cross(b - a, c - b) < 0;
  }
  double modulo(pt v) // |v| = sqrt(x2 + y2)
  {
    return sqrt(v.x * v.x + v.y * v.y);
  }
  double angle(pt a, pt b, pt c) // angulo entre os vetores ab e ac
  {
    // dot(ab , ac) / |ab| * |ac|
    pt ab = b - a; // vetor ab
    pt ac = c - a; // vetor ac
    double m1 = modulo(ab);
    double m2 = modulo(ac);
    double m3 = m1 * m2;
    return (dot(ab, ac) / m3); // retorna o cos do angulo em graus
  }
  pt rotate(pt p, double a) // rotacionar o ponto p em relacao a origem, em a graus, no sentido anti-horario
  {
    a = (a * PI) / 180;
    double xx = (cos(a) * p.x) + ((sin(a) * -1) * p.y);
    double yy = (sin(a) * p.x) + (cos(a) * p.y);
    pt ans = {xx, yy};
    return ans;
  }
  double polar(pt p) // polar angle
  {
    return atan2l(p.y, p.x);
  }
  bool cmp(pt a, pt b) // ordenar pontos pelo polar angle
  {
    return polar(a) < polar(b);
  }
  bool cmp_x(pt a, pt b) // ordenar os pontos pela coordenada x
  {
    if (a.x != b.x)
      return a.x < b.x;
    return a.y < b.y;
  }
  vector<pt> convex_hull(vector<pt> v)
  {
    sort(v.begin(), v.end(), cmp_x);
    pt p1 = v[0], p2 = v.back();
    vector<pt> up;
    vector<pt> down;
    up.pb(p1);
    down.pb(p1);
    for (int i = 1; i < v.size(); i++)
    {
      if (i == v.size() - 1 || cw(p1, v[i], p2))
      {
        while (up.size() >= 2 && !cw(up[up.size() - 2], up[up.size() - 1], v[i]))
          up.pop_back();
        up.pb(v[i]);
      }
    }
    for (int i = 1; i < v.size(); i++)
    {
      if (i == v.size() - 1 || ccw(p1, v[i], p2))
      {
        while (down.size() >= 2 && !ccw(down[down.size() - 2], down[down.size() - 1], v[i]))
          down.pop_back();
        down.pb(v[i]);
      }
    }
    int start = 0, limit = 0; // para por em ans no sentido anti-horario e a partir de start
    for (int i = 1; i < down.size(); i++)
      if ((down[i].y < down[start].y) || (down[i].y == down[start].y && down[i].x < down[start].x))
        start = i;
    if (!start)
      limit = 1;
    vector<pt> ans;
    for (int i = start; i < down.size() - 1; i++)
      ans.pb(down[i]);
    for (int i = up.size() - 1; i >= limit; i--)
      ans.pb(up[i]);
    for (int i = 1; i < start; i++)
      ans.pb(down[i]);
    return ans;
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n, t = 0;
  while (cin >> n)
  {
    cout << "caso " << t << ":" << endl;
    vector<p::pt> v(n);
    for (int i = 0; i < n; i++)
      cin >> v[i].x >> v[i].y;
    vector<p::pt> ans = p::convex_hull(v);
    for (auto const &i : ans)
      cout << i.x << " " << i.y << endl;
    cout << endl;
    t++;
  }
  return 0;
}
// conceitos importantes:
// 1- poligono: uma figura plana que possui no minimo 3 lados e 3 angulos
// 2- poligono convexo: um poligono cujo todos os seus angulos internos sao menores do que 180 graus

// convex hull:
// dados n pontos em um plano, o objetivo e achar o menor poligono convexo que possui todos os n pontos dados
// Graham's Scan, complexidade O(n * log(n))

// ideia do algoritimo:
// 1- ache 2 pontos a e b tal que, a e o ponto mais a esquerda e b o ponto mais a direita do conjunto dado
// 2- a e b devem pertencer ao convex hull
// 3- desenhar uma linha ab, essa linha ira separar os outros pontos em 2 conjuntos s1 (superior) e s2 (inferior).
// 4- a e b pertencem aos dois conjuntos
// 5- agora para os conjuntos s1 e s2, achamos o convex hull dos dois conjuntos.
// 6- para isso, ordene todos os pontos pela cordenada x
// 7- para cada ponto, se o ponto dado pertence ao conjunto superior, verificamos o angulo formado pela linha
//    que liga o penultimo ponto e o ultimo ponto do convex hull superior, com a linha que conecta o
//    ultimo ponto do convex hull e o ponto atual. Se o angulo nao for no sentido horario,
//    removemos o ponto mais recente adicionado ao convex hull superior, pois o ponto atual sera capaz
//    de conter o ponto anterior, uma vez que seja adicionado ao convex hull.
// 8- fazer o mesmo para o conjunto inferior
// 9- ao final teremos o conjunto de pontos que formam o convex hull dos n pontos
```

1.8 dynamic ch

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 100005
#define double long double
#define mod 1000000007

const double eps = 1e-9;

struct pt
{
  double x, y;
  pt operator-(pt p) { return {x - p.x, y - p.y}; }
  bool eq(double a, double b) const
  {
    return abs(a - b) <= eps;
  }
  double operator^(const pt p) const { return x * p.y - y * p.x; }
  bool operator<(const pt p) const
  {
    if (!eq(x, p.x))
      return x < p.x;
    if (!eq(y, p.y))
      return y < p.y;
    return 0;
  }
  bool operator==(const pt p) const
  {
    return eq(x, p.x) and eq(y, p.y);
  }
};
double sarea(pt p, pt q, pt r)
{
  return ((q - p) ^ (r - q)) / 2;
}
bool ccw(pt p, pt q, pt r)
{
  return sarea(p, q, r) > eps;
}
// https://github.com/brunomaletta/Biblioteca/blob/master/Codigo/Problemas/dynamicHull.cpp
struct upper
{
  set<pt> se;
  set<pt>::iterator it;
  // 0 - fora
  // 1 - dentro
  // 2 - na borda
  int is_under(pt p)
  {
    it = se.lower_bound(p);
    if (it == se.end())
      return 0;
    if (it == se.begin())
      return p == *it ? 2 : 0;
    if (ccw(p, *it, *prev(it)))
      return 1;
    return ccw(p, *prev(it), *it) ? 0 : 2;
  }
  void insert(pt p)
  {
    if (is_under(p))
      return;
    if (it != se.end())
      while (next(it) != se.end() and !ccw(*next(it), *it, p))
        it = se.erase(it);
    if (it != se.begin())
      while (--it != se.begin() and !ccw(p, *it, *prev(it)))
        it = se.erase(it);
    se.insert(p);
  }
};
struct dyn_hull
{
  upper U, L;
  int is_inside(pt p)
  {
    int u = U.is_under(p), l = L.is_under({-p.x, -p.y});
    if (!u || !l)
      return 0;
    return max(u, l);
  }
  void insert(pt p)
  {
    U.insert(p);
    L.insert({-p.x, -p.y});
  }
  int size()
  {
    int ans = U.se.size() + L.se.size();
    return ans <= 2 ? ans / 2 : ans - 2;
  }
};
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  return 0;
}
// convex hull dinamico
// problema para usar: https://open.kattis.com/problems/hiringhelp
```

1.9 polygon area

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 200005
#define mod 998244353

double area(vector<pi> fig)
{
  double res = 0;
  for (unsigned i = 0; i < fig.size(); i++)
  {
    pi p = i ? fig[i - 1] : fig.back();
    pi q = fig[i];
    res += (p.fir - q.fir) * (p.sec + q.sec);
  }
  return fabs(res) / 2;
}
int cross(pi a, pi b)
{
  return a.fir * b.sec - a.sec * b.fir;
}
double area2(vector<pi> fig)
{
  double res = 0;
  for (unsigned i = 0; i < fig.size(); i++)
  {
    pi p = i ? fig[i - 1] : fig.back();
    pi q = fig[i];
    res += cross(p, q);
  }
  return fabs(res) / 2;
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  return 0;
}
// achar area de um poligono
// tomar cuiddado com a ordem
// percorrer os vertices em sentido horario ou anti-horario
```

1.10 points and vectors

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

```
#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 500001
#define mod 1000000007
#define PI acos(-1)

namespace p
{
  struct pt
  {
    double x, y;
    pt operator+(pt p) { return {x + p.x, y + p.y}; } // soma de pontos
    pt operator-(pt p) { return {x - p.x, y - p.y}; } // subtracao de pontos
    pt operator*(double d) { return {x * d, y * d}; } // multiplicacao por um double
    pt operator/(double d) { return {x / d, y / d}; } // divisao por um double
  };
  double dot(pt v, pt w) // produto escalar (dot product)
  {
    return v.x * w.x + v.y * w.y;
  }
  bool is_perp(pt v, pt w) // retorna se dois vetores sao perpendiculares (angulo 90 graus)
  {
    return dot(v, w) == 0;
  }
  double cross(pt v, pt w) // produto vetorial (cross product)
  {
    return v.x * w.y - v.y * w.x;
  }
  double dist(pt a, pt b) // distancia entre 2 pontos
  {
    pt c = a - b;
    return sqrt(c.x * c.x + c.y * c.y);
  }
  double dist2(pt a, pt b) // retorna o quadrado da distancia entre dois pontos
  {
    pt c = a - b;
    return c.x * c.x + c.y * c.y;
  }
  bool is_colinear(pt a, pt b, pt c) // retorna se os pontos a, b e c sao colineares
  {
    return cross(b - a, c - a) == 0;
  }
  bool ccw(pt a, pt b, pt c) // retorna se os pontos a,b e c estao no sentido anti horario
  {
    return cross(b - a, c - b) > 0;
  }
  bool cw(pt a, pt b, pt c) // retorna se os pontos a,b e c estao no sentido horario
  {
    return cross(b - a, c - b) < 0;
  }
  double modulo(pt v) // |v| = sqrt(x2 + y2)
  {
    return sqrt(v.x * v.x + v.y * v.y);
  }
  double angle(pt a, pt b, pt c) // angulo entre os vetores ab e ac
  {
    // dot(ab , ac) / |ab| * |ac|
    pt ab = b - a; // vetor ab
    pt ac = c - a; // vetor ac
    double m1 = modulo(ab);
    double m2 = modulo(ac);
    double m3 = m1 * m2;
    return (dot(ab, ac) / m3); // retorna o cos do angulo em graus
  }
  pt rotate(pt p, double a) // rotacionar o ponto p em relacao a origem, em a graus, no sentido anti-horario
  {
    a = (a * PI) / 180;
    double xx = (cos(a) * p.x) + ((sin(a) * -1) * p.y);
    double yy = (sin(a) * p.x) + (cos(a) * p.y);
    pt ans = {xx, yy};
    return ans;
  }
  double polar(pt p) // polar angle
  {
    return atan2l(p.y, p.x);
  }
  bool cmp(pt a, pt b) // ordenar pontos pelo polar angle
  {
    return polar(a) < polar(b);
  }
  bool cmp_x(pt a, pt b) // ordenar os pontos pela coordenada x
  {
    if (a.x != b.x)
      return a.x < b.x;
    return a.y < b.y;
  }
  pt polar_to_cartesian(double r, double theta) // r - distancia do centro, theta - polar angle
  {
    pt ans;
    ans.x = r * cos(double(theta) / 180 * PI); // assumindo que theta ta em graus, transforma pra radiano
    ans.y = r * sin(double(theta) / 180 * PI);
    return ans;
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  return 0;
}
```

1.11 minkowski

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 15
#define mod 1000000007

struct pt
{
  int x, y;
  bool operator<(pt ot)
  {
    if (x != ot.x)
      return x < ot.x;
    return y < ot.y;
  }
  void operator=(pt p) { x = p.x, y = p.y; }
  bool operator==(pt p) { return (x == p.x && y == p.y); }
  bool operator!=(pt p) { return (x != p.x || y != p.y); }
  pt operator+(const pt &p) { return {x + p.x, y + p.y}; }
  pt operator-(const pt &p) { return {x - p.x, y - p.y}; }
  pt operator*(int d) { return {x * d, y * d}; }
  pt operator/(int d) { return {x / d, y / d}; }
  int cross(pt ot) const { return x * ot.y - y * ot.x; }
```

```cpp
  int cross(pt a, pt b) const { return (a - *this).cross(b - *this); }
};
enum type
{
  outside,
  inside,
  boundary
};
int cross(pt v, pt w)
{
  return v.x * w.y - v.y * w.x;
}
bool ccw(pt a, pt b, pt c)
{
  return cross(b - a, c - b) > 0;
}
void radial_sort(vector<pt> &a)
{
  pt pivot = *min_element(a.begin(), a.end());
  auto cmp = [&](pt p, pt q)
  {
    if (p == pivot || q == pivot)
      return q != pivot;
    return ccw(pivot, p, q) > 0;
  };
  sort(a.begin(), a.end(), cmp);
}
vector<pt> trata(vector<pt> p)
{
  vector<pt> ans;
  for (int i = 0; i < p.size(); i++)
  {
    while (ans.size() >= 2 && ans.back().cross(p[i], ans.end()[-2]) == 0)
      ans.pop_back();
    ans.pb(p[i]);
  }
  if (ans.size() > 2 && ans.back().cross(p[0], ans.end()[-2]) == 0)
    ans.pop_back();
  return ans;
}
void prepare(vector<pt> &p)
{
  radial_sort(p); // sort points in counter-clockwise order
  p = trata(p);   // and the polygon dont have 3 colinear points
}
int sgn(int val)
{
  if (val > 0)
    return 1;
  else if (val < 0)
    return -1;
  return 0;
}
bool in_seg(pt p, pt a, pt b)
{
  // check if point p is in the line segment formed by a and b
  if (a.cross(b, p) == 0)
    return (p.x >= min(a.x, b.x) && p.x <= max(a.x, b.x) && p.y >= min(a.y, b.y)
         && p.y <= max(a.y, b.y));
  return 0;
}
bool in_tri(pt p, pt a, pt b, pt c)
{
  // check if point p is in the triangle formed by a, b and c
  int a1 = abs(a.cross(b, c));
  int a2 = abs(p.cross(a, b)) + abs(p.cross(a, c)) + abs(p.cross(b, c));
  return a1 == a2;
}
int in_polygon(vector<pt> &poly, pt p)
{
  int n = poly.size();
  if (n == 1)
    return (p == poly[0]) ? type::boundary : type::outside;
  if (n == 2)
    return (in_seg(p, poly[0], poly[1])) ? type::boundary : type::outside;
  if (poly[0].cross(poly[1], p) != 0 && sgn(poly[0].cross(poly[1], p)) != sgn(
      poly[0].cross(poly[1], poly[n - 1])))
    return type::outside;
  if (poly[0].cross(p, poly[n - 1]) != 0 && sgn(poly[0].cross(p, poly[n - 1]))
       != sgn(poly[0].cross(poly[1], poly[n - 1])))
    return type::outside;
  int l = 2, r = n - 1;
  if (poly[0].cross(poly[l], p) > 0)
  {
    while (l < r)
    {
      int mid = (l + r) >> 1;
      (poly[0].cross(poly[mid], p) <= 0) ? r = mid : l = mid + 1;
    }
  }
  if (!in_tri(p, poly[0], poly[l - 1], poly[l]))
    return type::outside;
  if (in_seg(p, poly[l - 1], poly[l]))
    return type::boundary;
  if (in_seg(p, poly[0], poly[1]))
    return type::boundary;
  if (in_seg(p, poly[0], poly[n - 1]))
    return type::boundary;
  return type::inside;
}
vector<pt> minkowski(vector<pt> a, vector<pt> b)
{
  prepare(a);
  prepare(b);
  a.push_back(a[0]);
  a.push_back(a[1]);
  b.push_back(b[0]);
  b.push_back(b[1]);
  vector<pt> ans;
  int i = 0, j = 0;
  while (i < a.size() - 2 || j < b.size() - 2)
  {
    ans.pb(a[i] + b[j]);
    auto c = cross(a[i + 1] - a[i], b[j + 1] - b[j]);
    if (c >= 0)
      i++;
    if (c <= 0)
      j++;
  }
  return ans;
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  vector<pt> v;
  for (int _ = 0; _ < 3; _++)
  {
    int n;
    cin >> n;
    vector<pt> p(n);
    for (int i = 0; i < n; i++)
      cin >> p[i].x >> p[i].y;
    if (_ == 0)
      v = p;
    else
      v = minkowski(v, p);
  }
  prepare(v);
  int q;
  cin >> q;
  while (q--)
  {
    pt p;
    cin >> p.x >> p.y;
    p.x *= 3, p.y *= 3;
    // ve se o ponto (3x, 3y) esta na bora, dentro ou fora do poligono v
    (in_polygon(v, p) != type::outside) ? cout << "YES\n" : cout << "NO\n";
  }
  return 0;
}
// problema exemplo:
// https://codeforces.com/contest/87/problem/E
```

1.12 halfplane intersection

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 100005
#define mod 1000000007

const long double eps = 1e-9;
const long double inf = 1e9;

struct pt
{
  long double x, y;
  pt(long double x = 0, long double y = 0) : x(x), y(y) {}
  friend pt operator+(pt p, pt q)
  {
    return pt(p.x + q.x, p.y + q.y);
  }
  friend pt operator-(pt p, pt q)
  {
    return pt(p.x - q.x, p.y - q.y);
  }
  friend pt operator*(pt p, long double k)
  {
    return pt(p.x * k, p.y * k);
  }
  friend long double dot(pt p, pt q)
  {
    return p.x * q.x + p.y * q.y;
  }
  friend long double cross(pt p, pt q)
  {
    return p.x * q.y - p.y * q.x;
  }
};
struct halfplane
{
  pt p, pq;
  long double angle;
  halfplane() {}
  halfplane(pt a, pt b) : p(a), pq(b - a)
  {
    angle = atan2l(pq.y, pq.x);
  }
  bool out(const pt &r)
  {
    return cross(pq, r - p) < -eps;
  }
  bool operator<(halfplane e) const
  {
    return angle < e.angle;
  }
  friend pt inter(halfplane s, halfplane t)
  {
    long double alpha = cross((t.p - s.p), t.pq) / cross(s.pq, t.pq);
    return s.p + (s.pq * alpha);
  }
};
vector<pt> hp_intersect(vector<halfplane> &h)
{
  pt box[4] = {pt(inf, inf), pt(-inf, inf), pt(-inf, -inf), pt(inf, -inf)}; //
      Bounding box in CCW order
  for (int i = 0; i < 4; i++)
  {
    halfplane aux(box[i], box[(i + 1) % 4]);
    h.pb(aux);
  }
  sort(h.begin(), h.end());
  deque<halfplane> dq;
  int len = 0;
  for (int i = 0; i < h.size(); i++)
  {
    while (len > 1 && h[i].out(inter(dq[len - 1], dq[len - 2])))
    {
      dq.pop_back();
      --len;
    }
    while (len > 1 && h[i].out(inter(dq[0], dq[1])))
    {
      dq.pop_front();
      --len;
    }
    if (len > 0 && fabsl(cross(h[i].pq, dq[len - 1].pq)) < eps)
    {
      if (dot(h[i].pq, dq[len - 1].pq) < 0.0)
      {
        return vector<pt>();
      }
      if (h[i].out(dq[len - 1].p))
      {
        dq.pop_back();
        --len;
      }
      else
      {
        continue;
      }
    }
    dq.push_back(h[i]);
    ++len;
  }
  while (len > 2 && dq[0].out(inter(dq[len - 1], dq[len - 2])))
  {
    dq.pop_back();
    --len;
  }
  while (len > 2 && dq[len - 1].out(inter(dq[0], dq[1])))
  {
    dq.pop_front();
    --len;
  }
  if (len < 3)
  {
    return vector<pt>();
  }
  vector<pt> ret(len);
  for (int i = 0; i + 1 < len; i++)
  {
    ret[i] = inter(dq[i], dq[i + 1]);
  }
  ret.back() = inter(dq[len - 1], dq[0]);
  return ret;
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int q;
  cin >> q; // quantidade de poligonos
  vector<halfplane> h;
  while (q--)
  {
    int n;
    cin >> n;
    vector<pt> v(n);
    for (int i = 0; i < n; i++)
    {
      cin >> v[i].x >> v[i].y;
    }
    for (int i = 0; i < n; i++)
    {
      int j = (i + 1) % n;
      h.pb(halfplane(v[i], v[j]));
    }
  }
  vector<pt> ans = hp_intersect(h);
  if (ans.size() == 0)
  {
    cout << "0.0\n";
    return 0;
  }
  long double res = 0;
  for (int i = 0; i < ans.size(); i++) // area da interseccao
  {
    pt p = (i) ? ans[i - 1] : ans.back();
    pt q = ans[i];
    res += (p.x - q.x) * (p.y + q.y);
  }
  double resp = abs(res) / 2;
  cout << fixed << setprecision(15) << resp << endl;
  return 0;
}
// half-plane intersection

// definicoes:
// half-plane - regiao planar que consiste de todos os pontos que estao de um
    lado de uma reta
// geralmente podem ser descritos da seguninte forma
// conjuntos dos pontos (x, y) que satisfazem algo do tipo:
// ax + by + c <= 0 ou ax + by + c >= 0
// da pra representar as retas e os half-planes atraves de um ponto (que ta na
    reta) e o vetor de direcao
// e dai pros half-planes, considerando que e a regiao da esquerda em relacao ao
     vetor de direcao

// alem disso, considerar uma bounding box sendo um retangulo, pra caso a
    inserseccao dos halfplanes nao seja "fechada"

// https://open.kattis.com/problems/bigbrother
// qual a area que voce pode botar uma camera dentro do poligono
// tal que de um ponto escolhido, e possivel ver todos o poligono
// dai considerar todos os halfplanes de arestas do poligono
// e achar a interseccao de todos esses halfplanes

// https://www.codechef.com/problems/CHN02
// achar a area da interseccao de varios poligonos convexos
// considerar todos os halfplanes de arestas do poligono
// e achar a interseccao de todos esses halfplanes
```

2 Strings

2.1 min suffix

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class string>
using ordered_set = tree<string, null_type, less<string>, rb_tree_tag,
    tree_order_statistics_node_update>;

#define int long long int
#define endl ’\n’
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 2001
#define mod 1000000007

int max_suffix(string s, bool mi = false)
{
  s.push_back(*min_element(s.begin(), s.end()) - 1);
  int ans = 0;
  for (int i = 1; i < s.size(); i++)
  {
    int j = 0;
    while (ans + j < i and s[i + j] == s[ans + j])
      j++;
    if (s[i + j] > s[ans + j])
    {
      if (!mi or i != s.size() - 2)
        ans = i;
    }
    else if (j)
      i += j - 1;
  }
  return ans;
}
int min_suffix(string s)
{
  for (auto &i : s)
    i *= -1;
  s.push_back(*max_element(s.begin(), s.end()) + 1);
  return max_suffix(s, true);
}
int max_cyclic_shift(string s)
{
  int n = s.size();
  for (int i = 0; i < n; i++)
    s.pb(s[i]);
  return max_suffix(s);
}
int min_cyclic_shift(string s)
{
  for (auto &i : s)
    i *= -1;
  return max_cyclic_shift(s);
}
// retorna a posicao de inicio menor/maior sufixo/shift de uma string
```

2.2 manacher

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

#define PI acos(-1)
#define int long long int
#define pb push_back
#define pi pair<int, int>
#define fir first
#define sec second
#define MAXN 100001
#define mod 1000000007

vector<int> d1;
vector<int> d2;

void manacher(string s)
{
  d1.resize(s.size());
  d2.resize(s.size());
  int l = 0, r = -1;
  for (int i = 0; i < s.size(); i++)
  {
    int k = (i > r) ? 1 : min(d1[l + r - i], r - i + 1);
    while (0 <= i - k && i + k < s.size() && s[i - k] == s[i + k])
      k++;
    d1[i] = k;
    k = k - 1;
    if (i + k > r)
      l = i - k, r = i + k;
  }
  l = 0, r = -1;
  for (int i = 0; i < s.size(); i++)
  {
    int k = (i > r) ? 0 : min(d2[l + r - i + 1], r - i + 1);
    while (0 <= i - k - 1 && i + k < s.size() && s[i - k - 1] == s[i + k])
      k++;
    d2[i] = k;
    k = k - 1;
    if (i + k > r)
      l = i - k - 1, r = i + k;
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  string s;
  cin >> s;
  manacher(s);
  return 0;
}

// algoritimo de manacher

// motivacao: dada uma string s, encontre todos os pares (l, r) tal que, a
     substring s[l,r]
// e palindroma.

// para cada posicao (0 <= i < s.size()), vamos encontrar os valores de d1[i] e
     d2[i],
// sendo estes o numero de palindromos com comprimentos impares e com
     comprimentos pares
// e com i sendo a posicao central desses palindromos

// algoritimo mais facil:
// para cada posicao (0 <= i < s.size()), ele tenta aumentar a resposta em 1
// ate q nao seja mais possivel
// while(s[i - curr] == s[i + curr])
// complexidade O(N^2)

// algoritimo de manacher:
// para cada posicao (0 <= i < s.size()):
// seja o par (l, r) os extremos da substring palindroma que possui o maior r
     entre todas as encontradas ate entao
// se i > r, o fim do ultimo palindromo foi antes de i: iremos rodar o
     algoritimo mais facil mais facil e ir ate o limite.
// caso contrario, so precisamos rodar o algoritimo a partir de onde nao foi
     percorrido previamente.
// ao final se o r atual e maior do que o nosso antigo r, atualizamos o par (l,
     r)
// por incrivel que pareca, a complexidade e O(N)

// voltando para a motivacao:
// se temos os valores de d1[i] e d2[i]:
// a substring s[i - k, i + k] e palindroma, para todo (0 <= k < d1[i])
// a substring s[i - k - 1, i + k] e palindroma, para todo (0 <= k < d2[i])
// dai temos todos os intervalos
// note que a complexidade do algoritimo de manacher e O(N),
// mas como a quantidade maxima de palindromos em uma string e n^2,
// imprimir todos os intervalos consequentemente teria complexidade O(N^2) no
     pior caso
```

2.3 de bruijin

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 500005
#define mod 1000000009

int n, m, k, sz;
string ans, ss, path;
vector<int> d;
set<string> st;

void dfs(string s)
{
  if (ans.size() + path.size() == sz) // a sagacidade aqui
  {
    ans += path;
    cout << ans << endl;
    exit(0);
  }
  for (auto const &i : d)
  {
    string t = s;
    t.pb('0' + i);
    if (!st.count(t))
    {
      st.insert(t);
      string nxt = t.substr(1);
      path.pb('0' + i);
      dfs(nxt);
      path.pop_back();
      ans.pb('0' + i);
      if (ans.size() == sz)
      {
        cout << ans << endl;
        exit(0);
      }
    }
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  srand(time(NULL));
  cin >> n >> m >> k;
  d.resize(m);
  for (int i = 0; i < m; i++)
  {
    cin >> d[i];
  }
  sz = n + k - 1;
  if (n >= 40) // n grande -> a probabilidade de colisao eh muito baixa
  {
    string s;
    for (int i = 0; i < sz; i++)
    {
      char c = '0' + d[rand() % m];
      s.pb(c);
    }
    cout << s << endl; // vai uma string gerada no random e gg
    return 0;
  }
  // n pequeno -> vamo achar um caminho euleriano
  for (int i = 1; i < n; i++)
  {
    ss.pb('0' + d[0]);
  }
  dfs(ss);
  ans += ss;
  while (ans.size() > sz)
    ans.pop_back();
  cout << ans << endl;
  return 0;
}
// vou escrever pq achei mto dahora esse problema
// https://codeforces.com/gym/102001/problem/C

// o problema basicamente eh:
// ache uma string s, minimizando o tamanho dessa string
// tal que ela tem k substrings distintas de tamanho n
// ai vai ser tipo:
// achar uma string na qual todas as substrings de tamanho n sao distintas

// alem disso, o alfabeto contem m letras
// essa string vai ter comprimento n + k - 1

// essa string eh chamada de de Bruijn sequence pro caso de k = m^n
// dai o que queremos eh basicamente achar um prefixo de uma de Bruijn sequence,
     pro k < m^n

// dai da pra transformar em um problema de achar um caminho euleriano num grafo
     direcionado
// montar um grafo no qual os vertices sao strings de tamanho n - 1
// e existe uma aresta direcionada u -> v se:
// v pode ser obtida adicionando um char no final de u, e tirando o primeiro
    char de u
```

2.4 stringhashing2

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 2001
#define mod 1000000009

struct modint
{
  int val;
  modint(int v = 0) { val = v % mod; }
  int pow(int y)
  {
    modint x = val;
    modint z = 1;
    while (y)
    {
      if (y & 1)
        z *= x;
      x *= x;
      y >>= 1;
    }
    return z.val;
  }
  int inv() { return pow(mod - 2); }
  void operator=(int o) { val = o % mod; }
  void operator=(modint o) { val = o.val % mod; }
  void operator+=(modint o) { *this = *this + o; }
  void operator-=(modint o) { *this = *this - o; }
  void operator*=(modint o) { *this = *this * o; }
  void operator/=(modint o) { *this = *this / o; }
  bool operator==(modint o) { return val == o.val; }
  bool operator!=(modint o) { return val != o.val; }
  int operator*(modint o) { return ((val * o.val) % mod); }
  int operator/(modint o) { return (val * o.inv()) % mod; }
  int operator+(modint o) { return (val + o.val) % mod; }
  int operator-(modint o) { return (val - o.val + mod) % mod; }
};
struct string_hashing
{
  modint d;
  modint h;
  vector<modint> pref;
  vector<modint> pot;

  string_hashing() {}
  string_hashing(int base, string &s)
  {
    d = base;
    pref.resize(s.size() + 1);
    pref[0] = 0;
    for (int i = 0; i < s.size(); i++)
    {
      modint val = pref[i] * d;
      pref[i + 1] = val + s[i];
    }
    h = pref[s.size()];
    pot.resize(s.size() + 1);
    pot[0] = 1;
    for (int i = 1; i <= s.size(); i++)
      pot[i] = pot[i - 1] * d;
  }
  modint get(int l, int r)
  {
    return pref[r + 1] - (pref[l] * pot[r - l + 1]);
  }
  modint append(modint hb, int blen)
  {
    h = hb + (h * pot[blen]);
    return h;
  }
};
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  string s;
  cin >> s;
  string_hashing h(256, s); // (base, string)
  // string_hashing h(227, s); // (base, string)
  return 0;
}
```

2.5 aho corasick

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define DEBUG 0
#define MAXN 5001
#define mod 1000000007

namespace aho
{
  int go(int v, char ch);
  const int K = 26; // tamanho do alfabeto
  struct trie
  {
    char me;            // char correspondente ao no atual
    int go[K];          // proximo vertice que eu devo ir estando em um estado (
         v, c)
    int down[K];        // proximo vertice da trie
    int is_leaf = 0;    // se o vertice atual da trie eh uma folha (fim de uma
         ou mais strings)
    int parent = -1;    // no ancestral do no atual
  int link = -1;      // link de sufixo do no atual (outro no com o maior
       matching de sufixo)
  int exit_link = -1; // folha mais proxima que pode ser alcancada a partir de
        v usando links de sufixo
  trie(int p = -1, char ch = '$') : parent(p), me(ch)
  {
    fill(begin(go), end(go), -1);
    fill(begin(down), end(down), -1);
  }
};
vector<trie> ac;
void init() // criar a raiz da trie
{
  ac.resize(1);
}
void add_string(string s) // adicionar string na trie
{
  int v = 0;
  for (auto const &ch : s)
  {
    int c = ch - 'a';
    if (ac[v].down[c] == -1)
    {
      ac[v].down[c] = ac.size();
      ac.emplace_back(v, ch);
    }
    v = ac[v].down[c];
  }
  ac[v].is_leaf++;
}
int get_link(int v) // pegar o suffix link saindo de v
{
  if (ac[v].link == -1)
    ac[v].link = (!v || !ac[v].parent) ? 0 : go(get_link(ac[v].parent), ac[v].
         me);
  return ac[v].link;
}
int go(int v, char ch) // proximo estado saindo do estado(v, ch)
{
  int c = ch - 'a';
  if (ac[v].go[c] == -1)
  {
    if (ac[v].down[c] != -1)
      ac[v].go[c] = ac[v].down[c];
    else
      ac[v].go[c] = (!v) ? 0 : go(get_link(v), ch);
  }
  return ac[v].go[c];
}
int get_exit_link(int v) // suffix link mais proximo de v que seja uma folha
{
  if (ac[v].exit_link == -1)
  {
    int curr = get_link(v);
    if (!v || !curr)
      ac[v].exit_link = 0;
    else if (ac[curr].is_leaf)
      ac[v].exit_link = curr;
    else
      ac[v].exit_link = get_exit_link(curr);
  }
  return ac[v].exit_link;
}
int query(string s) // query O(n + ans)
{
  int ans = 0, curr = 0, at;
  for (auto const &i : s)
  {
    curr = go(curr, i);
    ans += ac[curr].is_leaf;
    at = get_exit_link(curr);
    while (at)
    {
      ans += ac[at].is_leaf;
      at = get_exit_link(at);
    }
  }
  return ans;
```

```
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n, q;
  cin >> n >> q;
  aho::init();
  for (int i = 0; i < n; i++)
  {
    string s;
    cin >> s;
    aho::add_string(s);
  }
  while (q--)
  {
    string t;
    cin >> t;
    cout << aho::query(t) << endl;
  }
  return 0;
}
// automato de aho-corasick
// imagine o seguinte problema:
// temos um conjunto de n strings
// e q queries para processar
// em cada uma das q queries, voce recebe uma string s
// e quer saber, o numero de ocorrencias de
// alguma string do conjunto como
// substring de s e em tempo linear
```

2.6 z-function

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 2001
#define mod 1000000007

vector<int> z_function(string &s)
{
  int n = s.size();
  vector<int> z(n);
  z[0] = n;
  for (int i = 1, l = 0, r = 0; 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;
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  string s;
  cin >> s;
  vector<int> z = z_function(s);
}
// z-function
// calcula para cada i:
// z[i] = o tamanho de lcp(s, s.substr(i, n - i))
// lcp -> longest comom prefix
```

2.7 substring fft

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define PI acos(-1)
#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define MAXN 100005
#define mod 1000000007
#define cd complex<double>

const double eps = 1e-12;
const int alphabet_size = 26;

namespace fft
{
  void dft(vector<cd> &a)
  {
    int n = a.size();
    if (n == 1)
      return;
    vector<cd> a0(n / 2), a1(n / 2);
    for (int i = 0; 2 * i < n; i++)
    {
      a0[i] = a[2 * i];
      a1[i] = a[2 * i + 1];
    }
    dft(a0);
    dft(a1);
    double ang = 2 * PI / n;
    cd w(1), wn(cos(ang), sin(ang));
    for (int i = 0; 2 * i < n; i++)
    {
      a[i] = a0[i] + w * a1[i];
      a[i + n / 2] = a0[i] - w * a1[i];
      w *= wn;
    }
  }
  void inverse_dft(vector<cd> &a)
  {
    int n = a.size();
    if (n == 1)
      return;
    vector<cd> a0(n / 2), a1(n / 2);
    for (int i = 0; 2 * i < n; i++)
    {
      a0[i] = a[2 * i];
      a1[i] = a[2 * i + 1];
    }
    inverse_dft(a0);
    inverse_dft(a1);
    double ang = 2 * PI / n * -1;
    cd w(1), wn(cos(ang), sin(ang));
    for (int i = 0; 2 * i < n; i++)
    {
      a[i] = a0[i] + w * a1[i];
      a[i + n / 2] = a0[i] - w * a1[i];
      a[i] /= 2;
      a[i + n / 2] /= 2;
      w *= wn;
    }
  }
  vector<double> mul(vector<cd> a, vector<cd> b)
  {
    int n = 1;
    vector<cd> fa(a.begin(), a.end()), fb(b.begin(), b.end());
    while (n < a.size() + b.size())
      n <<= 1;
    fa.resize(n);
    fb.resize(n);
    dft(fa);
    dft(fb);
    for (int i = 0; i < n; i++)
      fa[i] *= fb[i];
    inverse_dft(fa);
    vector<double> ans(n);
    for (int i = 0; i < n; i++)
      ans[i] = fa[i].real();
    return ans;
  }
} // namespace fft
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  string s, t;
  cin >> s >> t;
  int n = s.size(), m = t.size();
  reverse(t.begin(), t.end());
  vector<cd> a(n);
  vector<cd> b(m);
  for (int i = 0; i < n; i++)
  {
    int ch = s[i] - ’a’;
    double ang = (2 * PI * ch) / alphabet_size;
    a[i] = cd(cos(ang), sin(ang));
  }
  for (int i = 0; i < m; i++)
  {
    int ch = t[i] - ’a’;
    double ang = (2 * PI * ch) / alphabet_size;
    b[i] = cd(cos(ang), -sin(ang));
  }
  vector<double> ans = fft::mul(a, b);
  int matches = 0;
  for (int i = m - 1; i < n; i++)
    matches += (abs(ans[i] - m) <= eps);
  cout << matches << endl;
  return 0;
}
// number of matches of a pattern in string
// using fft
```

2.8 suffix array

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl ’\n’
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define pci pair<char, int>
```

```
#define fir first
#define sec second
#define MAXN 500001
#define mod 1000000007

struct suffix_array
{
  int n, k;
  string s;
  vector<int> p, c, lcp;
  vector<pci> a;

  void radix(vector<pii> &v)
  {
    {
      int n = v.size();
      vector<int> cnt(n);
      for (auto const &i : v)
        cnt[i.fir.sec]++;
      vector<pii> ans(n);
      vector<int> pos(n);
      pos[0] = 0;
      for (int i = 1; i < n; i++)
        pos[i] = pos[i - 1] + cnt[i - 1];
      for (auto const &i : v)
      {
        int k = i.fir.sec;
        ans[pos[k]] = i;
        pos[k]++;
      }
      v = ans;
    }
    {
      int n = v.size();
      vector<int> cnt(n);
      for (auto const &i : v)
        cnt[i.fir.fir]++;
      vector<pii> ans(n);
      vector<int> pos(n);
      pos[0] = 0;
      for (int i = 1; i < n; i++)
        pos[i] = pos[i - 1] + cnt[i - 1];
      for (auto const &i : v)
      {
        int k = i.fir.fir;
        ans[pos[k]] = i;
        pos[k]++;
      }
      v = ans;
    }
  }
  suffix_array(string &st)
  {
    s = st;
    s += '$'; // menor do que todos os chars da string st
    n = s.size();
    p.resize(n);
    c.resize(n);
    a.resize(n);
    for (int i = 0; i < n; i++)
      a[i] = {s[i], i};
    sort(a.begin(), a.end());
    for (int i = 0; i < n; i++)
      p[i] = a[i].sec;
    c[p[0]] = 0;
    for (int i = 1; i < n; i++)
      (a[i].fir == a[i - 1].fir) ? c[p[i]] = c[p[i - 1]] : c[p[i]] = c[p[i - 1]]
           + 1;
    k = 0;
    while ((1 << k) < n)
    {
      vector<pii> v(n);
      for (int i = 0; i < n; i++)
        v[i] = {{c[i], c[(i + (1 << k)) % n]}, i};
      radix(v); // pode usar std::sort()
      for (int i = 0; i < n; i++)
        p[i] = v[i].sec;
      c[p[0]] = 0;
      for (int i = 1; i < n; i++)
        (v[i].fir == v[i - 1].fir) ? c[p[i]] = c[p[i - 1]] : c[p[i]] = c[p[i -
             1]] + 1;
      k++;
    }
  }
  void build_lcp()
  {
    lcp.resize(n);
    k = 0;
    for (int i = 0; i < n - 1; i++)
    {
      int idx = c[i], j = p[idx - 1];
      while (s[i + k] == s[j + k])
        k++;
      lcp[idx] = k;
      k = max(k - 1, 0ll);
    }
  }
};
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  string s;
  cin >> s;
  int n = s.size();
  suffix_array sa(s);
  for (int i = 0; i <= s.size(); i++) // sufix array
    cout << sa.p[i] << " ";
  cout << endl;
  sa.build_lcp();
  for (int i = 1; i <= s.size(); i++) // lcp entre 2 suffixos adjacentes no
       suffix array
    cout << sa.lcp[i] << " ";
  cout << endl;
  // queries de dada uma string t, diga quantas occorrencias tem de t como
       substring em s
  int q;
  cin >> q;
  while (q--)
  {
    string t;
    cin >> t;
    int i = 0, f = n, m, lb, ub;
    while (i < f)
    {
      m = (i + f) / 2;
      (t <= s.substr(sa.p[m], t.size())) ? f = m : i = m + 1;
    }
    ub = i, i = 0, f = n;
    while (i < f)
    {
      m = (i + f) / 2;
      (t >= s.substr(sa.p[m], t.size())) ? i = m + 1 : f = m;
    }
    lb = i;
    if (s.substr(sa.p[lb], t.size()) == t)
      lb++;
    cout << lb - ub << endl;
  }
  return 0;
}
```

2.9 stringhashing

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

```
#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 1000001

// https://github.com/brunomaletta/Biblioteca/blob/master/Codigo/Strings/
     hashingLargeMod.cpp
const int MOD = (1ll << 61) - 1;
int P;
int mulmod(int a, int b)
{
  const static int LOWER = (1ll << 30) - 1, GET31 = (1ll << 31) - 1;
  int l1 = a & LOWER, h1 = a >> 30, l2 = b & LOWER, h2 = b >> 30;
  int m = l1 * h2 + l2 * h1, h = h1 * h2;
  int ans = l1 * l2 + (h >> 1) + ((h & 1) << 60) + (m >> 31) + ((m & GET31) <<
       30) + 1;
  ans = (ans & MOD) + (ans >> 61), ans = (ans & MOD) + (ans >> 61);
  return ans - 1;
}
mt19937_64 rng(chrono::steady_clock::now().time_since_epoch().count());
int uniform(int l, int r)
{
  uniform_int_distribution<int> uid(l, r);
  return uid(rng);
}
struct string_hashing
{
  vector<int> h, p;
  string_hashing() {}
  string_hashing(string s) : h(s.size()), p(s.size())
  {
    p[0] = 1, h[0] = s[0];
    for (int i = 1; i < s.size(); i++)
      p[i] = mulmod(p[i - 1], P), h[i] = (mulmod(h[i - 1], P) + s[i]) % MOD;
  }
  int get(int l, int r)
  {
    int hash = h[r] - (l ? mulmod(h[l - 1], p[r - l + 1]) : 0);
    return hash < 0 ? hash + MOD : hash;
  }
  int append(int h, int hb, int blen)
  {
    return (hb + mulmod(h, p[blen])) % MOD;
  }
};
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n;
  cin >> n;
  P = uniform(256, MOD - 1);
  vector<string_hashing> v(n);
  vector<string_hashing> v_rev(n);
  vector<int> sz(n);
  int ans = 0;
  for (int i = 0; i < n; i++)
  {
    string s;
    cin >> s;
    v[i] = string_hashing(s);
    sz[i] = s.size();
    ans += (s.size() * n);
    ans += (s.size() * n);
    reverse(s.begin(), s.end());
    v_rev[i] = string_hashing(s);
  }
  unordered_map<int, int> mp;
  for (int i = 0; i < n; i++)
  {
    for (int j = 1; j <= sz[i]; j++)
      mp[v[i].get(0, j - 1)]++;
  }
  for (int i = 0; i < n; i++)
  {
    int acc = 0;
    for (int j = sz[i]; j >= 1; j--)
    {
      int curr = mp[v_rev[i].get(0, j - 1)];
      ans -= ((curr - acc) * j * 2);
      acc = curr;
    }
  }
  cout << ans << endl;
}
// https://codeforces.com/contest/1902/problem/E
// solucao usando hash mod 2^61 - 1
```

2.10 suffix automaton

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 100001
#define mod 998244353

namespace sa
{
  struct state
  {
    int len, suf_link;
    map<char, int> nxt;
  };

  vector<int> term;
  state st[2 * MAXN];
  int dp[2 * MAXN];
  int sz, last;

  void init()
  {
    memset(dp, -1, sizeof(dp));
    st[0].len = 0;
    st[0].suf_link = -1;
    sz++;
    last = 0;
  }
  void get_link(int curr, int p, char c)
  {
    while (p != -1 && !st[p].nxt.count(c))
    {
      st[p].nxt[c] = curr;
      p = st[p].suf_link;
    }
    if (p == -1)
    {
      st[curr].suf_link = 0;
      return;
    }
    int q = st[p].nxt[c];
    if (st[p].len + 1 == st[q].len)
    {
      st[curr].suf_link = q;
      return;
    }
    int clone = sz;
    sz++;
    st[clone].len = st[p].len + 1;
    st[clone].nxt = st[q].nxt;
    st[clone].suf_link = st[q].suf_link;
    while (p != -1 && st[p].nxt[c] == q)
    {
      st[p].nxt[c] = clone;
      p = st[p].suf_link;
    }
    st[q].suf_link = clone;
    st[curr].suf_link = clone;
  }
  void build(string &s)
  {
    for (auto const &c : s)
    {
      int curr = sz;
      sz++;
      st[curr].len = st[last].len + 1;
      get_link(curr, last, c);
      last = curr;
    }
    // achar os estados terminais
    // um estado terminal e aquele que representa um sufixo da string s
    int p = last;
    while (p != -1)
    {
      term.pb(p);
      p = st[p].suf_link;
    }
  }
  void dfs2(int v)
  {
    if (dp[v] != -1)
      return;
    dp[v] = 1;
    for (auto const &u : st[v].nxt)
    {
      if (!u.sec)
        continue;
      dfs2(u.sec);
      dp[v] += dp[u.sec];
    }
  }
  void dfs(int v, int k, int &at, string &curr)
  {
    if (at == k)
      return;
    for (auto const &u : st[v].nxt)
    {
      if (!u.sec)
        continue;
      if (at + dp[u.sec] < k)
      {
        at += dp[u.sec];
        continue;
      }
      curr.pb(u.fir);
      at++;
      dfs(u.sec, k, at, curr);
      if (at == k)
        return;
      curr.pop_back();
    }
  }
  void find_kth(int k)
  {
    int at = 0;
    string curr = "";
    dfs(0, k, at, curr);
    cout << curr << endl;
  }
} // namespace sa
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  string s;
  cin >> s;
  sa::init();
  sa::build(s);
  sa::dfs2(0);
  int q;
  cin >> q;
  while (q--)
  {
    int k;
    cin >> k;
    sa::find_kth(k);
  }
  return 0;
}
// https://cp-algorithms.com/string/suffix-automaton.html
// suffix automaton
// definicao: um suffix automaton de uma string s e um automato finito
     deterministico
// que aceita todos os suffixos da string s.
// ou seja:
// um suffix automaton eh um grafo aciclico orientado
// tal que, um vertice representa um estado
// e uma aresta representa uma transicao (um caractere a mais em relacao ao
     estado(suffixo) atual)
// t0 -> estado inicial(string vazia), e todos os demais estados podem ser
     alcancados a partir de t0
// o suffix automaton minimiza o numero de vertices

// a propiedade mais importante de um suffix automaton eh a de que
// ele contem informacoes sobre todas as substrings de s
// pois, qualquer caminho comecando do estado t0 corresponde a uma substring de
     s

// conceitos:

// 1 - endpos
// seja t uma substring de s, endpos(t) eh o conjunto de todas os indices(
     posicoes)
// na string s no qual todas as ocorrencias de t acabam
// por exemplo, se s = "abcbc" e t = "bc"
// logo endpos(t) = {2, 4}
// com isso se duas duas substrings t1 e t2 possuem os seus endpos iguais,
// chamamos de endpos-equivalent e dai podemos extrair algumas informacoes
// info 1: se duas substrings u e w u.size() <= w.size(), se u eh um sufixo de w
     , logo endpos(u) esta contido em endpos(w)
// info 2: se duas substrings u e w u.size() <= w.size(), se u nao eh um sufixo
     de w, logo nao existe interseccao entre endpos(u) e endpos(w)

// 2 - suffix link
// seja v algum estado != t0,  sabemos que v corresponde a classe de strings que
      possui os mesmos endpos
// seja w a maior dessas strings, com isso, todas as demais sao suffixos de w
// com isso um suffix_link(v) corresponde ao maior suffix de w que esta em outra
      classe de equivalencia pelos endpos
// com isso podemos abstrair algumas informacoes:
// info 1: os suffix links foram uma arvore enraizada em t0
// info 2: se construirmos uma arvore usando os sets endpos, a estrutura sera a
     arvore com os suffix links

// com isso, vamos ao algoritimo
// 1 - vai ser online, e iremos adicionar os caracteres de 1 por 1, da esquerda
     para a direita
// 2 - com isso para adicionar um novo char, seja v o ultimo estado que
     adicionamos antes do atual, adicionamos uma aresta
// do proximo em relacao a ele e iremos procurar pelo suffix link para adicionar
// 3 - complexidade O(n) ou O(n log k), se usarmos uma map para guardar as
     transicoes partindo de um estado

// exemplos de aplicacoes:

// 1 - checar se t aparece em s como substring:
// construa o suffix automaton de s, e vamos tentar fazer um caminho partindo de
      t0
// se em algum momento, nao existir transicao, logo nao existe
// se conseguir chegar no final, existe

// 2 - numero de substrings diferentes de s
// constura o suffix automaton de s, sabemos que, cada substring de s
     corresponde a um caminho no automato
// com isso, o numero de substrings distintas eh o numero de caminhos diferentes
      que comecam de t0
// e terminam em algum canto
// isso pode ser calculado facilmente com uma dpzinha
// 3 - tamanho total de todas as substrings distintas de s
// similar a solucao passada, podemos fazer isso com uma dpzinha :)

// 4 - a k-esima menor substring lexicografica
// a k-esima menor substring lexicograficamente corresponde ao k-esimo path no
     suffix automaton
// se considerarmos as transicoes sempre indo do menor char para o maior durante
      o percurso

// 5 - o menor cyclic shift
// construa o suffix automaton da string s + s (duplicada)
// com isso o suffix automaton vai conter todos os cyclic shifts da string s
// e agora o problema eh reduzido para: encontre o menor caminho
     lexicograficamente de tamanho s.size()

// 6 - numero de ocorrencias de uma substring t em s
// construa o suffix automaton da string s
// com isso, quando criamos um no que nao seja o t0 nem um clone
// inicializamos cnt[v] = 1
// depois vamos percorrer todo os estados em ordem decrescente de len
// e aplicando cnt[link(v)] += cnt[v]
// no final, para responder a query basta fazer o caminho ate o estado que
     quisermos e printar o cnt dele

// e mais uma porrada de aplicacoes alem dessas :)

// example of a problem: https://www.spoj.com/problems/SUBLEX/
// ver qual a k-th string lexicografica sem repeticao
// note que o k pode ser gigante
// ideia: calcular dp[v] -> quantidade de caminhos que comecam em v
// dai para cada query roda um dfs, sendo que, so vou pro proximo estado se at +
      dp[u] >= k
// caso contrario, posso ignorar
```

