Library

Tricks

1. Memory optimization of bitset solutions.
2. Square root optimization of knapsack/"$3k$ trick": Assume you have n rocks with nonnegative integer weights $a_1, a_2, \ldots, a_n$ such that $a_1 + a_2 + \ldots + a_n = m$.You want to find out if there is a way to choose some rocks such that their total weight is w.Suppose there are three rocks with equal weights a,a,a.Notice that it doesn't make any difference if we replace these three rocks with two rocks with weights $a, 2a$.We can repeat this process of replacing until there are at most two rocks of each weight. The sum of weights is still m,so there can be only $\sqrt{m}$ rocks(see next point).
3. Number of unique elements in a partition: Assume there are n nonnegative integers $a_1 + a_2 + \ldots + a_n = m$,Then there are only $O(\sqrt{m})$ distinct values.
4. Removing elements from a knapsack:

```
Adding a new item is classical:

1 # we go from large to small so that the already updated dp values won't affect any
calculations
2 for (int i = dp.size() - 1; i >= weight; i--) {
3     dp[i] += dp[i - weight];
4 }
```

```
To undo what we just did, we can simply do everything backwards:

1 # this moves the array back to the state as it was before the item was added
2 for (int i = weight; i < dp.size(); i++) {
3     dp[i] -= dp[i - weight];
4 }
```

5. $O(n^2)$ complexity of certain tree DPs:

```
function calc_dp (u):
    for each child v of u:
        calc_dp(v)
    for each child v of u:
        for each child w of u other than v:
            for i in 0..length(dp[v])-1:
                for j in in 0..length(dp[w])-1:
                    # calculate something
                    # typically based on dp[v][i] and dp[w][j]
                    # commonly assign to dp[u][i + j]

calc_dp(root)
```

6. $O(n \times k)$ complexity of certain tree DPs: Suppose instead of the vector being the length of the subtree of u,it is the minimum of k and the length of the subtree of u.

```
function calc_dp (u):
    for each child v of u:
        calc_dp(v)
    dp[u]=[0]
    for each child v of u:
        temp=[0,0,...,0]
        for i in 0..length(dp[u])-1:
            for j in in 0..length(dp[v])-1:
                if i+j<K:
                    # calculate something
                    # typically based on dp[u][i] and dp[v][j]
                    # commonly assign to temp[i + j]
        pop elements from temp until length(temp)<=K
        dp[u]=temp

calc_dp(root)
```

7. $O(n)$ complexity for some tree DPs: If you merge two subtrees in $O(min(depth_1, depth_2))$,you get exactly $n - treeDepth$ operations in total.This is because every node is merged exactly once! We can imagine that when states in a are merged into b, they just disappear.
8. How to precompute inverse:

```
for (int i = 1; i < N; ++i) {
    inv[i] = (i == 1) ? 1 : mod - 1ll * inv[mod % i] * (mod / i) % mod;
}
```

9. Formula and tips:
    1. there are $O(\frac{n}{log(n)})$ primes up to n.
    2. the nth primes is $O(n \times log(n))$.
    3. $a \equiv b \pmod{p} \iff p \mid b - a$
    4. $a \equiv b \pmod{m}, a \equiv b \pmod{n} \to a \equiv b \pmod{[m, n]}$

5. $(k,m)=d, ka_1 \equiv ka_2 \pmod{m} \rightarrow a_1 \equiv a_2 \pmod{\frac{m}{d}}$
6. $k \times C_n^k = n \times C_{n-1}^{k-1}$
7. $C_k^n \times C_m^k = C_m^n \times C_{m-n}^{m-k} \ (m-k<m-n)$
8. $\sum_0^n C_n^i = 2^n$
9. $\sum_{k=0}^n (-1)^k \times C_n^k = 0$
10. $C_n^k + C_n^{k+1} = C_{n+1}^{k+1}$
11. $\sum_{k=0}^m C_{n+k}^k = C_{n+m+1}^m$
12. $Tolerance$:

由三圆图可得公式：$S_1 \cup S_2 \cup S_3 = S_1 + S_2 + S_3 - (S_1 \cap S_2 + S_1 \cap S_3 + S_2 \cap S_3) + S_1 \cap S_2 \cap S_3$。

推广可得：

$$\bigcup_{i=1}^{m} S_i = \sum_{i=1}^{m} S_i - (S_1 \cap S_2 + S_1 \cap S_3 + \cdots + S_{m-1} \cap S_m) + \cdots + (-1)^{m-1} \bigcap_{i=1}^{m} S_i$$