2.11 kmp

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define MAXN 100005
#define mod 998244353

string s;
int n, m;
string a, b;
int c[MAXN][26];

vector<int> kmp(string &s)
{
  int n = s.size();
  vector<int> p(n);
  for (int i = 1; i < n; i++)
  {
    int j = p[i - 1];
    while (j > 0 && s[i] != s[j])
      j = p[j - 1];
    if (s[i] == s[j])
      j++;
    p[i] = j;
  }
  return p;
}
void compute(string s)
{
  s.pb('*');
  vector<int> p = kmp(s);
  for (int i = 0; i < s.size(); i++)
  {
    for (int cc = 0; cc < 26; cc++)
    {
      int j = i;
      while (j > 0 && 'a' + cc != s[j])
        j = p[j - 1];
      if ('a' + cc == s[j])
        j++;
      c[i][cc] = j;
    }
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  string s;
  cin >> s;
  compute(s);
  return 0;
}
// kmp
// algoritmo eh online, vai coonstruindo da esquerda pra direita
// calcula pi[i], a seguinte funcao:
// seja a substring s.substr(0, i + 1)
// pi[i] = tamanho do maior prefixo que tbm eh um sufixo dessa substring

// dai por exemplo
// da pra contar a quantidade de matchings de s em t
// so concatenar as strings fazendo: t = s + "*" + t
// dai contar as posicoes com pi[i] = s.size()

// tambem eh possivel construir um automato do kmp
// do tipo
// se meu pi[i] == x, e leio a letra c
// dai devo ir pro estado p[i] == y
// as transicoes podem ser computadas e isso pode ser muito util
```

2.12 rabin-karp

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define DEBUG 0
#define MAXN 100001

const int p = 31;
const int mod = 1e9 + 9;

int multiplicate(int x, int y)
{
  return (x * y) % mod;
}
int subtract(int a, int b)
{
  return (a - b < 0) ? a - b + mod : a - b;
}
int sum(int a, int b)
{
  return (a + b >= mod) ? a + b - mod : a + b;
}
vector<int> rabin_karp(string s, string t)
{
  int n = s.size(), m = t.size();
  vector<int> pot(n);
  pot[0] = 1;
  for (int i = 1; i < n; i++)
    pot[i] = multiplicate(pot[i - 1], p);
  vector<int> pref(n + 1, 0);
  for (int i = 0; i < n; i++)
  {
    int val = multiplicate(pref[i], p);
    pref[i + 1] = sum(s[i], val);
  }
  int hs = 0;
  for (int i = 0; i < m; i++)
  {
    int val = multiplicate(hs, p);
    hs = sum(t[i], val);
  }
  vector<int> ans;
  for (int i = 0; i + m - 1 < n; i++)
  {
    int cur_h = subtract(pref[i + m], multiplicate(pref[i], pot[m]));
    if (cur_h == hs)
      ans.pb(i);
  }
  return ans;
}
signed main()
{
  string s, t;
  cin >> s >> t;
  vector<int> ans = rabin_karp(s, t);
  for (auto const &i : ans)
    cout << i << " " << i + t.size() - 1 << endl;
  return 0;
}
// rabin-karp for pattern matching
// given two string s and t, determine all occurrences of t in s
// 1- calcule the hash of string t
// 2- calcule the prefix hash of string s
// 3- compare every substring of s with length |t|
// 4- store all occurrences in a vector and return this vector
// complexity: O(|t| + |s|)
```

3 Binary Search and Ternary Search

3.1 parallel binary search

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 300006
#define mod 1000000007

int n;
int b[MAXN];

void reset()
{
  for (int i = 0; i < MAXN; i++)
    b[i] = 0;
}
int sum(int r)
{
  int ret = 0;
  for (; r >= 0; r = (r & (r + 1)) - 1)
    ret += b[r];
  return ret;
}
void add(int idx, int delta)
{
  for (; idx < MAXN; idx = idx | (idx + 1))
    b[idx] += delta;
}
void update(int l, int r, int x)
{
  add(l, x);
  add(r + 1, -x);
}
void upd(int l, int r, int x)
{
  if (l <= r)
  {
    update(l, r, x);
    return;
  }
  update(l, MAXN - 2, x);
  update(0, r, x);
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n, m;
  cin >> n >> m;
  vector<vector<int>> adj(n);
  for (int i = 0; i < m; i++)
  {
    int x;
    cin >> x;
    x--;
    adj[x].pb(i);
  }
  vector<int> need(n);
  for (int i = 0; i < n; i++)
  {
    cin >> need[i];
  }
  int q;
  cin >> q;
  vector<pii> qry(q);
  for (int i = 0; i < q; i++)
  {
    cin >> qry[i].sec.fir >> qry[i].sec.sec >> qry[i].fir;
    qry[i].sec.fir--, qry[i].sec.sec--;
  }
  vector<int> l(n);
  vector<int> r(n);
  vector<vector<int>> on(q);
  for (int i = 0; i < n; i++)
  {
    l[i] = 0;
    r[i] = q;
  }
  while (1)
  {
    bool ok = 1;
    for (int i = 0; i < n; i++)
    {
      if (l[i] < r[i])
      {
        ok = 0;
        int mid = (l[i] + r[i]) >> 1;
        on[mid].pb(i);
      }
    }
    if (ok)
      break;
    reset();
    for (int mid = 0; mid < q; mid++)
    {
      upd(qry[mid].sec.fir, qry[mid].sec.sec, qry[mid].fir);
      for (auto const &j : on[mid])
      {
        int val = 0;
        for (auto const &k : adj[j])
        {
          val += sum(k);
          if (val >= need[j])
            break;
        }
        (val >= need[j]) ? r[j] = mid : l[j] = mid + 1;
      }
      on[mid].clear();
    }
  }
  for (int i = 0; i < n; i++)
  {
    if (l[i] >= q)
      cout << "NIE\n";
    else
      cout << l[i] + 1 << endl;
  }
  return 0;
}
// busca binaria paralela
// https://www.spoj.com/problems/METEORS/

// tem n member states e m sectors
// cada sector ta associado a uma member state
// cada query incrementa o range [l[i], r[i]] de sectors por a[i]
// seja q[i] a soma de todos os v[j], sendo j um sector associado ao member
     state i
// qual o primeiro momento no qual q[i] >= min_qt[i]
// para todos os i

// a sagacidade vai ser fzr uma busca binaria pra cada resposta
// primeiro vc faz todas a primeira iteracao de cada busca binaria
// depois cada segunda iteracao de cada bb
// e assim vai

// ai a bb eh so tipo
// a soma apos a query mid ja deu bom pra aquele member state?
```

3.2 Aplications

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

#define lli long long int
#define pb push_back
#define in insert
#define pi pair<int, int>
#define pii pair<int, pi>
#define mp make_pair
#define fir first
#define sec second
#define MAXN 1001

// 1 - ts para double
long double ts()
{
  long double l = 0, r = DBL_MAX;
  for (int i = 0; i < 2000; i++)
  {
    long double l1 = (l * 2 + r) / 3.0;
    long double l2 = (l + 2 * r) / 3.0;
    if (possible(l1))
      r = l2;
    else
      l = l1;
  }
  return l;
}
// 2- bb para double
long double bb()
{
  long double i = 0, f = DBL_MAX, m;
  while (f - i > 0.000000001)
  {
    m = (i + f) / 2.0;
    if (possible(m))
      f = m;
    else
      i = m;
  }
  return i;
}
// 3 - bb pra int
lli bb()
{
  lli i = 0, f = INT_MAX, m;
  while (i < f)
  {
    m = (i + f) / 2;
    if (possible(m))
      f = m;
    else
      i = m + 1;
  }
  return i;
}
// 4 - ts pra int (valor minimo da funcao f(x)), sendo x um inteiro
int l = 1, r = INT_MAX;
while (r - l > 15)
{
  int l1 = (l * 2 + r) / 3;
  int l2 = (l + 2 * r) / 3;
  (calc(l1) < calc(l2)) ? r = l2 : l = l1;
}
for (int i = l; i <= r; i++)
// vejo qual a melhor opcao de l ate r em o(n)


// busca ternaria para int, usando busca binaria:
int l = 0, r = 1e9;
while (l < r)
{
  int mid = (l + r) >> 1;
  (calc(mid) < calc(mid + 1)) ? r = mid : l = mid + 1;
}
return calc(l);
```

3.3 UpperBound

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
```

```
#define sec second
#define DEBUG 0
#define MAXN 1000001
#define mod 1000000007
// last element <= x
vector<int> k(MAXN);
int upper(int l, int r, int x)
{
  while (l < r)
  {
    int mid = (l + r + 1) >> 1;
    (k[mid] <= x) ? l = mid : r = mid - 1;
  }
  return k[l];
}
```

3.4 STL

```
// lower - primeiro maior ou igual a x
// upper - ultimo menor ou igual a x

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

#define lli long long int
#define pb push_back

vector <int> v ;
int main()
{
    int n , aux ;
    cin >> n ;

    for (int i = 0 ; i < n ; i++)
    {
        cin >> aux ;
        v.pb(aux);
    }

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

    int q ;
    cin >> q ;

    while (q--)
    {
        cin >> aux ;
        vector <int> :: iterator low = lower_bound (v.begin() , v.end() , aux) ;
        vector <int> :: iterator up = upper_bound (v.begin() , v.end() , aux) ;

        cout << (low - v.begin()) << " " << (up - v.begin()) - 1 << endl ;
    }

    return 0 ;
}
```

3.5 TS

```
// busca ternaria
// divide em 3 partes, 2 mids
// mid1 = l + (r-l)/3
// mid2 = r - (r-l)/3
#include <bits/stdc++.h>
using namespace std;

#define lli long long int
#define pb push_back
#define in insert
#define pi pair<int, int>
#define pii pair<int, pi>
#define mp make_pair
#define fir first
#define sec second
#define MAXL 100001

int n, key;
vector<int> ar;

int ts()
{
  int l = 0, r = n - 1;
  while (r >= l)
  {
    int mid1 = l + (r - l) / 3;
    int mid2 = r - (r - l) / 3;
    if (ar[mid1] == key)
      return mid1;
    if (ar[mid2] == key)
      return mid2;
    if (key < ar[mid1])
      r = mid1 - 1;
    else if (key > ar[mid2])
      l = mid2 + 1;
    else
    {
      l = mid1 + 1;
      r = mid2 - 1;
    }
  }
  return -1; // nao encontrado
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> n;
  ar.resize(n);
  for (int i = 0; i < n; i++)
    cin >> ar[i];
  sort(ar.begin(), ar.end());
  cin >> key;
  cout << ts() << endl;
  return 0;
}
```

3.6 BS

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

#define lli long long int
#define pb push_back

vector <int> v;
int binarysearch (int n , int x)
{
    int i = 0 ;
    int f = n - 1 ;
    int m ;

    while(i <= f)
    {
        m = (i + f) / 2 ;

        if(v[m] == x) return m + 1 ;
        if(v[m] < x)  i = m + 1 ;
        if(v[m] > x)  f = m - 1 ;
    }

    return 0 ;
}
int main ()
{
    int n , aux , m ;

    cin >> n ;

    for (int i = 0 ; i < n ; i++)
    {
        cin >> aux ;
        v.pb(aux);
    }

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

    cin >> m ;
    cout << binarysearch(n , m) << endl ;

    return 0 ;
}
```

3.7 LowerBound

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define DEBUG 0
#define MAXN 1000001
#define mod 1000000007
// first element >= x
vector<int> k(MAXN);
int lower(int l, int r, int x) // first element >= x
{
  while (l < r)
  {
    int mid = (l + r) >> 1;
    (x <= k[mid]) ? r = mid : l = mid + 1;
  }
  return k[l];
}
```

4 Graph

4.1 Floyd Warshall

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

#define pb push_back
#define lli long long int
#define MAXN 10000
#define INF 999999

int n , m , a , b , c ;
int dist [MAXN][MAXN] ;

void floyd_warshall ()
{
    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]) ;
            }
        }
    }
}
void initialize ()
{
    for (int i = 0 ; i < n ; i++)
    {
        for (int j = 0 ; j < n ; j++)
        {
            if (i == j)
            {
                dist[i][j] = 0 ;
            }
            else
            {
                dist[i][j] = INF ;
            }

        }
    }
}
int main()
{
    cin >> n >> m ;

    initialize () ;

    for (int i = 0 ; i < m ; i++)
    {
        cin >> a >> b >> c ;
        dist [a][b] = min (dist[a][b] , c) ;
        dist [b][a] = min (dist[b][a] , c) ;
    }

    floyd_warshall () ;

    return 0;
}
```

4.2 reroot

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define PI acos(-1)
#define pb push_back
#define int long long int
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define DEBUG 0
#define MAXN 200001
#define mod 1000000007

int n;
vector<int> adj[MAXN];
int sz[MAXN];
int dp[MAXN];

int dfs(int u, int v)
{
  sz[u] = 1;
  for (auto const &i : adj[u])
    if (i != v)
      sz[u] += dfs(i, u);
  return sz[u];
}
```

```
void reroot(int u, int v)
{
  for (auto const &i : adj[u])
  {
    if (i != v)
    {
      int a = sz[u], b = sz[i];
      dp[i] = dp[u];
      dp[i] -= sz[u], dp[i] -= sz[i];
      sz[u] -= sz[i], sz[i] = n;
      dp[i] += sz[u], dp[i] += sz[i];
      reroot(i, u);
      sz[u] = a, sz[i] = b;
    }
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> n;
  for (int i = 0; i < n - 1; i++)
  {
    int a, b;
    cin >> a >> b;
    a--, b--;
    adj[a].pb(b);
    adj[b].pb(a);
  }
  dfs(0, -1);
  for (int i = 0; i < n; i++)
    dp[0] += sz[i]; // answer when tree is rooted on vertex 0
  reroot(0, -1);
  cout << *max_element(dp, dp + n) << endl;
  return 0;
}
// https://codeforces.com/contest/1187/problem/E
// f(v) = when tree is rooted at vertex v, the current
// answer is the sum of all subtrees sizes
// final answer = max(f(0), f(1), f(2), ..., f(n))
// easy approach: O(N^2)
// with reroot: O(N)
// 1 - run a dfs and calculate f(0)
// 2 - let be dp[i] = f(i)
// 3 - now, lets run a another dfs, and re-calculate the
// answer when tree is rooted at vertex i (dp[i])
// 4 - the final answer is the maximum value of dp[i]
```

4.3 Ford Fulkerson

```
// ford-fulkerson: obter qual o fluxo maximo de um vertice s ate um vertice d
// 1 - rodar um bfs para descobrir um novo caminho de s ate d
// 2 - apos isso pego a aresta de menor custo desse caminho e subtraio o valor
     dela nas outras arestas do caminho
// 3 - fluxo_maximo += custo da aresta de menor custo desse caminho
// 4 - rodar isso ate nao existirem mais caminhos disponiveis (com fluxo
     diferente de 0) entre s e d
// 5 - o fluxo maximo de s ate d sera a soma das arestas de menor custo de cada
     caminho feito

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

#define lli long long int
#define pb push_back
#define MAXN 10000
#define INF 999999

int n , m , a , b , c , s , d , max_flow , flow ;
vector <int> parent ;
vector <int> adj [MAXN] ;
int cost [MAXN][MAXN] ;
bool visited [MAXN] ;

void get_menor_custo (int v , int mincost)
{
    if (v == s)
    {
        flow = mincost ;
        return ;
    }
    else if (parent[v] != -1)
    {
        get_menor_custo(parent[v] , min(mincost , cost[parent[v]][v])) ;
        cost[parent[v]][v] -= flow ;
        cost[v][parent[v]] += flow ;
    }
}
void bfs ()
{
    visited[s] = true ;

    queue <int> q ;
    q.push(s) ;
    parent.assign(MAXN , -1) ;

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

        if (u == d)
        {
            break ;
        }

        for (int j = 0 ; j < adj[u].size() ; j++)
        {
            int v = adj[u][j] ;

            if (cost[u][v] > 0 && !visited[v])
            {
                visited[v] = true ;
                q.push(v) ;
                parent[v] = u ;
            }
        }
    }
}
int ford_fulkerson ()
{
    max_flow = 0 ;

    while (1)
    {
        flow = 0 ;
        memset(visited , false , sizeof(visited));

        bfs() ;
        get_menor_custo(d , INF) ;

        if (flow == 0)
        {
            break ;
        }

        max_flow += flow ;
    }

    return max_flow ;
}
int main ()
{
    ios_base::sync_with_stdio(false) ;
    cin.tie(NULL) ;

    cin >> n >> m ;

    for (int i = 0 ; i < m ; i++)
    {
        cin >> a >> b >> c ;
        adj[a].pb(b);
        adj[b].pb(a);
        cost[a][b] = c ;
    }
    cin >> s >> d ;

    cout << ford_fulkerson() << endl ;

    return 0 ;
}
```

4.4 stable matching

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

// #define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define pci pair<char, int>
#define fir first
#define sec second
#define MAXN 100005
#define mod 1000000007

bool in[1005][1005];
int tt[1005][1005];
int b[1005];
int ini[1005];

// a.size() <= b.size()
vector<int> stable_marriage(vector<vector<int>> &a, vector<vector<int>> &b)
{
  int n = a.size(), m = b.size();
  assert(a[0].size() == m and b[0].size() == n and n <= m);
  vector<int> match(m, -1), it(n, 0);
  vector<vector<int>> inv_b(m, vector<int>(n));
  for (int i = 0; i < m; i++)
    for (int j = 0; j < n; j++)
      inv_b[i][b[i][j]] = j;
  queue<int> q;
  for (int i = 0; i < n; i++)
    q.push(i);
  while (q.size())
  {
    int i = q.front();
    q.pop();
    int j = a[i][it[i]];
    if (match[j] == -1)
      match[j] = i;
    else if (inv_b[j][i] < inv_b[j][match[j]])
    {
      q.emplace(match[j]);
      it[match[j]]++;
      match[j] = i;
    }
    else
      q.emplace(i), it[i]++;
  }
  vector<int> ret(n);
  for (int i = 0; i < m; i++)
    if (match[i] != -1)
      ret[match[i]] = i;
  return ret;
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n, m, d, e;
  cin >> n >> m >> d >> e;
  if (n > m)
  {
    cout << "impossible\n";
    return 0;
  }
  vector<vector<int>> adj(n);
  vector<vector<int>> adj2(m);
  memset(b, -1, sizeof(b));
  memset(ini, -1, sizeof(ini));
  for (int i = 0; i < e; i++)
  {
    int s, k, t;
    cin >> s >> k >> t;
    k--, t--;
    if (t == -1)
    {
      tt[k][b[k]] += (s - ini[k]);
      b[k] = -1;
      ini[k] = -1;
    }
    else
    {
      if (b[k] != -1)
      {
        tt[k][b[k]] += (s - ini[k]);
      }
      if (!in[k][t])
      {
        in[k][t] = 1;
        adj[k].pb(t);
      }
      b[k] = t;
      ini[k] = s;
    }
  }
  for (int k = 0; k < n; k++)
  {
    if (b[k] != -1)
      tt[k][b[k]] += (d - ini[k]);
  }
  for (int k = 0; k < n; k++)
  {
    for (int t = 0; t < m; t++)
    {
      if (!in[k][t])
        adj[k].pb(t);
    }
  }
  for (int t = 0; t < m; t++)
  {
    vector<pi> curr;
    for (int k = 0; k < n; k++)
    {
      curr.pb({tt[k][t], k});
    }
    sort(curr.begin(), curr.end());
    for (auto const &i : curr)
      adj2[t].pb(i.sec);
  }
  vector<int> ans = stable_marriage(adj, adj2);
  for (auto const &i : ans)
    cout << i + 1 << " ";
  cout << endl;
}
// solucao pro: https://open.kattis.com/problems/jealousyoungsters

// stable marriage
// voce quer achar um matching em um grafo bipartido
// que todos os caras de um lado dao matching com algum do outro lado

// cada vertice tem um vector
// que diz qual a ordem de preferencia dele
// o que ele mais quer dar matching eh com v[0]
// o segundo com que ele mais quer dar matching eh com v[1]
// e assim vai

// quando a.size() <= b.size(), entao sempre existe um stable matching
```

4.5 bridges

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 400005
#define mod 1000000007

int n, m, timer;
vector<pi> edges;
vector<bool> is_bridge;
vector<pi> adj[MAXN];
int tin[MAXN];
int low[MAXN];
bool vis[MAXN];

void dfs(int v, int p)
{
  vis[v] = true;
  tin[v] = timer, low[v] = timer++;
  for (auto const &u : adj[v])
  {
    if (u.fir == p)
      continue;
    if (vis[u.fir])
    {
      low[v] = min(low[v], tin[u.fir]);
      continue;
    }
    dfs(u.fir, v);
    low[v] = min(low[v], low[u.fir]);
    if (low[u.fir] > tin[v])
      is_bridge[u.sec] = 1;
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> n >> m;
  is_bridge.resize(m);
  for (int i = 0; i < m; i++)
  {
    int a, b;
    cin >> a >> b;
    a--, b--;
    edges.pb({a, b});
    adj[a].pb({b, i});
    adj[b].pb({a, i});
  }
  memset(tin, -1, sizeof(tin));
  memset(low, -1, sizeof(low));
  for (int i = 0; i < n; i++)
  {
    if (!vis[i])
      dfs(i, -1);
  }
  return 0;
}
```

4.6 mincostflow

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 301
#define mod 1000000007
#define INF 1e9

namespace mcf
{
  struct edge
  {
    int to, capacity, cost, res;
  };

  int source, destiny;
  vector<edge> adj[MAXN];
  vector<int> dist;
  vector<int> parent;
  vector<int> edge_index;
  vector<bool> in_queue;

  void add_edge(int a, int b, int c, int d)
  {
    adj[a].pb({b, c, d, (int)adj[b].size()});      // aresta normal
    adj[b].pb({a, 0, -d, (int)adj[a].size() - 1}); // aresta do grafo residual
  }
  bool dijkstra(int s) // rodando o dijkstra, terei o caminho de custo minimo
  {                    // que eu consigo passando pelas arestas que possuem
       capacidade > 0
    dist.assign(MAXN, INF);
    parent.assign(MAXN, -1);
    edge_index.assign(MAXN, -1);
    in_queue.assign(MAXN, false);
    dist[s] = 0;
    queue<int> q;
    q.push(s);
    while (!q.empty())
    {
      int u = q.front(), idx = 0;
      q.pop();
      in_queue[u] = false;
      for (auto const &v : adj[u])
      {
        if (v.capacity && dist[v.to] > dist[u] + v.cost)
        {
          dist[v.to] = dist[u] + v.cost;
          parent[v.to] = u;
          edge_index[v.to] = idx;
          if (!in_queue[v.to])
          {
            in_queue[v.to] = true;
            q.push(v.to);
          }
        }
        idx++;
      }
    }
    return dist[destiny] != INF; // se eu cheguei em destiny por esse caminho,
         ainda posso passar fluxo
  }
  int get_cost()
  {
    int flow = 0, cost = 0;
    while (dijkstra(source)) // rodo um dijkstra para saber qual o caminho que
         irei agora
    {
      int curr_flow = INF, curr = destiny;
      while (curr != source) // com isso, vou percorrendo o caminho encontrado
           para achar a aresta "gargalo"
      {
        int p = parent[curr];
        curr_flow = min(curr_flow, adj[p][edge_index[curr]].capacity);
        curr = p;
      }
      flow += curr_flow;                 // fluxo que eu posso passar por esse
           caminho = custo da aresta "gargalo"
      cost += curr_flow * dist[destiny]; // quanto eu gasto para passar esse
           fluxo no caminho encontrado
      curr = destiny;
      while (curr != source) // apos achar a aresta gargalo, passamos o fluxo
           pelo caminho encontrado
      {
        int p = parent[curr];
        int res_idx = adj[p][edge_index[curr]].res;
        adj[p][edge_index[curr]].capacity -= curr_flow;
        adj[curr][res_idx].capacity += curr_flow;
        curr = p;
      }
    }
    return cost; // ao final temos a resposta :)
  }
} // namespace mcf
signed main()
{
  int n;
  cin >> n;
  int v[n][n];
  mcf::source = 0, mcf::destiny = (2 * n) + 1;
  for (int i = 0; i < n; i++)
  {
    for (int j = 0; j < n; j++)
    {
      cin >> v[i][j];
      mcf::add_edge(i + 1, j + n + 1, 1, v[i][j]);
    }
  }
  for (int i = 1; i <= n; i++)
    mcf::add_edge(mcf::source, i, 1, 0);
  for (int i = n + 1; i <= n + n; i++)
    mcf::add_edge(i, mcf::destiny, 1, 0);
  cout << mcf::get_cost << endl;
}
```

4.7 link cut tree vertex

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 100005
#define mod 998244353

struct node
{
  int p, ch[2];
  int val, sub, lazy;
  bool rev;
  int sz;
  node() {}
  node(int v) : p(-1), val(v), sub(v), rev(0), sz(1), lazy(0)
  {
    ch[0] = ch[1] = -1;
  }
};
struct link_cut_tree
{
  vector<node> t;

  link_cut_tree()
  {
    t.clear();
  }
  static int neutral()
  {
    return 0;
  }
  int merge(int a, int b)
  {
    return a ^ b;
  }
  void prop(int x)
  {
    if (t[x].lazy)
    {
      t[x].val = t[x].lazy;
      t[x].sub = t[x].val;
      if (t[x].ch[0] + 1)
        t[t[x].ch[0]].lazy = t[x].lazy;
      if (t[x].ch[1] + 1)
        t[t[x].ch[1]].lazy = t[x].lazy;
    }
    if (t[x].rev)
    {
      swap(t[x].ch[0], t[x].ch[1]);
      if (t[x].ch[0] + 1)
        t[t[x].ch[0]].rev ^= 1;
      if (t[x].ch[1] + 1)
        t[t[x].ch[1]].rev ^= 1;
    }
    t[x].lazy = 0, t[x].rev = 0;
  }
  void update(int x)
  {
    t[x].sz = 1, t[x].sub = t[x].val;
    for (int i = 0; i < 2; i++)
    {
      if (t[x].ch[i] + 1)
      {
        prop(t[x].ch[i]);
        t[x].sz += t[t[x].ch[i]].sz;
        t[x].sub = merge(t[x].sub, t[t[x].ch[i]].sub);
      }
    }
  }
  bool is_root(int x)
  {
    return t[x].p == -1 or (t[t[x].p].ch[0] != x and t[t[x].p].ch[1] != x);
  }
  void rotate(int x)
  {
    int p = t[x].p, pp = t[p].p;
    if (!is_root(p))
      t[pp].ch[t[pp].ch[1] == p] = x;
    bool d = t[p].ch[0] == x;
    t[p].ch[!d] = t[x].ch[d], t[x].ch[d] = p;
    if (t[p].ch[!d] + 1)
      t[t[p].ch[!d]].p = p;
    t[x].p = pp, t[p].p = x;
    update(p), update(x);
  }
  int splay(int x)
  {
    while (!is_root(x))
    {
      int p = t[x].p, pp = t[p].p;
      if (!is_root(p))
        prop(pp);
      prop(p), prop(x);
      if (!is_root(p))
        rotate((t[pp].ch[0] == p) ^ (t[p].ch[0] == x) ? x : p);
      rotate(x);
    }
    return prop(x), x;
  }
  int access(int v)
  {
    int last = -1;
    for (int w = v; w + 1; update(last = w), splay(v), w = t[v].p)
      splay(w), t[w].ch[1] = (last == -1 ? -1 : v);
    return last;
  }
  void make_tree(int v, int w = neutral()) // cria uma arvore com um unico vertice, sendo o vertice i
  {
    while (t.size() <= v)
      t.pb(node(neutral()));
    t[v] = node(w);
  }
  int find_root(int v) // acha a raiz da arvore do vertice v
  {
    access(v), prop(v);
    while (t[v].ch[0] + 1)
      v = t[v].ch[0], prop(v);
    return splay(v);
  }
  bool connected(int v, int w) // checa se v e w estao na mesma arvore (aka componente conexa)
  {
    access(v), access(w);
    return v == w ? true : t[v].p != -1;
  }
  void rootify(int v) // torna v a raiz de sua arvore
  {
    access(v);
    t[v].rev ^= 1;
  }
  int query(int v, int w) // query no caminho de v ate w
  {
    rootify(w), access(v);
    return t[v].sub;
  }
  void update(int v, int w, int x) // aplica o update em todos os vertices no caminho de v ate w
  {
    rootify(w), access(v);
    t[v].lazy = x;
  }
  void link(int v, int w) // adiciona a aresta v - w
  {
    rootify(w);
    t[w].p = v;
  }
  void cut(int v, int w) // remove a aresta v - w
  {
    rootify(w), access(v);
    t[v].ch[0] = t[t[v].ch[0]].p = -1;
  }
  int lca(int v, int w) // acha o lca(v, w)
  {
    access(v);
    return access(w);
  }
};

set<int> adj[MAXN];

signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n, q;
  cin >> n >> q;
  int x = 0;
  link_cut_tree l;
  for (int i = 0; i < n; i++)
  {
    // l.make_tree(i, i) // poderia fazer isso tbm, para criar o vertice i com valor i
    l.make_tree(i);
    l.update(i, i, i); // testa o update
  }
  while (q--)
  {
    int a, b, c;
    cin >> a >> b >> c;
    a = ((a * (1 + x)) % mod) % 2;
    b = ((b * (1 + x)) % mod) % n;
    c = ((c * (1 + x)) % mod) % n;
    if (a == 0)
    {
      adj[b].insert(c);
      adj[c].insert(b);
      l.link(b, c);
    }
    else
    {
      if (!l.connected(b, c))
      {
        x = 0;
        cout << x << endl;
        continue;
      }
      int d = l.query(b, c);
      d ^= b;
      d ^= c;
      x = 0;
      if (d >= 0 && d < n)
      {
        bool ok = 1;
        ok &= (adj[b].find(d) != adj[b].end());
        ok &= (adj[c].find(d) != adj[c].end());
        if (ok)
          x = d + 1;
      }
      cout << x << endl;
    }
  }
}
// problema exemplo: https://atcoder.jp/contests/abc350/tasks/abc350_g
// nesse caso, a link cut tree tem lazy para assign de valor
// e de queries de xor
```

4.8 strong orientation

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 1000005
#define mod 1000000007

int n, m, timer, comps, bridges;
vector<pi> edges;
vector<pi> adj[MAXN];
int tin[MAXN];
int low[MAXN];
```

```
bool vis[MAXN];
char orient[MAXN];

void find_bridges(int v)
{
  low[v] = timer, tin[v] = timer++;
  for (auto const &p : adj[v])
  {
    if (vis[p.sec])
      continue;
    vis[p.sec] = true;
    orient[p.sec] = (v == edges[p.sec].first) ? '>' : '<';
    if (tin[p.fir] == -1)
    {
      find_bridges(p.fir);
      low[v] = min(low[v], low[p.fir]);
      if (low[p.fir] > tin[v])
        bridges++;
    }
    else
    {
      low[v] = min(low[v], low[p.fir]);
    }
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> n >> m;
  for (int i = 0; i < m; i++)
  {
    int a, b;
    cin >> a >> b;
    a--, b--;
    edges.pb({a, b});
    adj[a].pb({b, i});
    adj[b].pb({a, i});
  }
  memset(tin, -1, sizeof(tin));
  memset(low, -1, sizeof(low));
  for (int v = 0; v < n; v++)
  {
    if (tin[v] == -1)
    {
      comps++;
      find_bridges(v);
    }
  }
  // numero minimo de scc = numero de componentes + numero de pontes
  cout << comps + bridges << endl;
  // > - a aresta foi orientada da esquerda pra direita
  // < - a aresta foi orientada da direita pra esquerda
  for (int i = 0; i < m; i++)
    cout << orient[i];
  cout << endl;
  return 0;
}
// to_test: https://szkopul.edu.pl/problemset/problem/nldsb4EW1YuZykBlf4lcZL1Y/
     site/?key=statement
// strong orientation:
// encontrar uma orientacao para as arestas tal que o numero
// minimo de scc e o menor possivel
```

4.9 centroid decomposition2

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 100005
#define mod 998244353

int n, k, resp;
vector<int> adj[MAXN];
vector<int> cnt;

namespace cd
{
  int sz;
  vector<int> subtree_size;
  vector<bool> visited;

  void dfs(int s, int f)
  {
    sz++;
    subtree_size[s] = 1;
    for (auto const &v : adj[s])
    {
      if (v != f && !visited[v])
      {
        dfs(v, s);
        subtree_size[s] += subtree_size[v];
      }
    }
  }
  int get_centroid(int s, int f)
  {
    bool is_centroid = true;
    int heaviest_child = -1;
    for (auto const &v : adj[s])
    {
      if (v != f && !visited[v])
      {
        if (subtree_size[v] > sz / 2)
          is_centroid = false;
        if (heaviest_child == -1 || subtree_size[v] > subtree_size[
             heaviest_child])
          heaviest_child = v;
      }
    }
    return (is_centroid && sz - subtree_size[s] <= sz / 2) ? s : get_centroid(
         heaviest_child, s);
  }
  void dfs2(int s, int f, int d)
  {
    while (d >= cnt.size())
      cnt.pb(0);
    cnt[d]++;
    for (auto const &v : adj[s])
    {
      if (v != f && !visited[v])
        dfs2(v, s, d + 1);
    }
  }
  void solve(int s)
  {
    vector<int> tot;
    for (auto const &v : adj[s])
    {
      if (visited[v])
        continue;
      cnt.clear();
      dfs2(v, s, 1);
      for (int i = 1; i < cnt.size(); i++)
      {
        if (k - i < tot.size() && k - i >= 1)
          resp += (cnt[i] * tot[k - i]);
      }
      for (int i = 1; i < cnt.size(); i++)
      {
        while (i >= tot.size())
          tot.pb(0);
        tot[i] += cnt[i];
      }
    }
    if (k < tot.size())
```

```
      resp += tot[k];
  }
  int decompose_tree(int s)
  {
    sz = 0;
    dfs(s, s);
    int cend_tree = get_centroid(s, s);
    visited[cend_tree] = true;
    solve(cend_tree);
    for (auto const &v : adj[cend_tree])
    {
      if (!visited[v])
        decompose_tree(v);
    }
    return cend_tree;
  }
  void init()
  {
    subtree_size.resize(n);
    visited.resize(n);
    decompose_tree(0);
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> n >> k;
  for (int i = 1; i < n; i++)
  {
    int a, b;
    cin >> a >> b;
    a--, b--;
    adj[a].pb(b);
    adj[b].pb(a);
  }
  cd::init();
  cout << resp << endl;
  return 0;
}
// https://codeforces.com/contest/161/problem/D
// problema: contar quantos pares de vertices (u, v) existem tal que dist(u, v)
     = k
// durante a decomposicao
// pega o centroid atual e resolve o problema pra ele
// isso eh:
// para cada centroid que eu achei, devo contar quantos caminhos
// de tamanho k passam por esse centroid
// somando todas essas respostas, a gente tem a resposta final
```

4.10 LCA

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define PI acos(-1)
#define pb push_back
#define int long long int
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define DEBUG 0
#define MAXN 100001
#define mod 1000000007

int n;
vector<int> adj[MAXN];
namespace lca
{
  int l, timer;
  vector<int> tin, tout, depth;
  vector<vector<int>> up;

  void dfs(int v, int p)
  {
    tin[v] = ++timer;
    up[v][0] = p;
    for (int i = 1; i <= l; i++)
      up[v][i] = up[up[v][i - 1]][i - 1];
    for (auto const &u : adj[v])
    {
      if (p == u)
        continue;
      depth[u] = depth[v] + 1;
      dfs(u, v);
    }
    tout[v] = ++timer;
  }
  bool is_ancestor(int u, int v)
  {
    return tin[u] <= tin[v] && tout[u] >= tout[v];
  }
  int binary_lifting(int u, int v)
  {
    if (is_ancestor(u, v))
      return u;
    if (is_ancestor(v, u))
      return v;
    for (int i = l; i >= 0; --i)
      if (!is_ancestor(up[u][i], v))
        u = up[u][i];
    return up[u][0];
  }
  void init()
  {
    tin.resize(n);
    tout.resize(n);
    depth.resize(n);
    timer = 0;
    l = ceil(log2(n));
    up.assign(n, vector<int>(l + 1));
    dfs(0, 0);
  }
  int dist(int s, int v)
  {
    int at = binary_lifting(s, v);
    return (depth[s] + depth[v] - 2 * depth[at]);
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> n;
  for (int i = 0; i < n - 1; i++)
  {
    int a, b;
    cin >> a >> b;
    a--, b--;
    adj[a].pb(b);
    adj[b].pb(a);
  }
  lca::init();
  return 0;
}
```

4.11 dsu rollback

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;
```

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 600005
#define mod 1000000007

namespace dsu
{
  struct rollback
  {
    int u, v, ranku, rankv;
  };

  int num_sets;
  int parent[MAXN];
  int rank[MAXN];
  stack<rollback> op;

  int Find(int i)
  {
    return (parent[i] == i) ? i : Find(parent[i]);
  }
  bool Union(int x, int y)
  {
    int xx = Find(x);
    int yy = Find(y);
    if (xx != yy)
    {
      num_sets--;
      if (rank[xx] > rank[yy])
        swap(xx, yy);
      op.push({xx, yy, rank[xx], rank[yy]});
      parent[xx] = yy;
      if (rank[xx] == rank[yy])
        rank[yy]++;
      return true;
    }
    return false;
  }
  void do_rollback()
  {
    if (op.empty())
      return;
    rollback x = op.top();
    op.pop();
    num_sets++;
    parent[x.v] = x.v;
    rank[x.v] = x.rankv;
    parent[x.u] = x.u;
    rank[x.u] = x.ranku;
  }
  void init(int n)
  {
    for (int i = 0; i < n; i++)
    {
      parent[i] = i;
      rank[i] = 0;
    }
    num_sets = n;
  }
}
namespace seg
{
  struct query
  {
    int v, u, is_bridge;
  };

  vector<vector<query>> t(4 * MAXN);
  int ans[MAXN];
```

```
  void add(int i, int l, int r, int ql, int qr, query q)
  {
    if (l > r || l > qr || r < ql)
      return;
    if (l >= ql && r <= qr)
    {
      t[i].push_back(q);
      return;
    }
    int mid = (l + r) >> 1;
    add((i << 1), l, mid, ql, qr, q);
    add((i << 1) | 1, mid + 1, r, ql, qr, q);
  }
  void dfs(int i, int l, int r)
  {
    for (query &q : t[i])
      if (dsu::Union(q.v, q.u))
        q.is_bridge = 1;
    if (l == r)
      ans[l] = dsu::num_sets;
    else
    {
      int mid = (l + r) >> 1;
      dfs((i << 1), l, mid);
      dfs((i << 1) | 1, mid + 1, r);
    }
    for (query q : t[i])
      if (q.is_bridge)
        dsu::do_rollback();
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n, q;
  cin >> n >> q;
  int time = 0;
  map<pi, int> tin;
  vector<int> queries;
  while (q--)
  {
    char t;
    cin >> t;
    if (t == ’?’)
    {
      queries.pb(++time);
    }
    else if (t == ’+’)
    {
      int a, b;
      cin >> a >> b;
      a--, b--;
      if (a > b)
        swap(a, b);
      tin[{a, b}] = ++time;
    }
    else
    {
      int a, b;
      cin >> a >> b;
      a--, b--;
      if (a > b)
        swap(a, b);
      seg::query kappa = {a, b, 0};
      seg::add(1, 0, MAXN - 1, tin[{a, b}], ++time, kappa);
      tin[{a, b}] = -1;
    }
  }
  for (auto const &i : tin)
  {
    if (i.sec != -1)
    {
      seg::query kappa = {i.fir.fir, i.fir.sec, 0};
      seg::add(1, 0, MAXN - 1, i.sec, ++time, kappa);
    }
  }
  dsu::init(n);
  seg::dfs(1, 0, MAXN - 1);
  for (auto const &i : queries)
    cout << seg::ans[i] << endl;
  return 0;
}
// https://codeforces.com/edu/course/2/lesson/7/3/practice/contest/289392/
    problem/C
// conectividade dinamica
// para uma query (u, v)
// podemos descrever em um intervalo [l, r]
// l = quando a aresta (u, v) foi adicionada
// r = quando a aresta (u, v) foi removida
// dai agora que temos um intervalo, podemos adicionar
// a query (u, v) em uma segtree "adaptada"
// no final rodamos um dfs nessa segtree e vamos atualizando as repostas das
    queries
// quando estamos em uma posicao na seg, dou union em todos os caras daquela
    posicao
// e em seguida chamo pros meus filhos, quando chego em uma folha, ela eh
    equivalente
// a uma unidade de "tempo", logo a resposta para aquele tempo eh a resposta
    atual no dsu
// e ao sair recursivamente, vou dando rollbacks no dsu
```

4.12 sack

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 100005
#define mod 1000000007

vector<int> adj[MAXN];
vector<int> v[MAXN];
int c[MAXN];
int cnt[MAXN];
int sz[MAXN];

void dfs_sz(int x, int p)
{
  sz[x] = 1;
  for (auto const &i : adj[x])
  {
    if (i != p)
    {
      dfs_sz(i, x);
      sz[x] += sz[i];
    }
  }
}
void modify(int c, int val)
{
  cnt[c] += val;
}
void dfs(int x, int p, bool keep)
{
  int best = -1, big_child = -1;
  for (auto const &i : adj[x])
  {
    if (i != p && sz[i] > best)
    {
      best = sz[i];
      big_child = i;
    }
  }
  for (auto const &i : adj[x])
  {
    if (i != p && i != big_child)
      dfs(i, x, 0);
  }
  if (big_child != -1)
  {
    dfs(big_child, x, 1);
    swap(v[x], v[big_child]); // O(1)
  }
  v[x].pb(x);
  modify(c[x], 1); // adiciona
  for (auto const &i : adj[x])
  {
    if (i != p && i != big_child)
    {
      for (auto const &j : v[i])
      {
        v[x].pb(j);
        modify(c[j], 1); // adiciona
      }
    }
  }
  // a cor c aparece cnt[c] vezes na subtree de x
  // dai vc pode fazer algo tendo essa informacao
  // seja responser queries ou algo do tipo aqui
  if (!keep)
  {
    for (auto const &i : v[x])
      modify(c[i], -1); // remove
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n;
  cin >> n;
  for (int i = 0; i < n; i++)
    cin >> c[i];
  for (int i = 0; i < n - 1; i++)
  {
    int a, b;
    cin >> a >> b;
    a--, b--;
    adj[a].pb(b);
    adj[b].pb(a);
  }
  dfs_sz(0, -1);
  dfs(0, -1, 0);
  cout << endl;
}
// https://codeforces.com/blog/entry/44351
// https://codeforces.com/blog/entry/67696

// problema motivacao:
// dada uma arvore
// cada vertice tem uma cor
// tenho consultas do tipo: quantos caras na subtree de v tem cor == x
// com sack da pra resolver isso em O(n * log(n)) (complexidade do dfs do sack)

// para outros problemas, basta mudar a funcao modify
// muito util em problemas em que vc precisa guardar algo da subarvore de v,
    para todo v
// seja pra resolver queries offline ou algo do tipo
```

4.13 two sat

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;
```

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 200005
#define mod 1000000007

struct two_sat
{
  int n;
  vector<vector<int>> g, gr;  // gr is the reversed graph
  vector<int> comp, ord, ans; // comp[v]: ID of the SCC containing node v
  vector<bool> vis;

  two_sat() {}
  two_sat(int sz)
  {
    n = sz;
    g.assign(2 * n, vector<int>());
    gr.assign(2 * n, vector<int>());
    comp.resize(2 * n);
    vis.resize(2 * n);
    ans.resize(2 * n);
  }
  void add_edge(int u, int v)
  {
    g[u].push_back(v);
    gr[v].push_back(u);
  }
  // int x, bool val: if 'val' is true, we take the variable to be x. Otherwise
       we take it to be x's complement (not x).
  void implies(int i, bool f, int j, bool g) // a -> b
  {
    add_edge(i + (f ? 0 : n), j + (g ? 0 : n));
    add_edge(j + (g ? n : 0), i + (f ? n : 0));
  }
  void add_clause_or(int i, bool f, int j, bool g) // At least one of them is
       true
  {
    add_edge(i + (f ? n : 0), j + (g ? 0 : n));
    add_edge(j + (g ? n : 0), i + (f ? 0 : n));
  }
  void add_clause_xor(int i, bool f, int j, bool g) // only one of them is true
  {
    add_clause_or(i, f, j, g);
    add_clause_or(i, !f, j, !g);
  }
  void add_clause_and(int i, bool f, int j, bool g) // both of them have the
       same value
  {
    add_clause_xor(i, !f, j, g);
  }
  void set(int i, bool f) // Set a variable
  {
    add_clause_or(i, f, i, f);
  }
  void top_sort(int u)
  {
    vis[u] = 1;
    for (auto const &v : g[u])
    {
      if (!vis[v])
        top_sort(v);
    }
    ord.push_back(u);
  }
  void scc(int u, int id)
  {
    vis[u] = 1;
    comp[u] = id;
    for (auto const &v : gr[u])
    {
      if (!vis[v])
        scc(v, id);
    }
  }
  bool solve()
  {
    fill(vis.begin(), vis.end(), 0);
    for (int i = 0; i < 2 * n; i++)
    {
      if (!vis[i])
        top_sort(i);
    }
    fill(vis.begin(), vis.end(), 0);
    reverse(ord.begin(), ord.end());
    int id = 0;
    for (const auto &v : ord)
    {
      if (!vis[v])
        scc(v, id++);
    }
    for (int i = 0; i < n; i++)
    {
      if (comp[i] == comp[i + n])
        return 0;
      ans[i] = (comp[i] > comp[i + n]) ? 1 : 0;
    }
    return 1;
  }
};
signed main()
{
}
//  https://codeforces.com/blog/entry/92977
//  https://codeforces.com/blog/entry/16205
//  https://cp-algorithms.com/graph/2SAT.html
```

4.14 Dijkstra

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define DEBUG 1
#define MAXN 1001
#define mod 1000000007

int n, m;
vector<pi> adj[MAXN];
bool visited[MAXN];
int dist[MAXN];

void dijkstra(int s)
{
  for (int i = 0; i < n; i++)
  {
    dist[i] = INT_MAX;
    visited[i] = false;
  }
  priority_queue<pi, vector<pi>, greater<pi>> q;
  dist[s] = 0;
  q.push({dist[s], s});
  while (!q.empty())
  {
    int v = q.top().second;
    q.pop();
    if (visited[v])
      continue;
    visited[v] = true;
    for (auto const &u : adj[v])
    {
      if (dist[u.sec] > dist[v] + u.fir)
      {
        dist[u.sec] = dist[v] + u.fir;
        q.push({dist[u.sec], u.sec});
      }
    }
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> n >> m;
  for (int i = 0; i < m; i++)
  {
    int a, b, c;
    cin >> a >> b >> c;
    a--, b--;
    adj[a].pb({c, b});
    adj[b].pb({c, a});
  }
  dijkstra(0);
}
```

4.15 scc

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 500005
#define mod 1000000007

int n, m;

bool vis[MAXN];
int root[MAXN];
vector<int> order;
vector<int> roots;
vector<int> comp;
vector<vector<int>> comps;
vector<int> adj[MAXN];
vector<int> adj_rev[MAXN];
vector<int> adj_scc[MAXN];

void dfs(int v)
{
  vis[v] = true;
  for (auto const &u : adj[v])
    if (!vis[u])
      dfs(u);
  order.pb(v);
}
void dfs2(int v)
{
  comp.pb(v);
  vis[v] = true;
  for (auto const &u : adj_rev[v])
    if (!vis[u])
      dfs2(u);
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> n >> m;
  for (int i = 0; i < m; i++)
  {
    int a, b;
    cin >> a >> b;
    adj[a].pb(b);
    adj_rev[b].pb(a);
  }
  for (int i = 0; i < n; i++)
  {
    if (!vis[i])
      dfs(i);
  }
  reverse(order.begin(), order.end());
  memset(vis, false, sizeof(vis));
  for (auto const &v : order)
  {
    if (!vis[v])
    {
      comp.clear();
      dfs2(v);
      comps.pb(comp);
      // making condensation graph
      /*
      int r = comp.back();
      for (auto const &u : comp)
        root[u] = r;
      roots.push_back(r);
      */
    }
  }
  // making condensation graph
  /*
  for (int v = 0; v < n; v++)
  {
    for (auto const &u : adj[v])
    {
      int root_v = roots[v];
      int root_u = roots[u];
      if (root_u != root_v)
        adj_scc[root_v].pb(root_u);
    }
  }
  */
  // printing scc
  cout << comps.size() << endl;
  for (auto const &comp : comps)
  {
    cout << comp.size() << " ";
    for (auto const &u : comp)
      cout << u << " ";
    cout << endl;
  }
  return 0;
}
// to test: https://judge.yosupo.jp/problem/scc
```

4.16 dominator tree

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
```