对于集合 S 的每个元素 x，有 $x, k(1 \le k \le n)$，

$$cnt_x = C_k^1 - C_k^2 + C_k^3 + \cdots + (-1)^{k-1} C_k^k = 1。$$

可得容斥原理的正确性，而根据组合数恒等式推广可知：

$$\sum_{k=0}^{n} (-1)^k C_n^k = 0$$

将等式 cnt_x 两边同减去 $C_k^0 = 1$ 即可得到上式。

0. Series:
    1. $e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots$
    2. $ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \ldots$
    3. $\sqrt{1+x} = 1 + \frac{x}{2} - \frac{x^2}{8} + \frac{2x^3}{32} - \frac{5x^4}{128} + \ldots$
    4. $\frac{1}{1-x} = \sum_{i \ge 0} x^i$
    5. $\frac{1}{1-ax} = \sum_{i \ge 0} a^i x^i$
    6. $\frac{1}{(1-x)^k} = \sum_{i \ge 0} \binom{i+k-1}{i} x^i$
    7. $(1+x)^k = \sum_{n \ge 0} \binom{k}{n} x^n$
    8. $log(P(x)) = \int \frac{P'(x)}{P(x)}$
1. 欧拉序求LCA:

    $LCA(u,v) = RMQ(first(u), first(v))$
2. MoTree:

```
dfs 一棵树，然后如果 dfs 到 x 点，就 `push_back(x)`，dfs 完 x 点，就直接 `push_back(-x)`

新加入的值是 x --->`add(x)`

新加入的值是 - x --->`del(x)`

新删除的值是 x --->`del(x)`

新删除的值是 - x --->`add(x)`
```

对于u和v，假设$in(u) < in(v)$：

若$LCA(u,v) == u$，则为$in(u)$到$in(v)$这段区间。

若$LCA(u,v) != u$，则为$out(u)$到$in(v)$，需要额外加上$LCA(u,v)$的贡献。

❗❗(括号序 $\neq$ 欧拉序)❗❗

3. LCS：将S_1中的字符替换为在S_2中所有出现的下标(按照降序)，转化为LIS。
4. LIS：树状数组/二分优化到$O(nlogn)$。
5. 第一类斯特林数(无符号)

   长度为n的排列构成m个圆(非空轮换)的方案数，记作$S_u(n,m)$。

   递推式：$S_u(n,m) = S_u(n-1,m-1) + S_u(n-1,m) * (n-1)$。

   边界：$S_u(n,0) = [n == 0]$。
6. 第二类斯特林数

   把n个不同的数划分为m个集合的方案数，要求不能为空集，记作$S(n,m)$。

   递推式：$S(n,m) = S(n-1,m-1) + S(n-1,m) \times m$。
7. 整数划分

   一个正整数n写成多个大于等于1且小于等于其本身的整数的和,其中各加数构成的集合为n的一个划分。

   递推式：$f(n,m) = f(n,m-1) + f(n-m,m)$
8. 卡特兰数

   进栈出栈顺序，对角线，三角形划分，n个节点构成不同的二叉树.....

   递推式：$H_{n+1} = \sum_{i=0}^{n} H_i H_{n-i}$

   通项：$H_n = \frac{C_{2n}^n}{n+1}$
9. 拉格朗日插值

   $f(x) = \sum_{i=1}^{n} y_i \prod_{j \neq i} \frac{x-x_j}{x_i-x_j}$
10. 四边形不等式

    区间包含单调性：若$l \leq l' \leq r' \leq r$，则$w(l',r') \leq w(l,r)$。

    四边形不等式：交叉不大于包含，$w(l,r') + w(l',r) \leq w(l,r) + w(l',r')$。

    由四边形不等式可推导出$1D/2D$决策单调性。

Head

```
#include <bits/extc++.h>
using namespace std;
#define rep(i,a,b) for (int i=a;i<=b;i++)
#define per(i,b,a) for (int i=b;i>=a;i--)
typedef long long ll;
typedef unsigned long long ull;
typedef double db;
typedef long double ldb;
typedef pair<int, int> PII;
typedef pair<long long, long long> PLL;
typedef pair<double, double> PDD;
typedef vector<int> VI;
typedef vector<long long> VLL;
mt19937_64 rng(random_device {}());
template<typename ...T>
void debug_out(T... args) {((cerr << args << " "), ...); cerr << '\n';}
#define pb push_back
#define eb emplace_back
#define fi first
#define se second
#define mp make_pair
#define bit(x) (1ll<<(x))
#define SZ(x) ((int)x.size())
#define all(x) x.begin(),x.end()
#define debug(...) cerr << "[" << #__VA_ARGS__ << "]:", debug_out(__VA_ARGS__)
ll powmod(ll a, ll b, const ll p) {ll res = 1; while (b) {if (b & 1) res = res * a % p; b >>=
1; a = a * a % p;} return res;}
ll gcd(ll a, ll b) {return b == 0 ? a : gcd(b, a % b);}

const int mod = 998244353;
const ll inf = 1ll << 55;
const double pi = acosl(-1);
const double eps = 1e-12;
const int maxn = 1e6 + 105;
const int N = 5005;

ll n, m, k;
void solve() {

}
int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    int tt = 1;
    cin >> tt;
    while (tt--) {
        solve();
    }
}
```

01trie

```
struct node {
    int son[2];
    int end;
    int sz;
} seg[maxn << 2];
int root, tot;
int n, m;

void insert(ll x) {
    int cnt = root;
    for (int i = 62; i >= 0; i--) {
        int w = (x >> i) & 1;
        if (seg[cnt].son[w] == 0) seg[cnt].son[w] = ++tot;
        cnt = seg[cnt].son[w];
        seg[cnt].sz++;
    }
    seg[cnt].end++;
}

ll query(ll x, ll k) {
    ll res = 0;
    int cnt = root;
    for (int i = 62; i >= 0; i--) {
        int w = (x >> i) & 1;
        if (seg[seg[cnt].son[w]].sz >= k) cnt = seg[cnt].son[w];
        else {
            k -= seg[seg[cnt].son[w]].sz;
            cnt = seg[cnt].son[abs(w - 1)];
            res += bit(i);
        }
    }
    return res;
}
```

2DTree(Benq)

```
/**
 * Description: Does not allocate storage for nodes with no data
 * Source: USACO Mowing the Field
 * Verification: ~
 */

const int SZ = 1<<17;
template<class T> struct node {
    T val = 0; node<T>* c[2];
    node() { c[0] = c[1] = NULL; }
```

```
    void upd(int ind, T v, int L = 0, int R = SZ-1) { // add v
        if (L == ind && R == ind) { val += v; return; }
        int M = (L+R)/2;
        if (ind <= M) {
            if (!c[0]) c[0] = new node();
            c[0]->upd(ind,v,L,M);
        } else {
            if (!c[1]) c[1] = new node();
            c[1]->upd(ind,v,M+1,R);
        }
        val = 0; F0R(i,2) if (c[i]) val += c[i]->val;
    }
    T query(int lo, int hi, int L = 0, int R = SZ-1) { // query sum of segment
        if (hi < L || R < lo) return 0;
        if (lo <= L && R <= hi) return val;
        int M = (L+R)/2; T res = 0;
        if (c[0]) res += c[0]->query(lo,hi,L,M);
        if (c[1]) res += c[1]->query(lo,hi,M+1,R);
        return res;
    }
    void UPD(int ind, node* c0, node* c1, int L = 0, int R = SZ-1) { // for 2D segtree
        if (L != R) {
            int M = (L+R)/2;
            if (ind <= M) {
                if (!c[0]) c[0] = new node();
                c[0]->UPD(ind,c0?c0->c[0]:NULL,c1?c1->c[0]:NULL,L,M);
            } else {
                if (!c[1]) c[1] = new node();
                c[1]->UPD(ind,c0?c0->c[1]:NULL,c1?c1->c[1]:NULL,M+1,R);
            }
        }
        val = (c0?c0->val:0)+(c1?c1->val:0);
    }
};

/**
 * Description: BIT of SegTrees. $x\in (0,SZ), y\in [0,SZ)$.
 * Memory: O(N\log^2 N)
 * Source: USACO Mowing the Field
 * Verification:
    * USACO Mowing the Field
    * http://www.usaco.org/index.php?page=viewproblem2&cpid=722 (13/15, 15/15 and 1857ms with
BumpAllocator)
 */

#include "../1D Range Queries (9.2)/SparseSeg (9.2).h"

template<class T> struct BITseg {
    node<T> seg[SZ];
    BITseg() { F0R(i,SZ) seg[i] = node<T>(); }
    void upd(int x, int y, int v) { // add v
        for (; x < SZ; x += x&-x) seg[x].upd(y,v); }
```

```
    T query(int x, int yl, int yr) {
        T res = 0; for (; x; x-=x&-x) res += seg[x].query(yl,yr);
        return res; }
    T query(int xl, int xr, int yl, int yr) { // query sum of rectangle
        return query(xr,yl,yr)-query(xl-1,yl,yr); }
};

/**
 * Description: SegTree of SegTrees. $x,y\in [0,SZ).$
 * Memory: O(N\log^2 N)
 * Source: USACO Mowing the Field
 * Verification:
    * http://www.usaco.org/index.php?page=viewproblem2&cpid=722 (9/15 w/ BumpAllocator)
    * http://www.usaco.org/index.php?page=viewproblem2&cpid=601 (4238 ms, 2907 ms w/
BumpAllocator)
 */

#include "../1D Range Queries (9.2)/SparseSeg (9.2).h"

template<class T> struct Node {
    node<T> seg; Node* c[2];
    Node() { c[0] = c[1] = NULL; }
    void upd(int x, int y, T v, int L = 0, int R = SZ-1) { // add v
        if (L == x && R == x) { seg.upd(y,v); return; }
        int M = (L+R)/2;
        if (x <= M) {
            if (!c[0]) c[0] = new Node();
            c[0]->upd(x,y,v,L,M);
        } else {
            if (!c[1]) c[1] = new Node();
            c[1]->upd(x,y,v,M+1,R);
        }
        seg.upd(y,v); // only for addition
        // seg.UPD(y,c[0]?&c[0]->seg:NULL,c[1]?&c[1]->seg:NULL);
    }
    T query(int x1, int x2, int y1, int y2, int L = 0, int R = SZ-1) { // query sum of
rectangle
        if (x1 <= L && R <= x2) return seg.query(y1,y2);
        if (x2 < L || R < x1) return 0;
        int M = (L+R)/2; T res = 0;
        if (c[0]) res += c[0]->query(x1,x2,y1,y2,L,M);
        if (c[1]) res += c[1]->query(x1,x2,y1,y2,M+1,R);
        return res;
    }
};
```