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 200005
#define mod 998244353

struct dominator_tree
{
  int n, t;
  vector<vector<int>> g, tree, rg, bucket;
  vector<int> dfs_l, dfs_r, idom, sdom, prv, pre, ancestor, label, preorder;

  dominator_tree(vector<vector<int>> &adj, int source)
  {
    int n = adj.size();
    g = adj;
    tree.resize(n);
    rg.resize(n);
    bucket.resize(n);
    dfs_l.resize(n);
    dfs_r.resize(n);
    idom.resize(n);
    sdom.assign(n, -1);
    prv.resize(n);
    pre.assign(n, -1);
    ancestor.assign(n, -1);
    label.resize(n);
    build(source);
  }
  void dfs(int v)
  {
    pre[v] = ++t;
    sdom[v] = label[v] = v;
    preorder.pb(v);
    for (auto const &nxt : g[v])
    {
      if (sdom[nxt] == -1)
      {
        prv[nxt] = v;
        dfs(nxt);
      }
      rg[nxt].pb(v);
    }
  }
  int eval(int v)
  {
    if (ancestor[v] == -1)
      return v;
    if (ancestor[ancestor[v]] == -1)
      return label[v];
    int u = eval(ancestor[v]);
    if (pre[sdom[u]] < pre[sdom[label[v]]])
      label[v] = u;
    ancestor[v] = ancestor[u];
    return label[v];
  }
  void dfs2(int v)
  {
    dfs_l[v] = t++;
    for (auto const &nxt : tree[v])
    {
      dfs2(nxt);
    }
    dfs_r[v] = t++;
  }
  void build(int s)
  {
    t = 0;
    dfs(s);
    if (preorder.size() == 1)
    {
      return;
    }
    int sz = preorder.size();
    for (int i = sz - 1; i >= 1; i--)
    {
      int w = preorder[i];
      for (auto const &v : rg[w])
      {
        int u = eval(v);
        if (pre[sdom[u]] < pre[sdom[w]])
          sdom[w] = sdom[u];
      }
      bucket[sdom[w]].push_back(w);
      ancestor[w] = prv[w];
      for (auto const &v : bucket[prv[w]])
      {
        int u = eval(v);
        idom[v] = (u == v) ? sdom[v] : u;
      }
      bucket[prv[w]].clear();
    }
    for (int i = 1; i < preorder.size(); i++)
    {
      int w = preorder[i];
      if (idom[w] != sdom[w])
        idom[w] = idom[idom[w]];
      tree[idom[w]].push_back(w);
    }
    idom[s] = sdom[s] = -1;
    t = 0;
    dfs2(s);
  }
  bool dominates(int u, int v)
  {
    if (pre[v] == -1)
      return 1;
    return dfs_l[u] <= dfs_l[v] && dfs_r[v] <= dfs_r[u];
  }
};
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n, m;
  cin >> n >> m;
  vector<vector<int>> adj(n);
  for (int i = 0; i < m; i++)
  {
    int a, b;
    cin >> a >> b;
    a--, b--;
    adj[a].pb(b);
  }
  dominator_tree d(adj, 0);
  vector<int> ans;
  for (int i = 0; i < n; i++)
  {
    if (d.dominates(i, n - 1))
      ans.pb(i);
  }
  cout << ans.size() << endl;
  for (int i = 0; i < ans.size(); i++)
    cout << ans[i] + 1 << " \n"[i == ans.size() - 1];
}
// https://tanujkhattar.wordpress.com/2016/01/11/dominator-tree-of-a-directed-graph/
// https://cses.fi/problemset/task/1703/ (problema desse codigo)
// https://codeforces.com/gym/100513/problem/L
// https://codeforces.com/contest/757/problem/F

// dado um vertice source s
// dizemos que u domina w, se todos os caminhos de
// s ate w passam pelo vertice u

// dizemos que u e um dominador imediato de w se u domina w
// e todos os demais dominadores de w, dominam u

// 1 - todo vertice (tirando o source) tem um dominador
// pois o source domina todos os demais vertices

// 2 - todo vertice (tirando o source) tem exatamente um
// unico dominador imediato

// se eu crio um grafo com todas as arestas do tipo
// (dominador imediato de w) - w
// para todos os vertices w que nao sao a source
// esse grafo eh uma arvore
// e eh a dominator tree
```

4.17 flow with minimum capacities

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 500001
#define mod 1000000007
#define INF 1e9

struct edge
{
  int to, from, flow, capacity;
};
struct dinic
{
  int n, src, sink;
  vector<vector<edge>> adj;
  vector<int> level;
  vector<int> ptr;

  dinic(int sz)
  {
    n = sz;
    adj.resize(n);
    level.resize(n);
    ptr.resize(n);
  }
  void add_edge(int a, int b, int c)
  {
    adj[a].pb({b, (int)adj[b].size(), c, c});
    adj[b].pb({a, (int)adj[a].size() - 1, 0, 0});
  }
  bool bfs()
  {
    level.assign(n, -1);
    level[src] = 0;
    queue<int> q;
    q.push(src);
    while (!q.empty())
    {
      int u = q.front();
      q.pop();
      for (auto at : adj[u])
      {
        if (at.flow && level[at.to] == -1)
        {
          q.push(at.to);
          level[at.to] = level[u] + 1;
        }
      }
    }
    return level[sink] != -1;
  }
  int dfs(int u, int flow)
  {
    if (u == sink || flow == 0)
      return flow;
    for (int &p = ptr[u]; p < adj[u].size(); p++)
    {
      edge &at = adj[u][p];
      if (at.flow && level[u] == level[at.to] - 1)
      {
        int kappa = dfs(at.to, min(flow, at.flow));
        at.flow -= kappa;
        adj[at.to][at.from].flow += kappa;
        if (kappa != 0)
          return kappa;
      }
    }
    return 0;
  }
  int run()
  {
    int max_flow = 0;
    while (bfs())
    {
      ptr.assign(n, 0);
      while (1)
      {
        int flow = dfs(src, INF);
        if (flow == 0)
          break;
        max_flow += flow;
      }
    }
    return max_flow;
  }
  vector<pii> cut_edges() // arestas do corte minimo
  {
    bfs();
    vector<pii> ans;
    for (int i = 0; i < n; i++)
    {
      for (auto const &j : adj[i])
      {
        if (level[i] != -1 && level[j.to] == -1 && j.capacity > 0)
          ans.pb({j.capacity, {i, j.to}});
      }
    }
    return ans;
  }
};

int dx[] = {1, -1, 0, 0};
int dy[] = {0, 0, 1, -1};

signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int h, w;
  cin >> h >> w;
  vector<string> v(h);
  for (int i = 0; i < h; i++)
  {
    cin >> v[i];
  }
  vector<pi> s[2];
  int cnt = 0;
  for (int i = 0; i < h; i++)
  {
    for (int j = 0; j < w; j++)
    {
      if (v[i][j] != '1')
        s[(i + j) % 2].pb({i, j});
      else if (v[i][j] == '2')
        cnt++;
    }
  }
  for (int i = 0; i < 2; i++)
  {
    sort(s[i].begin(), s[i].end());
  }
  int sz = s[0].size() + s[1].size() + 4;
  int src = s[0].size() + s[1].size();
  int src1 = s[0].size() + s[1].size() + 1;
  int sink = s[0].size() + s[1].size() + 2;
  int sink1 = s[0].size() + s[1].size() + 3;
  dinic d(sz);
  auto add_edge = [&](int a, int b, int r) // a quantidade de fluxo na aresta tem que ser <= r
  {
    d.add_edge(a, b, r);
  };
  auto add_edge2 = [&](int a, int b, int l, int r) // a quantidade de fluxo na aresta tem que estar em [l, r]
  {
    d.add_edge(a, b, r - l);
    d.add_edge(src1, b, l);
    d.add_edge(a, sink1, l);
  };
  for (int x = 0; x < s[0].size(); x++)
  {
    int i = s[0][x].fir, j = s[0][x].sec;
    if (v[i][j] == '2')
      add_edge2(src, x, 1, 1);
    else
      add_edge(src, x, 1);
  }
  for (int x = 0; x < s[1].size(); x++)
  {
    int i = s[1][x].fir, j = s[1][x].sec;
    if (v[i][j] == '2')
      add_edge2(s[0].size() + x, sink, 1, 1);
    else
      add_edge(s[0].size() + x, sink, 1);
  }
  for (int x = 0; x < s[0].size(); x++)
  {
    for (int d = 0; d < 4; d++)
    {
      pi curr = {s[0][x].fir + dx[d], s[0][x].sec + dy[d]};
      if (binary_search(s[1].begin(), s[1].end(), curr))
      {
        int y = lower_bound(s[1].begin(), s[1].end(), curr) - s[1].begin();
        add_edge(x, s[0].size() + y, 1);
      }
    }
  }
  // preciso tentar passar o fluxo desses 4 jeitos, e na ordem
  d.src = src1, d.sink = sink1;
  int i = d.run();
  d.src = src1, d.sink = sink;
  int j = d.run();
  d.src = src, d.sink = sink1;
  int k = d.run();
  d.src = src, d.sink = sink;
  int l = d.run();
  bool ok = 1;
  // pra poder checar se existe um jeito de passar fluxo
  // que satisfaz todas as constraints
  for (int i = 0; i < sz; i++)
  {
    for (auto const &j : d.adj[i])
    {
      if (i == src1 || j.to == sink1)
        ok &= (j.flow == 0);
    }
  }
  // e pra uma aresta com a restricao de [l, r]
  // se eu olhar quanto de fluxo foi passado na aresta com capacidade r - l que foi criada pra ela
  // ai tenho que foi passado l + (essa quantidade)
  (ok) ? cout << "Yes\n" : cout << "No\n";
  return 0;
}
// problema exemplo
// https://atcoder.jp/contests/abc285/tasks/abc285_g
```

```
// as celulas com 1 eu posso ignorar
// agr pras celulas com 2, eu preciso achar um matching dela com alguem
// so considerando os 2 e as ?

// entao a missao vira achar um matching (nao necessariamente maximo)
// mas que englobe todos os 2
// pode ter 2 de um lado e pode ter 2 do outro

// e se eu pudesse adicionar a seguinte constraint para algumas arestas:
// a quantidade de fluxo passada naquela aresta tem que ser entre [l, r]
// Maximum flow problem with minimum capacities
// ai da pra dale em resolver
```

4.18 centroid decomposition

```
// centroid de uma arvore -> e um no que ao ser removido da arvore, separaria as
// arvores resultantes de modo com que a maior arvore desse conjunto teria no maximo
// (n / 2) nos, sendo n o numero de nos da arvore. Para qualquer arvore com n nos,
// o centroid sempre existe.

// /////////////////////////////////////////////////////////////////////////////////////

// centroid decomposition -> muito util para tentar diminuir a complexidade em certos
// tipos de consultas a serem feitas, uma maneira melhor de organizar a arvore.

// algoritimo:
// 1) o centroid e a raiz dessa nova arvore
// 2) achar o centroid das arvores menores que surgiram com a remocao do centroid "pai"
// 3) por uma aresta entre o centroid "filho" e o centroid "pai"
// 4) repetir isso ate todos os nos serem removidos
// 5) ao final teremos a centroid tree

#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define PI acos(-1)
#define pb push_back
#define int long long int
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define DEBUG 0
#define MAXN 100001
#define mod 1000000007

int n;
vector<int> adj[MAXN];

namespace cd
{
  int sz;
  vector<int> adjl[MAXN];
  vector<int> father, subtree_size;
  vector<bool> visited;

  void dfs(int s, int f)
  {
    sz++;
    subtree_size[s] = 1;
    for (auto const &v : adj[s])
    {
      if (v != f && !visited[v])
      {
        dfs(v, s);
        subtree_size[s] += subtree_size[v];
      }
    }
  }
  int getCentroid(int s, int f)
  {
    bool is_centroid = true;
    int heaviest_child = -1;
    for (auto const &v : adj[s])
    {
      if (v != f && !visited[v])
      {
        if (subtree_size[v] > sz / 2)
          is_centroid = false;
        if (heaviest_child == -1 || subtree_size[v] > subtree_size[
             heaviest_child])
          heaviest_child = v;
      }
    }
    return (is_centroid && sz - subtree_size[s] <= sz / 2) ? s : getCentroid(
         heaviest_child, s);
  }
  int decompose_tree(int s)
  {
    sz = 0;
    dfs(s, s);
    int cend_tree = getCentroid(s, s);
    visited[cend_tree] = true;
    for (auto const &v : adj[cend_tree])
    {
      if (!visited[v])
      {
        int cend_subtree = decompose_tree(v);
        adjl[cend_tree].pb(cend_subtree);
        adjl[cend_subtree].pb(cend_tree);
        father[cend_subtree] = cend_tree;
      }
    }
    return cend_tree;
  }
  void init()
  {
    subtree_size.resize(n);
    visited.resize(n);
    father.assign(n, -1);
    decompose_tree(0);
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> n;
  for (int i = 0; i < n - 1; i++)
  {
    int a, b;
    cin >> a >> b;
    a--, b--;
    adj[a].pb(b);
    adj[b].pb(a);
  }
  cd::init();
  return 0;
}
```

4.19 TreeDiameter

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

#define PI acos(-1)
#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define MAXN 100001
#define mod 1000000007

int diameter, best;
vector<int> adj[MAXN];
bool visited[MAXN];

void dfs(int s, int c)
{
  if (c > diameter)
  {
    diameter = c;
    best = s;
  }
  visited[s] = true;
  for (auto const &i : adj[s])
    if (!visited[i])
      dfs2(i, c + 1);
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int q;
  cin >> q;
  while (q--)
  {
    int n;
    cin >> n;
    for (int i = 0; i < n; i++)
      adj[i].clear();
    for (int i = 0; i < n - 1; i++)
    {
      int a, b;
      cin >> a >> b;
      a--, b--;
      adj[b].pb(a);
      adj[a].pb(b);
    }
    diameter = 0, best = 0;
    memset(visited, false, sizeof(visited));
    dfs(1, 0);                           // achar o vertice mais distante a partir
         do vertice 0
    memset(visited, false, sizeof(visited));
    dfs(best, 0);                       // achar o mais distante a partir do
         primeiro vertice que achamos
    cout << diameter << endl;
  }
  return 0;
}
```

4.20 virtual tree

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define pci pair<char, int>
#define fir first
#define sec second
#define MAXN 200005
```

```
#define mod 998244353

int n, ans;
vector<int> adj[MAXN];
vector<int> with_color[MAXN];
vector<pi> virt[MAXN];
int a[MAXN];
int dp[MAXN];
int dp2[MAXN];
int pot[MAXN];

namespace lca
{
  int l, timer;
  vector<int> tin, tout, depth;
  vector<vector<int>> up;

  void dfs(int v, int p)
  {
    tin[v] = ++timer;
    up[v][0] = p;
    for (int i = 1; i <= l; i++)
      up[v][i] = up[up[v][i - 1]][i - 1];
    for (auto const &u : adj[v])
    {
      if (p == u)
        continue;
      depth[u] = depth[v] + 1;
      dfs(u, v);
    }
    tout[v] = ++timer;
  }
  bool is_ancestor(int u, int v)
  {
    return tin[u] <= tin[v] && tout[u] >= tout[v];
  }
  int find_lca(int u, int v)
  {
    if (is_ancestor(u, v))
      return u;
    if (is_ancestor(v, u))
      return v;
    for (int i = l; i >= 0; i--)
      if (!is_ancestor(up[u][i], v))
        u = up[u][i];
    return up[u][0];
  }
  void init()
  {
    tin.resize(n);
    tout.resize(n);
    depth.resize(n);
    timer = 0;
    l = ceil(log2(n));
    up.assign(n, vector<int>(l + 1));
    dfs(0, 0);
  }
  int dist(int s, int v)
  {
    int at = find_lca(s, v);
    return (depth[s] + depth[v] - 2 * depth[at]);
  }
}

// dp naive (fazer O(n) para cada cor)
// fixa uma cor c e faz uma dp on tree pra calcular
// quantas subtrees tem tal que todas as folhas sao da cor c
// e chamando o dfs saindo de qualquer vertice
// void dfs(int s, int p, int c)
// {
//   // dado que eu calculo o dp2[i] para cada filho
//   // para cada possivel subset (i1, i2, ..., ik) nao vazio de filhos
//   // acha o valor de dp2[i1] * dp2[i2] * ... * dp2[ik]
//   // no final eu quero a soma de todos esses valores, isso vai tar nessa
     variavel prod
//   int prod = 1;
//   for (auto const &i : adj[s])
//   {
//     if (i != p)
//     {
//       dfs(i, s, c);
//       prod = (prod * (dp2[i] + 1)) % mod;
//     }
//   }
//   prod = (prod - 1 + mod) % mod;
//   dp[s] = prod;
//   dp2[s] = prod;
//   if (a[s] == c)
//   {
//     dp[s] = (dp[s] + 1) % mod;
//     dp2[s] = (dp2[s] + 1) % mod;
//   }
//   for (auto const &i : adj[s])
//   {
//     if (i == p)
//       continue;
//     if (a[s] != c)
//       dp[s] = (dp[s] - dp2[i] + mod) % mod;
//   }
// }

// virtual tree
// dado um conjunto de vertices v
// montar uma arvore comprimida
// tal que escolhendo dois vertices do conjunto v[i] e v[j]
// lca(v[i], v[j]) tambem ta na arvore
// se o conjunto v tem k vertices
// entao a arvore comprimida tem menos do que 2k vertices
// O(k log(k)), sem considerar a complexidade de achar lca
int build_virt(vector<int> v)
{
  auto cmp = [&](int i, int j)
  {
    return lca::tin[i] < lca::tin[j];
  };
  sort(v.begin(), v.end(), cmp);
  for (int i = v.size() - 1; i > 0; i--)
  {
    v.pb(lca::find_lca(v[i], v[i - 1]));
  }
  sort(v.begin(), v.end(), cmp);
  v.erase(unique(v.begin(), v.end()), v.end());
  for (int i = 0; i < v.size(); i++)
  {
    virt[v[i]].clear();
  }
  for (int i = 1; i < v.size(); i++)
  {
    virt[lca::find_lca(v[i - 1], v[i])].clear();
  }
  for (int i = 1; i < v.size(); i++)
  {
    int parent = lca::find_lca(v[i - 1], v[i]);
    int d = lca::dist(parent, v[i]);
    virt[parent].pb({v[i], d});
  }
  return v[0];
}
void dfs(int s, int c) // dp naive, so que fazer isso na virtual tree
{
  int prod = 1;
  for (auto const &i : virt[s])
  {
    dfs(i.fir, c);
    prod = (prod * (dp2[i.fir] + 1)) % mod;
  }
  prod = (prod - 1 + mod) % mod;
  dp[s] = prod;
  dp2[s] = prod;
  if (a[s] == c)
  {
    dp[s] = (dp[s] + 1) % mod;
    dp2[s] = (dp2[s] + 1) % mod;
  }
  for (auto const &i : virt[s])
  {
    if (a[s] != c)
      dp[s] = (dp[s] - dp2[i.fir] + mod) % mod;
  }
  ans = (ans + dp[s]) % mod;
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  pot[0] = 1;
  for (int i = 1; i < MAXN; i++)
  {
    pot[i] = (pot[i - 1] * 2) % mod;
  }
  cin >> n;
  for (int i = 0; i < n; i++)
  {
    cin >> a[i];
    with_color[a[i]].pb(i);
  }
  for (int i = 1; i < n; i++)
  {
    int a, b;
    cin >> a >> b;
    a--, b--;
    adj[a].pb(b);
    adj[b].pb(a);
  }
  lca::init();
  for (int i = 1; i <= n; i++)
  {
    if (with_color[i].size() > 0)
    {
      int root = build_virt(with_color[i]);
      dfs(root, i);
    }
  }
  cout << ans << endl;
}
// https://atcoder.jp/contests/abc340/tasks/abc340_g
// problema legal
// dado uma arvore com n vertices (n <= 2 * 10^5)
// cada vertice tem uma cor a[i]
// conte quantas subarvores existem tal que:
// todas as folhas nessa subarvore tem a mesma cor

// sei codar em O(n^2), rodando um dfs pra cada cor
// dai montar a virtual tree para cada cor
// e rodar a dp naive na virtual tree

// tambem resolve o https://codeforces.com/gym/103960/problem/L da sub de 2022
```

4.21 dinic

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 705
#define mod 1000000007
#define INF 1e9

struct edge
{
  int to, from, flow, capacity, id;
};
struct dinic
{
  int n, src, sink;
  vector<vector<edge>> adj;
  vector<int> level;
  vector<int> ptr;

  dinic(int sz)
  {
    n = sz;
    adj.resize(n);
    level.resize(n);
    ptr.resize(n);
  }
  void add_edge(int a, int b, int c, int id)
  {
    adj[a].pb({b, (int)adj[b].size(), c, c, id});
    adj[b].pb({a, (int)adj[a].size() - 1, 0, 0, id});
  }
  bool bfs()
  {
    level.assign(n, -1);
    level[src] = 0;
    queue<int> q;
    q.push(src);
    while (!q.empty())
    {
      int u = q.front();
      q.pop();
      for (auto at : adj[u])
      {
        if (at.flow && level[at.to] == -1)
        {
          q.push(at.to);
          level[at.to] = level[u] + 1;
        }
      }
    }
    return level[sink] != -1;
  }
  int dfs(int u, int flow)
  {
    if (u == sink || flow == 0)
      return flow;
    for (int &p = ptr[u]; p < adj[u].size(); p++)
    {
      edge &at = adj[u][p];
      if (at.flow && level[u] == level[at.to] - 1)
      {
        int kappa = dfs(at.to, min(flow, at.flow));
        at.flow -= kappa;
        adj[at.to][at.from].flow += kappa;
        if (kappa != 0)
          return kappa;
      }
    }
    return 0;
  }
  int run()
  {
    int max_flow = 0;
    while (bfs())
    {
      ptr.assign(n, 0);
      while (1)
      {
        int flow = dfs(src, INF);
        if (flow == 0)
          break;
        max_flow += flow;
      }
    }
    return max_flow;
  }
  vector<pii> cut_edges() // arestas do corte minimo
  {
    bfs();
    vector<pii> ans;
    for (int i = 0; i < n; i++)
    {
      for (auto const &j : adj[i])
      {
        if (level[i] != -1 && level[j.to] == -1 && j.capacity > 0)
          ans.pb({j.capacity, {i, j.to}});
      }
    }
    return ans;
  }
  vector<int> flow_edges(int n, int m) // fluxo em cada aresta, na ordem da
      entrada
  {
    vector<int> ans(m);
    for (int i = 0; i < n; i++)
    {
      for (auto const &j : adj[i])
        if (!j.capacity)
          ans[j.id] = j.flow;
    }
    return ans;
  }
};
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n, m;
  cin >> n >> m;
  dinic d(n);
  for (int i = 0; i < m; i++)
  {
    int a, b, c;
    cin >> a >> b >> c;
    a--, b--;
    d.add_edge(a, b, c, i);
  }
  d.src = 0, d.sink = n - 1;
  cout << d.run() << endl;
  vector<int> ans = d.flow_edges(n, m);
  for (auto const &i : ans)
    cout << i << endl;
  return 0;
}
```

4.22 hall theorem

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, pi>
#define fir first
#define sec second
#define MAXN 1000006
#define mod 1000000007

int cnt[6];
int cnt_mask[1 << 6];
int tot_mask[1 << 6];

bool halls()
{
  for (int mask = 1; mask < (1 << 6); mask++)
  {
    int x = 0;
    for (int i = 0; i < 6; i++)
    {
      if (mask & (1 << i))
        x += cnt[i];
    }
    if (x < tot_mask[mask])
      return 0;
  }
  return 1;
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  string s;
  cin >> s;
  for (auto const &i : s)
  {
    cnt[i - 'a']++;
  }
  int n = s.size();
  vector<int> mask(n, (1 << 6) - 1); // a mask que diz quais chars nao podem
      aparecer naquela posicao
  int m;
  cin >> m;
  while (m--)
  {
    int i;
    string t;
    cin >> i >> t;
    i--;
    mask[i] = 0;
    for (auto const &j : t)
      mask[i] |= (1 << (j - 'a'));
  }
  for (auto const &i : mask)
  {
    cnt_mask[i]++;
  }
  for (int m = 0; m < (1 << 6); m++)
  {
    for (int s = m; s; s = (s - 1) & m) // soma dos cnt_mask de todas as
        submasks
    {
      tot_mask[m] += cnt_mask[s];
    }
  }
  if (!halls())
  {
    cout << "Impossible\n";
    return 0;
  }
  for (int i = 0; i < n; i++)
  {
    for (int m = 0; m < (1 << 6); m++)
    {
      if ((m & mask[i]) == mask[i])
        tot_mask[m]--;
    }
    for (int j = 0; j < 6; j++)
    {
      if ((mask[i] & (1 << j)) && cnt[j] > 0)
      {
        cnt[j]--;
        if (halls()) // faz s[i] = j e ve se o matching continua a existir
        {
          cout << (char)(j + 'a');
          break;
        }
        cnt[j]++;
      }
    }
  }
  cout << endl;
  return 0;
}
// https://codeforces.com/contest/1009/problem/G
// problema bem legal, que usa o teorema de hall
// dada uma string s, no qual cada char eh a, b, c, d, e ou f
// eu quero permutar essa string s de alguma forma, tal que a seguinte condicao
     eh satisfeita para todo indice i:
```

```
// cada indice tem um set de chars que sao proibidos de estar naquela posicao
// se existir multiplas solucoes, printe a menor string lexicograficamente que
    puder ser a resposta
```

4.23 Kruskal

```
// Algoritimo de kruskal - Achar a mst

// 1 - listar todas as arestas em ordem crescente.

// 2 - Cada aresta liga dois vertices x e y, checar se eles ja estao na mesma
    componente conexa
// (aqui, consideramos apenas as arestas ja colocadas na arvore).

// 3 - Se x e y estao na mesma componente, ignoramos a aresta e continuamos o
    procedimento
// (se a usassemos, formariamos um ciclo). Se estiverem em componentes distintas
    , colocamos a aresta
//na arvore e continuamos o procedimento.

// OBS: como a prioridade eh ordenar pelas menores distancias, basta botar o
    custo da aresta como
// first no vector das arestas para poder ordenar

// em suma: ordeno as arestas em ordem crescente com prioridade no custo, depois
     para cada aresta,
// se o find(x) != find(y) sendo x e y os vertices das arestas, eu adiciono eles
     a mst e dou um join
// nos dois, como as arestas tao ordenadas em ordem crescente, o primeiro que eu
     pego
// eh necessariamente a melhor opcao e assim a mst eh formada.

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

#define lli long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define mp make_pair
#define fir first
#define sec second
#define MAXN 100001

int n, m, a, b, c;
vector<pii> ar;
vector<pii> mst;
int pai[MAXN];
int peso[MAXN];

int find(int x)
{
  if (pai[x] == x)
  {
    return x;
  }
  return pai[x] = find(pai[x]);
}
void join(int a, int b)
{
  a = find(a);
  b = find(b);

  if (peso[a] < peso[b])
  {
    pai[a] = b;
  }
  else if (peso[b] < peso[a])
  {
    pai[b] = a;
  }
  else
  {
    pai[a] = b;
    peso[b]++;
  }
}
void initialize()
{
  for (int i = 1; i <= n; i++)
  {
    pai[i] = i;
  }
}
int main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);

  cin >> n >> m;

  for (int i = 0; i < m; i++)
  {
    cin >> a >> b >> c;
    ar.pb(mp(c, mp(a, b)));
  }

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

  initialize();

  int size = 0;

  for (int i = 0; i < m; i++)
  {
    if (find(ar[i].sec.fir) != find(ar[i].sec.sec))
    {
      join(ar[i].sec.fir, ar[i].sec.sec);
      mst.pb(mp(ar[i].fir, mp(ar[i].sec.fir, ar[i].sec.sec)));
    }
  }

  for (int i = 0; i < mst.size(); i++)
  {
    cout << mst[i].sec.fir << " " << mst[i].sec.sec << " " << mst[i].fir << endl
        ;
  }

  return 0;
}
```

4.24 hungarian

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 505
#define mod 998244353

struct hungarian
{
  int n, inf;
  vector<vector<int>> a;
  vector<int> u, v, p, way;

  hungarian(int n_) : n(n_), u(n + 1), v(n + 1), p(n + 1), way(n + 1)
  {
    a = vector<vector<int>>(n, vector<int>(n));
    inf = numeric_limits<int>::max();
  }
  void add_edge(int x, int y, int c)
  {
    a[x][y] = c;
  }
  pair<int, vector<int>> run()
  {
    for (int i = 1; i <= n; i++)
    {
      p[0] = i;
      int j0 = 0;
      vector<int> minv(n + 1, inf);
      vector<int> used(n + 1, 0);
      do
      {
        used[j0] = true;
        int i0 = p[j0], j1 = -1;
        int delta = inf;
        for (int j = 1; j <= n; j++)
        {
          if (!used[j])
          {
            int cur = a[i0 - 1][j - 1] - u[i0] - v[j];
            if (cur < minv[j])
              minv[j] = cur, way[j] = j0;
            if (minv[j] < delta)
              delta = minv[j], j1 = j;
          }
        }
        for (int j = 0; j <= n; j++)
        {
          if (used[j])
            u[p[j]] += delta, v[j] -= delta;
          else
            minv[j] -= delta;
        }
        j0 = j1;
      } while (p[j0] != 0);
      do
      {
        int j1 = way[j0];
        p[j0] = p[j1];
        j0 = j1;
      } while (j0);
    }
    vector<int> ans(n);
    for (int j = 1; j <= n; j++)
      ans[p[j] - 1] = j - 1;
    return make_pair(-v[0], ans);
  }
};
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n;
  cin >> n;
  vector<vector<int>> c(n, vector<int>(n));
  hungarian h(n);
  for (int i = 0; i < n; i++)
  {
    for (int j = 0; j < n; j++)
    {
      cin >> c[i][j];
      h.add_edge(i, j, c[i][j]);
    }
  }
  cout << h.run().fir << endl;
  return 0;
}
```

4.25 BFS

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define DEBUG 1
#define MAXN 1001
#define mod 1000000007

int n, m;
vector<int> adj[MAXN];
bool visited[MAXN];

void bfs(int s)
{
  queue<int> q;
  q.push(s);
  while (!q.empty())
  {
    int v = q.front();
    q.pop();
    if (visited[v])
      continue;
    visited[v] = true;
    for (auto const &u : adj[v])
      if (!visited[u])
        q.push(u);
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> n >> m;
  for (int i = 0; i < m; i++)
  {
    int a, b, c;
    cin >> a >> b >> c;
    a--, b--;
    adj[a].pb(b);
    adj[b].pb(a);
  }
  bfs(0);
}
```

4.26 hld edge

```
//https://www.spoj.com/problems/QTREE/
//Don't use cin/cout in this problem (gives TLE)
#include <bits/stdc++.h>
using namespace std;

#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define DEBUG 0
#define MAXN 10001
#define mod 1000000007

int n;
vector<pi> adj[MAXN];
vector<pi> edges;

namespace seg
```

```
{
  int seg[4 * MAXN];
  int lazy[4 * MAXN];
  int v[MAXN];

  int single(int x)
  {
    return x;
  }
  int neutral()
  {
    return -1;
  }
  int merge(int a, int b)
  {
    return max(a, b);
  }
  void add(int i, int l, int r, int diff)
  {
    seg[i] = (r - l + 1) * diff;
    if (l != r)
    {
      lazy[i << 1] = diff;
      lazy[(i << 1) | 1] = diff;
    }
    lazy[i] = -1;
  }
  void update(int i, int l, int r, int ql, int qr, int diff)
  {
    if (lazy[i] != -1)
      add(i, l, r, lazy[i]);
    if (l > r || l > qr || r < ql)
      return;
    if (l >= ql && r <= qr)
    {
      add(i, l, r, diff);
      return;
    }
    int mid = (l + r) >> 1;
    update(i << 1, l, mid, ql, qr, diff);
    update((i << 1) | 1, mid + 1, r, ql, qr, diff);
    seg[i] = merge(seg[i << 1], seg[(i << 1) | 1]);
  }
  int query(int l, int r, int ql, int qr, int i)
  {
    if (lazy[i] != -1)
      add(i, l, r, lazy[i]);
    if (l > r || l > qr || r < ql)
      return neutral();
    if (l >= ql && r <= qr)
      return seg[i];
    int mid = (l + r) >> 1;
    return merge(query(l, mid, ql, qr, i << 1), query(mid + 1, r, ql, qr, (i <<
         1) | 1));
  }
  void build(int l, int r, int i)
  {
    if (l == r)
    {
      seg[i] = single(v[l]);
      lazy[i] = -1;
      return;
    }
    int mid = (l + r) >> 1;
    build(l, mid, i << 1);
    build(mid + 1, r, (i << 1) | 1);
    seg[i] = merge(seg[i << 1], seg[(i << 1) | 1]);
    lazy[i] = -1;
  }
} // namespace seg
namespace hld
{
  int cur_pos;
  vector<int> parent, depth, heavy, head, pos, sz, up;

  int dfs(int s)
  {
    int size = 1, max_c_size = 0;
    for (auto const &c : adj[s])
    {
      if (c.fir != parent[s])
      {
        parent[c.fir] = s;
        depth[c.fir] = depth[s] + 1;
        int c_size = dfs(c.fir);
        size += c_size;
        if (c_size > max_c_size)
          max_c_size = c_size, heavy[s] = c.fir;
      }
    }
    return sz[s] = size;
  }
  void decompose(int s, int h)
  {
    head[s] = h;
    pos[s] = cur_pos++;
    seg::v[pos[s]] = up[s];
    for (auto const &c : adj[s])
    {
      if (c.fir != parent[s] && c.fir == heavy[s])
      {
        up[c.fir] = c.sec;
        decompose(heavy[s], h);
      }
    }
    for (auto const &c : adj[s])
    {
      if (c.fir != parent[s] && c.fir != heavy[s])
      {
        up[c.fir] = c.sec;
        decompose(c.fir, c.fir);
      }
    }
  }
  void init()
  {
    parent.assign(MAXN, -1);
    depth.assign(MAXN, -1);
    heavy.assign(MAXN, -1);
    head.assign(MAXN, -1);
    pos.assign(MAXN, -1);
    sz.assign(MAXN, 1);
    up.assign(MAXN, 0);
    cur_pos = 0;
    dfs(0);
    decompose(0, 0);
    seg::build(0, n - 1, 1);
  }
  int query_path(int a, int b)
  {
    int res = -1;
    for (; head[a] != head[b]; b = parent[head[b]])
    {
      if (depth[head[a]] > depth[head[b]])
        swap(a, b);
      res = max(res, seg::query(0, n - 1, pos[head[b]], pos[b], 1));
    }
    if (depth[a] > depth[b])
      swap(a, b);
    res = max(res, seg::query(0, n - 1, pos[a] + 1, pos[b], 1));
    return res;
  }
  void update_path(int a, int b, int x)
  {
    for (; head[a] != head[b]; b = parent[head[b]])
    {
      if (depth[head[a]] > depth[head[b]])
        swap(a, b);
      seg::update(1, 0, n - 1, pos[head[b]], pos[b], x);
    }
    if (depth[a] > depth[b])
      swap(a, b);
    seg::update(1, 0, n - 1, pos[a] + 1, pos[b], x);
  }
  void update_subtree(int a, int x)
  {
    seg::update(1, 0, n - 1, pos[a] + 1, pos[a] + sz[a] - 1, x);
  }
  int query_subtree(int a, int x)
  {
    return seg::query(0, n - 1, pos[a] + 1, pos[a] + sz[a] - 1, 1);
  }
} // namespace hld
signed main()
{
  int q;
  scanf("%d", &q);
  while (q--)
  {
    scanf("%d", &n);
    for (int i = 0; i < n; i++)
      adj[i].clear();
    edges.clear();
    for (int i = 0; i < n - 1; i++)
    {
      int a, b, c;
      scanf("%d %d %d", &a, &b, &c);
      a--, b--;
      adj[a].pb({b, c});
      adj[b].pb({a, c});
      edges.pb({a, b});
    }
    hld::init();
    while (true)
    {
      char k[10];
      scanf("%s", k);
      if (k[0] == ’Q’)
      {
        int a, b;
        scanf("%d %d", &a, &b);
        a--, b--;
        printf("%d\n", hld::query_path(a, b));
      }
      else if (k[0] == ’C’)
      {
        int a, b;
        scanf("%d %d", &a, &b);
        a--;
        hld::update_path(edges[a].fir, edges[a].sec, b);
      }
      else
      {
        break;
      }
    }
  }
  return 0;
}
```

4.27 dsu

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl ’\n’
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 2001
#define mod 1000000007
struct dsu
{
  int tot;
  vector<int> parent;
  vector<int> sz;

  dsu(int n)
  {
    parent.resize(n);
    sz.resize(n);
    tot = n;
    for (int i = 0; i < n; i++)
    {
      parent[i] = i;
      sz[i] = 1;
    }
  }
  int find_set(int i)
  {
    return parent[i] = (parent[i] == i) ? i : find_set(parent[i]);
  }
  void make_set(int x, int y)
  {
    x = find_set(x), y = find_set(y);
    if (x != y)
    {
      if (sz[x] > sz[y])
        swap(x, y);
      parent[x] = y;
      sz[y] += sz[x];
      tot--;
    }
  }
};
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n;
  cin >> n;
  dsu d(n);
  int a, b;
  cin >> a >> b;
  d.make_set(a, b);
  d.find_set(a);
}
```

4.28 MatrixDijkstra

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

#define lli long long int
#define pb push_back
#define MAXN 10000000
typedef pair <int , int> pii ;

int t ;
int dist [MAXN] ;
bool visited [MAXN] ;
vector <pii> adj_list [MAXN] ;

void dijkstra (int s)
{
    dist[s] = 0 ;

    priority_queue <pii , vector<pii> , greater<pii>> q ;

    q.push(pii(dist[s], s)) ;

    while(1)
    {
        int davez = -1 ;
        int menor = INT_MAX ;
        while(!q.empty())
        {
            int atual = q.top().second ;
            q.pop() ;

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

        }

        if(davez == -1)
        {
            break ;
        }

        visited[davez] = true ;

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

            if(dist[atual] > dist[davez] + distt)
            {
                dist[atual] = dist[davez] + distt ;
                q.push(pii(dist[atual] , atual)) ;
            }
        }
    }

}
void initialize ()
{
    for (int i = 0 ; i < t ; i++)
    {
        visited[i] = false ;
        dist[i] = INT_MAX ;
    }
}
int main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);

    int n , m ;
    cin >> n >> m ;
    t = n * m ;
    char array [t] ;

    for (int i = 0 ; i < t ; i++)
    {
        cin >> array[i] ;
    }

    for (int i = 0 ; i < t ; i++)
    {
        if (i >= m && array[i] != '#')
        {
            adj_list[i].pb(pii(1 , (i - m))) ;
        }
        if (i < (n * m) - m  && array[i] != '#')
        {
            adj_list[i].pb(pii(1 , (i + m))) ;
        }
        if (i % m != 0 && array[i] != '#')
        {
            adj_list[i].pb(pii(1 , (i - 1))) ;
        }
        if ((i + 1) % m != 0  && array[i] != '#')
        {
            adj_list[i].pb(pii(1 , (i + 1))) ;
        }

    }

    int q ;
    cin >> q ;

    while (q--)
    {
        int a , b , c , d , e ;
        cin >> a >> b >> c >> d >> e ;
        a-- , b-- , c-- , d-- ;

        int index1 = (m * a) + b ;
        int index2 = (m * c) + d ;

        adj_list[index1].pb(pii(e , index2)) ;
        adj_list[index2].pb(pii(e , index1)) ;
    }

    initialize () ;

    dijkstra(0) ;

    cout << dist[t - 1] << endl ;

    return 0 ;
}
```

4.29 articulation points

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 400005
#define mod 1000000007

int n, m, timer;
vector<int> adj[MAXN];
bool is_cutpoint[MAXN];
int tin[MAXN];
int low[MAXN];
bool vis[MAXN];

void dfs(int v, int p)
{
  vis[v] = true;
  tin[v] = timer, low[v] = timer++;
  int childs = 0;
  for (auto const &u : adj[v])
  {
    if (u == p)
      continue;
    if (vis[u])
    {
      low[v] = min(low[v], tin[u]);
    }
    else
    {
      dfs(u, v);
      low[v] = min(low[v], low[u]);
      if (low[u] >= tin[v] && p != -1)
        is_cutpoint[v] = true;
      childs++;
    }
  }
  if (p == -1 && childs > 1)
```

```
    is_cutpoint[v] = true;
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> n >> m;
  for (int i = 0; i < m; i++)
  {
    int a, b;
    cin >> a >> b;
    a--, b--;
    adj[a].pb(b);
    adj[b].pb(a);
  }
  memset(tin, -1, sizeof(tin));
  memset(low, -1, sizeof(low));
  for (int i = 0; i < n; i++)
  {
    if (!vis[i])
      dfs(i, -1);
  }
  return 0;
}
```

4.30 link cut tree edge

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 100005
#define mod 1000000007

struct node
{
  int p, ch[2];
  pii val, sub;
  bool rev;
  int sz, ar;
  int lazy;
  node() {}
  node(pii v, int ar_) : p(-1), val(v), sub(v), rev(0), sz(ar_), ar(ar_), lazy
      (0)
  {
    ch[0] = ch[1] = -1;
  }
};
struct link_cut_tree
{
  vector<node> t;
  map<pair<int, int>, int> aresta;
  int sz, n;

  link_cut_tree(int nn)
  {
    t.clear();
    n = nn;
    sz = 0;
  }
  static pii neutral()
  {
    return {0, {0, 0}};
  }
  pii merge(pii a, pii b)
  {
    return max(a, b);
  }
  void prop(int x)
  {
    if (t[x].lazy)
    {
      if (t[x].ar)
        t[x].val.fir = t[x].lazy;
      t[x].sub = merge(t[x].sub, t[x].val);
      if (t[x].ch[0] + 1)
        t[t[x].ch[0]].lazy = t[x].lazy;
      if (t[x].ch[1] + 1)
        t[t[x].ch[1]].lazy = t[x].lazy;
    }
    if (t[x].rev)
    {
      swap(t[x].ch[0], t[x].ch[1]);
      if (t[x].ch[0] + 1)
        t[t[x].ch[0]].rev ^= 1;
      if (t[x].ch[1] + 1)
        t[t[x].ch[1]].rev ^= 1;
    }
    t[x].lazy = 0, t[x].rev = 0;
  }
  void update(int x)
  {
    t[x].sz = t[x].ar, t[x].sub = t[x].val;
    for (int i = 0; i < 2; i++)
    {
      if (t[x].ch[i] + 1)
      {
        prop(t[x].ch[i]);
        t[x].sz += t[t[x].ch[i]].sz;
        t[x].sub = merge(t[x].sub, t[t[x].ch[i]].sub);
      }
    }
  }
  bool is_root(int x)
  {
    return t[x].p == -1 or (t[t[x].p].ch[0] != x and t[t[x].p].ch[1] != x);
  }
  void rotate(int x)
  {
    int p = t[x].p, pp = t[p].p;
    if (!is_root(p))
      t[pp].ch[t[pp].ch[1] == p] = x;
    bool d = t[p].ch[0] == x;
    t[p].ch[!d] = t[x].ch[d], t[x].ch[d] = p;
    if (t[p].ch[!d] + 1)
      t[t[p].ch[!d]].p = p;
    t[x].p = pp, t[p].p = x;
    update(p), update(x);
  }
  int splay(int x)
  {
    while (!is_root(x))
    {
      int p = t[x].p, pp = t[p].p;
      if (!is_root(p))
        prop(pp);
      prop(p), prop(x);
      if (!is_root(p))
        rotate((t[pp].ch[0] == p) ^ (t[p].ch[0] == x) ? x : p);
      rotate(x);
    }
    return prop(x), x;
  }
  int access(int v)
  {
    int last = -1;
    for (int w = v; w + 1; update(last = w), splay(v), w = t[v].p)
      splay(w), t[w].ch[1] = (last == -1 ? -1 : v);
    return last;
  }
  void make_tree(int v, pii w = neutral(), int ar = 0)
  {
    while (t.size() <= v)
      t.pb(node(neutral(), 0));
    t[v] = node(w, ar);
  }
  int find_root(int v)
  {
    access(v), prop(v);
    while (t[v].ch[0] + 1)
      v = t[v].ch[0], prop(v);
    return splay(v);
  }
  bool conn(int v, int w)
  {
    access(v), access(w);
    return v == w ? true : t[v].p != -1;
  }
  void rootify(int v)
  {
    access(v);
    t[v].rev ^= 1;
  }
  pii query(int v, int w)
  {
    rootify(w), access(v);
    return t[v].sub;
  }
  void update(int v, int w, int x)
  {
    rootify(w), access(v);
    t[v].lazy += x;
  }
  void link_(int v, int w)
  {
    rootify(w);
    t[w].p = v;
  }
  void link(int v, int w, pii x)
  {
    int id = n + sz++;
    aresta[make_pair(v, w)] = id;
    make_tree(id, x, 1);
    link_(v, id), link_(id, w);
  }
  void cut_(int v, int w)
  {
    rootify(w), access(v);
    t[v].ch[0] = t[t[v].ch[0]].p = -1;
  }
  void cut(int v, int w)
  {
    int id = aresta[make_pair(v, w)];
    cut_(v, id), cut_(id, w);
  }
  int lca(int v, int w)
  {
    access(v);
    return access(w);
  }
};
struct dsu
{
  int tot;
  vector<int> parent;
  vector<int> sz;

  dsu(int n)
  {
    parent.resize(n);
    sz.resize(n);
    tot = n;
    for (int i = 0; i < n; i++)
    {
      parent[i] = i;
      sz[i] = 1;
    }
  }
  int find_set(int i)
  {
    return parent[i] = (parent[i] == i) ? i : find_set(parent[i]);
  }
  void make_set(int x, int y)
  {
    x = find_set(x), y = find_set(y);
    if (x != y)
    {
      if (sz[x] > sz[y])
        swap(x, y);
      parent[x] = y;
      sz[y] += sz[x];
      tot--;
    }
  }
};
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int q;
  cin >> q;
  while (q--)
  {
    int n, m, qq;
    cin >> n >> m >> qq;
    vector<pii> e;
    for (int i = 0; i < m; i++)
    {
      int a, b, c;
      cin >> a >> b >> c;
      a--, b--;
      e.pb({c, {a, b}});
    }
    sort(e.begin(), e.end());
    dsu d(n);
    link_cut_tree l(n);
    for (int i = 0; i < n; i++)
    {
      l.make_tree(i);
    }
    int cost = 0;
    for (auto const &i : e)
    {
      if (d.find_set(i.sec.fir) != d.find_set(i.sec.sec))
      {
        d.make_set(i.sec.fir, i.sec.sec);
        l.link(i.sec.fir, i.sec.sec, i);
        cost += i.fir;
      }
    }
    while (qq--)
    {
      int a, b, c;
      cin >> a >> b >> c;
      a--, b--;
      pii mx = l.query(a, b);
      if (c < mx.fir)
      {
        cost -= mx.fir;
        cost += c;
        l.cut(mx.sec.fir, mx.sec.sec);
        // l.link(a, b, {c, {a, b}}); poderia fazer assim, mas quero testar o
             update
        l.link(a, b, {0, {a, b}});
        l.update(a, b, c);
      }
      cout << cost << endl;
    }
  }
  return 0;
}
// link cut tree com peso nas arestas
// solucao para o: https://codeforces.com/gym/101047/problem/I
// problema onde e dado um grafo inicial e algumas queries
// cada query adiciona uma nova aresta nesse grafo
// e o objetivo e achar a mst apos cada adicao de aresta

// implementacao baseada na: https://github.com/brunomaletta/Biblioteca/blob/
     master/Codigo/Grafos/LCT/lctAresta.cpp

// make_tree(v) cria uma nova arvore com um um unico vertice
// rootify(v) torna v a raiz de sua arvore
// cut(u, v) apaga a aresta u, v
// link(u, v, c) adiciona a aresta de u ate v com peso c
// query(v, w) retorna a aresta de maior peso no caminho de v ate w
// update(v, w, x) faz com que as arestas do caminho de v ate w passem a ter
```

```
     peso x
// operacoes tem complexidade O(log(n)) amortizado
```

4.31 hopcroft karp

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

// #define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 1000003
#define mod 998244353
#define INF 1e9

struct hopcroft_karp
{
  vector<int> match;
  vector<int> dist;
  vector<vector<int>> adj;
  int n, m, t;

  hopcroft_karp(int a, int b)
  {
    n = a, m = b;
    t = n + m + 1;
    match.assign(t, n + m);
    dist.assign(t, 0);
    adj.assign(t, vector<int>{});
  }
  void add_edge(int u, int v)
  {
    adj[u].pb(v);
    adj[v].pb(u);
  }
  bool bfs()
  {
    queue<int> q;
    for (int u = 0; u < n; u++)
    {
      if (match[u] == n + m)
        dist[u] = 0, q.push(u);
      else
        dist[u] = INF;
    }
    dist[n + m] = INF;
    while (!q.empty())
    {
      int u = q.front();
      q.pop();
      if (dist[u] < dist[n + m])
      {
        for (auto const &v : adj[u])
        {
          if (dist[match[v]] == INF)
          {
            dist[match[v]] = dist[u] + 1;
            q.push(match[v]);
          }
        }
      }
    }
    return dist[n + m] < INF;
  }
```

```
bool dfs(int u)
{
  if (u < n + m)
  {
    for (auto const &v : adj[u])
    {
      if (dist[match[v]] == dist[u] + 1 && dfs(match[v]))
      {
        match[v] = u;
        match[u] = v;
        return true;
      }
    }
    dist[u] = INF;
    return false;
  }
  return true;
}
vector<pi> run()
{
  int cnt = 0;
  while (bfs())
    for (int u = 0; u < n; u++)
      if (match[u] == n + m && dfs(u))
        cnt++;
  vector<pi> ans;
  for (int v = n; v < n + m; v++)
    if (match[v] < n + m)
      ans.pb({match[v], v});
  return ans;
}
vector<int> mvc() // minimum vertex cover
{
  vector<pi> ans = run();
  vector<bool> vis(n + m, 0);
  for (int i = 0; i < n; i++)
  {
    if (match[i] == n + m)
    {
      queue<int> q;
      q.push(i);
      while (!q.empty())
      {
        int x = q.front();
        q.pop();
        vis[x] = 1;
        for (auto const &y : adj[x])
        {
          if (!vis[y])
          {
            vis[y] = 1;
            q.push(match[y]);
          }
        }
      }
    }
  }
  vector<int> vc;
  for (int i = 0; i < n; i++)
  {
    if (!vis[i])
      vc.pb(i);
  }
  for (int i = n; i < n + m; i++)
  {
    if (vis[i])
      vc.pb(i);
  }
  return vc;
}
vector<pi> mec() // minimum edge cover
{
  vector<pi> ans = run();
  for (int i = 0; i < n + m; i++)
  {
    if (match[i] == n + m && adj[i].size() > 0)
    {
      if (i < n)
        ans.pb({i, adj[i][0]});
        else
          ans.pb({adj[i][0], i});
      }
    }
    return ans;
  }
};