笛卡尔树

```
int a[maxn], l[maxn], r[maxn], root;
int ans[maxn], tot;
void build() {
    stack<int> stk;
    for (int i = 1; i <= n; i++) {
        int last = 0;
        while (!stk.empty() && a[stk.top()] > a[i]) {
            last = stk.top();
            stk.pop();
        }
        if (stk.empty())
            root = i;
        else
            r[stk.top()] = i;
        l[i] = last;
        stk.push(i);
    }
}
void dfs(int c, int L, int R) {
    ans[c] = ++tot;
    if (l[c]) dfs(l[c], L, c - 1);
    if (r[c]) dfs(r[c], c + 1, R);
}
```

树哈希

```
basic_string<int> e[maxn];
ull hashv[maxn];
ull seed1, seed2, seed3, seed4;

ull f(ull x) { return x * x * x * seed1 + x * seed2; }
ull h(ull x) { return f(x) ^ ((x & seed3) >> 31) ^ ((x & seed4) << 31); }

void dfs1(int u, int fa) {
    hashv[u] = 1;
    for (auto v : e[u]) if (v != fa) {
            dfs1(v, u);
            hashv[u] += h(hashv[v]);
        }
}

void dfs2(int u, int fa, ull fv) {
// for each root
    hashv[u] += fv;
    for (auto v : e[u]) if (v != fa) {
            dfs2(v, u, h(hashv[u] - h(hashv[v])));
        }
}
```

```
void solve() {
    seed1 = rng(), seed2 = rng();
    seed3 = rng(), seed4 = rng();
    cin >> n;
    rep(i, 2, n) {
        int u, v;
        cin >> u >> v;
        e[u].pb(v);
        e[v].pb(u);
    }
    dfs1(1, 0);
    sort(hashv + 1, hashv + n + 1);
    n = unique(hashv + 1, hashv + n + 1) - hashv - 1;
    cout << n << '\n';
}
```

树链剖分&SegTree

```
int n, m, a[N];
vector<int> e[N];
int l[N], r[N], idx[N];
int sz[N], hs[N], tot, top[N], dep[N], fa[N];

struct info {
  int maxv, sum;
};

info operator + (const info &l, const info &r) {
  return (info){max(l.maxv, r.maxv), l.sum + r.sum};
}

struct node {
  info val;
} seg[N * 4];

// [l, r]

void update(int id) {
  seg[id].val = seg[id * 2].val + seg[id * 2 + 1].val;
}

void build(int id, int l, int r) {
  if (l == r) {
    // 1号点， DFS序中第1个点‼️
    seg[id].val = {a[idx[l]], a[idx[l]]};
  } else {
    int mid = (l + r) / 2;
    build(id * 2, l, mid);
    build(id * 2 + 1, mid + 1, r);
    update(id);
```

```
  }
}

void change(int id, int l, int r, int pos, int val) {
  if (l == r) {
    seg[id].val = {val, val};
  } else {
    int mid = (l + r) / 2;
    if (pos <= mid) change(id * 2, l, mid, pos, val);
    else change(id * 2 + 1, mid + 1, r, pos, val);
    update(id);
  }
}

info query(int id, int l, int r, int ql, int qr) {
  if (l == ql && r == qr) return seg[id].val;
  int mid = (l + r) / 2;
  if (qr <= mid) return query(id * 2, l, mid, ql, qr);
  else if (ql > mid) return query(id * 2 + 1, mid + 1, r, ql,qr);
  else {
    return query(id * 2, l, mid, ql, mid) +
      query(id * 2 + 1, mid + 1, r, mid + 1, qr);
  }
}

// 第一遍DFS，子树大小，重儿子，父亲，深度
void dfs1(int u,int f) {
  sz[u] = 1;
  hs[u] = -1;
  fa[u] = f;
  dep[u] = dep[f] + 1;
  for (auto v : e[u]) {
    if (v == f) continue;
    dfs1(v, u);
    sz[u] += sz[v];
    if (hs[u] == -1 || sz[v] > sz[hs[u]])
      hs[u] = v;
  }
}

// 第二遍DFS，每个点DFS序，重链上的链头的元素。
void dfs2(int u, int t) {
  top[u] = t;
  l[u] = ++tot;
  idx[tot] = u;
  if (hs[u] != -1) {
    dfs2(hs[u], t);
  }
  for (auto v : e[u]) {
    if (v != fa[u] && v != hs[u]) {
      dfs2(v, v);
    }
```

```
  }
  r[u] = tot;
}

int LCA(int u, int v) {
  while (top[u] != top[v]) {
    if (dep[top[u]] < dep[top[v]]) v = fa[top[v]];
    else u = fa[top[u]];
  }
  if (dep[u] < dep[v]) return u;
  else return v;
}

info query(int u,int v) {
  info ans{(int)-1e9, 0};
  while (top[u] != top[v]) {
    if (dep[top[u]] < dep[top[v]]) {
      ans = ans + query(1, 1, n, l[top[v]], l[v]);
      v = fa[top[v]];
    } else {
      ans = ans + query(1, 1, n, l[top[u]], l[u]);
      u = fa[top[u]];
    }
  }
  if (dep[u] <= dep[v]) ans = ans + query(1, 1, n, l[u], l[v]);
  else ans = ans + query(1, 1, n, l[v], l[u]);
  return ans;
}
```

树套树

```
struct node {
    int v;
    int l,r,rt;
    node(): v(0),l(0),r(0),rt(0) {}
} seg[maxn*20];
int tot;

ll y_query(int u,int l,int r,int ql,int qr) {
    if (!u) return 0;
    if (l==ql&&r==qr) {
        return seg[u].v;
    }
    int mid=(l+r)>>1;
    if (qr<=mid) return y_query(seg[u].l,l,mid,ql,qr);
    else if (ql>mid) return y_query(seg[u].r,mid+1,r,ql,qr);
    else return y_quer y(seg[u].l,l,mid,ql,mid)
        +y_query(seg[u].r,mid+1,r,mid+1,qr);
}

ll x_query(int u,int l,int r,int xl,int xr,int yl,int yr) {
```

```
    if (!u) return 0;
    if (xl==l&&xr==r) {
        return y_query(seg[u].rt,1,n,yl,yr);
    }
    int mid=(l+r)>>1;
    if (xr<=mid) return x_query(seg[u].l,l,mid,xl,xr,yl,yr);
    else if (xl>mid) return x_query(seg[u].r,mid+1,r,xl,xr,yl,yr);
    else return x_query(seg[u].l,l,mid,xl,mid,yl,yr)
        +x_query(seg[u].r,mid+1,r,mid+1,xr,yl,yr);
}

int y_modify(int u,int l,int r,int y,ll v) {
    if (!u) u=++tot;
    if (l==r) {
        seg[u].v+=v;
        return u;
    } else {
        int mid=(l+r)>>1;
        if (y<=mid) seg[u].l=y_modify(seg[u].l,l,mid,y,v);
        else seg[u].r=y_modify(seg[u].r,mid+1,r,y,v);
        seg[u].v=seg[seg[u].l].v+seg[seg[u].r].v;
        return u;
    }
}

int x_modify(int u,int l,int r,int x,int y,ll v) {
    if (!u) u=++tot;
    seg[u].rt=y_modify(seg[u].rt,1,n,y,v);
    if (l==r) return u;
    int mid=(l+r)>>1;
    if (x<=mid) seg[u].l=x_modify(seg[u].l,l,mid,x,y,v);
    else seg[u].r=x_modify(seg[u].r,mid+1,r,x,y,v);
    return u;
}
```