// minimum path cover on dag
// minimum set of paths such that each of the vertices belongs to exactly one
     path
vector<vector<int>> mpc(int n, vector<pi> &e)
{
  hopcroft_karp h(n, n);
  for (auto const &i : e)
    h.add_edge(i.fir, n + i.sec);
  vector<pi> mat = h.run();
  vector<int> prv(n, -1);
  vector<int> nxt(n, -1);
  for (int i = 0; i < mat.size(); i++)
  {
    nxt[mat[i].fir] = mat[i].sec - n;
    prv[mat[i].sec - n] = mat[i].fir;
  }
  vector<vector<int>> ans;
  for (int i = 0; i < n; i++)
  {
    if (prv[i] == -1 && nxt[i] == -1)
    {
      ans.pb({i});
    }
    else if (prv[i] == -1)
    {
      vector<int> curr;
      int x = i;
      while (1)
      {
        curr.pb(x);
        if (nxt[x] == -1)
          break;
        x = nxt[x];
      }
      ans.pb(curr);
    }
  }
  return ans;
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n, m;
  cin >> n >> m;
  vector<pi> e;
  for (int i = 0; i < m; i++)
  {
    int a, b;
    cin >> a >> b;
    a--, b--;
    e.pb({a, b});
  }
  vector<vector<int>> ans = mpc(n, e);
  cout << ans.size() << endl;
  for (auto const &v : ans)
  {
    for (auto const &i : v)
      cout << i + 1 << " ";
    cout << endl;
  }
  return 0;
}
```

4.32 cycle detection

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define PI acos(-1)
#define pb push_back
#define int long long int
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 205
#define MAXP 100001
#define mod 1000000007

int n, m, idx;
vector<int> cycles[MAXN];
vector<int> adj[MAXN];
int color[MAXN];
int parent[MAXN];
int ans[MAXN];

void dfs(int u, int p)
{
  if (color[u] == 2)
    return;
  if (color[u] == 1)
  {
    idx++;
    int curr = p;
    ans[curr] = idx;
    cycles[idx].pb(curr);
    while (curr != u)
    {
      curr = parent[curr];
      cycles[idx].pb(curr);
      ans[curr] = idx;
    }
    return;
  }
  parent[u] = p;
  color[u] = 1;
  for (auto const &v : adj[u])
    if (v != parent[u])
      dfs(v, u);
  color[u] = 2;
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> n >> m;
  for (int i = 0; i < m; i++)
  {
    int a, b;
    cin >> a >> b;
    a--, b--;
    adj[a].pb(b);
    adj[b].pb(a);
  }
  for (int i = 0; i < n; i++)
    if (!color[i])
      dfs(i, -1);
  cout << idx << endl;
  for (int i = 1; i <= idx; i++)
  {
    cout << cycles[i].size() << endl;
    for (auto const &j : cycles[i])
      cout << j + 1 << " ";
    cout << endl;
  }
  return 0;
}
```

4.33 eulertour

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define DEBUG 1
#define MAXN 100001
#define mod 1000000009
#define d 31

int n, idx;
vector<int> adj[MAXN];
int euler[2 * MAXN];
int entrei[MAXN];
int sai[MAXN];

void euler_tour(int s, int f)
{
  euler[idx] = s;
  entrei[s] = idx;
  idx++;
  for (auto const &v : adj[s])
  {
    if (v == f)
      continue;
    euler_tour(v, s);
  }
  euler[idx] = s;
  sai[s] = idx;
  idx++;
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n;
  cin >> n;
  for (int i = 0; i < n - 1; i++)
  {
    int a, b;
    cin >> a >> b;
    a--, b--;
    adj[a].pb(b);
    adj[b].pb(a);
  }
  euler_tour(0, -1);
  for (int i = 0; i < 2 * n; i++)
    cout << euler[i] << " ";
  cout << endl;
  return 0;
}
// euler tour of a tree
// muito util para algumas coisas
// exemplos:
// 1- soma da subarvore de v(com update)
// usando segment trees, podemos fazer uma query(entrei[v], sai[v])
// 2- LCA
// lca(u, v) = query(entrei[u], entrei[v])
// usando uma query de minimo e considerando as profundidade dos vertices
// a resposta sera o vertice de profundidade minima que encontrarmos no
     intervalo
// 3- agilidade para remover arestas/vertices/subtrees da arvore
// basta apenas tratar o segmento equivalente do jeito que for necessario
// 4- reroot a tree
// basta apenas rotacionar o euler path
```

4.34 bipartite

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 500001
#define mod 1000000007

int n, m;
vector<int> adj[MAXN];

bool is()
{
  vector<int> c(n, -1);
  bool is = 1;
  queue<int> q;
  for (int st = 0; st < n; st++)
  {
    if (c[st] == -1)
    {
      q.push(st);
      c[st] = 0;
      while (!q.empty())
      {
        int v = q.front();
        q.pop();
        for (int u : adj[v])
        {
          if (c[u] == -1)
          {
            c[u] = c[v] ^ 1;
            q.push(u);
          }
          else
          {
            is &= (c[u] != c[v]);
          }
        }
      }
    }
  }
  return is;
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> n >> m;
  for (int i = 0; i < m; i++)
  {
    int a, b;
    cin >> a >> b;
    a--, b--;
    adj[a].pb(b);
    adj[b].pb(a);
  }
  cout << is() << endl;
  return 0;
}
```

4.35 mo dsu

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 50001
#define mod 1000000007

const int block = 224;

struct query
{
  int l, r, i;
};

vector<query> queries[block];

namespace dsu
{
  struct rollback
  {
    int u, v, ranku, rankv;
  };

  int num_sets;
  int parent[MAXN];
  int rank[MAXN];
  stack<rollback> op;

  int Find(int i)
  {
    return (parent[i] == i) ? i : Find(parent[i]);
  }
  bool Union(int x, int y, bool can_rollback)
  {
    int xx = Find(x);
    int yy = Find(y);
    if (xx != yy)
    {
      num_sets--;
      if (rank[xx] > rank[yy])
        swap(xx, yy);
      if (can_rollback)
        op.push({xx, yy, rank[xx], rank[yy]});
      parent[xx] = yy;
      if (rank[xx] == rank[yy])
        rank[yy]++;
      return true;
    }
    return false;
  }
  void do_rollback()
  {
    if (op.empty())
      return;
    rollback x = op.top();
    op.pop();
    num_sets++;
    parent[x.v] = x.v;
    rank[x.v] = x.rankv;
    parent[x.u] = x.u;
    rank[x.u] = x.ranku;
  }
  void rollback_all()
  {
    while (!op.empty())
      do_rollback();
  }
  void init(int n)
  {
    for (int i = 0; i < n; i++)
    {
      parent[i] = i;
      rank[i] = 0;
    }
    num_sets = n;
  }
} // namespace dsu
bool cmp(query a, query b)
{
  return a.r < b.r;
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n, m;
  cin >> n >> m;
  vector<pi> edges(m);
  for (int i = 0; i < m; i++)
  {
    cin >> edges[i].fir >> edges[i].sec;
    edges[i].fir--;
    edges[i].sec--;
  }
  int q;
  cin >> q;
  for (int i = 0; i < q; i++)
  {
    int l, r;
    cin >> l >> r;
    l--, r--;
    queries[l / block].pb({l, r, i});
  }
  for (int i = 0; i < block; i++)
  {
    if (queries[i].size())
      sort(queries[i].begin(), queries[i].end(), cmp);
  }
  vector<int> ans(q);
  for (int i = 0; i < block; i++)
  {
    if (!queries[i].size())
      continue;
    dsu::init(n);
    int limit = (i + 1) * block;
    for (auto const &j : queries[i])
    {
      while (j.r >= limit)
        dsu::Union(edges[limit].fir, edges[limit].sec, false), limit++;
      for (int k = j.l; k <= min(((i + 1) * block) - 1, j.r); k++)
        dsu::Union(edges[k].fir, edges[k].sec, true);
      ans[j.i] = dsu::num_sets;
      dsu::rollback_all();
    }
  }
  for (auto const &i : ans)
    cout << i << endl;
  return 0;
}
// https://codeforces.com/edu/course/2/lesson/7/3/practice/contest/289392/
     problem/B
// temos que fazer algo parecido com um mo algorithm
// sendo que a operacao eh um union/rollback do dsu
// podemos aplicar a seguinte ideia:
// - separamos os queries em blocos (pelo l) de tamanho sqrt(n)
// - para cada bloco, ordenamos esse bloco em ordem crescente do r
// - com isso, em cada query eu posso fazer:
// - de l ate (limite daquele bloco) - 1, adiciono na marra, podendo dar
     rollback
// - como o r eh crescente, para os valores de r que forem maior do que o limit
     daquele block
```

```
// - eu ja posso deixar adicionado para sempre sem precisar dar rollback
// fica algo que funciona em coisas que voce pode dar rollback
```

4.36 Prim

```
// algoritimo de prim

// 1 - definir a distancia de cada vertice como infinito (similar ao dijkstra).
// 2 - definir a distancia de 0 para o source(0).
// 3 - Em cada passo, encontrar o vertice u, que ainda nao foi processado, que
     possua a menor das distancias.
// 4 - ao termino fazer a soma de todas as distancias e encontrar qual a soma
     das distancias na MST.

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

#define lli long long int
#define pb push_back
#define pii pair<int, int>
#define mp make_pair
#define MAXN 100001
#define INF 999999
#define sec second
#define fir first

int n, m, a, b, c;
vector<pii> adj[MAXN];
int dist[MAXN];
bool processed[MAXN];

void prim()
{
  for (int i = 0; i < n; i++)
  {
    dist[i] = INF;
  }

  dist[0] = 0;

  priority_queue<pii, vector<pii>, greater<pii>> q;
  q.push(pii(dist[0], 0));

  while (1)
  {
    int davez = -1;

    while (!q.empty())
    {
      int atual = q.top().sec;
      q.pop();

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

    if (davez == -1)
    {
      break;
    }

    processed[davez] = true;

    for (int i = 0; i < adj[davez].size(); i++)
    {
      int distt = adj[davez][i].fir;
      int atual = adj[davez][i].sec;

      if (dist[atual] > distt && !processed[atual])
      {
        dist[atual] = distt;
        q.push(pii(dist[atual], atual));
      }
    }
  }

  int ans = 0;

  for (int i = 0; i < n; i++)
  {
    ans += dist[i];
  }

  cout << ans << endl;
}
int main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);

  cin >> n >> m;

  for (int i = 0; i < m; i++)
  {
    cin >> a >> b >> c;
    a--;
    b--;
    adj[a].pb(mp(c, b));
    adj[b].pb(mp(c, a));
  }

  prim();

  return 0;
}
```

4.37 tree isomorfism

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 501
#define mod 1000000007

int curr_hash = 1;
map<vector<int>, int> mp;

struct hash_tree
{
  pi h;
  int n;
  vector<int> c, sz, large_comp;
  vector<vector<int>> adj;

  hash_tree(vector<vector<int>> &a)
  {
    n = a.size();
    adj = a;
  }
  void dfs(int s, int p)
  {
    sz[s] = 1;
    large_comp[s] = 0;
    for (auto const &v : adj[s])
    {
      if (v != p)
      {
        dfs(v, s);
        sz[s] += sz[v];
        large_comp[s] = max(large_comp[s], sz[v]);
      }
    }
    large_comp[s] = max(large_comp[s], n - sz[s]);
  }
  int dfs2(int s, int p)
  {
    if (s == -1)
      return -1;
    vector<int> child;
    for (auto const &v : adj[s])
    {
      if (v != p)
        child.pb(dfs2(v, s));
    }
    sort(child.begin(), child.end());
    if (!mp[child])
      mp[child] = curr_hash++;
    return mp[child];
  }
  pi get_hash()
  {
    sz.assign(n, 0);
    large_comp.assign(n, 0);
    dfs(0, -1);
    int best = 1e18;
    for (int i = 0; i < n; i++)
    {
      if (large_comp[i] < best)
      {
        best = large_comp[i];
        c.clear();
      }
      if (large_comp[i] == best)
        c.pb(i);
    }
    while (c.size() < 2)
      c.pb(-1);
    h.fir = dfs2(c[0], -1);
    h.sec = dfs2(c[1], -1);
    if (h.fir > h.sec)
      swap(h.fir, h.sec);
    return h;
  }
};
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int q;
  cin >> q;
  while (q--)
  {
    int n;
    cin >> n;
    vector<vector<int>> a(n);
    vector<vector<int>> b(n);
    for (int i = 0; i < n - 1; i++)
    {
      int x, y;
      cin >> x >> y;
      x--, y--;
      a[x].pb(y);
      a[y].pb(x);
    }
    for (int i = 0; i < n - 1; i++)
    {
      int x, y;
      cin >> x >> y;
      x--, y--;
      b[x].pb(y);
      b[y].pb(x);
    }
    (hash_tree(a).get_hash() == hash_tree(b).get_hash()) ? cout << "YES\n" :
         cout << "NO\n";
  }
  return 0;
}
// https://www.spoj.com/problems/TREEISO/
// https://www.beecrowd.com.br/judge/en/problems/view/1229
// hash de arvores
// para descobrir se duas arvores sao isomorfas

// 1 - achar todos os centroides da arvore (toda arvore tem no maximo 2
     centroides)
// 2 - achar o hashing com a arvore enraizada em cada centroid
// 3 - dai o hashing da arvore eh um pair ordenado, indicando o hashing de cada
     enraizamento no centroid
```

4.38 caminhoeuleriano

```
// caminho euleriano em um grafo
// passa por todas as arestas apenas uma unica vez e percorre todas elas
// condicao de existencia:
// todos os vertices possuem grau par (ciclo euleriano) comeca e acaba no mesmo
     vertice
// ou
// apenas 2 vertices possuem grau impar, todos os outros possuem grau par ou ==
     0.
// comeca num vertice de grau impar e termina num vertice de grau impar nesse
     caso.
// solucao:
// rodar um dfs com map de visited para as arestas
// no final por o source no vector path
// ao final teremos o caminho inverso no vector path
// note que o caminho inverso tambem e um caminho valido

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

#define lli long long int
#define pb push_back
#define in insert
#define pi pair<int, int>
#define pd pair<double, int>
#define pib pair<pi, bool>
#define mp make_pair
#define fir first
#define sec second
#define MAXN 10001
#define MAXL 1000001
#define mod 1000000007

int n, m, start;
vector<int> path;
vector<int> adj[MAXN];
map<pi, bool> visited;

void dfs(int s)
{
  for (int i = 0; i < adj[s].size(); i++)
  {
    int v = adj[s][i];
    if (!visited[mp(s, v)])
    {
      visited[mp(s, v)] = true;
      visited[mp(v, s)] = true;
      dfs(v);
    }
  }
  path.pb(s);
}
bool check()
{
  int odd = 0;
  for (int i = 0; i < n; i++)
    if (adj[i].size() & 1)
      odd++, start = i;
  return (odd == 0 || odd == 2);
}
```

```
signed main()
{
  cin >> n >> m;
  for (int i = 0; i < m; i++)
  {
    int a, b;
    cin >> a >> b;
    adj[a].pb(b);
    adj[b].pb(a);
  }
  start = 0;
  bool ok = check();
  (ok) ? cout << "Yes\n" : cout << "No\n";
  if (ok)
  {
    dfs(start);
    for (int i = 0; i < path.size(); i++)
      cout << path[i] << " ";
    cout << "\n";
  }
  return 0;
}
```

4.39 hld

```
// https://codeforces.com/contest/343/problem/D
#include <bits/stdc++.h>
using namespace std;

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define DEBUG 0
#define MAXN 500001
#define mod 1000000007

struct segtree
{
  int n;
  vector<int> v;
  vector<int> seg;
  vector<int> lazy;

  segtree() {}
  segtree(int sz)
  {
    n = sz;
    seg.assign(4 * n, 0);
    lazy.assign(4 * n, -1);
  }
  int single(int x)
  {
    return x;
  }
  int neutral()
  {
    return 0;
  }
  int merge(int a, int b)
  {
    return a + b;
  }
  void add(int i, int l, int r, int diff)
  {
    seg[i] = (r - l + 1) * diff;
    if (l != r)
    {
      lazy[i << 1] = diff;
      lazy[(i << 1) | 1] = diff;
    }
    lazy[i] = -1;
  }
  void update(int i, int l, int r, int ql, int qr, int diff)
  {
    if (lazy[i] != -1)
      add(i, l, r, lazy[i]);
    if (l > r || l > qr || r < ql)
      return;
    if (l >= ql && r <= qr)
    {
      add(i, l, r, diff);
      return;
    }
    int mid = (l + r) >> 1;
    update(i << 1, l, mid, ql, qr, diff);
    update((i << 1) | 1, mid + 1, r, ql, qr, diff);
    seg[i] = merge(seg[i << 1], seg[(i << 1) | 1]);
  }
  int query(int l, int r, int ql, int qr, int i)
  {
    if (lazy[i] != -1)
      add(i, l, r, lazy[i]);
    if (l > r || l > qr || r < ql)
      return neutral();
    if (l >= ql && r <= qr)
      return seg[i];
    int mid = (l + r) >> 1;
    return merge(query(l, mid, ql, qr, i << 1), query(mid + 1, r, ql, qr, (i <<
        1) | 1));
  }
};
struct hld
{
  int n, cur_pos;
  segtree seg;
  vector<vector<int>> adj;
  vector<int> parent, depth, heavy, head, pos, sz;

  int dfs(int s)
  {
    int size = 1, max_c_size = 0;
    for (auto const &c : adj[s])
    {
      if (c != parent[s])
      {
        parent[c] = s;
        depth[c] = depth[s] + 1;
        int c_size = dfs(c);
        size += c_size;
        if (c_size > max_c_size)
          max_c_size = c_size, heavy[s] = c;
      }
    }
    return sz[s] = size;
  }
  void decompose(int s, int h)
  {
    head[s] = h;
    pos[s] = cur_pos++;
    if (heavy[s] != -1)
      decompose(heavy[s], h);
    for (int c : adj[s])
    {
      if (c != parent[s] && c != heavy[s])
        decompose(c, c);
    }
  }
  hld(vector<vector<int>> &g)
  {
    n = g.size();
    adj = g;
    seg = segtree(n);
    parent.assign(n, -1);
    depth.assign(n, -1);
    heavy.assign(n, -1);
    head.assign(n, -1);
    pos.assign(n, -1);
    sz.assign(n, 1);
    cur_pos = 0;
    dfs(0);
    decompose(0, 0);
  }
  int query_path(int a, int b)
  {
    int res = 0;
    for (; head[a] != head[b]; b = parent[head[b]])
    {
      if (depth[head[a]] > depth[head[b]])
        swap(a, b);
      res += seg.query(0, n - 1, pos[head[b]], pos[b], 1);
    }
    if (depth[a] > depth[b])
      swap(a, b);
    res += seg.query(0, n - 1, pos[a], pos[b], 1);
    return res;
  }
  void update_path(int a, int b, int x)
  {
    for (; head[a] != head[b]; b = parent[head[b]])
    {
      if (depth[head[a]] > depth[head[b]])
        swap(a, b);
      seg.update(1, 0, n - 1, pos[head[b]], pos[b], x);
    }
    if (depth[a] > depth[b])
      swap(a, b);
    seg.update(1, 0, n - 1, pos[a], pos[b], x);
  }
  void update_subtree(int a, int x)
  {
    seg.update(1, 0, n - 1, pos[a], pos[a] + sz[a] - 1, x);
  }
  int query_subtree(int a)
  {
    return seg.query(0, n - 1, pos[a], pos[a] + sz[a] - 1, 1);
  }
  int lca(int a, int b)
  {
    if (pos[a] < pos[b])
      swap(a, b);
    return (head[a] == head[b]) ? b : lca(parent[head[a]], b);
  }
};
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n;
  cin >> n;
  vector<vector<int>> adj(n);
  for (int i = 0; i < n - 1; i++)
  {
    int a, b;
    cin >> a >> b;
    a--, b--;
    adj[a].pb(b);
    adj[b].pb(a);
  }
  hld h(adj);
  int q;
  cin >> q;
  while (q--)
  {
    int a, b;
    cin >> a >> b;
    b--;
    if (a == 1)
    {
      h.update_subtree(b, 1);
    }
    if (a == 2)
    {
      h.update_path(0, b, 0);
    }
    if (a == 3)
    {
      cout << h.query_path(b, b) << endl;
    }
  }
```

```
  return 0;
}
```

4.40 caminhoeuleriano2

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 500005
#define mod 998244353

int deg[MAXN];
bool vis[MAXN];
vector<int> path;
vector<pi> edges, edges2;
vector<pi> ans;
vector<pi> adj[MAXN];
set<pi> a[MAXN];

void dfs2(int s)
{
  while (a[s].size() > 0)
  {
    auto v = (*a[s].begin());
    a[s].erase(v);
    a[v.fir].erase({s, v.sec});
    dfs2(v.fir);
  }
  path.pb(s);
}
void dfs(int i)
{
  vis[i] = 1;
  for (auto const &j : adj[i])
  {
    if (!vis[j.fir])
    {
      dfs(j.fir);
      if (deg[j.fir])
      {
        ans.pb(edges[j.sec]);
        deg[j.fir] ^= 1;
        deg[i] ^= 1;
      }
    }
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int q;
  cin >> q;
  while (q--)
  {
    int n, m;
    cin >> n >> m;
    edges.clear();
    edges2.clear();
    ans.clear();
    for (int i = 0; i < n; i++)
    {
      adj[i].clear();
      vis[i] = 0;
      deg[i] = 0;
    }
    for (int i = 0; i < m; i++)
    {
      int a, b, c;
      cin >> a >> b >> c;
      a--, b--;
      if (c == 1) // edges obrigatorios
      {
        edges2.pb({a, b});
        deg[a] ^= 1;
        deg[b] ^= 1;
      }
      else // edges nao obrigatorios
      {
        edges.pb({a, b});
        adj[a].pb({b, edges.size() - 1});
        adj[b].pb({a, edges.size() - 1});
      }
    }
    for (int i = 0; i < n; i++)
    {
      if (!vis[i])
        dfs(i);
    }
    bool ok = 1;
    for (int i = 0; i < n; i++)
    {
      if (deg[i])
        ok = 0;
    }
    if (!ok)
    {
      cout << "NO\n";
      continue;
    }
    for (int i = 0; i < n; i++)
    {
      a[i].clear();
    }
    // monta o grafo final e acha o ciclo euleriano
    // funciona para grafos com self loops e multiple edges
    int id = 0;
    for (auto [u, v] : ans)
    {
      a[u].insert({v, id});
      a[v].insert({u, id});
      id++;
    }
    for (auto [u, v] : edges2)
    {
      a[u].insert({v, id});
      a[v].insert({u, id});
      id++;
    }
    path.clear();
    dfs2(0);
    cout << "YES\n";
    cout << path.size() - 1 << endl;
    for (int i = 0; i < path.size(); i++)
      cout << path[i] + 1 << " ";
    cout << endl;
  }
  return 0;
}
// https://codeforces.com/contest/1994/problem/F
// dado um grafo, tem edges que sao obrigatorios de manter
// e outros q posso remover
// quero fazer com que um grafo tenha um ciclo euleriano
// no qual o grau de cada vertice eh par
// se tiver solucao, eu quero imprimir o ciclo euleriano
```

4.41 DFS

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

#define MAXN 500000

int n , m ;
int visited [MAXN] ;
vector <int> adj_list [MAXN] ;

void dfs (int x)
{
    for (int i = 0 ; i < adj_list[x].size() ; i++)
    {
        int v = adj_list[x][i] ;

        if(visited[v] == -1)
        {
            visited[v] = visited[x] ;
            dfs(v) ;
        }
    }
}
void initialize ()
{
    for (int i = 1 ; i <= n ; i++)
    {
        visited[i] = -1 ;
    }
}
int main ()
{
    int a , b ;

    cin >> n >> m ;

    initialize();

    for (int i = 1 ; i <= m ; i++)
    {
        cin >> a >> b ;

        adj_list[a].push_back(b) ;
        adj_list[b].push_back(a) ;
    }

    dfs(1) ;

    return 0;
}
```

4.42 block-cut-tree

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 500001
#define mod 1000000007

struct dsu
{
  vector<pi> parent;
  vector<int> rank;
  vector<int> bipartite;
  void reset(int v)
  {
    parent[v] = {v, 0};
    rank[v] = 0;
    bipartite[v] = 1;
  }
  dsu(int n)
  {
    parent.resize(n);
    rank.resize(n);
    bipartite.resize(n);
    for (int v = 0; v < n; v++)
      reset(v);
  }
  dsu() {}
  pi find_set(int v)
  {
    if (v != parent[v].fir)
    {
      int parity = parent[v].sec;
      parent[v] = find_set(parent[v].fir);
      parent[v].sec ^= parity;
    }
    return parent[v];
  }
  void add_edge(int a, int b)
  {
    pi pa = find_set(a);
    a = pa.fir;
    int x = pa.sec;
    pi pb = find_set(b);
    b = pb.fir;
    int y = pb.sec;
    if (a == b)
    {
      if (x == y)
        bipartite[a] = 0;
    }
    else
    {
      if (rank[a] < rank[b])
        swap(a, b);
      parent[b] = {a, x ^ y ^ 1};
      bipartite[a] &= bipartite[b];
      if (rank[a] == rank[b])
        rank[a]++;
    }
  }
  bool is_bipartite(int v)
  {
    return bipartite[find_set(v).fir];
  }
};
struct block_cut_tree
{
  // Source: https://github.com/brunomaletta/Biblioteca/blob/master/Codigo/
      Grafos/blockCutTree.cpp
  // Cria a block-cut tree, uma arvore com os blocos
  // e os pontos de articulacao
  // Blocos sao componentes 2-vertice-conexos maximais
  // Uma 2-coloracao da arvore eh tal que uma cor sao
  // os blocos, e a outra cor sao os pontos de art.
  // Funciona para grafo nao conexo
  //
  // art[i] responde o numero de novas componentes conexas
  // criadas apos a remocao de i do grafo g
  // Se art[i] >= 1, i eh ponto de articulacao
  //
  // Para todo i <= blocks.size()
  // blocks[i] eh uma componente 2-vertce-conexa maximal
  // edgblocks[i] sao as arestas do bloco i
  // tree[i] eh um vertice da arvore que corresponde ao bloco i
  // tree - eh a propia block-cut tree
  // pos[i] responde a qual vertice da arvore vertice i pertence
  // Arvore tem no maximo 2n vertices
  //
  // O(n + m)
  vector<vector<int>> g, blocks, tree;
  vector<vector<pi>> edgblocks;
  stack<int> s;
  stack<pi> s2;
  vector<int> id, art, pos;

  block_cut_tree(vector<vector<int>> g_) : g(g_)
  {
    int n = g.size();
    id.resize(n, -1), art.resize(n), pos.resize(n);
    build();
  }
  int dfs(int i, int &t, int p = -1)
  {
    int lo = id[i] = t++;
    s.push(i);
    if (p != -1)
    {
      s2.emplace(i, p);
    }
    for (int j : g[i])
    {
      if (j != p and id[j] != -1)
        s2.emplace(i, j);
    }
    for (int j : g[i])
    {
      if (j != p)
      {
        if (id[j] == -1)
        {
          int val = dfs(j, t, i);
          lo = min(lo, val);
          if (val >= id[i])
          {
            art[i]++;
            blocks.emplace_back(1, i);
            while (blocks.back().back() != j)
            {
              blocks.back().pb(s.top());
              s.pop();
            }
            edgblocks.emplace_back(1, s2.top());
            s2.pop();
            pi aux = {j, i};
            while (edgblocks.back().back() != aux)
            {
              edgblocks.back().pb(s2.top());
              s2.pop();
            }
          }
          // if (val > id[i]) aresta i-j eh ponte
        }
        else
        {
          lo = min(lo, id[j]);
        }
      }
    }
    if (p == -1 and art[i])
    {
      art[i]--;
    }
    return lo;
  }
  void build()
  {
    int t = 0;
    for (int i = 0; i < g.size(); i++)
    {
      if (id[i] == -1)
        dfs(i, t, -1);
    }
    tree.resize(blocks.size());
    for (int i = 0; i < g.size(); i++)
    {
      if (art[i])
        pos[i] = tree.size(), tree.emplace_back();
    }
    for (int i = 0; i < blocks.size(); i++)
    {
      for (int j : blocks[i])
      {
        if (!art[j])
          pos[j] = i;
        else
          tree[i].pb(pos[j]), tree[pos[j]].pb(i);
      }
    }
  }
};
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n, m;
  cin >> n >> m;
  vector<vector<int>> adj(n);
  for (int i = 0; i < m; i++)
  {
    int a, b;
    cin >> a >> b;
    a--, b--;
    adj[a].pb(b);
    adj[b].pb(a);
  }
  block_cut_tree bt(adj);
  vector<vector<int>> g(n);
  dsu d(n);
  for (auto const &v : bt.edgblocks)
  {
    vector<int> guys;
    for (auto const &j : v)
    {
      guys.pb(j.fir);
      guys.pb(j.sec);
      d.add_edge(j.fir, j.sec);
    }
    bool bip = 1;
    for (auto const &j : guys)
    {
      bip &= d.is_bipartite(j);
    }
    if (bip)
    {
      for (auto const &j : v)
      {
        g[j.fir].pb(j.sec);
        g[j.sec].pb(j.fir);
      }
    }
    for (auto const &j : guys)
    {
      d.reset(j);
    }
  }
  vector<bool> vis(n, 0);
  vector<bool> c(n, 0);
  int a = 0, b = 0;
  for (int i = 0; i < n; i++)
  {
    if (vis[i])
      continue;
    int x = 1, y = 0;
    queue<int> q;
    q.push(i);
    vis[i] = 1;
    while (!q.empty())
    {
      int k = q.front();
      q.pop();
      for (auto const &i : g[k])
      {
        if (!vis[i])
        {
          vis[i] = 1;
          c[i] = c[k] ^ 1;
          (c[i] == 1) ? y++ : x++;
          q.push(i);
        }
      }
    }
    a += (x * (x - 1)) / 2;
    a += (y * (y - 1)) / 2;
    b += (x * y);
  }
  cout << a << " " << b << endl;
  return 0;
}
// https://codeforces.com/gym/103934/problem/M
// pares (a, b) com a < b
// contar pares (a, b) tal que todos os caminhos de a para b possuem distancia
     impar
// contar pares (a, b) tal que todos os caminhos de a para b possuem distancia
     par

// grafo biconexo (ou 2-vertice-conexo) - nao tem ponto de articulacao
// blocos - sao subgrafos biconexos maximais (sem ponto de articulacao)

// block graph
// grafo que tem um vertice para cada bloco do grafo G
// e uma aresta entre dois vertices tal que os blocos correspondentes tem um
     vertice em comum

// block-cut tree
// um ponto de articulacao eh um vertice que esta em dois ou mais blocos
// a estrutura dos blocos e dos pontos de articulacao de um grafo conectado pode
      ser descrita por uma arvore chamada de arvore de block-cut tree
// essa arvore tem um vertice para cada bloco e para cada ponto de articulacao
     do grafo dado.
// tem uma aresta na block-cut tree para cada par (bloco, ponto de articulacao),
      tal que esse ponto de articulacao ta no bloco

// para o problema:
// para um grafo nao bipartido que e biconexo, tem caminhos de tamanho impar e
     par entre qualquer par de vertices

// um caminho em um grafo G, tem meio que um caminho equivalente na sua block-
     cut tree
// da pra pensar em resolver para cada bloco

// resolvendo pra cada bloco:
// o bloco tem que ser bipartido
// quando o bloco nao eh bipartido, eu nao considero as arestas dele

// considerando o grafo restante sendo bipartido
// da pra resolver pra cada componente conexa
// caminhos entre vertices de mesma cor tem paridade impar
// caminhos entre vertices de cor diferente tem paridade par

// https://codeforces.com/gym/102512/problem/A
// ter queries do tipo
// quantos pontos de articulacao desconectam u e v
// dai monta a block cut tree
// para cada ponto de articulacao, seta a pos[i] dele como 1 na arvore
// e o valor dos demais vertices como 0
// dai responde uma query com hld (ou com lca tbm sai)
```

4.43 mo trees edges

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
```

```cpp
#define fir first
#define sec second
#define MAXN 200001
#define mod 998244353

struct qry
{
  int l, r, ini, id;
};

int n, q;
vector<pi> adj[MAXN];
int v[MAXN];
int cnt[MAXN];
int freq[MAXN];
int in_block[MAXN];
int tin[MAXN];
int tout[MAXN];
int depth[MAXN];
int up[MAXN][25];
vector<int> t;
vector<qry> qq;

void dfs(int s, int p, int par_edge)
{
  v[s] = par_edge;
  tin[s] = t.size();
  up[s][0] = p;
  for (int i = 1; i < 25; i++)
    up[s][i] = up[up[s][i - 1]][i - 1];
  t.pb(s);
  for (auto const &i : adj[s])
  {
    if (i.fir == p)
      continue;
    depth[i.fir] = depth[s] + 1;
    dfs(i.fir, s, i.sec);
  }
  tout[s] = t.size();
  t.pb(s);
}
bool is(int u, int v)
{
  return tin[u] <= tin[v] && tout[u] >= tout[v];
}
int lca(int u, int v)
{
  if (is(u, v))
    return u;
  if (is(v, u))
    return v;
  for (int i = 24; i >= 0; i--)
  {
    if (!is(up[u][i], v))
      u = up[u][i];
  }
  return up[u][0];
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> n >> q;
  for (int i = 0; i < n - 1; i++)
  {
    int a, b, c;
    cin >> a >> b >> c;
    a--, b--;
    adj[a].pb({b, c});
    adj[b].pb({a, c});
  }
  dfs(0, 0, 0);
  for (int i = 0; i < q; i++)
  {
    int x, y;
    cin >> x >> y;
    x--, y--;
    int l = lca(x, y);
    if (tin[x] > tin[y])
      swap(x, y);
    if (l == x)
      qq.pb({tin[x], tin[y], x, i});
    else
      qq.pb({tout[x], tin[y], -1, i});
  }
  int block = sqrt(n) + 1;
  auto cmp = [&](qry x, qry y)
  {
    if (x.l / block != y.l / block)
      return x.l / block < y.l / block;
    return x.r < y.r;
  };
  sort(qq.begin(), qq.end(), cmp);
  vector<int> ans(q);
  int cl = 0, cr = 0, resp = 0;
  auto add2 = [&](int x)
  {
    if (v[x] >= MAXN)
      return;
    freq[v[x]]++;
    if (freq[v[x]] == 1)
      in_block[v[x] / block]++;
  };
  auto rem2 = [&](int x)
  {
    if (v[x] >= MAXN)
      return;
    freq[v[x]]--;
    if (freq[v[x]] == 0)
      in_block[v[x] / block]--;
  };
  auto add = [&](int x)
  {
    cnt[x]++;
    if (cnt[x] == 2)
      rem2(x);
    else
      add2(x);
  };
  auto rem = [&](int x)
  {
    cnt[x]--;
    if (cnt[x] == 1)
      add2(x);
    else
      rem2(x);
  };
  for (int i = 0; i < q; i++)
  {
    int idx = qq[i].id;
    int l = qq[i].l;
    int r = qq[i].r;
    int ini = qq[i].ini;
    while (cl < l)
      rem(t[cl++]);
    while (cl > l)
      add(t[--cl]);
    while (cr <= r)
      add(t[cr++]);
    while (cr > r + 1)
      rem(t[--cr]);
    if (ini != -1)
      rem(ini);
    for (int b = 0;; b++)
    {
      if (in_block[b] != block)
      {
        ans[idx] = b * block;
        while (freq[ans[idx]])
          ans[idx]++;
        break;
      }
    }
    if (ini != -1)
      add(ini);
  }
  for (auto const &i : ans)
    cout << i << endl;
  return 0;
}
// https://codeforces.com/gym/100962/attachments (problema F)
// mo em arvore com peso nos edges

// nesse problema em especifico: dado uma arvore, responder queries de mex
// no caminho entre u e v, considerando os pesos de arestas no caminho de u pra
     v

// mo em arvores
// acha o euler tour da arvore com tin e tout
// desconsidera no mo os indices duplicados no range
// e bem parecido com o de peso nos vertices
// considera v[i] -> peso do edge que liga ao meu pai na arvore
// dai pra query com o lca == u, nao tenho que considerar v[u] ([tin[u], tin[v
     ]], dps removendo v[u])
// e pra query com o lca != u, so fazer ela normalmente ([tout[u], tin[v]])
```

4.44 mo trees

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 500001
#define mod 998244353

struct qry
{
  int l, r, lca, id;
};

int n, q;
vector<int> adj[MAXN];
int v[MAXN];
int cnt[MAXN];
int freq[MAXN];
int tin[MAXN];
int tout[MAXN];
int depth[MAXN];
int up[MAXN][25];
vector<int> t;
vector<qry> qq;

void dfs(int s, int p)
{
  tin[s] = t.size();
  up[s][0] = p;
  for (int i = 1; i < 25; i++)
    up[s][i] = up[up[s][i - 1]][i - 1];
  t.pb(s);
  for (auto const &i : adj[s])
  {
    if (i == p)
      continue;
    depth[i] = depth[s] + 1;
    dfs(i, s);
  }
  tout[s] = t.size();
  t.pb(s);
}
bool is(int u, int v)
{
  return tin[u] <= tin[v] && tout[u] >= tout[v];
}
int lca(int u, int v)
{
  if (is(u, v))
    return u;
  if (is(v, u))
    return v;
  for (int i = 24; i >= 0; i--)
  {
    if (!is(up[u][i], v))
      u = up[u][i];
  }
  return up[u][0];
}
void compress()
{
  vector<int> vals;
  for (int i = 0; i < n; i++)
    vals.pb(v[i]);
  sort(vals.begin(), vals.end());
  vals.erase(unique(vals.begin(), vals.end()), vals.end());
  for (int i = 0; i < n; i++)
    v[i] = lower_bound(vals.begin(), vals.end(), v[i]) - vals.begin();
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  while (cin >> n >> q)
  {
    t.clear();
    qq.clear();
    depth[0] = 0;
    memset(cnt, 0, sizeof(cnt));
    memset(freq, 0, sizeof(freq));
    for (int i = 0; i < n; i++)
    {
      adj[i].clear();
      cin >> v[i];
    }
    compress();
    for (int i = 0; i < n - 1; i++)
    {
      int a, b;
      cin >> a >> b;
      a--, b--;
      adj[a].pb(b);
      adj[b].pb(a);
    }
    dfs(0, 0);
    for (int i = 0; i < q; i++)
    {
      int x, y;
      cin >> x >> y;
      x--, y--;
      int l = lca(x, y);
      if (tin[x] > tin[y])
        swap(x, y);
      if (l == x)
        qq.pb({tin[x], tin[y], -1, i});
      else
        qq.pb({tout[x], tin[y], l, i});
    }
    int block = sqrt(n) + 1;
    auto cmp = [&](qry x, qry y)
    {
      if (x.l / block != y.l / block)
        return x.l / block < y.l / block;
      return x.r < y.r;
    };
    sort(qq.begin(), qq.end(), cmp);
    vector<int> ans(q);
    int cl = 0, cr = 0, resp = 0;
    auto add2 = [&](int x)
    {
      freq[v[x]]++;
      if (freq[v[x]] == 1)
        resp++;
    };
    auto rem2 = [&](int x)
    {
      freq[v[x]]--;
      if (freq[v[x]] == 0)
        resp--;
    };
    auto add = [&](int x)
    {
      cnt[x]++;
      if (cnt[x] == 2)
        rem2(x);
      else
        add2(x);
    };
    auto rem = [&](int x)
    {
      cnt[x]--;
      if (cnt[x] == 1)
        add2(x);
      else
        rem2(x);
    };
    for (int i = 0; i < q; i++)
    {
      int idx = qq[i].id;
      int l = qq[i].l;
      int r = qq[i].r;
      int lc = qq[i].lca;
      while (cl < l)
        rem(t[cl++]);
      while (cl > l)
        add(t[--cl]);
      while (cr <= r)
        add(t[cr++]);
      while (cr > r + 1)
        rem(t[--cr]);
      if (lc != -1)
        add(lc);
      ans[idx] = resp;
      if (lc != -1)
        rem(lc);
    }
    for (auto const &i : ans)
      cout << i << endl;
  }
  return 0;
}
// https://www.spoj.com/problems/COT2/
// quantos caras distintos em um path entre u e v
// mo em arvores
// acha o euler tour da arvore com tin e tout
// desconsidera no mo os indices duplicados no range

// para queries em subtree eh mais simples:
// apenas saber o tamanho da subtree de i
// fazer o euler tour apenas com o tin
// e fzr a query pro range tin[i] ate tin[i] + sz[i] - 1

// pra queries de path com peso nos edges:
// https://codeforces.com/gym/100962/attachments (problema F)
// considera v[i] -> peso do edge que liga ao meu pai na arvore
// dai pra query com o lca == u, nao tenho que considerar v[u] ([tin[u], tin[v
     ]], dps removendo v[u])
// e pra query com o lca != u, so fazer ela normalmente ([tout[u], tin[v]])
```

4.45 erdos gallai

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

#define int long long int
#define endl ’\n’
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define pci pair<char, int>
#define fir first
#define sec second
#define MAXN 100005
#define mod 998244353

bool erdos_gallai(vector<int> &v)
{
  int sum = 0, n = v.size();
  for (auto const &i : v)
  {
    sum += i;
  }
  if (sum % 2)
  {
    return false;
  }
  sort(v.rbegin(), v.rend());
  vector<int> suf(n + 1, 0);
  int qt = 0, ptr = n;
  sum = 0;
  for (int i = n - 1; i >= 0; i--)
  {
    if (v[i] >= i)
    {
      qt++;
    }
    else
    {
      sum += v[i];
      ptr = i;
    }
    while (ptr < n && v[ptr] >= i)
    {
      qt++;
      sum -= v[ptr];
      ptr++;
    }
    suf[i] = sum + (qt * i);
  }
  sum = 0;
  bool ok = 1;
  for (int i = 0; i < n; i++)
  {
    sum += v[i];
    int curr = i * (i + 1) + suf[i + 1];
    ok &= (sum <= curr);
  }
  return ok;
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int q;
  cin >> q;
  while (q--)
  {
    int n;
    cin >> n;
    vector<int> v(n);
    for (int i = 0; i < n; i++)
      cin >> v[i];
    (erdos_gallai(v)) ? cout << "Y\n" : cout << "N\n";
  }
  return 0;
}
// https://codeforces.com/gym/101726/problem/A
// erdos gallai
// dado uma sequencia de n inteiros d[0], d[1], ..., d[n - 1]
// quero saber se existe um grafo simples e undirected com n vertices
// tal que o grau do i-esimo vertice e igual a d[i]
```

4.46 rmq tree

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

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

#define int long long int
#define endl ’\n’
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 5005
#define mod 998244353

struct lca
{
  int n;
  vector<vector<int>> adj;
  vector<int> vec;
  int l, timer;
  vector<int> tin, tout, depth;
  vector<vector<pi>> up;

  void dfs(int v, int p)
  {
    tin[v] = ++timer;
    up[v][0] = {p, min(vec[v], vec[p])};
    for (int i = 1; i <= l; i++)
    {
      up[v][i].fir = up[up[v][i - 1].fir][i - 1].fir;
      up[v][i].sec = min(up[v][i - 1].sec, up[up[v][i - 1].fir][i - 1].sec);
    }
    for (auto const &u : adj[v])
    {
      if (p == u)
        continue;
      depth[u] = depth[v] + 1;
      dfs(u, v);
    }
    tout[v] = ++timer;
  }
  bool is_ancestor(int u, int v)
  {
    return tin[u] <= tin[v] && tout[u] >= tout[v];
  }
  int find_lca(int u, int v)
  {
    if (is_ancestor(u, v))
      return u;
    if (is_ancestor(v, u))
      return v;
    for (int i = l; i >= 0; i--)
      if (!is_ancestor(up[u][i].fir, v))
        u = up[u][i].fir;
    return up[u][0].fir;
  }
  int dist(int s, int v)
  {
    int at = find_lca(s, v);
    return (depth[s] + depth[v] - 2 * depth[at]);
  }
  int solve(int u, int d)
  {
    int ans = vec[u];
    for (int i = l; i >= 0; i--)
    {
      if (d & (1 << i))
      {
        ans = min(ans, up[u][i].sec);
        u = up[u][i].fir;
      }
    }
    return ans;
  }
  int rmq(int u, int v)
  {
    int l = find_lca(u, v);
    return min(solve(u, dist(u, l)), solve(v, dist(v, l)));
  }
  lca(vector<vector<int>> &_adj, vector<int> &_vec)
  {
    adj = _adj;
    vec = _vec;
    n = adj.size();
    tin.resize(n);
    tout.resize(n);
    depth.resize(n);
    timer = 0;
    l = ceil(log2(n));
    up.assign(n, vector<pi>(l + 1));
    dfs(0, 0);
  }
};
signed main()
{
}
// valores nos vertices
// rmq considerando o caminho entre os vertices u e v
```

4.47 Topological Sort

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

#define lli long long int
#define pb push_back
#define MAXN 10000

int n , m , a , b ;
vector <int> adj [MAXN] ;
int grau [MAXN];
vector <int> order ;

bool topological_sort ()
{
    int ini = 0 ;

    while (ini < order.size())
    {
        int atual = order[ini] ;
        ini++ ;

        for (int i = 0 ; i < adj[atual].size() ; i++)
        {
            int v = adj[atual][i] ;
            grau[v]-- ;

            if (grau[v] == 0)
            {
                order.pb(v) ;
            }
        }
    }

    return (order.size() == n) ? true : false ;
}
int main ()
{
    ios_base::sync_with_stdio(false) ;
    cin.tie(NULL) ;

    cin >> n >> m ;

    for (int i = 1 ; i <= m ; i++)
    {
        cin >> a >> b ;
        grau[a]++ ;
        adj[b].pb(a) ;
    }

    for (int i = 1 ; i <= n ; i++)
    {
        if (grau[i] == 0)
        {
            order.pb(i) ;
        }
    }

    if (topological_sort())
    {
        for (int i = 0 ; i < order.size() ; i++)
        {
            cout << order[i] << " " ;
        }

        cout << endl ;
    }
    else
    {
        cout << "Impossible\n" ;
    }

    return 0 ;
}
```

4.48 Grafo Bipartido

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 200006
#define mod 1000000007

struct dsu
{
  vector<pi> parent;
  vector<int> rank;
  vector<int> bipartite;

  dsu(int n)
  {
    parent.resize(n);
    rank.resize(n);
    bipartite.resize(n);
    for (int v = 0; v < n; v++)
    {
      parent[v] = {v, 0};
      rank[v] = 0;
      bipartite[v] = 1;
    }
  }
  dsu() {}
  pi find_set(int v)
  {
    if (v != parent[v].fir)
    {
      int parity = parent[v].sec;
      parent[v] = find_set(parent[v].fir);
      parent[v].sec ^= parity;
    }
    return parent[v];
  }
  void add_edge(int a, int b)
  {
    pi pa = find_set(a);
    a = pa.fir;
    int x = pa.sec;
    pi pb = find_set(b);
    b = pb.fir;
    int y = pb.sec;
    if (a == b)
    {
      if (x == y)
        bipartite[a] = 0;
    }
    else
    {
      if (rank[a] < rank[b])
        swap(a, b);
      parent[b] = {a, x ^ y ^ 1};
      bipartite[a] &= bipartite[b];
      if (rank[a] == rank[b])
        rank[a]++;
    }
  }
  bool is_bipartite(int v)
  {
    return bipartite[find_set(v).fir];
  }
};
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  return 0;
}
```

5 Miscellaneous

5.1 inversion count

```
// seja S = a1, a2 , ... , an
// uma inversao S e um par (i,j) com i < j e ai > aj

// Solucao O(n2) nao ideal:
//for(int i=0;i<n;i++)
//      for(int j=i+1;j<n;j++)
//              if(v[i]>v[j]) ans++;

// Em vez de trabalharmos com o vetor inteiro(n2), vamos dividir o vetor ao meio
      e trabalhar com suas metades,
// que chamaremos de u1 e u2.

// Queremos saber o valor de inv, o numero de inversoes em v. Ha tres tipos de
     inversoes (i,j)(i,j) em v:
// aquelas em que i e j estao ambos em u1, aquelas em que i e j estao ambos em
     u2 e aquelas
// em que i esta em u1 e j esta em u2.
#include <bits/stdc++.h>
using namespace std;

#define lli long long int
#define pb push_back
#define in insert
#define pi pair<int, int>
#define pii pair<int, pi>
#define mp make_pair
#define fir first
#define sec second
#define MAXN 100001
#define INF 1000000000

int merge_sort(vector<int> &v)
{
  int ans = 0;

  if (v.size() == 1)
  {
    return 0;
  }

  vector<int> u1, u2;

  for (int i = 0; i < v.size() / 2; i++)
  {
    u1.pb(v[i]);
  }
  for (int i = v.size() / 2; i < v.size(); i++)
  {
    u2.pb(v[i]);
  }

  ans += merge_sort(u1);
  ans += merge_sort(u2);

  u1.pb(INF);
  u2.pb(INF);

  int ini1 = 0, ini2 = 0;

  for (int i = 0; i < v.size(); i++)
  {
    if (u1[ini1] <= u2[ini2])
    {
      v[i] = u1[ini1];
      ini1++;
    }
    else
    {
      v[i] = u2[ini2];
      ini2++;
      ans += u1.size() - ini1 - 1;
    }
  }

  return ans;
}
signed main()
{
  int n;
  cin >> n;
  vector<int> v(n);
  for (int i = 0; i < n; i++)
    cin >> v[i];
  cout << merge_sort(v) << endl;
  return 0;
}
```

5.2 two pointers

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 3005
#define mod 1000000007

const int inf = LLONG_MAX;
stack<pii> s[2];
void add(int x, int i)
{
  int mn = inf, mx = -inf;
  if (!s[i].empty())
  {
    mn = min(mn, s[i].top().sec.fir);
    mx = max(mx, s[i].top().sec.sec);
  }
  mn = min(mn, x);
  mx = max(mx, x);
  s[i].push({x, {mn, mx}});
}
void change()
{
  while (!s[1].empty())
  {
    int x = s[1].top().fir;
    s[1].pop();
    add(x, 0);
  }
}
void rem()
{
  if (!s[0].size())
    change();
  s[0].pop();
}
int q()
{
  int mn = inf, mx = -inf;
  for (int i = 0; i < 2; i++)
  {
    if (!s[i].empty())
    {
      mn = min(mn, s[i].top().sec.fir);
      mx = max(mx, s[i].top().sec.sec);
    }
  }
  if (mn == inf)
    return 0;
  return mx - mn;
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n, k;
  cin >> n >> k;
  vector<int> v(n);
  for (int i = 0; i < n; i++)
    cin >> v[i];
  int ans = 0, i = 0;
  for (int j = 0; j < n; j++)
  {
    add(v[j], 1);
    while (q() > k)
    {
      rem();
      i++;
    }
    ans += (j - i + 1);
  }
  cout << ans << endl;
  return 0;
}
// https://codeforces.com/edu/course/2/lesson/9/2/practice/contest/307093/
     problem/F
// Given an array of n integers, Let's say that a segment of this array is good
// if the difference between the maximum and minimum elements on this segment is
      at most k
// Your task is to find the number of different good segments
// amazing trick using stack
```

5.3 bitmasks

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

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

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 200005
#define mod 998244353

signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n, mask;
  vector<int> masks;

  // quantidade de bits setados na mask
  cout << __builtin_popcount(mask) << endl;

  // para printar o valor do bit i
  for (int i = 0; i < n; i++)
    cout << ((mask >> i) & 1) << " ";
  cout << endl;

  // quando eh necessario percorrer todas as submasks ate (1 << n)
  // e fazer algo com todas as submasks dessa mask
  // util em problemas de dp com mask por exemplo
  for (int i = 0; i < n; i++)
  {
    for (int j = 0; j < (1 << n); j++)
    {
      if ((j >> i & 1) == 0)
      {
        //alguma coisa aqui sabendo que a mask(j) eh uma submask de(j ^ 1 << i)
      }
    }
  }

  // para percorrer por todas as submasks de uma mask
  for (int s = mask; s; s = (s - 1) & mask)
  {
    // alguma coisa aqui sabendo que s eh uma submask de mask
  }

  // quando eh necessario percorrer todas as submasks ate (1 << n)
  // e fazer algo com todas as submasks dessa mask O(3^n)
  // util em problemas de dp com mask por exemplo
  for (int m = 0; m < (1 << n); m++)
  {
    for (int s = m; s; s = (s - 1) & m)
    {
      // alguma coisa aqui sabendo que mask s eh uma submask de m
    }
  }

  // comprimindo as masks de um vector baseada em uma mask qualquer
  for (int i = 0; i < masks.size(); i++)
  {
    int compressed = 0, curr_bit = 0;
    for (int j = 0; j < n; j++)
    {
      if (!(mask & (1LL << j)))
        continue;
      if (masks[i] & (1LL << j))
        compressed |= (1LL << curr_bit);
      curr_bit++;
    }
    // alguma coisa sabendo que a mask compressed eh a mask comprimida da mask
         atual
  }
  return 0;
}
```

5.4 sprague grundy

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 500009
#define mod 1000000001

vector<int> v = {2, 3, 4, 5, 6};
unordered_map<int, bool> vis;
unordered_map<int, int> dp;

int g(int x) // achar o grundy number na marra
{
    if (x == 0)
        return 0;
    vector<bool> ok(4, 0);
    int mex = 0;
    for (auto const &i : v)
    {
        int curr = g(x / i);
        if (curr < 4)
            ok[curr] = 1;
        while (ok[mex])
            mex++;
    }
    vis[x] = 1;
    return dp[x] = mex;
}
int solve(int x) // padraozin
{
    vector<int> ini = {0, 1, 2, 2, 3, 3, 0, 0, 0, 0, 0, 0};
    while (x >= 12)
        x /= 12;
    return ini[x];
}
signed main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    int q;
    cin >> q;
    while (q--)
    {
        int n;
        cin >> n;
        int x = 0;
        for (int i = 0; i < n; i++)
        {
            int k;
            cin >> k;
            x ^= solve(k);
        }
        (x > 0) ? cout << "Henry\n" : cout << "Derek\n";
    }
    return 0;
}
```

```
/*
game theory (um exemplo simples de problema pra ficar no repo)

- pro nim classico
- existem n pilhas cada uma possui x[i] blocos
- em uma play posso escolher uma pilha e tirar uma quantidade qualquer de blocos
      dela
- quem ganha?
- o jogador que comeca ganha se o xor dos tamanhos das pilhas for != 0

- teorema sprague-grundy (transformar um jogo qualquer em nim)
- seja v um estado que eu tou do jogo, podemos calcular o grundy number desse
     estado
- seja o conjuntos de estados adjacentes a v {u1, u2, ..., un}
- g(v) = mex(g(u1), g(u2), ..., g(un))
- se v nao tem nenhum extado adjacente, entao g(v) = 0
- g(v) -> grundy number do estado v
- com isso se tivemos varios estados iniciais (varias pilhas)
- podemos simplesmente achar o grundy number de cada um deles e depois saber
     quem ganha
- pelo valor do xor dos grundy numbers

- exemplo: floor division game
- existem n numeros e em uma play posso escolher um deles e dividir por 2, 3, 4,
      5 ou 6
- quem ganha?
- achar o grundy number de cada um dos n numeros
- se o xor for != 0, ganha quem comeca jogando
- caso contrario, o outro jogador ganha

- as vzs e util tbm ver se existe um padrao (em caso de altas constantes)
- notando o padrao, da pra achar o grundy number de forma mais eficiente e
     resolver o problema
*/
```

5.5 coordinate compression

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 500005
#define mod 1000000007

void compress(vector<int> &v)
{
  vector<int> val;
  for (auto const &i : v)
    val.pb(i);
  sort(val.begin(), val.end());
  val.erase(unique(val.begin(), val.end()), val.end());
  for (auto &i : v)
    i = lower_bound(val.begin(), val.end(), i) - val.begin();
}
```

5.6 meetinthemiddle

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define DEBUG 0
#define MAXN 1000001

int n, t;
vector<int> v;
vector<int> a;
vector<int> b;

void solve2(int i, int j, int k)
{
  if (i == j)
  {
    b.pb(k);
    return;
  }
  solve2(i + 1, j, k);
  solve2(i + 1, j, k + v[i]);
}
void solve(int i, int j, int k)
{
  if (i == j)
  {
    a.pb(k);
    return;
  }
  solve(i + 1, j, k);
  solve(i + 1, j, k + v[i]);
}
int upper(int l, int r, int x)
{
  while (l < r)
  {
    int mid = (l + r + 1) >> 1;
    (b[mid] <= x) ? l = mid : r = mid - 1;
  }
  return b[l];
}
int meetinthemiddle()
{
  solve(0, (n >> 1) + 1, 0);
  solve2((n >> 1) + 1, n, 0);
  sort(b.begin(), b.end());
  int ans = 0;
  for (auto const &i : a)
  {
    if (i > t)
      continue;
    ans = max(ans, i);
    int kappa = i + upper(0, b.size() - 1, t - i);
    if (kappa <= t)
      ans = max(ans, kappa);
  }
  return ans;
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> n >> t;
  v.resize(n);
  for (int i = 0; i < n; i++)
    cin >> v[i];
  cout << meetinthemiddle() << endl;
  return 0;
}
```

5.7 segment covering

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, pi>
#define fir first
#define sec second
#define MAXN 500005
#define mod 1000000007

int st[MAXN + 1][21];

void naive(vector<pi> &v) // guloso do point covering
{
  // os segmentos precisam estar ordenados pelo .second
  int n = v.size(), last = -1;
  vector<int> ans;
  for (int i = 0; i < n; i++)
  {
    if (v[i].fir > last)
    {
      ans.pb(v[i].sec);
      last = v[i].sec;
    }
  }
}
bool can(int l, int r, int x)
{
  for (int i = 20; i >= 0; i--)
  {
    if (x & (1 << i))
      l = st[l][i];
  }
  return l > r;
}
void solve(vector<pi> &v, int a, int b) // segment covering com binary lifting (
     da pra fazer point covering de forma bem similar)
{
  for (int i = 0; i <= MAXN; i++)
  {
    st[i][0] = i;
  }
  for (auto const &i : v)
  {
    st[i.fir][0] = max(st[i.fir][0], i.sec + 1);
  }
  for (int i = 1; i <= MAXN; i++) // se um segmento com l menor tem um r maior
  {
    st[i][0] = max(st[i][0], st[i - 1][0]);
  }
  for (int i = 1; i < 21; i++)
  {
    for (int v = 0; v <= MAXN; v++)
      st[v][i] = st[st[v][i - 1]][i - 1];
  }
  int lo = 1, hi = v.size();
  while (lo < hi) // busca binaria na resposta
  {
    int mid = (lo + hi) >> 1;
    (can(a, b, mid)) ? hi = mid : lo = mid + 1;
  }
  if (can(a, b, lo))
    cout << lo << endl;
  else
    cout << "-1\n";
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  return 0;
}
// um tipo de problema que eu achei bem legal
// https://codeforces.com/problemset/problem/1175/E
// https://codeforces.com/gym/101221 (Problem K)
// https://vjudge.net/contest/512192#problem/E

// problema exemplo:
// dado um conjunto de n segmentos [l[i], r[i]]
// qual o numero minimo de pontos que vc pode escolher, tal que:
// para cada segmento [l[i], r[i]], pelo menos um ponto escolhido ta nesse
     segmento
// tem a solucao gulosa em O(N)
// mas que da pra ser otimizada com binary lifting/sparse table (em caso de ter
     varias queries sobre o conjunto de segmentos)
// depois de adicionar um ponto a resposta, acho o nxt dele: o proximo ponto que
      irei colocar na resposta depois de adicionar ele

// outro problema exemplo:
// dado um conjunto de n segmentos [l[i], r[i]]
// voce quer selecionar o numero minimo de segmentos do conjunto
// para cobrir todo o segmento [a, b]
// bem parecido, tem uma solucao gulosa normal
// e se tu quer fazer varias queries, otimiza com binary lifting/sparse table
```

5.8 sum hash

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define MAXN 300005
#define mod 998244353

signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  mt19937_64 rng(chrono::steady_clock::now().time_since_epoch().count());
  int n, k;
  cin >> n >> k;
  vector<int> h(k + 1, 0);
  for (int i = 1; i < k; i++)
  {
    h[i] = rng();
    h[k] -= h[i];
  }
  vector<int> v(n);
  int sum = 0, ans = 0;
  map<int, int> mp;
  mp[0] = 0;
  for (int i = 0; i < n; i++)
  {
    cin >> v[i];
    sum += h[v[i]];
    if (mp.find(sum) != mp.end())
      ans = max(ans, i - mp[sum] + 1);
    else
```

```
      mp[sum] = i + 1;
  }
  cout << ans << endl;
}
// solucao pra C da final brasileira da maratona de 2023
// dado um array com n inteiros, cada a[i] ta entre 1 e k
// qual o maior tamanho de um subarray no qual todos os numeros de 1 ate k
// tem a mesma frequencia nesse subarray
```

5.9 tower of hanoi

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

// #define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 300005
#define mod 998244353

vector<pair<char, char>> ans;

void solve(int n, char a, char b, char c)
{
  if (n == 0)
    return;
  solve(n - 1, a, c, b);
  ans.pb({a, b});
  solve(n - 1, c, b, a);
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n, k;
  cin >> n >> k;
  solve(n, 'A', 'C', 'B');
  if (ans.size() > k)
  {
    cout << "N\n";
    return 0;
  }
  cout << "Y\n";
  k -= ans.size();
  if (k % 2 == 0)
  {
    for (int i = 0; i < (k / 2); i++)
    {
      cout << "A B\n";
      cout << "B A\n";
    }
  }
  else
  {
    for (int i = 0; i < (k / 2) - 1; i++)
    {
      cout << "A B\n";
      cout << "B A\n";
    }
    cout << "A B\n";
    cout << "B C\n";
    cout << "C A\n";
  }
  for (auto const &i : ans)
    cout << i.fir << " " << i.sec << endl;
  return 0;
}
// torre de hanoi
// 3 pilhas, sendo uma pilha com n discos e as outras duas pilhas vazias
// em cada movimento, vc tira o disco do topo de uma pilha e poe no topo de
     outra pilha
// desde que o raio do disco seja menor do que o raio do disco que ta no topo da
     outra pilha
// os n discos tem raios distintos aos pares
// fazer com que todos os discos vao parar em outra pilha

// https://codeforces.com/gym/101879/problem/I
// resolver a torre de hanoi com k movimentos
// se for possivel resolver, printar os movimentos feitos

// numero minimo pra resolver pros primeiros n
// 1, 3, 7, 15, 31, 63, 127, 255
// f(n) = 2^n - 1
```

5.10 prefix sum 2d

```
// https://cses.fi/problemset/task/1652
#include <bits/stdc++.h>
using namespace std;

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define MAXN 200005
#define mod 1000000007

int v[1001][1001];
int p[1001][1001];

int qry(int x1, int y1, int x2, int y2)
{
  return p[x2 + 1][y2 + 1] - p[x2 + 1][y1] - p[x1][y2 + 1] + p[x1][y1];
}
signed main()
{
  int n, q;
  cin >> n >> q;
  for (int i = 0; i < n; i++)
  {
    for (int j = 0; j < n; j++)
    {
      char c;
      cin >> c;
      v[i][j] = (c == '*');
    }
  }
  for (int i = 0; i < n; i++)
  {
    for (int j = 0; j < n; j++)
    {
      p[i + 1][j + 1] = p[i][j + 1] + p[i + 1][j] - p[i][j];
      p[i + 1][j + 1] += v[i][j];
    }
  }
  while (q--)
  {
    int a, b, c, d;
    cin >> a >> b >> c >> d;
    a--, b--, c--, d--;
    cout << qry(a, b, c, d) << endl;
  }
  return 0;
}
// prefix sum 2d
// me enrolo pra codar toda vez, e bom deixar na lib
```

5.11 stack trick

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

#define PI acos(-1)
#define int long long int
#define pb push_back
#define pi pair<int, int>
#define fir first
#define sec second
#define MAXN 300001
#define mod 1000000007

int n;
vector<int> v;
vector<int> ans;

void solve()
{
  stack<pi> s;
  for (int i = n - 1; i >= 0; i--)
  {
    while (!s.empty() && s.top().fir <= v[i])
      s.pop();
    (!s.empty()) ? ans[i] = s.top().sec : ans[i] = -1;
    s.push({v[i], i});
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> n;
  v.resize(n);
  ans.resize(n);
  for (int i = 0; i < n; i++)
    cin >> v[i];
  solve();
  for (auto const &i : ans)
    cout << i << " ";
  cout << endl;
}
// WITHOUT SEGMENT TREE
// for each index (0 <= i < n), find another index (0 <= j < n)
// which v[j] > v[i] and j > i and j is as close as possible to i.
// if this index does not exist, print -1

/*
5
1 3 3 4 5
*/
/*
1 3 3 4 -1
*/
```

6 Theorems and Formulas

6.1 binomial theorem

```
## Binomial Theorem

### Theorem

$$
(x + y)^n = \sum_{k = 0}^{n} {n \choose k} x^{n - k}y^{k}
$$

in addition, we have:

$$
(x - y)^n = \sum_{k = 0}^{n} (-1)^k {n \choose k} x^{n - k}y^{k}
$$

$$
(1 + x)^n = \sum_{k = 0}^{n} {n \choose k} x^{k}
$$

### Cool Problem

[Fibonacci Fever](https://codeforces.com/gym/104412/problem/F)

Given n and k you're asked to compute (mod $10^9 + 7$):

$$
\sum\_{i = 1}^{n}f_i^k
$$

where $f_n$ is the n-th fibonacci number.

Recall that:

$$
f_n = \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2}\right)^n -  \frac{1}{\sqrt
     {5}} \left(\frac{1-\sqrt{5}}{2}\right)^n
$$

This is something like that:

$$
f_n = ca^n - cb^n
$$

In the end, we end up with something like:

$$
c^k (\sum_{j = 0}^{k} (-1)^{j} {k \choose j} \sum_{i = 1}^{n} (a^{k - j}b^{j})^{
     i} )
$$

PS: To handle $\sqrt{5}$ mod $10^9 + 7$, we need to store each number in the
     form $x = a + b\sqrt{5}$

[Code](https://github.com/jonh14lk/Competitive_Programming/blob/master/Math/
     binomial_theorem.cpp)
```

6.2 chicken mcnugget

```
## Chicken McNugget Theorem

For any two coprime numbers (n > 0, m > 0), the greatest integer that cannot be
     written in the form:

an + bm, (a >= 0, b >= 0)

is (n \* m) - n - m

## Consequence of the theorem

That there are exactly ((n - 1) \* (m - 1)) / 2 positive integers which cannot
     be expressed in the form an + bm, (a >= 0, b >= 0)

## Generalization

If n and m are not coprime, so all numbers that are not multiples of gcd(n, m)
     cannot be expressed in the form an + bm, (a >= 0, b >= 0)

in addition, you can consider n = (n / gcd(n, m)) and m = (m / gcd(n, m)), to
     find how many multiples of gcd(n, m) cannot be expressed, or to find the
     greatest multiple of gcd(n, m) that cannot be expressed

## Considering a > 0, b > 0

Considering (n > 0, m > 0), n and m are coprime:

let y = ((n \* m) + min(n, m)) - 1
```

The number of positive integers which cannot be expressed increases by (y / n)

The number of positive integers which cannot be expressed increases by (y / m)

you must not count the multiples of (n * m) more than once, just decrease number of positive integers which cannot be expressed by (y / (n * m))

Problems

- [Forming Compounds](https://codeforces.com/group/XrhoJtxCjm/contest/422716/problem/I)

6.3 graph notes

Bipartite Graph

A bipartite graph is a graph that does not contain any odd-length cycles.

Directed acyclic graph (DAG)

Is a directed graph with no directed cycles.

Independent Set

Is a set of vertices in a graph, no two of which are adjacent. That is, it is a set S of vertices such that for every two vertices in S, there is no edge connecting the two.

Clique

Is a subset of vertices of an undirected graph such that every two distinct vertices in the clique are adjacent.

Vertex Cover

Is a set of vertices that includes at least one endpoint of every edge of the graph.

Edge Cover

Is a set of edges such that every vertex of the graph is incident to at least one edge of the set.

Path Cover

Given a directed graph G = (V, E), a path cover is a set of directed paths such that every vertex v belongs to at least one path.

Koning's Theorem

In any bipartite graph, the number of edges in a maximum matching equals the number of vertices in a minimum vertex cover.

Properties

- Every tree is a bipartite graph.
- Any NxM grid is a bipartite graph.
- A set of vertices is a vertex cover if and only if its complement is an independent set.
- The number of vertices of a graph is equal to its minimum vertex cover number plus the size of a maximum independent set.
- In bipartite graphs, the size of the minimum edge cover is equal to the size of the maximum independent set
- In bipartite graphs, the size of the minimum edge cover plus the size of the minimum vertex cover is equal to the number of vertices.
- In bipartite graphs, maximum clique size is two.

Min-cut

The smallest total weight of the edges which if removed would disconnect the source from the sink.

Max-flow min-cut theorem

In a flow network, the maximum amount of flow passing from the source to the sink is equal to the total weight of the edges in a minimum cut.

Maximum flow with vertex capacities

In other words, the amount of flow passing through a vertex cannot exceed its capacity. To find the maximum flow, we can transform the problem into the maximum flow problem by expanding the network. Each vertex v is replaced by v-in and v-out, where v-in is connected by edges going into v and v-out is connected to edges coming out from v. Then assign capacity c(v) to the edge connecting v-in and v-out.

Undirected edge-disjoint paths problem

We are given an undirected graph G = (V, E) and two vertices s and t, and we have to find the maximum number of edge-disjoint s-t paths in G.

Undirected vertex-disjoint paths problem

We are given an undirected graph G = (V, E) and two vertices s and t, and we have to find the maximum number of vertex-disjoint (except for s and t) paths in G.

Menger's theorem

The maximum number of edge-disjoint s-t paths in an undirected graph is equal to the minimum number of edges in an s-t cut-set.

Undirected vertex-disjoint paths solution

We can construct a network N=(V,E) from G with vertex capacities, where the capacities of all vertices and all edges are 1. Then the value of the maximum flow is equal to the maximum number of independent paths from s to t.

Minimum vertex-disjoint path cover in directed acyclic graph (DAG)

Given a directed acyclic graph G=(V, E), we are to find the minimum number of vertex-disjoint paths to cover each vertex in V. We can construct a bipartite graph G' from G. Each vertex v is replaced by v-in and v-out, where v-in is connected by edges going into v and v-out is connected to edges coming out from v. Then it can be shown that G' has a matching M of size m if and only if G has a vertex-disjoint path cover C of containing m edges and n-m paths.

Minimum general path cover in directed acyclic graph (DAG)

A general path cover is a path cover where a vertex can belong to more than one path. A minimum general path cover may be smaller than a minimum vertex-disjoint path cover. A minimum general path cover can be found almost like a minimum vertex-disjoint path cover. It suffices to add some new edges to the matching graph so that there is an edge a - b always when there is a path from a to b in the original graph.

Dilworth's theorem and maximum antichain

An antichain is a set of nodes of a graph such that there is no path from any node to another node using the edges of the graph. Dilworth's theorem states that in a directed acyclic graph, the size of a minimum general path cover equals the size of a maximum antichain.

Or in other words: For a DAG G that if has edges from vertex i -> vertex j and vertex j -> k, then it also has a edge from vertex i -> vertex k, the size of a minimum path cover is equal to the size of a maximum independent set.

Maximum weighted antichain

(https://atcoder.jp/contests/abc354/tasks/abc354_g) In this problem, each vertex has a cost a[i]. The cost of an antichain is equal to the sum of the costs of the vertices present in it. We need to find the maximum cost of a antichain. We can construct the same bipartite of the maximum antichain problem from a dag G, these edges have an infinite capacity. We also need to create a source vertex and a sink, and we need to add edges source -> v-in with capacity a[v] and v-out -> sink with capacity a[v]. The answer is equal to the sum of all a[i] minus the maximum flow on this network.

Hall's Theorem

Hall's theorem can be used to find out whether a bipartite graph has a matching

```
    that contains all left or right nodes. Assume that we want to find a
    matching that contains all left nodes. Let X be any set of left nodes and
    let f(X) be the set of their neighbors. According to Hall's theorem, a
    matching that contains all left nodes exists exactly when for
each X, the condition |X| <= |f(X)| holds.

## References

- [Competitive Programmer's Handbook](https://cses.fi/book/book.pdf)
- [(Graph Theory) - Wikipedia](https://en.wikipedia.org/wiki/Graph_theory)
- [(Medium Article) - Solving Minimum Path Cover on a DAG](https://
    towardsdatascience.com/solving-minimum-path-cover-on-a-dag-21b16ca11ac0)

## Extra (Getting Confidence Trick)

[2019-2020 ACM-ICPC Brazil Subregional Programming Contest, problem G](https://
    codeforces.com/gym/102346/problem/G)

<p>If you need to maximize a number x = (a * b * c * ...), then you can write it
     as x = (e^log(a) * e^log(b) * e^log(c) * ...), and then the number is x =
    e^(log(a) + log(b) + log(c) + ...), and the problem now becomes a problem
    of maximizing the sum of (log(a) + log(b) + log(c) + ...).<p/>

Use exp() and log() C++ functions :)
```

7 Structures

7.1 lower bound segtree

```cpp
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 300005
#define mod 998244353

const int inf = 1e18;

struct segtree
{
  int n;
  vector<int> v;
  vector<int> seg;

  segtree(vector<int> &vv)
  {
    v = vv;
    n = v.size();
    seg.assign(4 * n, 0);
    build(0, n - 1, 1);
  }
  int single(int x)
  {
    return x;
  }
  int neutral()
  {
    return 1e18;
  }
  int merge(int a, int b)
  {
    return max(a, b);
  }
  void update(int i, int l, int r, int q, int x)
  {
    if (l == r)
    {
      seg[i] = single(x);
      return;
    }
    int mid = (l + r) >> 1;
    if (q <= mid)
      update(i << 1, l, mid, q, x);
    else
      update((i << 1) | 1, mid + 1, r, q, x);
    seg[i] = merge(seg[i << 1], seg[(i << 1) | 1]);
  }
  int query(int l, int r, int ql, int qr, int i, int x)
  {
    if (l == r)
      return (seg[i] >= x && l >= ql) ? l : -1;
    int mid = (l + r) >> 1, at = -1;
    if (seg[i << 1] >= x && mid >= ql)
      at = query(l, mid, ql, qr, i << 1, x);
    if (at == -1)
      at = query(mid + 1, r, ql, qr, (i << 1) | 1, x);
    return at;
  }
  void build(int l, int r, int i)
  {
    if (l == r)
    {
      seg[i] = single(v[l]);
      return;
    }
    int mid = (l + r) >> 1;
    build(l, mid, i << 1);
    build(mid + 1, r, (i << 1) | 1);
    seg[i] = merge(seg[i << 1], seg[(i << 1) | 1]);
  }
  int qry(int x, int l)
  {
    return query(0, n - 1, l, n - 1, 1, x);
  }
  void upd(int x, int l)
  {
    update(1, 0, n - 1, x, l);
  }
};
signed main()
{
  int n, q;
  cin >> n >> q;
  vector<int> v(n);
  for (int i = 0; i < n; i++)
    cin >> v[i];
  segtree st(v);
  while (q--)
  {
    int t;
    cin >> t;
    if (t == 2)
    {
      int x, l;
      cin >> x >> l; // find the minimum index j such that j >= l and v[j] >= x
      cout << st.qry(x, l) << endl;
    }
    else
    {
      int a, b;
      cin >> a >> b; // v[a] = b;
      st.upd(a, b);
    }
  }
}
```

7.2 Segtree2

```cpp
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 500001
#define mod 1000000007

struct segtree
{
  int n;
  vector<int> seg;

  int neutral()
  {
    return 0;
  }
  int merge(int a, int b)
  {
    return a + b;
  }
  void build(vector<int> &v)
  {
    n = 1;
    while (n < v.size())
      n <<= 1;
    seg.assign(n << 1, neutral());
    for (int i = 0; i < v.size(); i++)
      seg[i + n] = v[i];
    for (int i = n - 1; i; i--)
      seg[i] = merge(seg[i << 1], seg[(i << 1) | 1]);
  }
  void upd(int i, int value)
  {
    seg[i += n] += value;
    for (i >>= 1; i; i >>= 1)
      seg[i] = merge(seg[i << 1], seg[(i << 1) | 1]);
  }
  int qry(int l, int r)
  {
    int ansl = neutral(), ansr = neutral();
    for (l += n, r += n + 1; l < r; l >>= 1, r >>= 1)
    {
      if (l & 1)
        ansl = merge(ansl, seg[l++]);
      if (r & 1)
        ansr = merge(seg[--r], ansr);
    }
    return merge(ansl, ansr);
  }
};
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  return 0;
}
// iterative segtree without lazy propagation
```

7.3 SegTree pa

```cpp
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 100005
#define mod 1000000007

struct lazy_node
{
  int n, a, d;

  int sum()
  {
    int an = a + (d * (n - 1));
    return ((a + an) * n) >> 1;
  }
  void merge(lazy_node to_add)
  {
    a += to_add.a;
    d += to_add.d;
  }
};
struct segtree
{
  vector<int> seg;
  vector<lazy_node> lazy;
  vector<bool> lazy_status;

  segtree(int n)
  {
    seg.resize(4 * n);
    lazy.resize(4 * n);
    lazy_status.resize(4 * n);
    build(0, n - 1, 1);
  }
  int single(int x)
  {
    return x;
  }
  int neutral()
  {
    return 0;
  }
  int merge(int a, int b)
  {
    return a + b;
  }
  void add(int i, int l, int r, lazy_node to_add)
  {
    seg[i] += to_add.sum();
    if (l != r)
    {
      int mid = (l + r) >> 1;
      lazy[i << 1].merge({mid - l + 1, to_add.a, to_add.d});
      lazy_status[i << 1] = 1;
      int diff = (mid + 1) - l, a = to_add.a, d = to_add.d;
      lazy[(i << 1) | 1].merge({r - (mid + 1) + 1, a + (d * diff), d});
      lazy_status[(i << 1) | 1] = 1;
    }
    lazy[i] = {r - l + 1, 0, 0};
    lazy_status[i] = 0;
  }
  void update(int i, int l, int r, int ql, int qr, lazy_node to_add)
  {
    if (lazy_status[i])
      add(i, l, r, lazy[i]);
    if (l > r || l > qr || r < ql)
      return;
    if (l >= ql && r <= qr)
    {
      int diff = l - ql, a = to_add.a, d = to_add.d;
      lazy_node curr = {r - l + 1, a + (d * diff), d};
      add(i, l, r, curr);
      return;
    }
    int mid = (l + r) >> 1;
    update(i << 1, l, mid, ql, qr, to_add);
    update((i << 1) | 1, mid + 1, r, ql, qr, to_add);
    seg[i] = merge(seg[i << 1], seg[(i << 1) | 1]);
  }
  int query(int l, int r, int ql, int qr, int i)
  {
    if (lazy_status[i])
      add(i, l, r, lazy[i]);
    if (l > r || l > qr || r < ql)
      return neutral();
    if (l >= ql && r <= qr)
      return seg[i];
    int mid = (l + r) >> 1;
    return merge(query(l, mid, ql, qr, i << 1), query(mid + 1, r, ql, qr, (i <<
        1) | 1));
  }
  void build(int l, int r, int i)
  {
    seg[i] = 0;
    lazy_status[i] = 0;
    lazy[i] = {r - l + 1, 0, 0};
    if (l == r)
      return;
    int mid = (l + r) >> 1;
    build(l, mid, i << 1);
    build(mid + 1, r, (i << 1) | 1);
  }
};
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n, q;
  cin >> n >> q;
  segtree s(n);
  while (q--)
  {
    int t;
    cin >> t;
    if (t == 1)
    {
      int l, r, a, d;
      cin >> l >> r >> a >> d;
      l--, r--;
      s.update(1, 0, n - 1, l, r, {r - l + 1, a, d});
    }
    else
    {
      int x;
      cin >> x;
      x--;
      cout << s.query(0, n - 1, x, x, 1) << endl;
    }
  }
  return 0;
}
// queries of:
// add an arithmetic progression to a segment [l, r]
// print current value of a given element
```

7.4 min queue

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define MAXN 1005
#define mod 998244353

namespace min_queue
{
  deque<pi> q;
  int l, r;

  void init()
  {
    l = r = 1;
    q.clear();
  }
  void push(int v)
  {
    while (!q.empty() && v < q.back().fir)
      q.pop_back();
    q.pb({v, r});
    r++;
  }
  void pop()
  {
    if (!q.empty() && q.front().sec == l)
      q.pop_front();
    l++;
  }
  int getmin()
  {
    return q.front().fir;
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n, m;
  cin >> n >> m;
  vector<int> v(n);
  for (int i = 0; i < n; i++)
    cin >> v[i];
  int l = 0, r = m - 1;
  cout << l << " " << r << endl;
  for (int i = l; i <= r; i++)
    min_queue::push(v[i]);
  cout << min_queue::getmin() << " ";
  l++, r++;
  while (r < n)
  {
    min_queue::pop();
    min_queue::push(v[r]);
    cout << min_queue::getmin() << " ";
    l++, r++;
  }
  cout << endl;
  return 0;
}
// minimum of each subarray of length m (m <= n)
```

7.5 mo update

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define PI acos(-1)
#define pb push_back
#define int long long int
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 100005
#define mod 1000000007

int n, q;
int v[MAXN];
int vv[MAXN];

namespace mo
{
  struct query
  {
    int idx, l, r, t;
  };
  struct update
  {
    int i, prevx, x;
  };

  int block;
  vector<query> queries;
  vector<update> updates;
  vector<int> ans;

  bool cmp(query x, query y)
  {
    if (x.l / block != y.l / block)
      return x.l / block < y.l / block;
    if (x.r / block != y.r / block)
      return x.r / block < y.r / block;
    return x.t < y.t;
  }
  void run()
  {
    block = 3153; // (2 * n) ^ 0.666
    sort(queries.begin(), queries.end(), cmp);
    ans.resize(queries.size());
    int cl = 0, cr = -1, sum = 0, t = 0;
    auto add = [&](int x)
    {
      sum += x;
    };
    auto rem = [&](int x)
    {
      sum -= x;
    };
    for (int i = 0; i < queries.size(); i++)
    {
      while (cl > queries[i].l)
      {
        cl--;
        add(v[cl]);
      }
      while (cr < queries[i].r)
      {
        cr++;
        add(v[cr]);
      }
      while (cl < queries[i].l)
      {
        rem(v[cl]);
        cl++;
      }
      while (cr > queries[i].r)
      {
        rem(v[cr]);
        cr--;
      }
      while (t > queries[i].t)
      {
        t--;
        if (queries[i].l <= updates[t].i && queries[i].r >= updates[t].i)
        {
          rem(updates[t].x);
          add(updates[t].prevx);
        }
        v[updates[t].i] = updates[t].prevx;
      }
      while (t < queries[i].t)
      {
        if (queries[i].l <= updates[t].i && queries[i].r >= updates[t].i)
        {
          rem(updates[t].prevx);
          add(updates[t].x);
        }
        v[updates[t].i] = updates[t].x;
        t++;
      }
      ans[queries[i].idx] = sum;
    }
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> n >> q;
  for (int i = 0; i < n; i++)
  {
    cin >> v[i];
    vv[i] = v[i];
  }
  for (int i = 0; i < q; i++)
  {
    int type;
    cin >> type;
    if (type == 1)
    {
      mo::update curr;
      cin >> curr.i >> curr.x;
      curr.prevx = vv[curr.i];
      vv[curr.i] = curr.x;
      mo::updates.pb(curr);
    }
    else
    {
      mo::query curr;
      cin >> curr.l >> curr.r;
      curr.r--;
      curr.idx = mo::queries.size();
      curr.t = mo::updates.size();
      mo::queries.pb(curr);
    }
  }
  mo::run();
  for (auto const &i : mo::ans)
    cout << i << endl;
}
// to test: https://codeforces.com/edu/course/2/lesson/4/1/practice/contest
    /273169/problem/A
// 1 i v - set the element with index i to v
// 2 l r - calculate the sum of elements with indices from l to r - 1
// n, q <= 100000
// runs in 467ms
```

7.6 treap2

```cpp
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 1000005
#define mod 1000000007

vector<int> ans;

struct treap
{
  int data, priority;
  int sz;
  bool lazy;
  treap *l, *r;
};
int size(treap *node)
{
  return (!node) ? 0 : node->sz;
}
void recalc(treap *node)
{
  if (!node)
    return;
  node->sz = 1;
  if (node->l)
    node->sz += node->l->sz;
  if (node->r)
    node->sz += node->r->sz;
}
void lazy_propagation(treap *node)
{
  if (!node || !(node->lazy))
    return;
  swap(node->l, node->r);
  if (node->l)
    node->l->lazy ^= 1;
  if (node->r)
    node->r->lazy ^= 1;
  node->lazy = 0;
}
void merge(treap *&t, treap *l, treap *r)
{
  lazy_propagation(l);
  lazy_propagation(r);
  if (!l)
    t = r;
  else if (!r)
    t = l;
  else if (l->priority > r->priority)
    merge(l->r, l->r, r), t = l;
  else
    merge(r->l, l, r->l), t = r;
  recalc(t);
}
void split(treap *t, treap *&l, treap *&r, int n)
{
  if (!t)
    return void(l = r = 0);
  lazy_propagation(t);
  if (size(t->l) >= n)
    split(t->l, l, t->l, n), r = t;
  else
    split(t->r, t->r, r, n - size(t->l) - 1), l = t;
  recalc(t);
}
void reverse(treap *&t, int l, int r)
{
  treap *a0, *a1, *b0, *b1;
  split(t, a0, a1, l);
  split(a1, b0, b1, r - l + 1);
  b0->lazy ^= 1;
  merge(t, a0, b0);
  merge(t, t, b1);
}
void shift(treap *&t, int l, int r)
{
  treap *a0, *a1, *b0, *b1, *c0, *c1;
  split(t, a0, a1, l);
  split(a1, b0, b1, r - l + 1);
  split(b0, c0, c1, r - l);
  merge(t, a0, c1);
  merge(t, t, c0);
  merge(t, t, b1);
}
void dfs(treap *t)
{
  if (!t)
    return;
  lazy_propagation(t);
  dfs(t->l);
  ans.pb(t->data);
  dfs(t->r);
}
treap *create_node(int data, int priority)
{
  treap *ret = new treap;
  ret->data = data;
  ret->priority = priority;
  ret->l = 0;
  ret->r = 0;
  ret->sz = 1;
  ret->lazy = 0;
  return ret;
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  srand(time(NULL));
  treap *t = 0;
  int n, m, q;
  cin >> n >> q >> m;
  for (int i = 0; i < n; i++)
  {
    int k;
    cin >> k;
    merge(t, t, create_node(k, rand()));
  }
  while (q--)
  {
    int ty, l, r;
    cin >> ty >> l >> r;
    l--, r--;
    (ty == 1) ? shift(t, l, r) : reverse(t, l, r);
  }
  dfs(t);
  while (m--)
  {
    int i;
    cin >> i;
    i--;
    cout << ans[i] << " ";
  }
  cout << endl;
  return 0;
}
```

7.7 segtree lazy

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 200005
#define mod 998244353

struct segtree
{
  int n;
  vector<int> v;
  vector<int> seg;
  vector<int> lazy;

  segtree(int sz)
  {
    n = sz;
    seg.assign(4 * n, 0);
    lazy.assign(4 * n, 0);
    // v = vv; // for build
    // build(0, n - 1, 1); // for build
  }
  int single(int x)
  {
    return x;
  }
  int neutral()
  {
    return 0;
  }
  int merge(int a, int b)
  {
    return a + b;
  }
  void add(int i, int l, int r, int diff)
  {
    seg[i] += (r - l + 1) * diff;
    if (l != r)
    {
      lazy[i << 1] += diff;
      lazy[(i << 1) | 1] += diff;
    }
    lazy[i] = 0;
  }
  void update(int i, int l, int r, int ql, int qr, int diff)
  {
    if (lazy[i])
      add(i, l, r, lazy[i]);
    if (l > r || l > qr || r < ql)
      return;
    if (l >= ql && r <= qr)
    {
      add(i, l, r, diff);
      return;
    }
    int mid = (l + r) >> 1;
    update(i << 1, l, mid, ql, qr, diff);
    update((i << 1) | 1, mid + 1, r, ql, qr, diff);
    seg[i] = merge(seg[i << 1], seg[(i << 1) | 1]);
  }
  int query(int l, int r, int ql, int qr, int i)
  {
    if (lazy[i])
      add(i, l, r, lazy[i]);
    if (l > r || l > qr || r < ql)
      return neutral();
    if (l >= ql && r <= qr)
      return seg[i];
    int mid = (l + r) >> 1;
    return merge(query(l, mid, ql, qr, i << 1), query(mid + 1, r, ql, qr, (i <<
        1) | 1));
  }
  void build(int l, int r, int i)
  {
    if (l == r)
    {
      seg[i] = single(v[l]);
      return;
    }
    int mid = (l + r) >> 1;
    build(l, mid, i << 1);
    build(mid + 1, r, (i << 1) | 1);
    seg[i] = merge(seg[i << 1], seg[(i << 1) | 1]);
  }
  int qry(int l, int r)
  {
    return query(0, n - 1, l, r, 1);
  }
  void upd(int l, int r, int x)
  {
    update(1, 0, n - 1, l, r, x);
  }
};
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n, q;
  cin >> n >> q;
  segtree s(n);
  while (q--)
  {
    int t;
    cin >> t;
    if (t == 1)
    {
      int l, r, x;
      cin >> l >> r >> x;
      s.upd(l, r, x);
    }
    else
    {
      int l, r;
      cin >> l >> r;
      cout << s.qry(l, r) << endl;
    }
  }
  return 0;
}
```

7.8 SegTree

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

#define PI acos(-1)
#define pb push_back
#define int long long int
#define mp make_pair
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define MAXN 100001
#define MAXL 100
```

```
#define mod 1000000007

vector<int> seg;
vector<int> v;

int single(int x)
{
  return x;
}
int neutral()
{
  return 0;
}
int merge(int a, int b)
{
  return a + b;
}
void update(int i, int l, int r, int q, int x)
{
  if (l == r)
  {
    seg[i] = single(x);
    return;
  }
  int mid = (l + r) >> 1;
  if (q <= mid)
    update(i << 1, l, mid, q, x);
  else
    update((i << 1) | 1, mid + 1, r, q, x);
  seg[i] = merge(seg[i << 1], seg[(i << 1) | 1]);
}
int query(int l, int r, int ql, int qr, int i)
{
  int mid = (l + r) >> 1;
  if (l > r || l > qr || r < ql)
    return neutral();
  if (l >= ql && r <= qr)
    return seg[i];
  return merge(query(l, mid, ql, qr, i << 1), query(mid + 1, r, ql, qr, (i << 1)
        | 1));
}
void build(int l, int r, int i)
{
  if (l == r)
  {
    seg[i] = single(v[l]);
    return;
  }
  int mid = (l + r) >> 1;
  build(l, mid, i << 1);
  build(mid + 1, r, (i << 1) | 1);
  seg[i] = merge(seg[i << 1], seg[(i << 1) | 1]);
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n, q;
  cin >> n >> q;
  v.resize(n);
  seg.resize(4 * n);
  for (int i = 0; i < n; i++)
    cin >> v[i];
  build(0, n - 1, 1);
  while (q--)
  {
    int l, r;
    int t;
    cin >> t >> l >> r;
    if (t == 2)
      cout << query(0, n - 1, l, r - 1, 1) << endl;
    else
      update(1, 0, n - 1, l, r);
  }
}
```

7.9 fenwick2D

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl ’\n’
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 1025
#define mod 1000000007

int b[MAXN][MAXN];
int vv[MAXN][MAXN];

int qry(int x, int y)
{
  int sum = 0;
  for (; x >= 0; x = (x & (x + 1)) - 1)
  {
    for (int yy = y; yy >= 0; yy = (yy & (yy + 1)) - 1)
      sum += b[x][yy];
  }
  return sum;
}
void add(int x, int y, int v)
{
  for (; x < MAXN; x = x | (x + 1))
  {
    for (int yy = y; yy < MAXN; yy = yy | (yy + 1))
      b[x][yy] += v;
  }
}
int qry2(int x1, int y1, int x2, int y2)
{
  return qry(x2, y2) - qry(x2, y1 - 1) - qry(x1 - 1, y2) + qry(x1 - 1, y1 - 1);
}
void add2(int x1, int y1, int x2, int y2, int v)
{
  add(x1, y1, v);
  add(x1, y2 + 1, -v);
  add(x2 + 1, y1, -v);
  add(x2 + 1, y2 + 1, v);
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int q;
  cin >> q;
  while (q--)
  {
    int n;
    cin >> n;
    for (int i = 0; i < n; i++) // reseta
    {
      for (int j = 0; j < n; j++)
      {
        add(i, j, -vv[i][j]);
        vv[i][j] = 0;
      }
    }
    while (1)
    {
      string s;
      cin >> s;
      if (s == "SET")
      {
        int a, b, c;
        cin >> a >> b >> c;
        add(a, b, -vv[a][b]);
        vv[a][b] = c;
        add(a, b, vv[a][b]);
      }
      else if (s == "SUM")
      {
        int a, b, c, d;
        cin >> a >> b >> c >> d; // c >= a e d >= b
        cout << qry2(a, b, c, d) << endl;
      }
      else
      {
        break;
      }
    }
  }
  return 0;
}
// to test: https://www.spoj.com/problems/MATSUM/
```

7.10 mergesorttree

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define DEBUG 0
#define MAXN 100001
#define mod 1000000007

vector<int> seg[4 * MAXN];
int v[MAXN];

void update(int i, int l, int r, int q, int x)
{
  if (l == r)
  {
    seg[i].clear();
    seg[i].pb(x);
    return;
  }
  int mid = (l + r) >> 1;
  if (q <= mid)
    update(i << 1, l, mid, q, x);
  else
    update((i << 1) | 1, mid + 1, r, q, x);
  // a merge do c++ une os dois vectors, deixando ele ordenado em O(n)
  seg[i].clear();
  merge(seg[i << 1].begin(), seg[i << 1].end(), seg[(i << 1) | 1].begin(), seg[(
       i << 1) | 1].end(), back_inserter(seg[i]));
}
int query(int l, int r, int ql, int qr, int i, int x)
{
  int mid = (l + r) >> 1;
  if (l > r || l > qr || r < ql)
    return 0;
  if (l >= ql && r <= qr) // quantidade de elementos maiores do que x no range
       atual
    return seg[i].end() - upper_bound(seg[i].begin(), seg[i].end(), x);
  return query(l, mid, ql, qr, i << 1, x) + query(mid + 1, r, ql, qr, (i << 1) |
        1, x);
}
void build(int l, int r, int i)
{
  if (l == r)
  {
    seg[i].pb(v[l]);
    return;
  }
  int mid = (l + r) >> 1;
  build(l, mid, i << 1);
  build(mid + 1, r, (i << 1) | 1);
  // a merge do c++ une os dois vectors, deixando ele ordenado em O(n)
  merge(seg[i << 1].begin(), seg[i << 1].end(), seg[(i << 1) | 1].begin(), seg[(
       i << 1) | 1].end(), back_inserter(seg[i]));
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  return 0;
}
// merge sort tree
// a segment tree with ordered vectors in range nodes

// example:
// number of elements > x in a range [l, r]

// memory: O(n * log n)
// query: O(log^2 n)
```

7.11 treap

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

// #define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 1000005
#define mod 1000000007

struct treap
{
  int data, priority;
  int sz, lazy2;
  bool lazy;
  treap *l, *r, *parent;
};
int size(treap *node)
{
  return (!node) ? 0 : node->sz;
}
void recalc(treap *node)
{
  if (!node)
    return;
  node->sz = 1;
  node->parent = 0;
  if (node->l)
    node->sz += node->l->sz, node->l->parent = node;
  if (node->r)
```

```
    node->sz += node->r->sz, node->r->parent = node;
}
void lazy_propagation(treap *node)
{
  if (node == NULL)
    return;
  if (node->lazy2)
  {
    if (node->l)
      node->l->lazy2 += node->lazy2;
    if (node->r)
      node->r->lazy2 += node->lazy2;
    node->data += node->lazy2;
    node->lazy2 = 0;
  }
  if (node->lazy)
  {
    swap(node->l, node->r);
    if (node->l)
      node->l->lazy = !node->l->lazy;
    if (node->r)
      node->r->lazy = !node->r->lazy;
    node->lazy = 0;
  }
}
void split(treap *t, treap *&l, treap *&r, int n)
{
  if (!t)
    return void(l = r = 0);
  lazy_propagation(t);
  if (size(t->l) >= n)
    split(t->l, l, t->l, n), r = t;
  else
    split(t->r, t->r, r, n - size(t->l) - 1), l = t;
  recalc(t);
}
void merge(treap *&t, treap *l, treap *r)
{
  lazy_propagation(l);
  lazy_propagation(r);
  if (!l)
    t = r;
  else if (!r)
    t = l;
  else if (l->priority > r->priority)
    merge(l->r, l->r, r), t = l;
  else
    merge(r->l, l, r->l), t = r;
  recalc(t);
}
void troca(treap *&t, int l, int r, int ll, int rr)
{
  treap *a0, *a1, *b0, *b1, *c0, *c1, *d0, *d1;
  split(t, a0, a1, l);
  split(a1, b0, b1, r - l + 1);
  ll -= (r + 1);
  rr -= (r + 1);
  split(b1, c0, c1, ll);
  split(c1, d0, d1, rr - ll + 1);
  merge(t, a0, d0);
  merge(t, t, c0);
  merge(t, t, b0);
  merge(t, t, d1);
}
void add(treap *&t, int l, int r)
{
  treap *a0, *a1, *b0, *b1;
  split(t, a0, a1, l);
  split(a1, b0, b1, r - l + 1);
  b0->lazy ^= 1;
  b0->lazy2 += 1;
  merge(t, a0, b0);
  merge(t, t, b1);
}
void solve(int x)
{
  x = x % 26;
  char c = x + 'a';
  cout << c;
}
void dfs(treap *t)
{
  if (!t)
    return;
  lazy_propagation(t);
  dfs(t->l);
  solve(t->data);
  dfs(t->r);
}
treap *create_node(int data, int priority)
{
  treap *ret = new treap;
  ret->data = data;
  ret->priority = priority;
  ret->l = 0;
  ret->r = 0;
  ret->sz = 1;
  ret->lazy = 0;
  ret->lazy2 = 0;
  ret->parent = 0;
  return ret;
}
void goup(treap *&ans, treap *t) // vai pra raiz da arvore
{
  if (!t->parent)
  {
    ans = t;
    return;
  }
  goup(ans, t->parent);
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  srand(time(NULL));
  int q;
  cin >> q;
  while (q--)
  {
    int n, m;
    string s;
    cin >> s >> m;
    n = s.size();
    treap *t = 0;
    for (auto const &i : s)
    {
      int x = i - 'a';
      merge(t, t, create_node(x, rand()));
    }
    while (m--)
    {
      int a, b, c, d;
      cin >> a >> b >> c >> d;
      a--, b--, c--, d--;
      add(t, a, b);
      add(t, c, d);
      troca(t, a, b, c, d);
    }
    dfs(t);
    cout << endl;
  }
  return 0;
}
// https://vjudge.net/contest/478186#problem/E
// - lazy propagation
// - reverse range with lazy propagation
// - swap ranges with equal lenght

// extra:
// - save node parent
```

7.12 bit2d

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

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

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 1000001
#define mod 1000000007

// source: https://github.com/tfg50/Competitive-Programming/blob/master/
     Biblioteca/Data%20Structures/Bit2D.cpp
struct bit2d
{
  vector<int> ord;
  vector<vector<int>> t;
  vector<vector<int>> coord;

  bit2d(vector<pi> &pts) // recebe todos os pontos que vao ser inseridos pra
       construir, mas nao insere eles
  {
    sort(pts.begin(), pts.end());
    for (auto const &a : pts)
    {
      if (ord.empty() || a.fir != ord.back())
        ord.pb(a.fir);
    }
    t.resize(ord.size() + 1);
    coord.resize(t.size());
    for (auto &a : pts)
    {
      swap(a.fir, a.sec);
    }
    sort(pts.begin(), pts.end());
    for (auto &a : pts)
    {
      swap(a.fir, a.sec);
      for (int on = upper_bound(ord.begin(), ord.end(), a.fir) - ord.begin(); on
           < t.size(); on += on & -on)
      {
        if (coord[on].empty() || coord[on].back() != a.sec)
          coord[on].push_back(a.sec);
      }
    }
    for (int i = 0; i < t.size(); i++)
      t[i].assign(coord[i].size() + 1, 0);
  }
  void add(int x, int y, int v) // v[a][b] += v
  {
    for (int xx = upper_bound(ord.begin(), ord.end(), x) - ord.begin(); xx < t.
         size(); xx += xx & -xx)
    {
      for (int yy = upper_bound(coord[xx].begin(), coord[xx].end(), y) - coord[
           xx].begin(); yy < t[xx].size(); yy += yy & -yy)
        t[xx][yy] += v;
    }
  }
  int qry(int x, int y) // soma de todos os v[a][b] com (a <= x && b <= y)
  {
    int ans = 0;
    for (int xx = upper_bound(ord.begin(), ord.end(), x) - ord.begin(); xx > 0;
         xx -= xx & -xx)
    {
      for (int yy = upper_bound(coord[xx].begin(), coord[xx].end(), y) - coord[
           xx].begin(); yy > 0; yy -= yy & -yy)
        ans += t[xx][yy];
    }
    return ans;
  }
  int qry2(int x1, int y1, int x2, int y2)
  {
    return qry(x2, y2) - qry(x2, y1 - 1) - qry(x1 - 1, y2) + qry(x1 - 1, y1 - 1)
        ;
  }
  void add2(int x1, int y1, int x2, int y2, int v)
  {
    add(x1, y1, v);
    add(x1, y2 + 1, -v);
    add(x2 + 1, y1, -v);
    add(x2 + 1, y2 + 1, v);
  }
};
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  return 0;
}
```