线段树合并

```
ll n, m, k;
vector<int> e[maxn];
int tot, col[maxn];
struct node {
    ll maxv, cnt, l, r;
    node(): maxv(0), l(0), r(0), cnt(0) {}
} seg[maxn * 20];

void upd(int rt) {
    if (seg[seg[rt].l].maxv > seg[seg[rt].r].maxv) {
        seg[rt].maxv = seg[seg[rt].l].maxv;
        seg[rt].cnt = seg[seg[rt].l].cnt;
    } else if (seg[seg[rt].l].maxv < seg[seg[rt].r].maxv) {
        seg[rt].maxv = seg[seg[rt].r].maxv;
```

```
        seg[rt].cnt = seg[seg[rt].r].cnt;
    } else {
        seg[rt].maxv = seg[seg[rt].r].maxv;
        seg[rt].cnt = seg[seg[rt].r].cnt + seg[seg[rt].l].cnt;
    }
}

int modify(int rt, int l, int r, int pos) {
    if (rt == 0) rt = ++tot;
    if (l == r) {
        seg[rt].maxv++;
        seg[rt].cnt = pos;
    } else {
        int mid = (l + r) >> 1;
        if (pos <= mid)
            seg[rt].l = modify(seg[rt].l, l, mid, pos);
        else
            seg[rt].r = modify(seg[rt].r, mid + 1, r, pos);
        upd(rt);
    }
    return rt;
}

int merge(int u, int v, int l, int r) {
    if (!u) return v;
    if (!v) return u;
    if (l == r) {
        seg[u].maxv += seg[v].maxv;
        return u;
    } else {
        int mid = (l + r) >> 1;
        seg[u].l = merge(seg[u].l, seg[v].l, l, mid);
        seg[u].r = merge(seg[u].r, seg[v].r, mid + 1, r);
        upd(u);
        return u;
    }
}

ll query(int rt, int l, int r) {
    return seg[rt].cnt;
}

void split(int &p, int &q, int s, int t, int l, int r) {
    if (t < l || r < s) return;
    if (!p) return;
    if (l <= s && t <= r) {
        q = p;
        p = 0;
        return;
    }
    if (!q) q = New();
    int m = s + t >> 1;
```

```
    if (l <= m) split(ls[p], ls[q], s, m, l, r);
    if (m < r) split(rs[p], rs[q], m + 1, t, l, r);
    push_up(p);
    push_up(q);
}

void solve() {
    cin >> n;
    vector<int> rt(n + 1);
    rep(i, 1, n) {
        cin >> col[i];
        rt[i] = modify(0, 1, n, col[i]);
    }
    rep(i, 2, n) {
        int u, v; cin >> u >> v;
        e[u].pb(v), e[v].pb(u);
    }
    vector<ll> ans(n + 1);
    function<void(int, int)> dfs = [&](int u, int f) {
        for (auto v : e[u]) if (v != f) {
                dfs(v, u);
                rt[u] = merge(rt[u], rt[v], 1, n);
            }
        ans[u] = query(rt[u], 1, n);
    };
    dfs(1, 0);
    rep(i, 1, n) cout << ans[i] << " \n"[i == n];
}
```

BIT-binarySearch

```
struct BIT {
    ll a[maxn];
    int sz;
    BIT(int x): sz(x) {};
    BIT() {};
    void resize(int x) { sz = x; }
    ll query(ll d) {
        ll res = 0, sum = 0;
        for (int i = 18; i >= 0; i--) {
            if (res + bit(i) <= sz && sum + a[res + bit(i)] <= d) {
                sum += a[res + bit(i)];
                res += bit(i);
            }
        }
        return res;
    }
    void modify(int pos, ll d) {
        for (int i = pos; i <= sz; i += (i & -i)) {
            a[i] += d;
        }
```

```
    }
};
```