7.13 fenwick3

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

//#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define MAXN 200005
#define mod 1000000007

int v[MAXN];

namespace bit
{
  ordered_set<int> bit[MAXN];

  int query(int r, int a, int b)
  {
    int ret = 0, curr = r;
    for (; r >= 0; r = (r & (r + 1)) - 1)
      ret += (bit[r].order_of_key(b + 1) - bit[r].order_of_key(a));
    return ret;
  }
  void add(int idx, int delta)
  {
    for (; idx < MAXN; idx = idx | (idx + 1))
      bit[idx].insert(delta);
  }
  void rem(int idx, int delta)
  {
    for (; idx < MAXN; idx = idx | (idx + 1))
      bit[idx].erase(delta);
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  return 0;
}
// ideia da merge sort tree na bit (fica mais rapido)
// so fazer uma bit de ordered set ou vector(se nao tiver update)
// add -> adiciona o numero delta na posicao idx
// rem -> remove o numero delta na posicao idx
// query -> retorna o numero de elementos tal que posicao <= r && (a <= num <= b
     )
```

7.14 segtree2d

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 1003
#define mod 1000000007

struct segtree2d
{
  int n, m;
  vector<vector<int>> seg;

  int neutral()
  {
    return 0;
  }
  int merge(int a, int b)
  {
    return a + b;
  }
  segtree2d(int nn, int mm)
  {
    n = nn, m = mm;
    seg = vector<vector<int>>(2 * n, vector<int>(2 * m, neutral()));
  }
  int qry(int x1, int y1, int x2, int y2)
  {
    int ret = neutral();
    int y3 = y1 + m, y4 = y2 + m;
    for (x1 += n, x2 += n; x1 <= x2; ++x1 /= 2, --x2 /= 2)
    {
      for (y1 = y3, y2 = y4; y1 <= y2; ++y1 /= 2, --y2 /= 2)
      {
        if (x1 % 2 == 1 and y1 % 2 == 1)
          ret = merge(ret, seg[x1][y1]);
        if (x1 % 2 == 1 and y2 % 2 == 0)
          ret = merge(ret, seg[x1][y2]);
        if (x2 % 2 == 0 and y1 % 2 == 1)
          ret = merge(ret, seg[x2][y1]);
        if (x2 % 2 == 0 and y2 % 2 == 0)
          ret = merge(ret, seg[x2][y2]);
      }
    }
    return ret;
  }
  void upd(int x, int y, int val)
  {
    int y2 = y += m;
    for (x += n; x; x /= 2, y = y2)
    {
      if (x >= n)
        seg[x][y] = val;
      else
        seg[x][y] = merge(seg[2 * x][y], seg[2 * x + 1][y]);
      while (y /= 2)
        seg[x][y] = merge(seg[x][2 * y], seg[x][2 * y + 1]);
    }
  }
};
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int q;
  cin >> q;
  while (q--)
  {
    int n;
    cin >> n;
    segtree2d st(n, n); // matriz NxN
    while (1)
    {
      string s;
      cin >> s;
      if (s == "SET")
      {
        int a, b, c;
        cin >> a >> b >> c;
        st.upd(a, b, c);
      }
      else if (s == "SUM")
      {
        int a, b, c, d;
        cin >> a >> b >> c >> d; // c >= a e d >= b
        cout << st.qry(a, b, c, d) << endl;
      }
      else
      {
        break;
      }
    }
  }
  return 0;
}
// to test: https://www.spoj.com/problems/MATSUM/
```

7.15 persistent seg2

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

//#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 200005
#define mod 1000000007
#define PI acos(-1)

struct node
{
  int item, l, r;
  node() {}
  node(int l, int r, int item) : l(l), r(r), item(item) {}
};
int n, q;
vector<node> seg;
vector<int> roots;

void init()
{
  seg.resize(1);
}
int newleaf(int vv)
{
  int p = seg.size();
  seg.pb(node(0, 0, vv));
  return p;
}
int newpar(int l, int r)
{
  int p = seg.size();
  seg.pb(node(l, r, seg[l].item + seg[r].item));
  return p;
}
int upd(int i, int l, int r, int pos)
{
  if (l == r)
    return newleaf(seg[i].item + 1);
  int mid = (l + r) >> 1;
  if (pos <= mid)
    return newpar(upd(seg[i].l, l, mid, pos), seg[i].r);
  return newpar(seg[i].l, upd(seg[i].r, mid + 1, r, pos));
}
int build(int l, int r)
{
  if (l == r)
    return newleaf(0);
  int mid = (l + r) >> 1;
  return newpar(build(l, mid), build(mid + 1, r));
}
int qry(int vl, int vr, int l, int r, int k)
{
  if (l == r)
    return l;
  int mid = (l + r) >> 1;
  int c = seg[seg[vr].l].item - seg[seg[vl].l].item;
  if (c >= k)
    return qry(seg[vl].l, seg[vr].l, l, mid, k);
  return qry(seg[vl].r, seg[vr].r, mid + 1, r, k - c);
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> n >> q;
  vector<int> v(n);
  set<int> vals;
  for (int i = 0; i < n; i++)
  {
    cin >> v[i];
    vals.insert(v[i]);
  }
  int mx = 1;
  map<int, int> mp, mpr;
  for (auto const &i : vals)
  {
    mp[i] = mx;
    mpr[mx] = i;
    mx++;
  }
  init();
  roots.pb(build(0, mx));
  for (auto const &i : v)
    roots.pb(upd(roots.back(), 0, mx, mp[i]));
  while (q--)
  {
    char c;
    cin >> c;
    if (c == 'Q')
    {
      int l, r, k;
      cin >> l >> r >> k;
      l--, r--;
      cout << mpr[qry(roots[l], roots[r + 1], 0, mx, k)] << endl;
    }
    else
    {
      int x;
      cin >> x;
      x--;
      swap(v[x], v[x + 1]);
      int a = upd(roots[x], 0, mx, mp[v[x]]);
      int b = upd(a, 0, mx, mp[v[x + 1]]);
      roots[x + 1] = a, roots[x + 2] = b;
    }
  }
  return 0;
}
// https://neps.academy/br/exercise/127
// queries de k-esimo menor em um range
// e fazer um swap entre v[i] e v[i + 1]
```

7.16 rmq

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

#define lli long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 200005
#define mod 998244353

struct rmq
{
  vector<int> v;
  bool is_max;
  int n;
  static const int b = 30;
  vector<int> mask, t;

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

```
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n;
  cin >> n;
  vector<int> v(n);
  for (int i = 0; i < n; i++)
    cin >> v[i];
  rmq r_min(v, 0);
  rmq r_max(v, 1);
  lli ans = 0;
  {
    vector<pi> q;
    q.pb({0, n - 1});
    while (!q.empty())
    {
      int l = q.back().fir;
      int r = q.back().sec;
      int max_pos = r_max.qry(l, r);
      q.pop_back();
      lli qt = (max_pos - l + 1) * 1ll * (r - max_pos + 1);
      ans += (v[max_pos] * 1ll * qt);
      if (max_pos > l)
        q.pb({l, max_pos - 1});
      if (max_pos < r)
        q.pb({max_pos + 1, r});
    }
  }
  {
    vector<pi> q;
    q.pb({0, n - 1});
    while (!q.empty())
    {
      int l = q.back().fir;
      int r = q.back().sec;
      int min_pos = r_min.qry(l, r);
      q.pop_back();
      lli qt = (min_pos - l + 1) * 1ll * (r - min_pos + 1);
      ans -= (v[min_pos] * 1ll * qt);
      if (min_pos > l)
        q.pb({l, min_pos - 1});
      if (min_pos < r)
        q.pb({min_pos + 1, r});
    }
  }
  cout << ans << endl;
  return 0;
}
// https://github.com/brunomaletta/Biblioteca/blob/master/Codigo/Estruturas/rmq.
    cpp
// O(n) pra buildar, query O(1)
// qry(l, r) -> retorna o indice do menor elemento no range [l, r]
// query(l, r) -> retorna o menor elemento no range [l, r]

// problema exemplo: https://codeforces.com/contest/817/problem/D
```

7.17 color update

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 500001
#define mod 1000000007

const int inf = 1e15;

struct color_upd
{
#define left fir
#define right sec.fir
#define color sec.sec
  set<pii> ranges;
  vector<pii> erased;

  color_upd(int n) // inicialmente, todo mundo pintado com a cor inf
  {
    // nao usar cores negativas!!!!!!!!!
    ranges.insert({0, {n - 1, inf}});
  }
  int get(int i) // qual a cor do elemento na posicao i
  {
    auto it = ranges.upper_bound({i, {1e18, 1e18}});
    if (it == ranges.begin())
      return -1;
    it--;
    return (*(it)).color;
  }
  void del(int l, int r) // apaga o intervalo [l, r]
  {
    erased.clear();
    auto it = ranges.upper_bound({l, {0, 0}});
    if (it != ranges.begin())
    {
      it--;
    }
    while (it != ranges.end())
    {
      if ((*(it)).left > r)
        break;
      else if ((*(it)).right >= l)
        erased.push_back(*it);
      it++;
    }
    if (erased.size() > 0)
    {
      int sz = erased.size();
      auto it = ranges.lower_bound({erased[0].left, {0, 0}});
      auto it2 = ranges.lower_bound({erased[sz - 1].left, {0, 0}});
      pii ini = *it, fim = *it2;
      it2++;
      ranges.erase(it, it2);
      pii upd1 = {ini.left, {l - 1, ini.color}};
      pii upd2 = {r + 1, {fim.right, fim.color}};
      erased[0].left = max(erased[0].left, l);
      erased[sz - 1].right = min(erased[sz - 1].right, r);
      if (upd1.left <= upd1.right)
        ranges.insert(upd1);
      if (upd2.left <= upd2.right)
        ranges.insert(upd2);
    }
  }
  void add(int a, int b, int c) // nao ter dois intervalos adjacentes com a
      mesma cor no set de ranges
  {
    auto it = ranges.lower_bound({a, {b, 0}});
    pii aa = {-1, {-1, -1}};
    pii bb = {-1, {-1, -1}};
    if (it != ranges.end())
    {
      if ((*it).color == c && (*it).left == b + 1)
      {
        aa = *it;
        b = (*it).right;
      }
    }
    if (it != ranges.begin())
    {
      it--;
      if ((*it).color == c && (*it).right == a - 1)
      {
        bb = *it;
        a = (*it).left;
      }
    }
    ranges.erase(aa);
    ranges.erase(bb);
    ranges.insert({a, {b, c}});
  }
  void upd(int a, int b, int c) // pinta o intervalo [a, b] com a cor c
  {
    del(a, b);
    add(a, b, c);
  }
};
struct segtree
{
  vector<int> seg;
  vector<int> lazy;

  segtree(int n)
  {
    seg.resize(4 * n, 0);
    lazy.assign(4 * n, 0);
  }
  int single(int x)
  {
    return x;
  }
  int neutral()
  {
    return 0;
  }
  int merge(int a, int b)
  {
    return a + b;
  }
  void add(int i, int l, int r, int diff)
  {
    seg[i] += (r - l + 1) * diff;
    if (l != r)
    {
      lazy[i << 1] += diff;
      lazy[(i << 1) | 1] += diff;
    }
    lazy[i] = 0;
  }
  void update(int i, int l, int r, int ql, int qr, int diff)
  {
    if (lazy[i])
      add(i, l, r, lazy[i]);
    if (l > r || l > qr || r < ql)
      return;
    if (l >= ql && r <= qr)
    {
      add(i, l, r, diff);
      return;
    }
    int mid = (l + r) >> 1;
    update(i << 1, l, mid, ql, qr, diff);
    update((i << 1) | 1, mid + 1, r, ql, qr, diff);
    seg[i] = merge(seg[i << 1], seg[(i << 1) | 1]);
  }
  int query(int l, int r, int ql, int qr, int i)
  {
    if (lazy[i])
      add(i, l, r, lazy[i]);
    if (l > r || l > qr || r < ql)
      return neutral();
    if (l >= ql && r <= qr)
      return seg[i];
    int mid = (l + r) >> 1;
    return merge(query(l, mid, ql, qr, i << 1), query(mid + 1, r, ql, qr, (i <<
         1) | 1));
  }
};
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n, q;
  cin >> n >> q;
  color_upd c = color_upd(n);
  segtree s = segtree(n);
  for (int i = 0; i < n; i++)
    c.upd(i, i, i + 1);
  while (q--)
  {
    int t;
    cin >> t;
    if (t == 1)
    {
      int l, r, x;
      cin >> l >> r >> x;
      l--, r--;
      c.upd(l, r, x);
      for (auto const &i : c.erased)
        s.update(1, 0, n - 1, i.left, i.right, abs(x - i.color));
    }
    else
    {
      int l, r;
      cin >> l >> r;
      l--, r--;
      cout << s.query(0, n - 1, l, r, 1) << endl;
    }
  }
  return 0;
}
// https://codeforces.com/contest/444/problem/C
```

7.18 segtree max seg sum

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define MAXN 200005
#define mod 1000000007

struct item
{
  int sg_max, pref_max, suf_max, sg_min, pref_min, suf_min, sum;
};
struct segtree
{
  int n;
  vector<item> seg;

  item single(int x)
  {
    return {max(0ll, x), max(0ll, x), max(0ll, x), min(0ll, x), min(0ll, x), min
         (0ll, x), x};
  }
  item neutral()
  {
    return {0, 0, 0, 0, 0, 0, 0};
  }
  segtree() {}
  segtree(int sz)
  {
    n = sz;
    seg.assign(4 * n, neutral());
  }
  item merge(item a, item b)
  {
    item ret;
    ret.sg_max = max({a.sg_max, b.sg_max, a.suf_max + b.pref_max});
    ret.pref_max = max(a.pref_max, a.sum + b.pref_max);
    ret.suf_max = max(b.suf_max, b.sum + a.suf_max);
    ret.sg_min = min({a.sg_min, b.sg_min, a.suf_min + b.pref_min});
    ret.pref_min = min(a.pref_min, a.sum + b.pref_min);
    ret.suf_min = min(b.suf_min, b.sum + a.suf_min);
    ret.sum = a.sum + b.sum;
    return ret;
  }
  void update(int i, int l, int r, int q, int x)
  {
    if (l == r)
    {
      seg[i] = single(x);
      return;
    }
    int mid = (l + r) >> 1;
    if (q <= mid)
      update(i << 1, l, mid, q, x);
    else
      update((i << 1) | 1, mid + 1, r, q, x);
    seg[i] = merge(seg[i << 1], seg[(i << 1) | 1]);
  }
  item query(int l, int r, int ql, int qr, int i)
  {
    int mid = (l + r) >> 1;
    if (l > r || l > qr || r < ql)
      return neutral();
    if (l >= ql && r <= qr)
      return seg[i];
    return merge(query(l, mid, ql, qr, i << 1), query(mid + 1, r, ql, qr, (i <<
         1) | 1));
  }
};
signed main()
{
  return 0;
}
// segtree for maximum segment sum
// me enrolo pra codar toda vez, e bom deixar na lib
```

7.19 fenwick

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 200005
#define mod 998244353

struct fenw
{
  int n;
  vector<int> bit;
  fenw() {}
  fenw(int sz)
  {
    n = sz;
    bit.assign(sz + 1, 0);
  }
  int qry(int r) // query de prefixo a[0] + a[1] + ...  a[r]
  {
    int ret = 0;
    for (int i = r + 1; i > 0; i -= i & -i)
      ret += bit[i];
    return ret;
  }
  void upd(int r, int x) // a[r] += x
  {
    for (int i = r + 1; i <= n; i += i & -i)
      bit[i] += x;
  }
  int bs(int x) // retorna o maior indice i (i < n) tal que: qry(i) < x
  {
    int i = 0, k = 0;
    while (1 << (k + 1) <= n)
      k++;
    while (k >= 0)
    {
      int nxt_i = i + (1 << k);
      if (nxt_i <= n && bit[nxt_i] < x)
      {
        i = nxt_i;
        x -= bit[i];
      }
      k--;
    }
    return i - 1;
  }
};
```

7.20 implicit seg

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 200005
#define mod 998244353

struct implcit_seg
{
  int l, r;
  int sum, lazy;
  implcit_seg *left_child = nullptr;
  implcit_seg *right_child = nullptr;

  implcit_seg(int l, int r) : l(l), r(r)
  {
    sum = 0;
    lazy = 0;
  }
  void check_childs()
  {
    if (!left_child && l != r)
    {
      int mid = (l + r) >> 1;
      left_child = new implcit_seg(l, mid);
      right_child = new implcit_seg(mid + 1, r);
    }
  }
  void add(int x)
  {
    sum += (r - l + 1) * x;
    if (l != r)
    {
      check_childs();
      left_child->lazy += x;
      right_child->lazy += x;
    }
    lazy = 0;
  }
  void upd(int ql, int qr, int x)
  {
    add(lazy);
    if (l > r || l > qr || r < ql)
      return;
    if (l >= ql && r <= qr)
    {
      add(x);
      return;
    }
    check_childs();
    left_child->upd(ql, qr, x);
    right_child->upd(ql, qr, x);
    sum = left_child->sum + right_child->sum;
  }
  void upd(int k, int x)
  {
    sum += x;
    check_childs();
    if (left_child)
    {
      if (k <= left_child->r)
        left_child->upd(k, x);
      else
        right_child->upd(k, x);
    }
  }
  int qry(int ql, int qr)
  {
    add(lazy);
    if (l > r || l > qr || r < ql)
      return 0;
    if (l >= ql && r <= qr)
      return sum;
    check_childs();
    return left_child->qry(ql, qr) + right_child->qry(ql, qr);
  }
};
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n, q;
  cin >> n >> q;
  implcit_seg *s = new implcit_seg(0, n - 1);
  while (q--)
  {
    int t;
    cin >> t;
    if (t == 1)
    {
      int l, r, x;
      cin >> l >> r >> x;
      if (l == r - 1) // point update
        s->upd(l, x);
      else // range update
        s->upd(l, r - 1, x);
    }
    else
    {
      int l, r;
      cin >> l >> r;
      cout << s->qry(l, r - 1) << endl; // range sum
    }
  }
  return 0;
}
```

7.21 fenwick2

```
// fenwick com update pro range [l, r]
// complexidade O(q * log(n)) com a criacao de duas bits ao inves de uma
#include <bits/stdc++.h>
using namespace std;

#define PI acos(-1)
#define int long long int
#define pb push_back
#define mp make_pair
#define pi pair<string, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 100001
#define MAXL 20
#define mod 998244353

int n;
vector<int> bit, bit2;

void add1(int idx, int delta)
{
  for (; idx < n; idx = idx | (idx + 1))
    bit[idx] += delta;
}
void add2(int idx, int delta)
{
  for (; idx < n; idx = idx | (idx + 1))
    bit2[idx] += delta;
}
void update_range(int val, int l, int r)
{
  add1(l, val);
  add1(r + 1, -val);
  add2(l, val * (l - 1));
  add2(r + 1, -val * r);
}
int sum1(int r)
{
  int ret = 0;
  for (; r >= 0; r = (r & (r + 1)) - 1)
    ret += bit[r];
  return ret;
}
int sum2(int r)
{
  int ret = 0;
  for (; r >= 0; r = (r & (r + 1)) - 1)
    ret += bit2[r];
  return ret;
}
int sum(int x)
{
  return (sum1(x) * x) - sum2(x);
}
int range_sum(int l, int r)
{
  return sum(r) - sum(l - 1);
}
int main()
{
  bit.assign(MAXN, 0); // inicializar sempre
  bit2.assign(MAXN, 0); // inicializar sempre
  update_range(x, l, r); // pra cada elemento em [l, r] += x
  range_sum(l, r);  // soma de [l, r]
}
```

7.22 mo

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

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

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

#define PI acos(-1)
#define pb push_back
#define int long long int
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 500005
#define mod 1000000007

int n, q;
int v[MAXN];

namespace mo
{
  struct query
  {
    int idx, l, r;
  };

  int block;
  vector<query> queries;
  vector<int> ans;

  bool cmp(query x, query y)
  {
    if (x.l / block != y.l / block)
      return x.l / block < y.l / block;
    return x.r < y.r;
  }
  void run()
  {
    block = (int)sqrt(n);
    sort(queries.begin(), queries.end(), cmp);
    ans.resize(queries.size());
    int cl = 0, cr = -1, sum = 0;
    auto add = [&](int x)
    {
      sum += x;
    };
    auto rem = [&](int x)
    {
      sum -= x;
    };
    for (int i = 0; i < queries.size(); i++)
    {
      while (cl > queries[i].l)
      {
        cl--;
        add(v[cl]);
      }
      while (cr < queries[i].r)
      {
        cr++;
        add(v[cr]);
      }
      while (cl < queries[i].l)
      {
        rem(v[cl]);
        cl++;
      }
      while (cr > queries[i].r)
      {
        rem(v[cr]);
        cr--;
      }
      ans[queries[i].idx] = sum;
    }
  }
}
```

```
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> n >> q;
  for (int i = 0; i < n; i++)
    cin >> v[i];
  for (int i = 0; i < q; i++)
  {
    mo::query curr;
    cin >> curr.l >> curr.r;
    curr.r--;
    curr.idx = i;
    mo::queries.pb(curr);
  }
  mo::run();
  for (auto const &i : mo::ans)
    cout << i << endl;
}
// to test: https://judge.yosupo.jp/problem/static_range_sum
```

7.23 persistent seg

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define MAXN 100003
#define mod 1000000007

int v[MAXN];

namespace seg
{
  struct node
  {
    int item, lazy, lazy_status, l, r;
    node() {}
    node(int l, int r, int lazy, int lazy_status, int item) : l(l), r(r), lazy(
         lazy), lazy_status(lazy_status), item(item) {}
  };

  vector<node> seg;
  vector<int> roots;

  void init()
  {
    seg.resize(1);
  }
  int neutral()
  {
    return 0;
  }
  int merge(int a, int b)
  {
    return a + b;
  }
  int newleaf(int vv)
  {
    int p = seg.size();
    seg.pb(node(0, 0, 0, 0, vv));
    return p;
  }
  int newparent(int l, int r)
  {
    int p = seg.size();
    seg.pb(node(l, r, 0, 0, merge(seg[l].item, seg[r].item)));
    return p;
  }
  int newkid(int i, int diff, int l, int r)
  {
    int p = seg.size();
    seg.pb(node(seg[i].l, seg[i].r, seg[i].lazy + diff, 1, seg[i].item + ((r - l
          + 1) * diff)));
    return p;
  }
  void add(int i, int l, int r)
  {
    if (!seg[i].lazy_status)
      return;
    if (l != r)
    {
      int mid = (l + r) >> 1;
      seg[i].l = newkid(seg[i].l, seg[i].lazy, l, mid);
      seg[i].r = newkid(seg[i].r, seg[i].lazy, mid + 1, r);
    }
    seg[i].lazy = 0;
    seg[i].lazy_status = 0;
  }
  int update(int i, int l, int r, int ql, int qr, int diff)
  {
    if (l > r || l > qr || r < ql)
      return i;
    if (l >= ql && r <= qr)
      return newkid(i, diff, l, r);
    add(i, l, r);
    int mid = (l + r) >> 1;
    return newparent(update(seg[i].l, l, mid, ql, qr, diff), update(seg[i].r,
         mid + 1, r, ql, qr, diff));
  }
  int query(int l, int r, int ql, int qr, int i)
  {
    if (l > r || l > qr || r < ql)
      return neutral();
    if (l >= ql && r <= qr)
      return seg[i].item;
    add(i, l, r);
    int mid = (l + r) >> 1;
    return merge(query(l, mid, ql, qr, seg[i].l), query(mid + 1, r, ql, qr, seg[
         i].r));
  }
  int build(int l, int r)
  {
    if (l == r)
      return newleaf(v[l]);
    int mid = (l + r) >> 1;
    return newparent(build(l, mid), build(mid + 1, r));
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n, q;
  cin >> n >> q;
  for (int i = 0; i < n; i++)
    cin >> v[i];
  seg::init();
  int root = seg::build(0, n - 1);
  seg::roots.pb(root);
  while (q--)
  {
    char t;
    cin >> t;
    if (t == 'C')
    {
      int l, r, d;
      cin >> l >> r >> d;
      l--, r--;
      int root = seg::update(seg::roots.back(), 0, n - 1, l, r, d);
      seg::roots.pb(root);
    }
    else if (t == 'Q')
    {
      int l, r;
      cin >> l >> r;
      l--, r--;
      cout << seg::query(0, n - 1, l, r, seg::roots.back()) << endl;
    }
    else if (t == 'H')
    {
      int l, r, d;
      cin >> l >> r >> d;
      l--, r--;
      cout << seg::query(0, n - 1, l, r, seg::roots[d]) << endl;
    }
    else
    {
      int d;
      cin >> d;
      while (seg::roots.size() > d + 1)
        seg::roots.pop_back();
    }
  }
  return 0;
}
// https://www.spoj.com/problems/TTM/
// rollback segtree to a time stamp t
```

7.24 binary lifting

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 500001
#define mod 1000000007

struct item
{
  int nxt, sum;
};

int n, q;
int v[MAXN];
item st[MAXN][21];

signed main()
{
  cin >> n >> q;
  for (int i = 0; i < n; i++)
    cin >> v[i];
  for (int i = 0; i < n; i++)
  {
    st[i][0].nxt = min(i + 1, n - 1);
    st[i][0].sum = v[st[i][0].nxt];
  }
  for (int i = 1; i < 21; i++)
  {
    for (int v = 0; v < n; v++)
    {
      st[v][i].nxt = st[st[v][i - 1].nxt][i - 1].nxt;
      st[v][i].sum = st[v][i - 1].sum + st[st[v][i - 1].nxt][i - 1].sum;
    }
  }
  while (q--)
  {
    int l, r;
    cin >> l >> r;
    r--;
    int ans = v[l], len = r - l;
    for (int i = 20; i >= 0; i--)
    {
      if (len & (1 << i))
      {
        ans += st[l][i].sum;
        l = st[l][i].nxt;
      }
    }
    cout << ans << endl;
  }
  return 0;
}
// simple range sum query with binary lifting
// https://judge.yosupo.jp/problem/static_range_sum
```

7.25 sparsetable

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define PI acos(-1)
#define pb push_back
#define int long long int
#define pi pair<int, int>
#define pii pair<int, pair<int, pi>>
#define fir first
#define sec second
#define DEBUG 0
#define MAXN 10005
#define mod 1000000007

struct rmq
{
  bool is_min;
  vector<vector<int>> st;
  vector<int> log;

  int f(int a, int b) { return (is_min) ? min(a, b) : max(a, b); }
  int qry(int l, int r) { return f(st[l][log[r - l + 1]], st[r - (1 << log[r - l
        + 1]) + 1][log[r - l + 1]]); }
  rmq(vector<int> &v, bool flag)
  {
    is_min = flag;
    int n = v.size();
    log.resize(n + 1);
    log[1] = 0;
    for (int i = 2; i <= n; i++)
      log[i] = log[i / 2] + 1;
    int m = log[n] + 2;
    st.assign(n + 1, vector<int>(m, 0));
    for (int i = 0; i < n; i++)
      st[i][0] = v[i];
    for (int j = 1; j < m; j++)
    {
      for (int i = 0; i + (1 << j) <= n; i++)
        st[i][j] = f(st[i][j - 1], st[i + (1 << (j - 1))][j - 1]);
    }
  }
};
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
}
```

8 Utils

8.1 execution time

```
// https://codeforces.com/blog/entry/57647
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 300005
#define mod 1000000007

signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  // just call clock in the beginning
  clock_t time = clock();

  // ...
  // ...
  // some code here ...
  // ...
  // ...

  // execution time:
  cout << setprecision(3) << fixed << (double)(clock() - time) / CLOCKS_PER_SEC
       << endl;
  return 0;
}
```

8.2 runner

```
# This script does the following:
# 1 - Generate a random testcase
# 2 - Run some "naive" code with this input
# 3 - Run your code with this input
# 4 - Compare the outputs
import os

naive = "brute.cpp" # path to naive code
code = "d.cpp" # path to your code
generator = "g.cpp" # path to test generator

def compile_codes():
    os.system('g++ ' + generator + ' -o generator -O2')
    os.system('g++ ' + naive + ' -o naive -O2')
    os.system('g++ ' + code + ' -o code -O2')

def generate_case():
    os.system('./generator > in');

def get_naive_output():
    output = os.popen('./naive <in').read()
    return output

def get_code_output():
    output = os.popen('./code <in').read()
    return output

def main():
    compile_codes()

    while True:
        generate_case()
        naive_output = get_naive_output()
        code_output = get_code_output()

        if naive_output == code_output:
            print('ACCEPTED')
        else :
            print('FAILED\n')
            print('ANSWER:')
            print(naive_output)
            print('\nCODE OUTPUT:')
            print(code_output)
            break

if __name__ == '__main__':
    main()
```

8.3 int128

```
// https://codeforces.com/blog/entry/75044
// functions to print and read a __int128 in c++
__int128 read()
{
  __int128 x = 0, f = 1;
  char ch = getchar();
  while (ch < '0' || ch > '9')
  {
    if (ch == '-')
      f = -1;
    ch = getchar();
  }
  while (ch >= '0' && ch <= '9')
  {
    x = x * 10 + ch - '0';
    ch = getchar();
  }
  return x * f;
}
void print(__int128 x)
{
  if (x < 0)
  {
    cout << "-";
    x = -x;
  }
  if (x > 9)
    print(x / 10);
  char at = (x % 10) + '0';
  cout << at;
}
```

8.4 runner2

```
# This script does the following:
# 1 - Run a code with all inputs files from a folder
# 2 - Compare the output for each test case with the answer
import os

code = "a.cpp" # Path to your code
input_folder = "input" # Path to folder which the input files are
output_folder = "output" # Path to folder which the output files are
input_prefix = "L_" # prefix of all input files names
output_prefix = "L_" # prefix of all input files names
tests = 56 # Number of test cases

def compile_code():
    os.system('g++ ' + code + ' -o code -O2')

def get_ans(output):
    out = open(output, "r")
    ret = out.read()
    out.close()
    return ret

def get_code_output(input):
    output = os.popen('./code <' + input).read()
    return output

def main():
    compile_code()
    # tests indexed from 1
    for i in range (1, tests + 1):
        ans = get_ans(output_folder + '/' + output_prefix + str(i))
        code_output = get_code_output(input_folder + '/' + input_prefix + str(i)
            )
        print('Case' + str(i) + ': ')
        if ans == code_output:
            print('ACCEPTED')
        else :
            print('FAILED\n')
            print('ANSWER:')
            print(ans)
            print('\nCODE OUTPUT:')
            print(code_output)
        print()

if __name__ == '__main__':
    main()
```

8.5 rand

```
// https://codeforces.com/blog/entry/61587
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 300005
#define mod 1000000007

signed main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    mt19937 rng(chrono::steady_clock::now().time_since_epoch().count());

    int n = 10;
    vector<int> v(n);
    for (int i = 0; i < n; i++)
        v[i] = i;
    shuffle(v.begin(), v.end(), rng); // random shuffle

    int x = rng() % 10; // better than rand()
    cout << x << endl;

    // random integer values on the closed interval [a, b]
    int y = uniform_int_distribution<int>(3, 77)(rng);
    cout << y << endl;

    return 0;
}
```

9 Dynamic programming and common problems

9.1 divideandconquer

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 200005
#define mod 1000000007

int s[8005];
int dp[3005][8005];

int cost(int l, int r)
{
  return (s[r + 1] - s[l]) * (r - l + 1);
}
void compute(int l, int r, int optl, int optr, int i)
{
  if (l > r)
    return;
  int mid = (l + r) >> 1;
  pair<int, int> ans = {1e18, -1}; // dp, k
  for (int q = optl; q <= min(mid, optr); q++)
  {
    if (q > 0)
      ans = min(ans, {dp[i - 1][q - 1] + cost(q, mid), q});
    else
      ans = min(ans, {cost(q, mid), q});
  }
  dp[i][mid] = ans.fir;
  compute(l, mid - 1, optl, ans.sec, i);
  compute(mid + 1, r, ans.sec, optr, i);
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n, g;
  cin >> n >> g;
  for (int i = 1; i <= n; i++)
  {
    cin >> s[i];
    s[i] += s[i - 1];
  }
  for (int i = 0; i <= g; i++)
  {
    for (int j = 0; j <= n; j++)
      dp[i][j] = 1e18;
  }
  for (int i = 1; i <= g; i++)
    compute(0, n - 1, 0, n - 1, i);
  cout << dp[g][n - 1] << endl;
  return 0;
}
// https://codeforces.com/gym/103536/problem/A
// https://codeforces.com/contest/321/problem/E

// otimizacao de dp usando divide and conquer
// para dps do tipo:
// dp[i][j] = min(dp[i - 1][k] + c(k, j)), para algum k <= j
// considerando opt(i, j) o menor valor de k que minimiza dp[i][j]
// podemos calcular opt(i, j) usando divide and conquer
// isso diminuiria a complexidade para O(k * n * log(n))
```

9.2 suffix sum

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 100005
#define mod 1000000007

int n, k;
int a[101];
int dp[101][100005];
int sum[101][100005];

int solve(int i, int j);

void calc(int i)
{
  sum[i][0] = solve(i, 0);
  for (int j = 1; j <= k; j++)
    sum[i][j] = (sum[i][j - 1] + solve(i, j)) % mod;
}
int solve(int i, int j)
{
  if (i == n)
    return (j == 0) ? 1 : 0;
  if (dp[i][j] != -1)
    return dp[i][j];
  int ans = 0, limit = min(j, a[i]);
  if (sum[i + 1][j] == -1)
    calc(i + 1);
  ans = (ans + sum[i + 1][j]) % mod;
  if (j - limit - 1 >= 0)
    ans = (ans - sum[i + 1][j - limit - 1] + mod) % mod;
  return dp[i][j] = ans;
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> n >> k;
  for (int i = 0; i < n; i++)
    cin >> a[i];
  memset(dp, -1, sizeof(dp));
  memset(sum, -1, sizeof(sum));
  cout << solve(0, k) << endl;
  return 0;
}
// uma dp que tem uma recorrencia do tipo:
// dp[i][j] = dp[i + 1][l] + dp[i + 1][l + 1] + ... + dp[i + 1][r]
// o jeito de fazer isso com soma de sufixo sem ter que codar a dp iterativa :)
```

9.3 steiner tree

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 4007
#define mod 998244353

const int inf = 1e18;

int n, m, k, sz;
int dx[] = {-1, 1, 0, 0};
int dy[] = {0, 0, 1, -1};
int a[101][101];

vector<int> g[201];
int w[201][201];
int dist[201][201];
int par_floyd[201][201];
int dp[201][1 << 8];
char anss[201][201];
vector<pi> par[201][1 << 8];
int pos[10];

int steiner()
{
  // floyd warshall
  for (int i = 0; i < sz; i++)
  {
    for (int j = 0; j < sz; j++)
    {
      if (i == j)
      {
        dist[i][j] = 0;
      }
      else
      {
        dist[i][j] = w[i][j];
        par_floyd[i][j] = j;
      }
    }
  }
  for (int k = 0; k < sz; k++)
  {
    for (int i = 0; i < sz; i++)
    {
      for (int j = 0; j < sz; j++)
      {
        if (dist[i][j] > dist[i][k] + dist[k][j])
        {
          dist[i][j] = dist[i][k] + dist[k][j];
          par_floyd[i][j] = par_floyd[i][k];
        }
      }
    }
  }
  for (int i = 0; i < sz; i++)
  {
    for (int j = 0; j < (1 << k); j++)
      dp[i][j] = inf;
  }
  for (int i = 0; i < k; i++)
  {
    for (int j = 0; j < sz; j++)
    {
      dp[j][1 << i] = dist[pos[i]][j];
      par[j][1 << i] = {{pos[i], 0}};
    }
  }
  for (int mask = 2; mask < (1 << k); mask++)
  {
    for (int i = 0; i < sz; i++)
    {
      for (int mask2 = mask; mask2 > 0; mask2 = (mask2 - 1) & mask)
      {
        int mask3 = mask ^ mask2;
        if (dp[i][mask] > dp[i][mask2] + dp[i][mask3])
        {
          dp[i][mask] = dp[i][mask2] + dp[i][mask3];
          par[i][mask] = {{i, mask2}, {i, mask3}};
        }
      }
      for (int j = 0; j < sz; j++)
      {
        if (dp[j][mask] > dp[i][mask] + dist[i][j])
        {
          dp[j][mask] = dp[i][mask] + dist[i][j];
          par[j][mask] = {{i, mask}};
        }
      }
    }
  }
  // preciso somar a[i / m][i % m] pq tou usando a[i][j]
  // como peso de uma aresta (i, j) -> (x, y)
  // mas eh especifico para esse problema do garden
  int ans = inf, best = -1;
  for (int i = 0; i < sz; i++)
  {
    int curr = a[i / m][i % m] + dp[i][(1 << k) - 1];
    if (curr < ans)
    {
      ans = curr;
      best = i;
    }
  }
  // recuperar resposta
  queue<pi> q;
  q.push({best, (1 << k) - 1});
  while (!q.empty())
  {
    auto [i, mask] = q.front();
    q.pop();
    anss[i / m][i % m] = 'X';
    for (auto [j, mask2] : par[i][mask])
    {
      // marcar o caminho de j para i feito pelo floyd warshall
      int st = j, en = i;
      while (st != en)
      {
        st = par_floyd[st][en];
        anss[st / m][st % m] = 'X';
      }
      q.push({j, mask2});
    }
  }
  return ans;
}
signed main()
{
  ios::sync_with_stdio(false);
  cin.tie(0);
  cin >> n >> m >> k;
  for (int i = 0; i < n; i++)
  {
    for (int j = 0; j < m; j++)
    {
      cin >> a[i][j];
      anss[i][j] = '.';
    }
  }
  sz = (n * m);
  for (int i = 0; i < sz; i++)
  {
    for (int j = 0; j < sz; j++)
      w[i][j] = inf;
  }
  // montando o grafo
  // (i, j) -> (x, y) com peso a[i][j]
  for (int i = 0; i < n; i++)
  {
    for (int j = 0; j < m; j++)
    {
      for (int d = 0; d < 4; d++)
      {
        int x = i + dx[d];
        int y = j + dy[d];
        if (x >= 0 && x < n && y >= 0 && y < m)
          w[(i * m) + j][(x * m) + y] = a[i][j];
      }
    }
  }
  // posicoes importantes
  for (int i = 0; i < k; i++)
  {
    int x, y;
    cin >> x >> y;
    x--, y--;
    pos[i] = (x * m) + y;
  }
  cout << steiner() << endl;
  for (int i = 0; i < n; i++)
  {
    for (int j = 0; j < m; j++)
      cout << anss[i][j];
    cout << endl;
  }
  return 0;
}
// https://codeforces.com/problemset/problem/152/E
// dada uma matriz representando um jardim
// a posicao (i, j) tem a[i][j] flores
// com isso, quero cobrir algumas posicoes com concreto
// quando cobrimos uma posicao (i, j) com concreto, "matamos" as a[i][j] flores
     daquela posicao
// existem k <= 7 posicoes importantes, que devem ser cobertas com concreto
// alem disso, posso cobrir qualquer outra posicao com concreto
// alem disso, para duas posicoes a e b que sao importantes, deve existir um
     caminho
// de a ate b passando somente por posicoes cobertas por concreto.
// quero minimizar o numero de flores mortas, satisfazendo essas condicoes

// o que queremos nesse caso, eh uma steiner tree
// dado um grafo, com peso nas arestas
// e um subconjunto de vertices
// queremos achar uma arvore de menor peso que contenha todos os vertices do
     subconjunto
// mas essa arvore pode conter tambem outros vertices que nao estao no
     subconjunto
// minimizando o peso total das arestas da arvore

// um outro problema de steiner tree: https://codeforces.com/gym/101908/problem/
     J
```

9.4 lcs

```
//Dadas duas sequencias s1 e s2, uma de tamanho n e outra de tamanho m, qual a
     maior subsequencia comum as duas?

// uma subsequencia de s e um subconjunto dos elementos de s na mesma ordem em
     que apareciam antes.
// isto significa que {1, 3, 5} e uma subsequencia de {1, 2, 3, 4, 5}, mesmo 1
     nao estando do lado do 3.
#include <bits/stdc++.h>
using namespace std;

#define lli long long int
#define pb push_back
#define in insert
#define pi pair<int, int>
#define pii pair<int, pi>
#define mp make_pair
#define fir first
#define sec second
#define MAXN 1001
#define INF 1000000000
int v1[MAXN];
int v2[MAXN];
int dp[MAXN][MAXN];

void lcs(int m, int n)
{
  for (int i = 0; i <= m; i++)
  {
    for (int j = 0; j <= n; j++)
    {
      if (i == 0 || j == 0) //se uma das sequencias for vazia
        dp[i][j] = 0;
      else if (v1[i - 1] == v2[j - 1]) // se eh igual, adiciono a lcs e subtraio
           dos dois
        dp[i][j] = dp[i - 1][j - 1] + 1;
      else
        dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]); // se nao retorno o maximo
             entre tirar um dos dois caras
    }
  }
  cout << dp[m][n] << endl;
}
signed main()
{
  int n, m;
  cin >> n >> m;
  for (int i = 0; i < n; i++)
    cin >> v1[i];
  for (int i = 0; i < m; i++)
    cin >> v2[i];
  lcs(n, m);
  return 0;
}
```

9.5 bitwise digit dp

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 65
#define mod 1000000007

int a, b;
// current bit / i is bigger than l / i is lower than r / j is bigger than l / j
      is lower than r
int dp[MAXN][2][2][2][2];

int solve(int i, int j, int k, int l, int m)
{
  if (i < 0)
    return (j && k && l && m) ? 1 : 0;
  if (dp[i][j][k][l][m] != -1)
    return dp[i][j][k][l][m];
  int ret = 0;
  int ll = a & (1LL << i);
  int rr = b & (1LL << i);
  if ((j || !ll) && (l || !ll))
    ret += solve(i - 1, j, (rr) ? 1 : k, l, (rr) ? 1 : m);
  if ((k || rr) && (l || !ll))
    ret += solve(i - 1, (!ll) ? 1 : j, k, l, (rr) ? 1 : m);
  if ((m || rr) && (j || !ll))
```

```
    ret += solve(i - 1, j, (rr) ? 1 : k, (!ll) ? 1 : l, m);
  return dp[i][j][k][l][m] = ret;
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int q;
  cin >> q;
  while (q--)
  {
    cin >> a >> b;
    a--, b++;
    memset(dp, -1, sizeof(dp));
    if (a == -1)
      cout << solve(60, 1, 0, 1, 0) << endl;
    else
      cout << solve(60, 0, 0, 0, 0) << endl;
  }
  return 0;
}
// https://codeforces.com/contest/1245/problem/F
// https://codeforces.com/blog/entry/88064
// count the number of pairs (i, j) which (i & j) == 0
```

9.6 aliens trick

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 500001
#define mod 1000000007

int n, k, l;
string s;

pi solve(vector<int> &v, int lambda)
{
  // associar um custo lambda para ser subtraido quando realizamos uma operacao
  // dp[i] - melhor profit que tivemos considerando as i primeiras posicoes
  // cnt[i] - quantas operacoes utilizamos para chegarno valor de dp[i]
  vector<int> dp(n + 1);
  vector<int> cnt(n + 1);
  dp[0] = 0;
  cnt[0] = 0;
  for (int i = 1; i <= n; i++)
  {
    dp[i] = dp[i - 1];
    cnt[i] = cnt[i - 1];
    int id = i - 1;
    dp[i] += v[id];
    int lo = max(0ll, id - l + 1);
    int s = dp[lo] + (id - lo + 1) - lambda;
    if (s > dp[i])
    {
      dp[i] = s;
      cnt[i] = cnt[lo] + 1;
    }
  }
  return {dp[n], cnt[n]};
}
int aliens_trick(vector<int> &v)
{
  int l = 0, r = n;
  while (l < r)
  {
    int mid = (l + r) >> 1;
    pi ans = solve(v, mid);
    (ans.sec > k) ? l = mid + 1 : r = mid;
  }
  pi ans = solve(v, l);
  return ans.fir + (l * k);
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> n >> k >> l >> s;
  vector<int> a(n);
  vector<int> b(n);
  for (int i = 0; i < n; i++)
  {
    a[i] = 1, b[i] = 0;
    if (s[i] >= 'A' && s[i] <= 'Z')
    {
      a[i] ^= 1;
      b[i] ^= 1;
    }
  }
  cout << n - max(aliens_trick(a), aliens_trick(b)) << endl;
  return 0;
}
// https://codeforces.com/contest/1279/problem/F
```

9.7 Digitdp

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define fir first
#define sec second
#define MAXN 2001
#define mod 1000000007

int dp[20][20 * 9][2]; // a,b <= 10^18
vector<int> dig;

int solve(int i, int j, int k)
{
  if (i == dig.size())
    return (k) ? dp[i][j][k] = j : dp[i][j][k] = 0;
  if (dp[i][j][k] != -1)
    return dp[i][j][k];
  int sum = 0;
  if (k)
    for (int f = 0; f <= 9; f++)
      sum += solve(i + 1, j + f, k);
  if (!k)
    for (int f = 0; f <= dig[i]; f++)
      sum += solve(i + 1, j + f, (dig[i] != f) ? 1 : 0);
  return dp[i][j][k] = sum;
}
void get_digits(int n)
{
  dig.clear();
  while (n)
  {
    dig.pb(n % 10);
    n = n / 10;
  }
  reverse(dig.begin(), dig.end());
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int a, b;
  cin >> a >> b;
  get_digits(a);
  memset(dp, -1, sizeof(dp));
  int aa = solve(0, 0, 0);
  get_digits(b + 1);
  memset(dp, -1, sizeof(dp));
  int bb = solve(0, 0, 0);
  cout << bb - aa << endl;
  return 0;
}
```

9.8 broken profile

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define PI acos(-1)
#define pb push_back
#define int long long int
#define pi pair<int, int>
#define pii pair<int, pair<int, pi>>
#define fir first
#define sec second
#define DEBUG 0
#define MAXN 1001
#define mod 1000000007

int n;
vector<int> validmasks;
int dp[MAXN][1 << 4];

void init() // preprocess valid masks
{
  for (int mask = 0; mask < (1 << 7); mask++)
  {
    int nxt_mask = 0, prev_mask = 0, valid = true;
    for (int k = 0; k < 7; k++)
    {
      if (mask & (1 << k))
      {
        if (k <= 3)
        {
          int idx = k, idx2 = k;
          if (nxt_mask & (1 << idx) || prev_mask & (1 << idx2))
            valid = false;
          prev_mask = prev_mask | (1 << idx);
          nxt_mask = nxt_mask | (1 << idx2);
        }
        else
        {
          int idx = k - 4, idx2 = idx + 1;
          if (nxt_mask & (1 << idx) || nxt_mask & (1 << idx2))
            valid = false;
          nxt_mask = nxt_mask | (1 << idx);
          nxt_mask = nxt_mask | (1 << idx2);
        }
      }
    }
    if (valid)
      validmasks.pb(mask);
  }
}
int solve(int i, int j)
{
  if (i == n)
    return (j == ((1 << 4) - 1)) ? 1 : 0;
  if (dp[i][j] != -1)
    return dp[i][j];
  int ret = 0;
  for (auto const &mask : validmasks)
  {
    int nxt_mask = 0, prev_mask = j, valid = true;
    for (int k = 0; k < 7; k++)
    {
      if (mask & (1 << k))
      {
        if (k <= 3)
        {
          int idx = k, idx2 = idx;
          if (prev_mask & (1 << idx) || nxt_mask & (1 << idx2))
            valid = false;
          prev_mask = prev_mask | (1 << idx);
          nxt_mask = nxt_mask | (1 << idx2);
        }
        else
        {
          int idx = k - 4, idx2 = idx + 1;
          if (nxt_mask & (1 << idx) || nxt_mask & (1 << idx2))
            valid = false;
          nxt_mask = nxt_mask | (1 << idx);
          nxt_mask = nxt_mask | (1 << idx2);
        }
      }
    }
    if (valid && prev_mask == ((1 << 4) - 1))
      ret += solve(i + 1, nxt_mask);
  }
  return dp[i][j] = ret;
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int q;
  cin >> q;
  init();
  for (int i = 1; i <= q; i++)
  {
    cin >> n;
    memset(dp, -1, sizeof(dp));
    cout << i << " " << solve(0, (1 << 4) - 1) << endl;
  }
  return 0;
}
// broken profile dp
// if you can fully fill an area with some figures
// finding number of ways to fully fill an area with some figures
// finding a way to fill an area with minimum number of figures
// ...
// https://www.spoj.com/problems/GNY07H/
// We wish to tile a 4xN grid with rectangles 2x1 (in either orientation)
// dp[i][mask]
// i denotes the current column
// mask denotes the situation of the previous column
// our mission is to fill all of the units of
// the previous column in a state [i][mask]
```

9.9 largest square

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

#define PI acos(-1)
#define pb push_back
#define int long long int
#define double long double
#define pi pair<int, int>
#define pii pair<pi, int>
```

```
#define fir first
#define sec second
#define MAXN 1001
#define mod 1000000007

signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n;
  cin >> n;
  int v[n][n];
  int dp[n][n];
  for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
      cin >> v[i][j];
  int ans = 0;
  for (int i = 0; i < n; i++)
  {
    for (int j = 0; j < n; j++)
    {
      dp[i][j] = v[i][j];
      if (i && j && dp[i][j])
        dp[i][j] = min({dp[i][j - 1], dp[i - 1][j], dp[i - 1][j - 1]}) + 1;
      ans = max(ans, dp[i][j]);
    }
  }
  cout << ans * ans << endl;
  return 0;
}
```

9.10 max matrix path

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

#define PI acos(-1)
#define pb push_back
#define int long long int
#define mp make_pair
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define MAXN 301
#define MAXL 20
#define mod 1000000007
#define INF 1000000001

int n;
int grid[MAXN][MAXN];
int dp[MAXN][MAXN];

int solve(int i, int j)
{
  if (i == n - 1 && j == n - 1)
    return grid[i][j];
  if (dp[i][j] != -1)
    return dp[i][j];
  if (i + 1 < n && j + 1 < n)
    return dp[i][j] = grid[i][j] + max(solve(i + 1, j), solve(i, j + 1));
  if (i + 1 < n)
    return dp[i][j] = grid[i][j] + solve(i + 1, j);
  if (j + 1 < n)
    return dp[i][j] = grid[i][j] + solve(i, j + 1);
}
signed main()
{
  cin >> n;
  for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
      cin >> grid[i][j];
  memset(dp, -1, sizeof(dp));
  cout << solve(0, 0) << endl;
  return 0;
}
```

9.11 largest-sum-contiguous-subarray

```
// dada uma sequencia s qual a maior soma que podemos obter escolhendo um
    subconjunto de termos adjacentes de s
// nesse caso o temos apenas duas opcoes
// nao usar o elemento v[i]
// ou
// usamos, adicionando a maior soma possivel que antes dele
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 200001
#define mod 1000000007

int kadane(vector<int> v)
{
  int n = v.size(), ans = 0, max_here = 0;
  for (int i = 0; i < n; i++)
  {
    max_here += v[i];
    if (ans < max_here)
      ans = max_here;
    if (max_here < 0)
      max_here = 0;
  }
  return ans;
}
int kadane_circular(vector<int> v)
{
  int n = v.size(), max_kadane = kadane(v);
  int max_wrap = 0, i;
  for (i = 0; i < n; i++)
  {
    max_wrap += v[i];
    v[i] = -v[i];
  }
  max_wrap += kadane(v);
  return max(max_wrap, max_kadane);
}
signed main()
{
  int n;
  cin >> n;
  vector<int> v(n);
  for (int i = 0; i < n; i++)
    cin >> v[i];
  cout << kadane_circular(v) << endl;
  return 0;
}
```

9.12 exchange arguments

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class T>
```

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 1001
#define mod 1000000009

const int inf = 1e18;

int n;
vector<pi> v;
int dp[MAXN][MAXN];
bool vis[MAXN][MAXN];

int solve(int i, int j)
{
  if (j == 0)
    return inf;
  if (i == n)
    return -inf;
  if (vis[i][j])
    return dp[i][j];
  int ans = -inf;
  ans = max(ans, solve(i + 1, j));
  int ot = min(v[i].sec, solve(i + 1, j - 1) - v[i].fir);
  ans = max(ans, ot);
  vis[i][j] = 1;
  return dp[i][j] = ans;
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> n;
  v.resize(n);
  for (int i = 0; i < n; i++)
  {
    cin >> v[i].fir >> v[i].sec;
    v[i].sec -= v[i].fir;
  }
  auto cmp = [&](pi a, pi b)
  {
    return (a.sec - b.fir) < (b.sec - a.fir);
  };
  sort(v.begin(), v.end(), cmp);
  memset(dp, -1, sizeof(dp));
  int ans = 0;
  for (int i = n; i >= 0; i--)
  {
    if (solve(0, i) >= 0)
    {
      ans = i;
      break;
    }
  }
  cout << ans << endl;
  return 0;
}
// problema:
// existem n caixas, cada uma tem um peso w[i] e uma resistencia r[i]
// voce deve escolher um subset de caixas e empilhar na ordem que vc quiser
// tal que: a soma dos pesos de todas as caixas acima de uma caixa seja menor ou
      igual a resistencia dessa caixa

// dp[i][j] - estou na caixa i e quero escolher mais j caixas para botar na
     pilha
// qual a maior resistencia restante que eu posso obter escolhendo essas j
     caixas

// a grande sacada pra achar a ordenacao otima antes da dp:
// para duas caixas a e b
// quando vai ser stonks botar a antes de b?
// r[a] - w[b] > r[b] - w[a]
// pois a resistencia reestante vai ser maior

// pra demais problemas de exchange argument, essa ideia pode se aplicar
// do tipo, ver o jeito otimo de resolver pro n = 2
// e fazer a ordenacao baseada nisso
```

9.13 tip

```
// dados os valores de moedas v1, v2, ... vn e possivel formar um valor m como
     combinacao de moedas
// para isso basta montar uma dp inicializada com -1
// nesse caso a dp so precisa de um parametro q e = valor restante ate o limite
// mas podem existir variacoes do problema q precise de mais coisas
// se em achar alguma combinacao valida retorna 1, se nao retorna 0
#include <bits/stdc++.h>
using namespace std;

#define lli long long int
#define pb push_back
#define in insert
#define pi pair<int, int>
#define pd pair<double, int>
#define pib pair<pi, bool>
#define mp make_pair
#define fir first
#define sec second
#define MAXN 200001
#define MAXL 10001
#define mod 1000000007

int dp[MAXN];
vector<int> v;

int solve(int rem)
{
  if (rem == 0)
    return 1;
  if (rem < 0)
    return 0;
  if (dp[rem] >= 0)
    return dp[rem];
  for (int i = 0; i < v.size(); i++)
    if (solve(rem - v[i]))
      return dp[rem - v[i]] = 1;
  return dp[rem] = 0;
}
signed main()
{
  int n, m;
  cin >> n >> m;
  v.resize(n);
  for (int i = 0; i < n; i++)
    cin >> v[i];
  memset(dp, -1, sizeof(dp));
  (solve(m)) ? cout << "Yes\n" : cout << "No\n";
  return 0;
}
```

9.14 Knapsack

```
//O problema mais classico de Programacao Dinamica talvez seja o Knapsack.
//De maneira geral, um ladrao ira roubar uma casa com uma mochila
//que suporta um peso s. Ele ve n objetos na casa e sabe estimar o peso pi    e
     o valor vi
//de cada objeto i. Com essas informacoes, qual o maior valor que o ladrao pode
     roubar sem rasgar sua mochila?
#include <bits/stdc++.h>
using namespace std;

#define lli long long int
#define pb push_back
#define in insert
```

```
#define pi pair<int, int>
#define pii pair<int, pi>
#define mp make_pair
#define fir first
#define sec second
#define MAXN 1001
#define INF 1000000000

int n, l;
int value[MAXN];
int peso[MAXN];
int dp[MAXN][MAXN];

int knapsack(int i, int limit)
{
  if (dp[i][limit] >= 0) // se ja foi calculado
  {
    return dp[i][limit];
  }

  if (i == n or !limit) // se chegou no fim do array ou chegou no limite
  {
    return dp[i][limit] = 0;
  }

  int nao_coloca = knapsack(i + 1, limit); // recursivamente pra caso eu nao
       coloque o objeto i

  if (peso[i] <= limit) // se eu consigo botar o objeto i
  {
    int coloca = value[i] + knapsack(i + 1, limit - peso[i]);
    return dp[i][limit] = max(coloca, nao_coloca);
  }

  return dp[i][limit] = nao_coloca;
}
signed main()
{
  cin >> l >> n;
  for (int i = 0; i < n; i++)
  {
    cin >> peso[i] >> value[i];
  }
  memset(dp, -1, sizeof(dp));
  cout << knapsack(0, l) << endl;
  return 0;
}
```

9.15 subsequences string

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

#define PI acos(-1)
#define int long long int
#define pb push_back
#define mp make_pair
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 100
#define MAXL 20
#define mod 998244353

void count(string a, string b)
{
  int m = a.size();
  int n = b.size();
  int dp[m + 1][n + 1] = {{0}};
  for (int i = 0; i <= n; ++i)
    dp[0][i] = 0;
  for (int i = 0; i <= m; ++i)
    dp[i][0] = 1;
  for (int i = 1; i <= m; i++)
  {
    for (int j = 1; j <= n; j++)
    {
      if (a[i - 1] == b[j - 1])
        dp[i][j] = dp[i - 1][j - 1] + dp[i - 1][j];
      else
        dp[i][j] = dp[i - 1][j];
    }
  }
  cout << dp[m][n] << endl;
}
signed main()
{
  string a, b;
  cin >> a >> b;
  count(a, b);
  return 0;
}
```

9.16 dynamic cht

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pf push_front
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 1000005
#define mod 1000000007

struct line
{
  mutable int m, b, p;
  bool operator<(const line &o) const
  {
    if (m != o.m)
      return m < o.m;
    return b < o.b;
  }
  bool operator<(const int x) const { return p < x; }
  int eval(int x) const { return m * x + b; }
  int inter(const line &o) const
  {
    int x = b - o.b, y = o.m - m;
    return (x / y) - ((x ^ y) < 0 && x % y);
  }
};
struct cht
{
  int INF = 1e18;
  multiset<line, less<>> l;
  void add(int m, int b)
  {
    auto y = l.insert({m, b, INF});
    auto z = next(y);
    if (z != l.end() && y->m == z->m)
    {
      l.erase(y);
      return;
    }
    if (y != l.begin())
    {
      auto x = prev(y);
      if (x->m == y->m)
        x = l.erase(x);
    }
    while (1)
    {
      if (z == l.end())
      {
        y->p = INF;
        break;
      }
      y->p = y->inter(*z);
      if (y->p < z->p)
        break;
      else
        z = l.erase(z);
    }
    if (y == l.begin())
      return;
    z = y;
    auto x = --y;
    while (1)
    {
      int ninter = x->inter(*z);
      if (ninter <= x->p)
        x->p = ninter;
      else
      {
        l.erase(z);
        break;
      }
      if (x == l.begin())
        break;
      y = x;
      x--;
      if (x->p < y->p)
        break;
      else
        l.erase(y);
    }
  }
  int get(int x)
  {
    if (l.empty())
      return 0;
    return l.lower_bound(x)->eval(x);
  }
};
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  return 0;
}
// sources:
// https://github.com/pauloamed/Training/blob/master/PD/cht.cpp
// https://github.com/brunomaletta/Biblioteca/blob/master/Codigo/DP/CHT-Dinamico.cpp

// cht dinamico
// dado uma coordenada x
// e um conjunto com varias equacoes lineares da forma: y = mx + c
// retorna o maior valor de y entre as equacoes do conjunto

// para o menor valor, multiplicar m e c de cada equacao por -1
// e multiplicar o resultado da query por -1

// problemas iniciais:
// https://atcoder.jp/contests/dp/tasks/dp_z
// https://codeforces.com/contest/1083/problem/E
```

9.17 lis

```
// dada uma sequencia s qualquer, descobrir o tamanho da maior subsequencia crescente de s
// uma subsequencia de s e qualquer subconjunto de elementos de s.
// Para cada novo numero, voce tem duas operacoes possiveis:
// 1 - Colocar o novo numero no topo de uma pilha se ele nao superar o que ja esta em seu topo;
// ou
// 2 - Criar uma nova pilha a direita de todas as outras e colocar o novo numero la.
// ao final do processo a nossa pilha tera os elementos da lis.
#include <bits/stdc++.h>
using namespace std;

#define lli long long int
#define pb push_back
#define in insert
#define pi pair<int, int>
#define pd pair<double, int>
#define pib pair<pi, bool>
#define mp make_pair
#define fir first
#define sec second
#define MAXN 200001
#define MAXL 1000001
#define mod 1000000007

vector<int> v;

int lis()
{
  vector<int> q;
  for (int i = 0; i < v.size(); i++)
  {
    vector<int>::iterator it = lower_bound(q.begin(), q.end(), v[i]);
    if (it == q.end())
      q.pb(v[i]);
    else
      *it = v[i];
  }
  for (int i = 0; i < q.size(); i++)
    cout << q[i] << " ";
  cout << endl;
  return q.size();
}
signed main()
{
  int n;
  cin >> n;
  v.resize(n);
  for (int i = 0; i < n; i++)
    cin >> v[i];
  cout << lis() << endl;
  return 0;
}
```

9.18 lichao

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define pci pair<char, int>
#define fir first
#define sec second
#define MAXN 100005
#define mod 1000000007

const int inf = 1e18;
```

```
struct line
{
  int a, b, id;
  int ch[2];
  line()
  {
    a = 0, b = inf, id = -1;
    ch[0] = -1, ch[1] = -1;
  }
  line(int aa, int bb, int i)
  {
    a = aa, b = bb, id = i;
    ch[0] = -1, ch[1] = -1;
  }
  int f(int x) { return a * x + b; }
};
struct save
{
  int a, b, id, new_id, p;
};
struct lichao
{
  int lo, hi, curr;
  vector<line> t;
  stack<save> st; // se nao precisar de rollback, pode tirar a stack (pra nao
       usar tanta memoria)

  lichao(int ll, int rr)
  {
    lo = ll, hi = rr;
    t.emplace_back();
  }
  int child(int p, int d)
  {
    if (t[p].ch[d] == -1)
    {
      t[p].ch[d] = t.size();
      t.emplace_back();
    }
    return t[p].ch[d];
  }
  bool cmp(line a, line b, int x)
  {
    if (a.f(x) != b.f(x)) // menor valor em x
      return a.f(x) < b.f(x);
    return a.id > b.id; // desempata pelo maior id
  }
  void add(int l, int r, line s, int p)
  {
    int mid = (l + r) >> 1;
    bool fl = cmp(s, t[p], l);
    bool fm = cmp(s, t[p], mid);
    bool fr = cmp(s, t[p], r);
    if (fm)
    {
      st.push({t[p].a, t[p].b, t[p].id, curr, p});
      swap(t[p], s);
      swap(t[p].ch, s.ch);
    }
    if (s.b == inf)
      return;
    if (fl != fm)
      add(l, mid - 1, s, child(p, 0));
    else if (fr != fm)
      add(mid + 1, r, s, child(p, 1));
  }
  pi query(int l, int r, int x, int p)
  {
    int mid = (l + r) >> 1;
    pi ans = {t[p].f(x), -t[p].id}; // como eu quero o maior id, basta negar e
         pegar o menor
    if (ans.fir == inf)
      return ans;
    if (x < mid)
      return min(ans, query(l, mid - 1, x, child(p, 0)));
    return min(ans, query(mid + 1, r, x, child(p, 1)));
  }
  void add(line s)
  {
    curr = s.id;
    add(lo, hi, s, 0);
  }
  pi qry(int x)
  {
    return query(lo, hi, x, 0);
  }
  void rollback(int id)
  {
    while (!st.empty() && st.top().new_id == id)
    {
      int p = st.top().p;
      t[p].a = st.top().a;
      t[p].b = st.top().b;
      t[p].id = st.top().id;
      st.pop();
    }
  }
};
signed main()
{
  lichao lt(0, 1e9 + 2);
  lt.add(line(3, 2, 0));
  lt.add(line(5, -6, 1));
  cout << lt.qry(10).fir << " " << -lt.qry(10).sec << endl;
}
// li-chao tree
// dado uma coordenada x
// e um conjunto com varias equacoes lineares da forma: y = ax + b
// retorna o menor valor de y entre as equacoes do conjunto
// O(log(hi - lo))
// no qual:
// lo -> menor valor possivel de um x que vai ser passado pra uma query
// hi -> maior valor possivel de um x que vai ser passado pra uma query
```

9.19 expected value

```
//https://atcoder.jp/contests/dp/tasks/dp_j
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define pb push_back
#define mp make_pair
#define pi pair<int, int>
#define pii pair<pi, int>
#define pci pair<char, int>
#define fir first
#define sec second
#define DEBUG 0
#define MAXN 301
#define mod 1000000007

int n;
vector<int> v;
vector<int> cnt(3);
double dp[MAXN][MAXN][MAXN];

double solve(int i, int j, int k)
{
  if (!i && !j && !k)
    return dp[i][j][k] = 0;
  if (dp[i][j][k] != -1)
    return dp[i][j][k];
  /*
  It is well-known from statistics that for the geometric distribution
  (counting number of trials before a success, where each independent trial is
       probability p)
    the expected value is i / p
    */
    double p = ((double)(i + j + k) / n);
    double ret = 1 / p; // expected number of trials before a success
    if (i)
    {
      double prob = (double)i / (i + j + k); // probabilidade de ser um prato com
          um sushi
      ret += (solve(i - 1, j, k) * prob);
    }
    if (j)
    {
      double prob = (double)j / (i + j + k); // probabilidade de ser um prato com
          dois sushis
      ret += (solve(i + 1, j - 1, k) * prob);
    }
    if (k)
    {
      double prob = (double)k / (i + j + k); // probabilidade de ser um prato com
          tres sushis
      ret += (solve(i, j + 1, k - 1) * prob);
    }
    return dp[i][j][k] = ret;
  }
  signed main()
  {
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    cin >> n;
    v.resize(n);
    for (int i = 0; i < n; i++)
      cin >> v[i], cnt[v[i] - 1]++;
    for (int i = 0; i < MAXN; i++)
      for (int j = 0; j < MAXN; j++)
        for (int k = 0; k < MAXN; k++)
          dp[i][j][k] = -1;
    cout << setprecision(15) << solve(cnt[0], cnt[1], cnt[2]) << endl;
    return 0;
  }
```

9.20 subset sum

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 100005
#define mod 1000000007

// subset sum com bitset de tamanho variado
// usada no https://codeforces.com/contest/1856/problem/E2
// with n <= 10^6
template <int len = 1>
int subset_sum(int n, int h)
{
  if (n >= len)
  {
    return subset_sum<std::min(len * 2, (int)MAXN)>(n, h);
  }
  bitset<len> dp;
  dp[0] = 1;
  for (auto const &x : w)
  {
    dp = dp | (dp << x);
  }
  return dp._Find_next(max(0ll, h - 1)); // retorna o proximo bit setado apos a
       posicao passada como parametro
}
int solve(vector<int> &w, int tot, int h)
{
  // tot -> soma de todos os elementos de w
  // h -> valor desejado
  // quero retornar o menor valor x >= h, tal que existe um subset com soma x em
      w
  if (!w.size())
    return 0;
  sort(w.rbegin(), w.rend());
  if (w[0] * 2 >= tot)
    return w[0];
  int n = w.size();
  w.pb(0);
  vector<int> aux;
  int p = 0;
  for (int i = 1; i <= n; i++)
  {
    if (w[i] != w[i - 1])
    {
      int cnt = i - p;
      int x = w[i - 1];
      int j = 1;
      while (j < cnt)
      {
        aux.pb(x * j);
        cnt -= j;
        j *= 2;
      }
      aux.pb(x * cnt);
      p = i;
    }
  }
  swap(aux, w);
  return subset_sum(tot, h);
}

int f[MAXN];      // f[i] -> quantos "itens" com valor i tem
bitset<MAXN> dp; // dp[i] = 1, se existe um subset com soma i
// garantir que a soma de todo mundo seja < MAXN
void subset_sum(vector<int> &v)
{
  for (auto const &i : v)
  {
    f[i]++;
  }
  dp[0] = 1;
  for (int i = 1; i < MAXN; i++)
  {
    while (f[i] > 2)
    {
      f[i * 2]++;
      f[i] -= 2;
    }
    while (f[i]--)
      dp |= (dp << i);
  }
}

// https://github.com/gabrielpessoa1/ICPC-Library/blob/master/code/Miscellaneous
     /SubsetSum.cpp
/*
 Given N non-negative integer weights w and a non-negative target t,
 computes the maximum S <= t such that S is the sum of some subset of the
      weights.
 O(N * max(w[i]))
*/
int knapsack(vector<int> w, int t)
{
  int a = 0, b = 0;
  while (b < w.size() && a + w[b] <= t)
  {
    a += w[b++];
  }
  if (b == w.size())
  {
    return a;
  }
  int m = *max_element(w.begin(), w.end());
  vector<int> u, v(2 * m, -1);
  v[a + m - t] = b;
  for (int i = b; i < w.size(); i++)
  {
    u = v;
    for (int x = 0; x < m; x++)
    {
      v[x + w[i]] = max(v[x + w[i]], u[x]);
    }
    for (int x = 2 * m; --x > m;)
    {
      for (int j = max(0ll, u[x]); j < v[x]; j++)
        v[x - w[j]] = max(v[x - w[j]], j);
    }
  }
  a = t;
  while (v[a + m - t] < 0)
  {
    a--;
  }
  return a;
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  return 0;
}
```

9.21 sos dp

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 100001
#define mod 1000000007

signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  // exemplos de sos dp para calcular f[x] para cada mask x
  // a[x] eh o valor de uma funcao a para uma mask x
  // complexidade: O(M * 2^M), M = numero de bits

  // Exemplo 1:
  // nesse caso, f[x] eh a funcao que soma:
  // todos os a[i], tal que, (x & i) == i)
  // isso eh, i eh uma "mask filha" de x
  // pois todos os bits de i estao setados em x
  for (int mask = 0; mask < (1 << m); mask++)
  {
    f[mask] = a[mask];
  }
  for (int i = 0; i < m; ++i)
  {
    for (int mask = 0; mask < (1 << m); mask++)
    {
      if (mask & (1 << i))
        f[mask] += f[mask ^ (1 << i)];
    }
  }

  // Exemplo 2:
  // nesse caso, f[x] eh a funcao que soma:
  // todos os a[i], tal que, (x & i) == x)
  // isso eh, i eh uma "mask pai" de x
  // pois todos os bits de x estao setados em i
  for (int mask = 0; mask < (1 << m); mask++)
  {
    f[mask] = a[mask];
  }
  for (int i = 0; i < m; ++i)
  {
    for (int mask = 0; mask < (1 << m); mask++)
    {
      if (!(mask & (1 << i)))
        f[mask] += f[mask ^ (1 << i)];
    }
  }

  return 0;
}
// https://codeforces.com/blog/entry/45223
```

9.22 cht

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 1000005
#define mod 1000000007

struct line
{
  int m, b, p;
  line(int m, int b) : m(m), b(b) {}
  bool operator<(const line &o) const
  {
    if (m != o.m)
      return m < o.m;
    return b < o.b;
  }
  bool operator<(const int x) const { return p < x; }
  int eval(int x) const { return m * x + b; }
  int inter(const line &o) const
  {
    int x = b - o.b, y = o.m - m;
    return (x / y) - ((x ^ y) < 0 && x % y);
  }
};
struct cht
{
  int ptr;
  vector<line> a;
  cht() { ptr = 0; }
  void add(line l)
  {
    while (1)
    {
      if (a.size() >= 1 && a.back().m == l.m && l.b > a.back().b)
      {
        a.pop_back();
      }
      else if (a.size() >= 1 && a.back().m == l.m && l.b <= a.back().b)
      {
        break;
      }
      else if (a.size() >= 2 && a.back().inter(l) >= a[a.size() - 2].inter(a.
           back()))
      {
        a.pop_back();
      }
      else
      {
        a.pb(l);
        break;
      }
    }
  }
  int get(int x)
  {
    if (!a.size())
      return -inf;
    while (ptr + 1 < a.size() && a[ptr].eval(x) <= a[ptr + 1].eval(x))
      ptr++;
    return a[ptr].eval(x);
  }
};
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  return 0;
}
// cht
// queries ordenadas em ordem decrescente
// linhas ordenadas em ordem decrescente
```

10 STL

10.1 STL

```
1)Vector

vector <int> v; //Criacao o vector

v.push_back(10); //Adiciono o elemento 10 no final do vector v
v.size() // retorna o tamanho do vector
v.resize(10); //Muda o tamanho do vector v para 10.
v.pop_back(); //Apaga o ultimo elemento do vector V.
v.clear(); // apaga todos os elementos do vector v .
sort(v.begin(), v.end()); //Ordena todo o vector v


2)Pair

pair <string, int> p; // criando a pair relacionando um first com um second

p.first = "Joao"; // adicionando elementos
p.second = 8 ; //adicionando elementos

// utilidade: vector de pair
vector< pair <int, string> > v; // criando o vector v de pair
v.push_back(make_pair(a,b)); // dando push back em uma pair no vector usando
     make_pair
sort(v.begin(), v.end()); // tambem e possivel ordenar o vector de pair

3)Queue / FIla

queue <int> f; // criando a queque
f.push(10); // adiciona alguem na fila
f.pop(); // remove o elemento que esta na frente da fila
f.front(); // olha qual o elemento esta na frete da fila
f.empty() // retorna true se a fila estiver vazia e false se nao estiver vazia


4)Stack / Pilha

stack <int> p ; // criando a stack

pilha.push(x); //Adiciona o elemento x no topo da pilha
pilha.pop(); //Remove elemento do topo da pilha
pilha.top(); // retorna o elemento do topo da pilha
pilha.empty(); // verifica se a pilha esta vazia ou nao

5) Set

set <int> s ; // criando a set
// obs: a set nao adiciona elementos repetidos

s.insert(10); //Adiciona o elemento 10 no set
s.find(10) // Para realizar uma busca no set utilizamos o comando find,
o find retorna um ponteiro que aponta para o elemento procurado caso o elemento
     esteja no set ou para o final do set, caso o elemento procurado nao esteja
     no set , em complexidade O(log n)

if(s.find(10) != s.end()) // procurando pelo 10, se ele estiver no set
s.erase(10); //Apaga o elemento 10 do set em O(log n)
s.clear(); // Apaga todos os elementos
s.size(); //  Retorna a quantidade de elementos
s.begin(); // Retorna um ponteiro para o inicio do set
s.end(); // Retorna um ponteiro para o final do set


6)Map
map <string, int> m; //Cria uma variavel do tipo map que mapeia strings em int
// Em um map cada elemento esta diretamente ligado a um valor, ou seja, cada
     elemento armazenado no map possui um valor correspondente
// Se tivermos um map de strings em inteiros e inserimos os pair ("Joao", 1), ("
     Alana", 10), ("Rodrigo", 9)
// Caso facamos uma busca pela chave "Alana" receberemos o numero 10 como
     retorno.

m.insert(make_pair("Alana", 10)); //Inserimos uma variavel do tipo pair
     diretamente no map, O(log n)
M["Alana"] = 10; //Relacionando o valor 10 a chave "Alana"
if(m.find("Alana") != m.end()){ //Se a chave "Alana" foi inserida no map
cout << m["Alana"] << endl; //Imprime o valor correspondente a chave "Alana", no
      caso, o valor 10.
m.erase("Alana"); //Apaga o elemento que possui a chave "Alana" do map
m.clear(); // Apaga todos os elementos
m.size(); //  Retorna a quantidade de elementos
m.begin(); // Retorna um ponteiro para o inicio do map
m.end(); // Retorna um ponteiro para o final do map


7)Priority Queue
priority_queue <int> q; // declarando a priority queue
// Para utilizar a priority_queue do C++ e importante apenas saber que o maior
     elemento sempre estara na primeiro posicao.
// Com execao disso, todos os outros metodos sao semelhantes ao uso de uma queue
      comum, porem para manter a estrutura organizada, a complexidade da
     operacao de insercao e O(logn).
p.push(i) // adiciono o elemento i na priority_queue
p.pop(); // apago o primeiro da fila
p.top(); // vejo quem esta no topo
```

10.2 ordered set

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

#define int long long int
```

```
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define DEBUG false
#define MAXN 200002

template <class T> // template do ordered set
using ordered_set = tree<T, null_type, less<T>, rb_tree_tag,
     tree_order_statistics_node_update>;

signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  ordered_set<int> s; // ordered_set
  s.insert(1);
  s.insert(1);
  s.insert(2);
  s.insert(4);
  s.insert(3);
  for (auto const &i : s) // nao adiciona elementos repetidos, que nem o set
       normal
    cout << i << " ";
  cout << endl;
  cout << *(s.find_by_order(0)) << endl; // iterator do elemento 0
  cout << *(s.find_by_order(1)) << endl; // iterator do elemento 1
  cout << s.order_of_key(4) << endl;     // quantidade de elementos que sao
       menores do que 4
  cout << s.order_of_key(6) << endl;     // quantidade de elementos que sao
       menores do que 4
}
// find_by_order : O(log n), retorna (um iterator) qual o k-esimo elemento do
     set
// order_of_key: O(log n), retorna qual a quantidade de elementos menores do que
      x no set
```

11 Math

11.1 mobius2

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 200001
#define mod 998244353

int lpf[MAXN];
int mobius[MAXN];
int mp[MAXN];
vector<int> d[MAXN];

void calc_lpf()
{
  for (int i = 2; i < MAXN; i++)
  {
    if (!lpf[i])
    {
      for (int j = i; j < MAXN; j += i)
      {
        if (!lpf[j])
          lpf[j] = i;
      }
    }
  }
}
void calc()
{
  for (int i = 2; i < MAXN; i++) // divisores
  {
    for (int j = i; j < MAXN; j += i)
      d[j].pb(i);
  }
  calc_lpf();
  mobius[1] = 1;
  for (int i = 2; i < MAXN; i++)
  {
    if (lpf[i / lpf[i]] == lpf[i])
      mobius[i] = 0;
    else
      mobius[i] = -1 * mobius[i / lpf[i]];
  }
}
void add(int x, int dd) // adiciona dd em todos os val[i] que gcd(x, i) > 1
{
  for (auto const &i : d[x])
    mp[i] += dd;
}
int sum(int x) // valor de val[x]
{
  int ans = 0;
  for (auto const &i : d[x])
    ans += (mobius[i] * -1 * mp[i]);
  return ans;
}
signed main()
{
  return 0;
}
// mobius/inclusao-exclusao com os fatores primos
// a funcao de mobius eh definida como:
// mi(n) = 1, se n e um square-free com um numero par de fatores primos
// mi(n) = -1, se n e um square-free com um numero impar de fatores primos
// mi(n) = 0, caso nenhum dos dois
// square-free = nenhum fator primo aparece duas vezes ou mais

// dai pra problemas que da pra se fazer com inclusao-exclusao nos fatores
     primos
// tambem sai com mobius
```

11.2 binomial theorem

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

#pragma GCC optimize("O3")
#pragma GCC optimize("unroll-loops")
#pragma gcc optimize("Ofast")

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 100005
#define mod 1000000007

struct modint
{
  int val;
  modint(int v = 0) { val = (v < 0) ? v + mod : v; }
  int pow(int y)
  {
    modint x = val;
    modint z = 1;
    while (y)
    {
      if (y & 1)
        z *= x;
      x *= x;
      y >>= 1;
    }
    return z.val;
  }
  int inv() { return pow(mod - 2); }
  void operator=(int o) { val = o % mod; }
  void operator=(modint o) { val = o.val % mod; }
  void operator+=(modint o) { *this = *this + o; }
  void operator-=(modint o) { *this = *this - o; }
  void operator*=(modint o) { *this = *this * o; }
  void operator/=(modint o) { *this = *this / o; }
  bool operator==(modint o) { return val == o.val; }
  bool operator!=(modint o) { return val != o.val; }
  int operator*(modint o) { return ((val * o.val) % mod); }
  int operator/(modint o) { return (val * o.inv()) % mod; }
  int operator+(modint o) { return (val + o.val) % mod; }
  int operator-(modint o) { return (val - o.val + mod) % mod; }
};

modint f[MAXN];
modint invfat[MAXN];

void calc()
{
  f[0] = 1;
  for (int i = 1; i < MAXN; i++)
    f[i] = f[i - 1] * i;
  for (int i = 0; i < MAXN; i++)
    invfat[i] = f[i].inv();
}
modint ncr(int n, int k)
{
  modint ans = f[n] * invfat[k];
  ans *= invfat[n - k];
  return ans;
}

struct strange_modint // escrever um numero na forma: a + (b * sqrt(5))
{
  modint a, b;
  strange_modint() { a = 0, b = 0; }
  strange_modint(modint v) { a = v, b = 0; }
  strange_modint(modint v, modint v2) { a = v, b = v2; }
  strange_modint operator*(strange_modint o)
  {
    return strange_modint((a * o.a) + (b * o.b * 5), (a * o.b) + (b * o.a));
  }
  strange_modint operator+(strange_modint o)
  {
    return strange_modint(a + o.a, b + o.b);
  }
  strange_modint operator-(strange_modint o)
  {
    return strange_modint(a - o.a, b - o.b);
  }
  strange_modint pow(int y)
  {
    strange_modint x(a, b);
    strange_modint z(1, 0);
    while (y)
    {
      if (y & 1)
        z = z * x;
      x = x * x;
      y >>= 1;
    }
    return z;
  }
};

strange_modint matrix_ans[3][3];

void matrix_multiply(strange_modint a[3][3], strange_modint b[3][3])
{
  strange_modint res[3][3];
  for (int i = 0; i < 3; i++)
  {
    for (int j = 0; j < 3; j++)
    {
      for (int k = 0; k < 3; k++)
        res[i][j] = res[i][j] + (a[i][k] * b[k][j]);
    }
  }
  for (int i = 0; i < 3; i++)
  {
    for (int j = 0; j < 3; j++)
      a[i][j] = res[i][j];
  }
}
void matrix_pow(strange_modint mat[3][3], int m)
{
  for (int i = 0; i < 3; i++)
  {
    for (int j = 0; j < 3; j++)
      matrix_ans[i][j] = strange_modint((i == j));
  }
  while (m > 0)
  {
    if (m & 1)
      matrix_multiply(matrix_ans, mat);
    m = m / 2;
    matrix_multiply(mat, mat);
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  calc();
  int n, k;
  cin >> n >> k;
  strange_modint a(modint(1) / 2, modint(1) / 2);
  strange_modint b(modint(1) / 2, -(modint(1) / 2));
  strange_modint c(0, modint(1) / 5);
  strange_modint ans(0, 0);
  for (int j = 0; j <= k; j++)
  {
    strange_modint curr(modint(1000000006).pow(j), 0); // (-1)^j
    curr = curr * strange_modint(ncr(k, j), 0);
    strange_modint prod = a.pow(k - j) * b.pow(j);
    strange_modint s0(0, 0);
    strange_modint s1(1, 0);
    strange_modint mat[3][3] = {{prod, s0, s1}, {s1, s0, s0}, {s0, s0, s1}};
    matrix_pow(mat, n);
    curr = curr * (matrix_ans[0][0] + matrix_ans[0][2]);
    ans = ans + curr;
  }
  ans = ans * c.pow(k);
  cout << ans.a.val << endl;
  return 0;
}
// https://codeforces.com/gym/104412/problem/F
```

11.3 diophantine

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 200001
#define mod 998244353

namespace dio
```

```
{
  vector<pi> sols;

  int gcd(int a, int b, int &x, int &y)
  {
    if (b == 0)
    {
      x = 1, y = 0;
      return a;
    }
    int x1, y1, d = gcd(b, a % b, x1, y1);
    x = y1, y = x1 - y1 * (a / b);
    return d;
  }
  void one_sol(int a, int b, int c)
  {
    int x0, y0, g;
    g = gcd(abs(a), abs(b), x0, y0);
    if (c % g)
      return;
    x0 *= (c / g);
    y0 *= (c / g);
    if (a < 0)
      x0 *= -1;
    if (b < 0)
      y0 *= -1;
    sols.pb({x0, y0});
  }
  void more_sols(int a, int b, int c)
  {
    int g = __gcd(a, b);
    int x0 = sols[0].fir, y0 = sols[0].sec;
    for (int k = -200000; k <= 200000; k++)
    {
      int x = x0 + k * (b / g);
      int y = y0 - k * (a / g);
      sols.pb({x, y});
    }
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int a, b, c;
  cin >> a >> b >> c;
  dio::one_sol(a, b, c);
  if (!dio::sols.size())
  {
    cout << "No\n";
    return 0;
  }
  dio::more_sols(a, b, c);
  bool can = false;
  for (auto const &i : dio::sols)
    can |= (i.fir >= 0 && i.sec >= 0);
  (can) ? cout << "Yes\n" : cout << "No\n";
  return 0;
}
// equacoes do tipo:
// ax + by = c
// o caso a = 0 e b = 0, nao eh tratado nesse codigo
// nesse caso quero checar se equacao diofantina tem uma solucao
// com x >= 0 e y >= 0
```

11.4 primefactors

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

#define PI acos(-1)
#define int long long int
#define pb push_back
#define mp make_pair
#define pi pair<int, int>
#define fir first
#define sec second
#define MAXN 501
#define MAXL 20
#define mod 1000000007

vector<int> facts;
void primefactors(int n)
{
  while (n % 2 == 0)
  {
    facts.pb(2);
    n = n / 2;
  }
  for (int i = 3; i <= sqrt(n); i += 2)
  {
    while (n % i == 0)
    {
      facts.pb(i);
      n = n / i;
    }
  }
  if (n > 2)
    facts.pb(n);
}
signed main()
{
  int n;
  cin >> n;
  primefactors(n);
  sort(facts.begin(), facts.end());
  for (auto const &i : facts)
    cout << i << endl;
  return 0;
}
```

11.5 xor trie

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 500001
#define mod 1000000007

struct node
{
  int me, cnt, id;
  int down[2];
  node(int c = 0) : me(c)
  {
    cnt = 0;
    id = -1;
    fill(begin(down), end(down), -1);
  }
};
struct trie_xor
{
  vector<node> t;

  trie_xor()
  {
    t.resize(1);
  }
  void add(int n, int id)
  {
    int v = 0;
    t[v].cnt++;
    for (int i = 30; i >= 0; i--)
    {
      int bit = (n & (1 << i)) ? 1 : 0;
      if (t[v].down[bit] == -1)
      {
        t[v].down[bit] = t.size();
        t.emplace_back(bit);
      }
      v = t[v].down[bit];
      t[v].cnt++;
    }
    t[v].id = id;
  }
  void rem(int n, int id)
  {
    int v = 0;
    t[v].cnt--;
    for (int i = 30; i >= 0; i--)
    {
      int bit = (n & (1 << i)) ? 1 : 0;
      v = t[v].down[bit];
      t[v].cnt--;
    }
  }
  int qry(int n) // maximum xor with n
  {
    if (t[0].cnt == 0) // no element
      return -1;
    int v = 0;
    for (int i = 30; i >= 0; i--)
    {
      int bit = (n & (1 << i)) ? 0 : 1;
      int nxt = t[v].down[bit];
      if (nxt != -1 && t[nxt].cnt > 0)
        v = nxt;
      else
        v = t[v].down[bit ^ 1];
    }
    return t[v].id;
  }
};
signed main()
{
}
// alguns problemas:
// https://codeforces.com/problemset/problem/706/D
// https://codeforces.com/contest/1625/problem/D
// https://codeforces.com/contest/888/problem/G
```

11.6 totient

```
#define MAXN 100000

int phi[MAXN];

void calc()
{
  for (int i = 0; i < MAXN; i++)
    phi[i] = i;
  for (int i = 2; i < MAXN; i++)
  {
    if (phi[i] == i)
    {
      for (int j = i; j < MAXN; j += i)
        phi[j] -= phi[j] / i;
    }
  }
}
int calc_phi(int n)
{
  int ans = n;
  for (int i = 2; i * i <= n; i++)
  {
    if (n % i == 0)
    {
      while (n % i == 0)
        n /= i;
      ans -= ans / i;
    }
  }
  if (n > 1)
    ans -= ans / n;
  return ans;
}
```

11.7 matrix inverse and determinant

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 100005
#define mod 998244353

struct modint
{
  int val;
  modint(int v = 0) { val = (v + mod) % mod; }
  int pow(int y)
  {
    modint x = val;
    modint z = 1;
    while (y)
    {
      if (y & 1)
        z *= x;
      x *= x;
      y >>= 1;
    }
    return z.val;
  }
  int inv() { return pow(mod - 2); }
  void operator=(int o) { val = o % mod; }
  void operator=(modint o) { val = o.val % mod; }
  void operator+=(modint o) { *this = *this + o; }
  void operator-=(modint o) { *this = *this - o; }
  void operator*=(modint o) { *this = *this * o; }
  void operator/=(modint o) { *this = *this / o; }
  bool operator==(modint o) { return val == o.val; }
  bool operator!=(modint o) { return val != o.val; }
  int operator*(modint o) { return ((val * o.val) % mod); }
  int operator/(modint o) { return (val * o.inv()) % mod; }
  int operator+(modint o) { return (val + o.val) % mod; }
  int operator-(modint o) { return (val - o.val + mod) % mod; }
};
pair<int, modint> gauss(vector<vector<modint>> &a, int pivot_end)
{
  int n = a.size(), m = a[0].size();
  int rank = 0, curr = pivot_end;
  if (curr == -1)
    curr = m;
  modint det = 1;
  for (int j = 0; j < curr; j++)
  {
    int idx = -1;
    for (int i = rank; i < n; i++)
    {
      if (a[i][j].val != 0)
      {
        idx = i;
        break;
      }
    }
    if (idx == -1)
    {
      det = 0;
      continue;
    }
    if (rank != idx)
    {
      det *= -1;
      swap(a[rank], a[idx]);
    }
    det *= a[rank][j];
    if (a[rank][j].val != 1)
    {
      modint coef = a[rank][j].inv();
      for (int k = j; k < m; k++)
        a[rank][k] *= coef;
    }
    int is = 0;
    for (int i = is; i < n; i++)
    {
      if (i == rank)
        continue;
      if (a[i][j].val != 0)
      {
        modint coef = a[i][j] / a[rank][j];
        for (int k = j; k < m; k++)
          a[i][k] -= a[rank][k] * coef;
      }
    }
    rank++;
  }
  return {rank, det};
}
vector<vector<modint>> inverse_matrix(vector<vector<modint>> a)
{
  int n = a.size();
  vector<vector<modint>> m(n, vector<modint>(2 * n));
  for (int i = 0; i < n; i++)
  {
    copy(begin(a[i]), end(a[i]), begin(m[i]));
    m[i][n + i] = 1;
  }
  auto [rank, det] = gauss(m, n);
  if (rank != n)
    return {};
  vector<vector<modint>> b(n);
  for (int i = 0; i < n; i++)
    copy(begin(m[i]) + n, end(m[i]), back_inserter(b[i]));
  return b;
}
modint determinant(vector<vector<modint>> a)
{
  return gauss(a, -1).sec;
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n;
  cin >> n;
  vector<vector<modint>> v(n, vector<modint>(n));
  for (int i = 0; i < n; i++)
  {
    for (int j = 0; j < n; j++)
      cin >> v[i][j].val;
  }
  if (determinant(v).val == 0)
  {
    cout << "-1\n";
    return 0;
  }
  vector<vector<modint>> ans = inverse_matrix(v);
  for (int i = 0; i < n; i++)
  {
    for (int j = 0; j < n; j++)
      cout << ans[i][j].val << " ";
    cout << endl;
  }
  return 0;
}
// https://judge.yosupo.jp/problem/inverse_matrix
// https://judge.yosupo.jp/problem/matrix_det
// como precisa de divisao, entao o mod tem que ser primo
```

11.8 gaussian elimination

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define double long double
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define DEBUG 1
#define MAXN 2001
#define mod 1000000007
#define EPS 1e-9

vector<double> ans;

int gauss(vector<vector<double>> a)
{
  int n = a.size(), m = a[0].size() - 1, ret = 1;
  ans.assign(m, 0);
  vector<int> where(m, -1);
  for (int col = 0, row = 0; col < m && row < n; col++, row++)
  {
    int sel = row;
    for (int i = row; i < n; i++)
      if (abs(a[i][col]) > abs(a[sel][col]))
        sel = i;
    if (abs(a[sel][col]) < EPS)
      continue;
    for (int i = col; i <= m; i++)
      swap(a[sel][i], a[row][i]);
    where[col] = row;
    for (int i = 0; i < n; i++)
    {
      if (i != row)
      {
        double c = a[i][col] / a[row][col];
        for (int j = col; j <= m; j++)
          a[i][j] -= a[row][j] * c;
      }
    }
  }
  for (int i = 0; i < m; i++)
  {
    if (where[i] != -1)
      ans[i] = (a[where[i]][m] / a[where[i]][i]);
    else
      ret = 2;
  }
  for (int i = 0; i < n; i++)
  {
    double sum = 0;
    for (int j = 0; j < m; j++)
      sum += (ans[j] * a[i][j]);
    if (abs(sum - a[i][m]) > EPS)
      ret = 0;
  }
  return ret; // 0 = nao existe solucao, 1 = existe uma solucao, 2 = existem
       multiplas solucoes
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  vector<vector<double>> a = {{1.0, 1.0, 20.0},  // 1x + 1y = 20
                              {3.0, 4.0, 72.0}}; // 3x + 4y = 72
  cout << gauss(a) << endl;
  for (auto const &i : ans) // x = 8 e y = 12
    cout << i << " ";
  cout << endl;
}
// eliminacao gaussiana
// para resolver sistemas com n equacoes e m incognitas

// para isso iremos utilizar uma representacao usando
// matrizes, no qual uma coluna extra e adicionada,
// representando os resultados de cada equacao.

// algoritimo:
// ideia: qualquer equacao pode ser reescrita como uma combinacao linear dela
     mesma
// 1- dividir a primeira linha(primeira equacao) por a[0][0]
// 2- adicionar a primeira linha as linhas restantes, de modo que, os
//    coeficientes da primeira coluna se tornem todos zeros, para que
//    isso aconteca, na i-esima linha devemos adicionar a primeira linha
//    multiplicada por (a[i][0] * -1)
// 3- com isso, o elemento a[0][0] = 1 e os demais elementos da primeira coluna
//    serao iguais a zero
// 4- continuamos o algoritimo a partir da etapa 1 novamente, dessa vez
//    com a segunda coluna e a segunda linha, dividindo a linha por a[1][1]
//    e assim sucessivamente
// 5- ao final, teremos a resposta

// complexidade O(min(n, m) * n * m);
// se n == m, logo a complexidade sera O(n^3)
```

11.9 extended euclidean

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 200001
#define mod 998244353

int gcd(int a, int b, int &x, int &y)
{
  if (b == 0)
  {
    x = 1;
    y = 0;
    return a;
  }
  int x1, y1;
  int d = gcd(b, a % b, x1, y1);
  x = y1;
  y = x1 - y1 * (a / b);
  return d;
}
signed main()
{
  int n;
  cin >> n;
  int k = 2;
  while (1)
  {
    int x, y;
    if (gcd(k, n, x, y) == 1)
    {
      x = ((x % n) + n) % n;
      cout << x << endl;
      return 0;
    }
    k++;
  }
  return 0;
}
// achar os numeros x e y tal que:
// a * x + b * y = gcd(a, b)

// problema exemplo:
// https://codeforces.com/group/btcK4I5D5f/contest/451372/problem/J
// dado um numero k
// quero achar um numero x, se possivel, tal que:
// (k * x) % n = 1

// k * x + n * y = 1
// se gcd(k, n) = 1, entao:
// k * x + n * y = gcd(k, n)
// note que, se gcd(k, n) > 1, logo nao existe solucao
```

11.10 primefactors2

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define PI acos(-1)
#define pb push_back
#define int long long int
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define DEBUG 0
#define MAXN 1000001
#define mod 1000000007

namespace primefactors
{
  bitset<MAXN> prime;
  vector<int> nxt(MAXN);
  vector<int> factors;

  void crivo()
  {
    prime.set();
    prime[0] = false, prime[1] = false;
    for (int i = 2; i < MAXN; i++)
    {
      if (prime[i])
      {
        nxt[i] = i;
        for (int j = 2; j * i < MAXN; j++)
        {
          prime[j * i] = false;
          nxt[j * i] = i;
        }
      }
    }
  }
  void fact(int n)
  {
    factors.clear();
    while (n > 1)
    {
      factors.pb(nxt[n]);
      n = n / nxt[n];
    }
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  return 0;
}
```

11.11 catalan

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 400005
#define mod 998244353

struct modint
{
  int val;
  modint(int v = 0) { val = v % mod; }
  int pow(int y)
  {
    modint x = val;
    modint z = 1;
    while (y)
    {
      if (y & 1)
        z *= x;
      x *= x;
      y >>= 1;
    }
    return z.val;
  }
  int inv() { return pow(mod - 2); }
  void operator=(int o) { val = o % mod; }
  void operator=(modint o) { val = o.val % mod; }
  void operator+=(modint o) { *this = *this + o; }
  void operator-=(modint o) { *this = *this - o; }
  void operator*=(modint o) { *this = *this * o; }
  void operator/=(modint o) { *this = *this / o; }
  bool operator==(modint o) { return val == o.val; }
  bool operator!=(modint o) { return val != o.val; }
  int operator*(modint o) { return ((val * o.val) % mod); }
  int operator/(modint o) { return (val * o.inv()) % mod; }
  int operator+(modint o) { return (val + o.val) % mod; }
  int operator-(modint o) { return (val - o.val + mod) % mod; }
};

modint f[MAXN];
modint inv[MAXN];
modint invfat[MAXN];

void calc()
{
  f[0] = 1;
  for (int i = 1; i < MAXN; i++)
  {
    f[i] = f[i - 1] * i;
  }
  invfat[0] = 1;
  for (int i = MAXN - 1; i >= 1; i--)
  {
    invfat[i] = modint(f[i]).inv();
  }
}
modint ncr(int n, int k)
{
  modint ans = f[n] * invfat[k];
  ans *= invfat[n - k];
  return ans;
}
modint catalan(int x)
{
  modint ans = modint(1) / modint(x + 1);
  ans *= ncr(2 * x, x);
  return ans;
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  calc();
  cout << catalan(10).val << endl;
  return 0;
}
// catalan numbers:
// 1, 1, 2, 5, 14, 42, 132, 429, 1430, ...
// c[i] = (1 / (n + 1)) * ncr(2n, n)

// existem c[i] regular bracket sequences de tamanho 2 * n
// o numero de binary trees completas com n + 1 folhas
// o numero de possibilidades de conectar 2n numeros em um circulo em n arestas
// e uma serie de coisas

// https://cp-algorithms.com/combinatorics/catalan-numbers.html
```

11.12 berlekamp massey

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 100005
#define mod 1000000007

struct modint
{
  int val;
  modint(int v = 0) { val = ((v % mod) + mod) % mod; }
  int pow(int y)
  {
    modint x = val;
    modint z = 1;
    while (y)
    {
      if (y & 1)
        z *= x;
      x *= x;
      y >>= 1;
    }
    return z.val;
  }
  int inv() { return pow(mod - 2); }
  void operator=(int o) { val = o % mod; }
  void operator=(modint o) { val = o.val % mod; }
  void operator+=(modint o) { *this = *this + o; }
  void operator-=(modint o) { *this = *this - o; }
  void operator*=(modint o) { *this = *this * o; }
  void operator/=(modint o) { *this = *this / o; }
  bool operator==(modint o) { return val == o.val; }
  bool operator!=(modint o) { return val != o.val; }
  int operator*(modint o) { return ((val * o.val) % mod); }
  int operator/(modint o) { return (val * o.inv()) % mod; }
  int operator+(modint o) { return (val + o.val) % mod; }
  int operator-(modint o) { return (val - o.val + mod) % mod; }
};

// berlekamp massey (muito roubado)
// mas precisa que o mod seja primo (para poder achar inverso)
// dado os n primeiros termos de uma recorrencia linear
// a[0], a[1], a[2], ..., a[n - 1]
// ele acha a reccorrencia linear mais curta que da matching com os n primeiros valores
vector<modint> berlekamp_massey(vector<modint> x)
{
  vector<modint> ls, cur;
  int lf, ld;
  for (int i = 0; i < x.size(); i++)
  {
    modint t = 0;
    for (int j = 0; j < cur.size(); j++)
    {
      t += (x[i - j - 1] * cur[j]);
    }
    if (modint(t - x[i]).val == 0)
      continue;
    if (cur.empty())
    {
      cur.resize(i + 1);
      lf = i;
      ld = (t - x[i]) % mod;
      continue;
    }
    modint k = -(x[i] - t);
    k *= modint(ld).inv();
    vector<modint> c(i - lf - 1);
    c.pb(k);
    for (auto const &j : ls)
    {
      modint curr = modint(j.val * -1) * k;
      c.pb(curr);
    }
    if (c.size() < cur.size())
      c.resize(cur.size());
    for (int j = 0; j < cur.size(); j++)
    {
      c[j] = c[j] + cur[j];
    }
    if (i - lf + ls.size() >= cur.size())
    {
      tie(ls, lf, ld) = make_tuple(cur, i, t - x[i]);
    }
    cur = c;
  }
  return cur;
}
modint get_nth(vector<modint> rec, vector<modint> dp, int n)
{
  int m = rec.size();
  vector<modint> s(m), t(m);
  s[0] = 1;
  if (m != 1)
    t[1] = 1;
  else
    t[0] = rec[0];
  auto mul = [&rec](vector<modint> v, vector<modint> w)
  {
    vector<modint> ans(2 * v.size());
    for (int j = 0; j < v.size(); j++)
    {
      for (int k = 0; k < v.size(); k++)
        ans[j + k] += v[j] * w[k];
    }
    for (int j = 2 * v.size() - 1; j >= v.size(); j--)
    {
      for (int k = 1; k <= v.size(); k++)
        ans[j - k] += ans[j] * rec[k - 1];
    }
    ans.resize(v.size());
    return ans;
  };
  while (n)
  {
    if (n & 1)
      s = mul(s, t);
    t = mul(t, t);
    n >>= 1;
  }
  modint ret = 0;
  for (int i = 0; i < m; i++)
    ret += s[i] * dp[i];
  return ret;
}
modint guess_nth_term(vector<modint> x, int n)
{
  if (n < x.size())
    return x[n];
  vector<modint> coef = berlekamp_massey(x); // coeficientes da recorrencia
  /*for (auto const &i : coef)
    cout << i.val << " ";
  cout << endl;*/
  if (coef.empty())
    return 0;
  return get_nth(coef, x, n);
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  vector<modint> vals;
  vals.pb(0);
  vals.pb(1);
  for (int i = 2; i <= 200; i++)
    vals.pb(vals[vals.size() - 1] + vals[vals.size() - 2]);
  int n;
  cin >> n;
  cout << guess_nth_term(vals, n).val << endl;
  return 0;
}
// exemplo fibonacci
```