BIT-2D

```
struct BIT {
    ll a[N][N];
    int X, Y;
    BIT(int x, int y): X(x), Y(y) {};
    BIT() {};
    void resize(int x, int y) { X = x; Y = y; }
    ll query(int x, int y) {
        ll res = 0;
        for (int i = x; i > 0; i -= (i & -i))
            for (int j = y; j > 0; j -= (j & -j))
                res += a[i][j];
        return res;
    }
    void modify(int x, int y, ll d) {
        for (int i = x; i <= X; i += (i & -i))
            for (int j = y; j <= Y; j += (j & -j))
                a[i][j] += d;
    }
};
```

BIT

```
struct BIT {
    ll a[maxn];
    int sz;
    BIT(int x): sz(x) {};
    BIT() {};
    void resize(int x) { sz = x; }
    ll query(int pos) {
        ll res = 0;
        for (int i = pos; i > 0; i -= (i & -i)) {
            res += a[i];
        }
        return res;
    }
    void modify(int pos, ll d) {
        for (int i = pos; i <= sz; i += (i & -i)) {
            a[i] += d;
        }
    }
};
```

CDQ

```
int ans[maxn], lev[maxn];
array<int, 5> v[maxn], tmp[maxn];

struct BIT {
    ll a[maxn];
    int sz;
    BIT(int x): sz(x) {};
    BIT() {};
    void resize(int x) {
        sz = x;
    }
    ll query(int pos) {
        ll res = 0;
        for (int i = pos; i > 0; i -= (i & -i)) {
            res += a[i];
        }
        return res;
    }
    void modify(int pos, ll d) {
        for (int i = pos; i <= sz; i += (i & -i)) {
            a[i] += d;
        }
    }
} c;

void solve(int l, int r) {
    if (l >= r) return;
    int mid = (l + r) / 2;
    solve(l, mid), solve(mid + 1, r);
    int i = l, j = mid + 1;
    int piv = l;
    while (i <= mid || j <= r) {
        if (i <= mid && (j > r || mp(v[i][1], v[i][2]) <= mp(v[j][1], v[j][2]))) {
            c.modify(v[i][2], v[i][3]);
            tmp[piv++] = v[i++];
        } else {
            v[j][4] += c.query(v[j][2]);
            tmp[piv++] = v[j++];
        }
    }
    rep(i, l, mid) c.modify(v[i][2], -v[i][3]);
    rep(i, l, r) v[i] = tmp[i];
}

void solve() {
    cin >> n >> k;
    c.resize(k);
    rep(i, 1, n) {
        int s, c, m;
        cin >> s >> c >> m;
```

```
        v[i] = {s, c, m, 1, 0};
    }
    v[0][0] = -1;
    sort(v + 1, v + n + 1);
    int cnt = 0;
    rep(i, 1, n) {
        if (v[i][0] == v[cnt][0] && v[i][1] == v[cnt][1] && v[i][2] == v[cnt][2]) v[cnt][3]++;
        else v[++cnt] = v[i];
    }
    solve(1, cnt);
    rep(i, 1, cnt) {
        ans[v[i][4] + v[i][3] - 1] += v[i][3];
    }
    rep(i, 0, n - 1) cout << ans[i] << '\n';
}
```

chairman tree

```
struct node {
    node *l, *r;
    ull val;
};

node* build(int l, int r) {
    node* p = new node();
    if (l == r) {
        p->l = p->r = nullptr;
        p->val = 0;
    } else {
        int mid = (l + r) >> 1;
        p->l = build(l, mid);
        p->r = build(mid + 1, r);
        p->val = 0;
    }
    return p;
}

ull query(node *v, int l, int r, int ql, int qr) {
    if (ql == l && qr == r) {
        return v->val;
    } else {
        int mid = (l + r) >> 1;
        if (qr <= mid)
            return query(v->l, l, mid, ql, qr);
        else if (ql > mid)
            return query(v->r, mid + 1, r, ql, qr);
        else
            return query(v->l, l, mid, ql, mid) ^ query(v->r, mid + 1, r, mid + 1, qr);
    }
}
```

```
node* update(node* v, int l, int r, int pos, ull x) {
    if (l == r) {
        node *p = new node();
        p->l = p->r = nullptr;
        p->val = v->val ^ x;
        return p;
    } else {
        int mid = (l + r) >> 1;
        node* p = new node();
        *p = *v;
        if (pos <= mid) p->l = update(v->l, l, mid, pos, x);
        else p->r = update(v->r, mid + 1, r, pos, x);
        p->val = p->l->val ^ p->r->val;
        return p;
    }
}
```