11.13 fraction

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 200006
#define mod 1000000007

struct fraction
{
  int x, y; // x / y

  fraction() {}
  fraction(int x, int y) : x(x), y(y) {}
  bool operator==(fraction o) { return (x * o.y == o.x * y); }
  bool operator!=(fraction o) { return (x * o.y != o.x * y); }
  bool operator>(fraction o) { return (x * o.y > o.x * y); }
  bool operator>=(fraction o) { return (x * o.y >= o.x * y); }
  bool operator<(fraction o) { return (x * o.y < o.x * y); }
  bool operator<=(fraction o) { return (x * o.y <= o.x * y); }
  fraction operator+(fraction o)
  {
    fraction ans;
    ans.y = (y == o.y) ? y : y * o.y;
    ans.x = (x) * (ans.y / y) + (o.x) * (ans.y / o.y);
    // ans.simplify();
    return ans;
  }
  fraction operator*(fraction o)
  {
    fraction ans;
    ans.x = x * o.x;
    ans.y = y * o.y;
    // ans.simplify();
    return ans;
  }
  fraction inv()
  {
    fraction ans = fraction(x, y);
    swap(ans.x, ans.y);
    return ans;
  }
  fraction neg()
  {
    fraction ans = fraction(x, y);
    ans.x *= -1;
    return ans;
  }
  void simplify()
  {
    if (abs(x) > 1e9 || abs(y) > 1e9) // slow simplification
    {
      int g = __gcd(y, x);
      x /= g;
      y /= g;
    }
  }
  // substraction and division can be easily done
};
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  return 0;
}
```

11.14 pollard rho

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int __int128
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 1000001
#define mod 998244353

int read() // __int128 functions
{
  int x = 0, f = 1;
  char ch = getchar();
  while (ch < '0' || ch > '9')
  {
    if (ch == '-')
      f = -1;
    ch = getchar();
  }
  while (ch >= '0' && ch <= '9')
  {
    x = x * 10 + ch - '0';
    ch = getchar();
  }
  return x * f;
}
void print(__int128 x) // __int128 functions
{
  if (x < 0)
  {
    cout << "-";
    x = -x;
  }
  stack<char> s;
  while (x)
  {
    s.push((x % 10) + '0');
    x = x / 10;
  }
  while (!s.empty())
  {
    cout << s.top();
    s.pop();
  }
}
namespace pollard_rho
{
  int multiplicate(int x, int y, int m)
  {
    return (x * y) % m;
  }
  int modpow(int x, int y, int m)
  {
    int z = 1;
    while (y)
    {
      if (y & 1)
        z = (z * x) % m;
      x = (x * x) % m;
      y >>= 1;
    }
    return z;
  }
  bool is_composite(int n, int a, int d, int s)
  {
    int x = modpow(a, d, n);
    if (x == 1 or x == n - 1)
      return false;
    for (int r = 1; r < s; r++)
    {
      x = multiplicate(x, x, n);
      if (x == n - 1LL)
        return false;
    }
    return true;
  };
  int miller_rabin(int n)
  {
    if (n < 2)
      return false;
    int r = 0, d = n - 1LL;
    while ((d & 1LL) == 0)
    {
      d >>= 1;
      r++;
    }
    for (int a : {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37})
    {
      if (n == a)
        return true;
      if (is_composite(n, a, d, r))
        return false;
    }
    return true;
  }
  int f(int x, int m)
  {
    return multiplicate(x, x, m) + 1;
  }
  int rho(int n)
  {
    int x0 = 1, t = 0, prd = 2;
    int x = 0, y = 0, q;
    while (t % 40 || __gcd(prd, n) == 1)
    {
      if (x == y)
      {
        x0++;
        x = x0;
        y = f(x, n);
      }
      q = multiplicate(prd, max(x, y) - min(x, y), n);
      if (q != 0)
        prd = q;
      x = f(x, n);
      y = f(y, n);
      y = f(y, n);
      t++;
    }
    return __gcd(prd, n);
  }
  vector<int> fact(int n)
  {
    if (n == 1)
      return {};
    if (miller_rabin(n))
      return {n};
    int x = rho(n);
    auto l = fact(x), r = fact(n / x);
    l.insert(l.end(), r.begin(), r.end());
    return l;
  }
}
signed main()
{
  //ios_base::sync_with_stdio(false);
  //cin.tie(NULL);
  while (1)
  {
    int n = read();
    if (n == 0)
      break;
    vector<int> factors = pollard_rho::fact(n);
    sort(factors.begin(), factors.end());
    int prev = -1, cnt = 0;
    for (auto const &i : factors)
    {
      if (prev != i)
      {
        if (prev != -1)
        {
          print(prev);
          printf("^");
          print(cnt);
          printf(" ");
        }
        prev = i;
        cnt = 0;
      }
      cnt++;
    }
    if (prev != -1)
    {
      print(prev);
      printf("^");
      print(cnt);
      printf(" ");
    }
    printf("\n");
  }
  return 0;
}
// sources:
// https://github.com/PauloMiranda98/Competitive-Programming-Notebook/blob/
     master/code/math/prime.h
// https://github.com/brunomaletta/Biblioteca/blob/master/Codigo/Matematica/
     pollardrho.cpp
// fast integer factorization with pollard-rho
// https://www.spoj.com/problems/FACT0/ - ok
// https://www.spoj.com/problems/FACT1/ - ok
// https://www.spoj.com/problems/FACT2/ - sigkill
// since the limit is at most 29 digits(in FACT2), we need to use __int128
```

11.15 baby step gigant step

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template <class string>
using ordered_set = tree<string, null_type, less<string>, rb_tree_tag,
     tree_order_statistics_node_update>;

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 2001
#define mod 1000000007

int bsgs(int a, int b, int m)
{
  if (a == 0 && b == 0)
    return 1;
  a %= m, b %= m;
  int k = 1, add = 0, g;
  while ((g = __gcd(a, m)) > 1) // fazer a e m serem coprimos
  {
    if (b == k)
      return add;
    if (b % g)
      return -1;
    b /= g, m /= g, ++add;
    k = (k * 1ll * a / g) % m;
  }
  int n = sqrt(m) + 1;
  int an = 1;
  for (int i = 0; i < n; i++)
    an = (an * 1ll * a) % m;
  unordered_map<int, int> vals;
  for (int q = 0, cur = b; q <= n; q++)
  {
    vals[cur] = q;
    cur = (cur * 1ll * a) % m;
  }
  for (int p = 1, cur = k; p <= n; p++)
  {
    cur = (cur * 1ll * an) % m;
    if (vals.count(cur))
    {
      int ans = n * p - vals[cur] + add;
      return ans;
    }
  }
  return -1;
}
```

```
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int q;
  cin >> q;
  while (q--)
  {
    int a, b, m;
    cin >> a >> b >> m;
    cout << bsgs(a, b, m) << endl;
  }
  return 0;
}
// menor x tal que: (a^x) % m = b % m
// a e m sao coprimos
// se nao forem coprimos tem como tratar
// complexidade: sqrt(m)
```

11.16 matrix exponentiation2

```
// https://www.spoj.com/problems/ITRIX12E/
// count some {f(0) + f(1) + ... + f(n)} with just one matrix exponentiation
// creates an extra dimension in the matrix and initializes that column with 1s

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

#define PI acos(-1)
#define pb push_back
#define mp make_pair
#define int long long int
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define MAXN 100001
#define MAXL 20
#define INF 200001
#define mod 1000000007

const int n = 11;
vector<vector<int>> ans(n, vector<int>(n));

vector<vector<int>> multiply(vector<vector<int>> a, vector<vector<int>> b)
{
  vector<vector<int>> res(n, vector<int>(n));
  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] = (res[i][j] + (((a[i][k] % mod) * (b[k][j] % mod)) % mod)) %
             mod;
    }
  }
  return res;
}
vector<vector<int>> expo(vector<vector<int>> mat, int m)
{
  for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
      ans[i][j] = (i == j);
  while (m > 0)
  {
    if (m & 1)
      ans = multiply(ans, mat);
    m = m / 2;
    mat = multiply(mat, mat);
  }
  return ans;
}
bool is_prime(int n)
{
  for (int i = 2; i < n; i++)
    if (!(n % i))
      return false;
  return true;
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int q;
  cin >> q;
  while (q--)
  {
    int k;
    cin >> k;
    int resp = 0;
    vector<vector<int>> mat(n, vector<int>(n, 0));
    for (int i = 1; i <= 9; i++)
      for (int j = 1; j <= 9; j++)
        if (is_prime(i + j))
          mat[i][j] = 1;
    for (int i = 0; i <= 10; i++)
      mat[i][10] = 1;
    vector<vector<int>> ans = expo(mat, k - 1);
    for (int i = 0; i < n; i++)
      for (int j = 0; j < n; j++)
        resp = (resp + ans[i][j]) % mod;
    cout << resp - 7 << endl;
  }
  return 0;
}
```

11.17 simplex

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

#define int long long int
#define pi pair<int, int>
#define fir first
#define sec second
#define mod 2147483647
#define pb push_back
#define double long double

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

const double eps = 1e-8;
const double inf = 1e18;
#define MP make_pair
#define ltj(X)                                              \
  if (s == -1 || MP(X[j], N[j]) < MP(X[s], N[s]))  \
  s = j

// resolve um problema de programacao linear para maximizar uma funcao  c[0] * x
     [0] + c[1] * x[1] + ... <= b (c[i] eh o coeficiente do i-esimo cara na
     funcao objetiva)
// sujeito a restricoes que tem a seguinte forma:
// a[0] * x[0] + a[1] * x[1] + ... <= b, (a[i] eh o coeficiente)
// ai todas as restricoes sao passadas nos vectors a e b
// complexidade: 2^n, mas na pratica pode ser melhor do que isso
struct lp_solver
{
  int m, n;
  vector<int> N, B;
  vector<vector<double>> D;

  lp_solver(const vector<vector<double>> &A, const vector<double> &b, const
       vector<double> &c) : m(b.size()), n(c.size()), N(n + 1), B(m), D(m + 2,
       vector<double>(n + 2))
  {
    for (int i = 0; i < m; i++)
    {
      for (int j = 0; j < n; j++)
        D[i][j] = A[i][j];
    }
    for (int i = 0; i < m; i++)
    {
      B[i] = n + i;
      D[i][n] = -1;
      D[i][n + 1] = b[i];
    }
    for (int j = 0; j < n; j++)
    {
      N[j] = j;
      D[m][j] = -c[j];
    }
    N[n] = -1;
    D[m + 1][n] = 1;
  }
  void pivot(int r, int s)
  {
    double *a = D[r].data(), inv = 1 / a[s];
    for (int i = 0; i < m + 2; i++)
    {
      if (i != r && abs(D[i][s]) > eps)
      {
        double *b = D[i].data(), inv2 = b[s] * inv;
        for (int j = 0; j < n + 2; j++)
          b[j] -= a[j] * inv2;
        b[s] = a[s] * inv2;
      }
    }
    for (int j = 0; j < n + 2; j++)
      if (j != s)
        D[r][j] *= inv;
    for (int i = 0; i < m + 2; i++)
      if (i != r)
        D[i][s] *= -inv;
    D[r][s] = inv;
    swap(B[r], N[s]);
  }
  bool simplex(int phase)
  {
    int x = m + phase - 1;
    for (;;)
    {
      int s = -1;
      for (int j = 0; j < n + 1; j++)
      {
        if (N[j] != -phase)
          ltj(D[x]);
      }
      if (D[x][s] >= -eps)
        return true;
      int r = -1;
      for (int i = 0; i < m; i++)
      {
        if (D[i][s] <= eps)
          continue;
        if (r == -1 || MP(D[i][n + 1] / D[i][s], B[i]) < MP(D[r][n + 1] / D[r][s], B[r]))
          r = i;
      }
      if (r == -1)
        return false;
      pivot(r, s);
    }
  }
  double solve()
  {
    int r = 0;
    for (int i = 1; i < m; i++)
    {
      if (D[i][n + 1] < D[r][n + 1])
        r = i;
    }
    if (D[r][n + 1] < -eps)
    {
      pivot(r, n);
      if (!simplex(2) || D[m + 1][n + 1] < -eps)
        return -inf;
      for (int i = 0; i < m; i++)
      {
        if (B[i] == -1)
        {
          int s = 0;
          for (int j = 1; j < n + 1; j++)
            ltj(D[i]);
          pivot(i, s);
        }
      }
    }
    bool ok = simplex(1);
    vector<double> x = vector<double>(n); // os valores escolhidos pra cada x[i] (se quiser eles tbm, so retornar)
    for (int i = 0; i < m; i++)
    {
      if (B[i] < n)
        x[B[i]] = D[i][n + 1];
    }
    return ok ? D[m][n + 1] : inf;
  }
};
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int k;
  cin >> k;
  vector<double> t(k), l(k), r(k);
  for (int i = 0; i < k; i++)
    cin >> t[i] >> l[i] >> r[i];
  int q;
  cin >> q;
  while (q--)
  {
    int aa, bb;
    cin >> aa >> bb;
    int p = aa * bb;
    vector<vector<double>> a;
    vector<double> b;
    vector<double> curr(k, 1);
    a.pb(curr);
    b.pb(bb);
    curr = vector<double>(k, -1);
    a.pb(curr);
    b.pb(-bb);
    a.pb(t);
    b.pb(p);
    curr = vector<double>(k, 0);
    for (int i = 0; i < k; i++)
    {
      curr[i] = 1;
      a.pb(curr);
      b.pb(r[i]);
      curr[i] = 0;
    }
    for (int i = 0; i < k; i++)
    {
      curr[i] = -1;
      a.pb(curr);
      b.pb(-l[i]);
      curr[i] = 0;
    }
    int x = a.size();
    lp_solver l(a, b, t);
    int ans = round(l.solve());
    if (ans == p)
      cout << "yes\n";
    else
      cout << "no\n";
  }
}
// solucao pro: https://open.kattis.com/problems/joiningflows
// source: https://github.com/kth-competitive-programming/kactl/blob/main/
```

```
    content/numerical/Simplex.h

// lembrete: quando eu quero adicionar algo com <= ao inves de >=, basta
    multiplicar os dois lados por -1 :)
// TODO: escrever melhor isso tudo depois
```

11.18 lagrange

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 600005
#define mod 1000000007

struct modint
{
  int val;
  modint(int v = 0) { val = v % mod; }
  int pow(int y)
  {
    modint x = val;
    modint z = 1;
    while (y)
    {
      if (y & 1)
        z *= x;
      x *= x;
      y >>= 1;
    }
    return z.val;
  }
  int inv() { return pow(mod - 2); }
  void operator=(int o) { val = o % mod; }
  void operator=(modint o) { val = o.val % mod; }
  void operator+=(modint o) { *this = *this + o; }
  void operator-=(modint o) { *this = *this - o; }
  void operator*=(modint o) { *this = *this * o; }
  void operator/=(modint o) { *this = *this / o; }
  bool operator==(modint o) { return val == o.val; }
  bool operator!=(modint o) { return val != o.val; }
  int operator*(modint o) { return ((val * o.val) % mod); }
  int operator/(modint o) { return (val * o.inv()) % mod; }
  int operator+(modint o) { return (val + o.val) % mod; }
  int operator-(modint o) { return (val - o.val + mod) % mod; }
};
struct lagrange
{
  int n;
  vector<modint> den;
  vector<modint> y;
  vector<modint> fat;
  vector<modint> inv_fat;

  lagrange(vector<modint> &v) // f(i) = v[i], gera um polinomio de grau n - 1
  {
    n = v.size();
    calc(n);
    calc_den(n);
    y = v;
  }
  void calc_den(int n)
  {
    den.resize(n);
    for (int i = 0; i < n; i++)
    {
      den[i] = inv_fat[n - i - 1] * inv_fat[i];
      if ((n - i - 1) % 2 == 1)
      {
        int x = (mod - den[i].val) % mod;
        den[i] = x;
      }
    }
  }
  void calc(int n)
  {
    fat.resize(n + 1);
    inv_fat.resize(n + 1);
    fat[0] = 1;
    inv_fat[0] = 1;
    for (int i = 1; i <= n; i++)
    {
      fat[i] = fat[i - 1] * i;
      inv_fat[i] = fat[i].inv();
    }
  }
  modint get_val(int x) // complexidade: O(n)
  {
    x %= mod;
    vector<modint> l(n);
    vector<modint> r(n);
    l[0] = 1, r[n - 1] = 1;
    for (int i = 1; i < n; i++)
    {
      modint cof = (x - (i - 1) + mod);
      l[i] = l[i - 1] * cof;
    }
    for (int i = n - 2; i >= 0; i--)
    {
      modint cof = (x - (i + 1) + mod);
      r[i] = r[i + 1] * cof;
    }
    modint ans = 0;
    for (int i = 0; i < n; i++)
    {
      modint cof = l[i] * r[i];
      ans += modint(cof * y[i]) * den[i];
    }
    return ans;
  }
  vector<modint> find_coefs() // encontra os coeficientes do polinomio
  {
    int nn = n;
    int d = nn - 1;
    vector<modint> c(nn, 0);
    for (int i = 0; i < y.size(); i++)
    {
      c[d] += (y[i] * den[i]);
    }
    for (int p = nn - 2; p >= 0; p--)
    {
      nn--;
      calc_den(nn);
      for (int i = 0; i <= p; i++)
      {
        y[i] -= (c[p + 1] * modint(i).pow(d));
        c[p] += (y[i] * den[i]);
      }
      d--;
    }
    return c;
  }
};
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n, k;
  cin >> n >> k;
  vector<modint> v;
  v.pb(0);
  int lim = k + 1;
  for (int i = 1; i <= lim; i++)
    v.pb(v.back() + modint(i).pow(k));
  lagrange l(v);
  cout << l.get_val(n).val << endl;
  return 0;
}
// https://codeforces.com/contest/622/problem/F
// https://codeforces.com/contest/1817/problem/C
// https://codeforces.com/gym/103388/problem/A
```

11.19 crt

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define MAXN 2000006
//#define mod 1000000007

namespace crt
{
  vector<pi> eq;

  int gcd(int a, int b, int &x, int &y)
  {
    if (b == 0)
    {
      x = 1, y = 0;
      return a;
    }
    int x1, y1, d = gcd(b, a % b, x1, y1);
    x = y1, y = x1 - y1 * (a / b);
    return d;
  }
  pi crt()
  {
    int a1 = eq[0].fir, m1 = eq[0].sec;
    a1 %= m1;
    for (int i = 1; i < eq.size(); i++)
    {
      int a2 = eq[i].fir, m2 = eq[i].sec;
      int g = __gcd(m1, m2);
      if (a1 % g != a2 % g)
        return {-1, -1};
      int p, q;
      gcd(m1 / g, m2 / g, p, q);
      int mod = m1 / g * m2;
      int x = (a1 * (m2 / g) % mod * q % mod + a2 * (m1 / g) % mod * p % mod) %
           mod;
      a1 = x;
      if (a1 < 0)
        a1 += mod;
      m1 = mod;
    }
    return {a1, m1};
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n;
  cin >> n;
  for (int i = 0; i < n; i++)
  {
    int a, b;
    cin >> a >> b;
    crt::eq.pb({a, b});
  }
  pi ans = crt::crt();
  if (ans.fir == -1)
    cout << "No solution\n";
  else
    cout << ans.fir << " " << ans.sec << endl;
  return 0;
}
// references:
// https://forthright48.com/chinese-remainder-theorem-part-2-non-coprime-moduli/
// https://cp-algorithms.com/algebra/chinese-remainder-theorem.html
// https://www.geeksforgeeks.org/chinese-remainder-theorem-set-1-introduction/

// teorema chines do resto(crt)
// para resolver sistemas de congruencias modulares
// o menor inteiro a que satisfaz:
// a mod p1 = x1
// a mod p2 = x2
// ...
// a mod pn = xn
// a funcao crt retorna um pair {a, mod}
// dai a solucao pode ser descrita como
// x = a % mod
// entao os valores possiveis sao:
// a, (a + mod), a + (2 * mod), a + (3 * mod), ...
```

11.20 fft

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

using namespace std;

#define PI acos(-1)
#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define MAXN 125001
#define mod 1000000007
#define cd complex<double>

namespace fft
{
  int n;

  void fft(vector<cd> &a, bool invert)
  {
    int n = a.size();
    for (int i = 1, j = 0; i < n; i++)
    {
      int bit = n >> 1;
      for (; j & bit; bit >>= 1)
        j ^= bit;
      j ^= bit;
      if (i < j)
        swap(a[i], a[j]);
    }
    for (int len = 2; len <= n; len <<= 1)
    {
      double ang = 2 * PI / len * (invert ? -1 : 1);
      cd wlen(cos(ang), sin(ang));
      for (int i = 0; i < n; i += len)
      {
        cd w(1);
        for (int j = 0; j < len / 2; j++)
        {
          cd u = a[i + j], v = a[i + j + len / 2] * w;
          a[i + j] = u + v;
          a[i + j + len / 2] = u - v;
          w *= wlen;
        }
      }
    }
    if (invert)
      for (cd &x : a)
        x /= n;
  }
  vector<int> mul(vector<int> a, vector<int> b)
  {
    vector<cd> fa(a.begin(), a.end()), fb(b.begin(), b.end());
    n = 1;
    while (n < a.size() + b.size())
      n <<= 1;
    fa.resize(n);
    fb.resize(n);
    fft(fa, false);
    fft(fb, false);
    for (int i = 0; i < n; i++)
      fa[i] *= fb[i];
    fft(fa, true);
    vector<int> ans(n);
    for (int i = 0; i < n; i++)
      ans[i] = round(fa[i].real());
    return ans;
  }
}
signed main()
{
  ios::sync_with_stdio(false);
  cin.tie(0);
  int n;
  cin >> n;
  int m = n + n;
  vector<int> a(m, 0);
  vector<int> aa(m, 0);
  for (int i = 0; i < n; i++)
  {
    cin >> a[i];
    aa[i] = (a[i] == 47) ? 1 : 0;
    a[i + n] = a[i];
    aa[i + n] = aa[i];
  }
  vector<int> b(m, 0);
  vector<int> bb(m, 0);
  for (int i = n - 1; i >= 0; i--)
  {
    cin >> b[i + n];
    bb[i + n] = (b[i + n] == 47) ? 1 : 0;
  }
  vector<int> ans1 = fft::mul(a, b);
  vector<int> ans2 = fft::mul(aa, bb);
  int ans = 0;
  for (int i = (m - 1); i < (m - 1) + n; i++)
  {
    if (ans2[i] > 0)
      continue;
    ans = max(ans1[i], ans); // produto escalar de algum cyclic shift
  }
  cout << ans << endl;
  return 0;
}
// https://algo.sk/br24/problem.php?problem=d3-badsquare
// exemplo do all possible scalar products
// dados dois arrays a e b de tamanho n
// quero computar o scalar product de todos os cyclics shifts de a com b
// duplicar o array a
// dar reverse no array b e adicionar n zeros no inicio
```

11.21 segmentedsieve

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define PI acos(-1)
#define pb push_back
#define int long long int
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define DEBUG 0
#define MAXN 1000003
#define mod 1000000007

vector<int> prime;

void segmentedsieve(int l, int r)
{
  int lim = sqrt(r);
  vector<bool> mark(lim + 1, false);
  vector<int> primes;
  for (int i = 2; i <= lim; ++i)
  {
    if (!mark[i])
    {
      primes.pb(i);
      for (int j = i * i; j <= lim; j += i)
        mark[j] = true;
    }
  }
  vector<bool> isprime(r - l + 1, true);
  for (int i : primes)
    for (int j = max(i * i, (l + i - 1) / i * i); j <= r; j += i)
      isprime[j - l] = false;
  if (l == 1)
    isprime[0] = false;
  for (int i = 0; i < isprime.size(); i++)
    if (isprime[i])
      prime.pb(i + l);
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int l, r;
  cin >> l >> r;
  segmentedsieve(l, r);
  for (auto const &i : prime)
    cout << i << " ";
  return 0;
}
```

11.22 gaussian elimination2

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 4007
#define mod 998244353
#define EPS 1e-9

bitset<MAXN> ans;

int gauss(vector<bitset<MAXN>> &a)
{
  ans.reset();
  int n = a.size(), m = a[0].size() - 1, ret = 1;
  vector<int> where(m, -1);
  for (int col = 0, row = 0; col < m && row < n; col++)
  {
    for (int i = row; i < n; i++)
    {
      if (a[i][col])
      {
        swap(a[i], a[row]);
        break;
      }
    }
    if (!a[row][col])
      continue;
    where[col] = row;
    for (int i = 0; i < n; i++)
      if (i != row && a[i][col])
        a[i] ^= a[row];
    ++row;
  }
  for (int i = 0; i < m; i++)
  {
    if (where[i] != -1)
      ans[i] = (a[where[i]][m] / a[where[i]][i]);
    else
      ret = 2;
  }
  for (int i = 0; i < n; i++)
  {
    double sum = 0;
    for (int j = 0; j < m; j++)
      sum += (ans[j] * a[i][j]);
    if (abs(sum - a[i][m]) > EPS)
      ret = 0;
  }
  return ret;
}
signed main()
{
  ios::sync_with_stdio(false);
  cin.tie(0);
  int m, n;
  cin >> m >> n;
  string s;
  getline(cin, s);
  auto get_id = [&](string st)
  {
    return n - stoi(st.substr(1));
  };
  vector<bitset<MAXN>> v(n + m);
  for (int i = 0; i < m; i++)
  {
    getline(cin, s);
    s = s.substr(1); // (
    if (i == m - 1)
      s = s.substr(0, s.size() - 1); // )
    else
      s = s.substr(0, s.size() - 5); // ) and
    istringstream input_stream(s);
    string t;
    while (input_stream >> t)
    {
      if (t == "not") // not var
      {
        input_stream >> t;
        v[i][n + get_id(t)] = v[i][n + get_id(t)] ^ 1;
      }
      else if (t != "or") // var
      {
        v[i][get_id(t)] = v[i][get_id(t)] ^ 1;
      }
    }
    v[i][MAXN - 1] = v[i][MAXN - 1] ^ 1;
  }
  for (int i = 0; i < n; i++)
  {
    v[i + m][i] = v[i + m][i] ^ 1;
    v[i + m][n + i] = v[i + m][n + i] ^ 1;
    v[i + m][MAXN - 1] = v[i + m][MAXN - 1] ^ 1;
  }
  if (gauss(v) == 0)
  {
    cout << "impossible\n";
    return 0;
  }
  string resp(n, 'F');
  int id = n - 1;
  for (int i = 0; i < (n + n); i++)
  {
    if (ans[i])
    {
      (i < n) ? resp[id] = 'T' : resp[id] = 'F';
    }
    id--;
    if (id < 0)
      id = n - 1;
  }
  cout << resp << endl;
  return 0;
}
// exemplo de solucao para o https://codeforces.com/gym/101908/problem/M
// esse codigo ja acha a menor solucao lexicograficamente (caso exista)
// caso a gente queira a maior lexicograficamente (que e o caso desse problema
    exemplo)
// basta considerar as variaveis na ordem contraria
```

11.23 stars and bars

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 100005
#define mod 1000000007

struct modint
{
  int val;
  modint(int v = 0) { val = v % mod; }
  int pow(int y)
  {
    modint x = val;
    modint z = 1;
    while (y)
    {
      if (y & 1)
        z = z * x;
      x = x * x;
      y >>= 1;
    }
    return z.val;
  }
  int inv() { return pow(mod - 2); }
  int operator*(modint o) { return ((val * o.val) % mod); }
  int operator/(modint o) { return (val * o.inv()) % mod; }
  int operator+(modint o) { return (val + o.val) % mod; }
  int operator-(modint o) { return (val - o.val + mod) % mod; }
};

modint ncr(int n, int k)
{
  // calcular combinacao para n grande
  // nesse problema n <= 10^12
  // em O(k)
  modint num = 1;
  modint den = 1;
  for (int i = 0; i < k; i++)
  {
    num = num * modint(n - i);
    den = den * modint(i + 1);
  }
  modint ans = num / den;
  return ans;
}
modint stars_and_bars(int n, int k)
{
  // para pares de inteiros n e k
  // enconte a quantidade de k-tuplas com soma == n
  // x1 + x2 + ... + xk = n
  return ncr(n + k - 1, k - 1);
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n, s;
  cin >> n >> s;
  vector<int> v(n);
  for (int i = 0; i < n; i++)
    cin >> v[i];
  modint all = stars_and_bars(s, n);
  modint to_sub = 0;
  for (int mask = 1; mask < (1 << n); mask++)
  {
    int sum = 0;
    for (int j = 0; j < n; j++)
    {
      if (mask & (1 << j))
        sum += (v[j] + 1);
    }
    if (sum <= s)
    {
      modint curr = stars_and_bars(s - sum, n);
      to_sub = (__builtin_popcount(mask) % 2) ? to_sub + curr : to_sub - curr;
    }
  }
  all = all - to_sub;
  cout << all.val << endl;
  return 0;
}
// stars and bars
// dado dois inteiros positivos n e k
// conte o numero de k-tuplas (x1, x2, .., xk) tal que
// x1 + x2 + ... + xk = n
// com x1, x2, ..., xk >= 0
// resposta = ncr(n + k - 1, k - 1)

// para k-tuplas com x1, x2, ..., xk > 0:
// resposta = ncr(n - 1, k - 1)

// problema exemplo:
// https://codeforces.com/contest/451/problem/E
// contar quantas k-tuplas com soma == n
// tal que: x[i] >= 0 e x[i] <= f[i]
// k <= 20

// solucao:
// conta tudo com stars and bars
// dai preciso subtrair todas as possibilidades invalidas (com pelo menos um i
     tal que x[i] > f[i])
// seja n(i) as possibilidades com x[i] > f[i]
// dai eu quero calcular a quantidade de elementos na uniao entre todos os n(i)
// dai da pra fzr usando a formulinha de uniao de conjuntos:
// n(A uniao B uniao C) = n(A) + n(B) + n(C) - n(A intersecao B) ... + n(A
     intersecao B intersecao C)
// itera por todos os 2^n subsets e calcula o que deve subtrair/somar com
     aqueles caras
```

11.24 fwht

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 1025
#define mod 998244353

struct modint
{
  int val;
  modint(int v = 0) { val = ((v % mod) + mod) % mod; }
  int pow(int y)
  {
    modint x = val;
    modint z = 1;
    while (y)
    {
      if (y & 1)
        z *= x;
      x *= x;
      y >>= 1;
    }
    return z.val;
  }
  int inv() { return pow(mod - 2); }
  void operator=(int o) { val = o % mod; }
  void operator=(modint o) { val = o.val % mod; }
  void operator+=(modint o) { *this = *this + o; }
  void operator-=(modint o) { *this = *this - o; }
  void operator*=(modint o) { *this = *this * o; }
  void operator/=(modint o) { *this = *this / o; }
  bool operator==(modint o) { return val == o.val; }
  bool operator!=(modint o) { return val != o.val; }
  int operator*(modint o) { return ((val * o.val) % mod); }
  int operator/(modint o) { return (val * o.inv()) % mod; }
  int operator+(modint o) { return (val + o.val) % mod; }
  int operator-(modint o) { return (val - o.val + mod) % mod; }
};

vector<modint> fwht(char op, vector<modint> f, bool inv = 0)
{
  int n = f.size();
  for (int k = 0; (n - 1) >> k; k++)
  {
    for (int i = 0; i < n; i++)
    {
      if (i >> k & 1)
      {
        int j = i ^ (1 << k);
        if (op == '^')
          f[j] += f[i], f[i] = f[j] - modint(2) * f[i];
        if (op == '|')
          f[i] += modint(inv ? -1 : 1) * f[j];
        if (op == '&')
          f[j] += modint(inv ? -1 : 1) * f[i];
      }
    }
  }
  if (op == '^' and inv)
  {
    for (auto &i : f)
    i /= n;
  }
  return f;
}
vector<modint> conv(char op, vector<modint> a, vector<modint> b)
{
  a = fwht(op, a, 0);
  b = fwht(op, b, 0);
  for (int i = 0; i < a.size(); i++)
  {
    a[i] *= b[i];
  }
  return fwht(op, a, 1);
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n;
  cin >> n;
  n = 1 << n;
  vector<modint> a(n);
  for (int i = 0; i < n; i++)
  {
    int x;
    cin >> x;
    a[i] = x;
  }
  vector<modint> b(n);
  for (int i = 0; i < n; i++)
  {
    int x;
    cin >> x;
    b[i] = x;
  }
  vector<modint> c = conv('^', a, b); // convolucao de xor
  for (auto const &i : c)
    cout << i.val << " ";
  cout << endl;
  vector<modint> d = conv('&', a, b); // convolucao de and
  for (auto const &i : d)
    cout << i.val << " ";
  cout << endl;
  return 0;
}
// o tipo ta como modint, mas tem como mudar para qualquer um
// usar preferencialmente tamanho como potencia de 2

// faz a convolucao de a com b
// c[k] = (a[i] * b[j]), com (i op j) = k
// op pode ser xor, and ou or

// para testar
// https://judge.yosupo.jp/problem/bitwise_xor_convolution
// https://judge.yosupo.jp/problem/bitwise_and_convolution
```

11.25 operadores binarios

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

#define lli long long int
#define pb push_back
#define in insert
#define pi pair<int, int>
#define pd pair<double, int>
#define pii pair<int, pi>
#define mp make_pair
#define fir first
#define sec second
#define MAXN 200001
#define mod 1000000007

void shifts ()
{
  bitset <4> bs;
  bs.reset();
  bs[2] = true;
  bs[3] = true;
  cout << bs << endl ; // 1100
  bs >>= 1; // 0110
  bs <<= 1; // 1100
  bs >>= 2; // 0011
  bs <<= 2; // 1100
  bs >>= 3; // 0001
  bs <<= 3; // 1000
  cout << bs << endl ;
}
void op_xor ()
{
  // 0 ^ 0 = 0
  // 0 ^ 1 = 1
  // 1 ^ 0 = 1
  // 1 ^ 1 = 0
  bitset <4> bs , bs2;
  bs.reset();
  bs2.reset();
  bs[2] = true;
  bs[3] = true;
  bs2[1] = true;
  bs2[3] = true;
  bs ^= bs2; // bs = bs ^ bs2
  cout << bs.count() << endl ;
}
void op_and ()
{
  // 0 & 0 = 0
  // 0 & 1 = 0
  // 1 & 0 = 0
  // 1 & 1 = 1
  bitset <4> bs , bs2;
  bs.reset();
  bs2.reset();
  bs[2] = true;
  bs[3] = true;
  bs2[1] = true;
  bs2[3] = true;
  bs &= bs2; // bs = bs & bs2
  cout << bs.count() << endl ;
}
void op_or ()
{
  // 0 | 0 = 0
  // 0 | 1 = 1
  // 1 | 0 = 1
  // 1 | 1 = 1
  bitset <4> bs , bs2;
  bs.reset(); // poe tudo 0
  bs2.reset();
  bs[2] = true;
  bs[3] = true;
  bs2[1] = true;
  bs2[3] = true;
  bs |= bs2; // bs = bs | bs2
  cout << bs.count() << endl ; // quantidade de 1
}
signed main()
{
  op_or();
  op_and();
  op_xor();
  shifts();
  return 0;
}
```

11.26 matrix exponentiation

```
// https://codeforces.com/gym/102644/problem/C
// achar o n-esimo termo da sequencia de fibonacci mod (10^9 + 7) em O(log(n))
// n <= 10^18
// podemos escrever a recorrencia de fibonnaci como uma exponenciacao de matriz
/*
  (  fib(n)   )       (1 1) ^ (n - 1)       (fib(1) = 1)
  (fib(n - 1))    =   (1 0)             *   (fib(0) = 1)
*/
// e possivel fazer essa exponenciacao em O(log(n)) com um algoritimo muito
     similar ao de exponenciacao rapida
// dai calculamos o n-esimo termo da sequencia de fibonacci mod (10^9 + 7) em O(
     log(n))

#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define PI acos(-1)
#define pb push_back
#define int long long int
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define DEBUG 0
#define MAXN 201
#define mod 1000000007

namespace matrix
{
  vector<vector<int>> ans;

  int multi(int x, int y)
  {
    return (x * y) % mod;
  }
  int sum(int a, int b)
  {
    return (a + b >= mod) ? a + b - mod : a + b;
  }
  vector<vector<int>> multiply(vector<vector<int>> a, vector<vector<int>> b)
  {
    vector<vector<int>> res(a[0].size(), vector<int>(b[0].size()));
    for (int i = 0; i < a.size(); i++)
    {
      for (int j = 0; j < b[0].size(); j++)
      {
        res[i][j] = 0;
        for (int k = 0; k < a[0].size(); k++)
          res[i][j] = sum(res[i][j], multi(a[i][k], b[k][j]));
      }
    }
    return res;
  }
  vector<vector<int>> expo(vector<vector<int>> mat, int m)
  {
    ans = vector<vector<int>>(mat.size(), vector<int>(mat[0].size()));
    for (int i = 0; i < mat.size(); i++)
      for (int j = 0; j < mat[0].size(); j++)
        ans[i][j] = (i == j);
    while (m > 0)
    {
      if (m & 1)
        ans = multiply(ans, mat);
      m = m / 2;
      mat = multiply(mat, mat);
    }
    return ans;
  }
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n;
  cin >> n;
  vector<vector<int>> mat = {{1, 1}, {1, 0}};
  vector<vector<int>> ans = matrix::expo(mat, n);
  cout << ans[0][1] << endl;
  return 0;
}
```

11.27 max xor subsequence

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<pi, int>
#define fir first
#define sec second
#define MAXN 300005
#define mod 1000000007

int modpow(int a, int b)
{
  int res = 1;
  while (b > 0)
  {
    if (b & 1)
      res = (res * a) % mod;
    a = (a * a) % mod;
    b >>= 1;
  }
  return res;
}

int all, qt;
int dp[33];

void add(int x)
{
  all++;
  for (int i = 32; i >= 0; i--)
  {
    if (x & (1ll << i))
    {
      if (dp[i] == 0)
      {
        dp[i] = x;
        qt++;
        return;
      }
      x ^= dp[i];
    }
  }
}
int get(int x) // qual o x-esimo menor valor de xor de uma subsequencia
{
  int tot = (1ll << qt), ans = 0;
  for (int i = 32; i >= 0; i--)
  {
    if (dp[i] > 0)
    {
      int d = tot / 2;
      if (d < x && !(ans & (1ll << i)))
        ans ^= dp[i];
      else if (d >= x && (ans & (1ll << i)))
        ans ^= dp[i];
      if (d < x)
```

```
        x -= d;
      tot /= 2;
    }
  }
  return ans;
}
bool check(int x) // se existe pelo menos uma subsequencia com xor x
{
  for (int i = 32; i >= 0; i--)
  {
    if (x & (1ll << i))
    {
      if (!dp[i])
        return 0;
      x ^= dp[i];
    }
  }
  return 1;
}
int count(int x) // quantas subsequencias tem xor x
{
  if (!check(x))
    return 0;
  return modpow(2, all - qt);
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n;
  cin >> n;
  vector<int> v(n);
  for (int i = 0; i < n; i++)
  {
    cin >> v[i];
    add(v[i]);
  }
  int x = get(1ll << qt); // maior xor possivel de uma subsequencia
  int y = get(1);         // maior xor possivel != 0 (o 0 sempre eh possivel - subsequencia vazia)
  return 0;
}
// referencia:
// https://codeforces.com/blog/entry/68953

// problemas:
// https://codeforces.com/gym/103708/problem/A
// https://codeforces.com/contest/959/problem/F
// https://codeforces.com/contest/1101/problem/G
// https://atcoder.jp/contests/abc283/tasks/abc283_g
```

11.28 divisors

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

#define PI acos(-1)
#define int long long int
#define pb push_back
#define pi pair<int, int>
#define fir first
#define sec second
#define MAXN 5001
#define mod 1000000007

signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n;
  cin >> n;
  int ans = 0;
  for (int i = 1; i <= sqrt(n); i++)
  {
    if (!(n % i))
    {
      ans++;
      if (n / i != i)
        ans++;
    }
  }
  cout << ans << endl;
}
```

11.29 ntt

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 250005
#define mod 998244353

struct modint
{
  int val;
  modint(int v = 0) { val = ((v % mod) + mod) % mod; }
  int pow(int y)
  {
    modint x = val;
    modint z = 1;
    while (y)
    {
      if (y & 1)
        z *= x;
      x *= x;
      y >>= 1;
    }
    return z.val;
  }
  int inv() { return pow(mod - 2); }
  void operator=(int o) { val = o % mod; }
  void operator=(modint o) { val = o.val % mod; }
  void operator+=(modint o) { *this = *this + o; }
  void operator-=(modint o) { *this = *this - o; }
  void operator*=(modint o) { *this = *this * o; }
  void operator/=(modint o) { *this = *this / o; }
  bool operator==(modint o) { return val == o.val; }
  bool operator!=(modint o) { return val != o.val; }
  int operator*(modint o) { return ((val * o.val) % mod); }
  int operator/(modint o) { return (val * o.inv()) % mod; }
  int operator+(modint o) { return (val + o.val) % mod; }
  int operator-(modint o) { return (val - o.val + mod) % mod; }
};
namespace fft
{
  // para o modulo ser valido
  // precisa ser primo
  // precisa possuir a forma c * 2^k + 1
  // 998244353 - possui a forma - c * 2^k + 1 e eh primo
  int n;
  int root = -1;
  int root_1 = -1;
  int pw = __builtin_ctz(mod - 1);
  int root_pw = (1 << pw);

  void find_root()
  {
    if (root != -1)
```

```
      return;
    int r = 2;
    while (!(modint(r).pow((1 << pw)) == 1 && modint(r).pow((1 << (pw - 1))) !=
        1))
      r++;
    root = r;
    root_1 = modint(root).inv();
  }
  void ntt(vector<modint> &a, bool invert)
  {
    find_root();
    int n = a.size();
    for (int i = 1, j = 0; i < n; i++)
    {
      int bit = n >> 1;
      for (; j & bit; bit >>= 1)
        j ^= bit;
      j ^= bit;
      if (i < j)
        swap(a[i], a[j]);
    }
    for (int len = 2; len <= n; len <<= 1)
    {
      modint wlen = (invert) ? root_1 : root;
      for (int i = len; i < root_pw; i <<= 1)
        wlen *= wlen;
      for (int i = 0; i < n; i += len)
      {
        modint w = 1;
        for (int j = 0; j < len / 2; j++)
        {
          modint u = a[i + j];
          modint v = a[i + j + len / 2] * w;
          a[i + j] = u + v;
          a[i + j + len / 2] = u - v;
          w *= wlen;
        }
      }
    }
    if (invert)
    {
      modint n_1 = modint(n).inv();
      for (int i = 0; i < a.size(); i++)
        a[i] *= n_1;
    }
  }
  vector<modint> mul(vector<modint> a, vector<modint> b)
  {
    n = 1;
    while (n < 2 * max(a.size(), b.size()))
      n <<= 1;
    a.resize(n);
    b.resize(n);
    ntt(a, false);
    ntt(b, false);
    for (int i = 0; i < n; i++)
      a[i] *= b[i];
    ntt(a, true);
    return a;
  }
} // namespace fft
  // https://codeforces.com/contest/1613/problem/F
```

11.30 modular arithmetic

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 500001
#define mod 1000000007

struct modint
{
  int val;
  modint(int v = 0) { val = v % mod; }
  int pow(int y)
  {
    modint x = val;
    modint z = 1;
    while (y)
    {
      if (y & 1)
        z *= x;
      x *= x;
      y >>= 1;
    }
    return z.val;
  }
  int inv() { return pow(mod - 2); }
  void operator=(int o) { val = o % mod; }
  void operator=(modint o) { val = o.val % mod; }
  void operator+=(modint o) { *this = *this + o; }
  void operator-=(modint o) { *this = *this - o; }
  void operator*=(modint o) { *this = *this * o; }
  void operator/=(modint o) { *this = *this / o; }
  bool operator==(modint o) { return val == o.val; }
  bool operator!=(modint o) { return val != o.val; }
  int operator*(modint o) { return ((val * o.val) % mod); }
  int operator/(modint o) { return (val * o.inv()) % mod; }
  int operator+(modint o) { return (val + o.val) % mod; }
  int operator-(modint o) { return (val - o.val + mod) % mod; }
};

modint f[MAXN];
modint inv[MAXN];
modint invfat[MAXN];

void calc()
{
  f[0] = 1;
  for (int i = 1; i < MAXN; i++)
  {
    f[i] = f[i - 1] * i;
  }
  inv[1] = 1;
  for (int i = 2; i < MAXN; ++i)
  {
    int val = mod / i;
    val = (inv[mod % i] * val) % mod;
    val = mod - val;
    inv[i] = val;
  }
  invfat[0] = 1;
  invfat[MAXN - 1] = modint(f[MAXN - 1]).inv();
  for (int i = MAXN - 2; i >= 1; i--)
  {
    invfat[i] = invfat[i + 1] * (i + 1);
  }
}
modint ncr(int n, int k) // combinacao
{
  modint ans = f[n] * invfat[k];
  ans *= invfat[n - k];
  return ans;
}
modint arr(int n, int k) // arranjo
{
  modint ans = f[n] * invfat[n - k];
  return ans;
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  return 0;
}
```

11.31 crivo

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

#define lli long long int
#define pb push_back
#define in insert
#define pi pair<int, int>
#define pd pair<double, int>
#define pii pair<int, pi>
#define mp make_pair
#define fir first
#define sec second
#define MAXN 100001
#define mod 1000000007

bitset <MAXN> prime;

void crivo ()
{
  prime.set();
  prime[0] = false;
  prime[1] = false;
  for (int i = 2 ; i < MAXN ; i++)
    if(prime[i])
      for(int j = 2 ; j * i < MAXN ; j++)
        prime[j * i] = false;
}
signed main()
{
  crivo();
  int q;
  cin >> q;
  while(q--)
  {
    int n;
    cin >> n;
    (prime[n]) ? cout << "YES\n" : cout << "NO\n" ;
  }
  return 0;
}
```

11.32 mobius

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

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

#define int long long int
#define endl '\n'
#define pb push_back
#define pi pair<int, int>
#define pii pair<int, pi>
#define fir first
#define sec second
#define MAXN 5000005
#define mod 1000000001

int lpf[MAXN];
int mobius[MAXN];
int g[MAXN];

void calc_lpf()
{
  for (int i = 2; i < MAXN; i++)
  {
    if (!lpf[i])
    {
      for (int j = i; j < MAXN; j += i)
      {
        if (!lpf[j])
          lpf[j] = i;
      }
    }
  }
}
void calc_mobius()
{
  calc_lpf();
  mobius[1] = 1;
  for (int i = 2; i < MAXN; i++)
  {
    if (lpf[i / lpf[i]] == lpf[i])
      mobius[i] = 0;
    else
      mobius[i] = -1 * mobius[i / lpf[i]];
  }
}
int count_pairs(int n)
{
  // f(n) -> contar pares (i, j) com __gcd(i, j) == 1 e 1 <= i, j <= n
  int ans = 0;
  for (int d = 1; d <= n; d++)
  {
    // quadrado pq sao pares (2 caras)
    // mas se fossem x caras seria (n / d)^x
    int sq = (n / d) * (n / d);
    int x = mobius[d] * sq;
    ans += x;
  }
  return ans;
}
int gcd_sum(int n)
{
  // soma de todos os gcd(i, j) com 1 <= i, j <= n
  int ans = 0;
  for (int k = 1; k <= n; k++) // fixa o valor do gcd(i, j) e conta quantos
     pares com gcd(i, j) == k
  {
    int lim = n / k;
    int curr = k * count_pairs(lim);
    ans += curr;
  }
  return ans;
}
int lcm_sum(int n)
{
  // soma de todos os lcm(i, j) com 1 <= i, j <= n
  for (int i = 1; i <= n; i++)
    g[i] = 0;
  for (int i = 1; i <= n; i++)
  {
    for (int j = i; j <= n; j += i)
      g[j] += (mobius[i] * j * i);
  }
  int ans = 0;
  for (int l = 1; l <= n; l++)
  {
    int cima = (1 + n / l) * (n / l);
    int f = (cima / 2) * (cima / 2);
    f *= g[l];
    ans += f;
  }
  return ans;
}
signed main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int q;
  cin >> q;
  calc_mobius();
  for (int i = 1; i <= q; i++)
  {
    int n;
    cin >> n;
    int ans = lcm_sum(n);
    for (int i = 1; i <= n; i++)
      ans -= i;
    ans /= 2;
    cout << "Case " << i << ": " << ans << endl;
  }
  return 0;
}
// https://codeforces.com/blog/entry/53925
// mobius inversion
// sejam f(x) e g(x) funcoes
// e g(x) e definida da seguinte maneira
// g(x) = soma dos f(d), no qual d eh um divisor de x

// temos que:
// f(n) = soma dos (g(d) * u(n / d)), no qual d eh um divisor de x
// u(x) -> mobius function

// propiedade legal:
// seja l(x) -> soma de u(d), para cada divisor d de x
// l(1) = 1
// l(x) = 0, x > 1

// problemas iniciais:
// https://vjudge.net/problem/AtCoder-abc162_e
// https://vjudge.net/problem/CodeChef-SMPLSUM
```