DSU on tree

```
void dfs(int x, int fa) {
    hs[x] = -1, w[x] = 1;
    l[x] = ++tot;
    id[tot] = x;
    for (auto y : g[x]) if (y != fa) {
            dfs(y, x);
            w[x] += w[y];
            if (hs[x] == -1 || w[y] > w[hs[x]])
                hs[x] = y;
        }
    r[x] = tot;
}

void dsu(int x, int fa, int keep) {
    for (auto y : g[x]) {
        if (y != hs[x] && y != fa) {
            dsu(y, x, 0);
        }
    }
    if (hs[x] != -1) dsu(hs[x], x, 1);

    for (auto y : g[x]) {
        if (y != hs[x] && y != fa) {
            for (int i = l[y]; i <= r[y]; i++) {

            }
        }
    }
    // add current node

    ans[x] = cnt;
```

```
    if (!keep) {
        // clear
    }
}
```

hash_table

```
struct Hash_table {
    static const int V = 1000003;
    int fst[V], nxt[V];
    int ctm, ptm[V], T;
    int val[V];
    ll key[V];
    void init() {T = 0, ctm++;}
    void insert(ll k, int v) {
        int s = k % V;
        if (ptm[s] != ctm) ptm[s] = ctm, fst[s] = -1;
        for (int i = fst[s]; i != -1; i = nxt[i]) if (key[i] == k) {
                return;
            }
        nxt[T] = fst[s], fst[s] = T, key[T] = k, val[T] = v;
        T++;
    }
    int query(ll k) {
        int s = k % V;
        if (ptm[s] != ctm) return -1;
        for (int i = fst[s]; i != -1; i = nxt[i]) {
            if (key[i] == k) return val[i];
        }
        return -1;
    }
};
```

HLD

```
struct HLD {
    int n;
    std::vector<int> siz, top, dep, parent, in, out, seq;
    std::vector<std::vector<int>> adj;
    int cur;

    HLD() {}
    HLD(int n) {
        init(n);
    }
    void init(int n) {
        this->n = n;
        siz.resize(n);
        top.resize(n);
        dep.resize(n);
```

```
        parent.resize(n);
        in.resize(n);
        out.resize(n);
        seq.resize(n);
        cur = 0;
        adj.assign(n, {});
    }
    void addEdge(int u, int v) {
        adj[u].push_back(v);
        adj[v].push_back(u);
    }
    void work(int root = 0) {
        top[root] = root;
        dep[root] = 0;
        parent[root] = -1;
        dfs1(root);
        dfs2(root);
    }
    void dfs1(int u) {
        if (parent[u] != -1) {
            adj[u].erase(std::find(adj[u].begin(), adj[u].end(), parent[u]));
        }

        siz[u] = 1;
        for (auto &v : adj[u]) {
            parent[v] = u;
            dep[v] = dep[u] + 1;
            dfs1(v);
            siz[u] += siz[v];
            if (siz[v] > siz[adj[u][0]]) {
                std::swap(v, adj[u][0]);
            }
        }
    }
    void dfs2(int u) {
        in[u] = cur++;
        seq[in[u]] = u;
        for (auto v : adj[u]) {
            top[v] = v == adj[u][0] ? top[u] : v;
            dfs2(v);
        }
        out[u] = cur;
    }
    int lca(int u, int v) {
        while (top[u] != top[v]) {
            if (dep[top[u]] > dep[top[v]]) {
                u = parent[top[u]];
            } else {
                v = parent[top[v]];
            }
        }
        return dep[u] < dep[v] ? u : v;
```

```
    }

    int dist(int u, int v) {
        return dep[u] + dep[v] - 2 * dep[lca(u, v)];
    }

    int jump(int u, int k) {
        if (dep[u] < k) {
            return -1;
        }

        int d = dep[u] - k;

        while (dep[top[u]] > d) {
            u = parent[top[u]];
        }

        return seq[in[u] - dep[u] + d];
    }

    bool isAncester(int u, int v) {
        return in[u] <= in[v] && in[v] < out[u];
    }

    int rootedParent(int u, int v) {
        std::swap(u, v);
        if (u == v) {
            return u;
        }
        if (!isAncester(u, v)) {
            return parent[u];
        }
        auto it = std::upper_bound(adj[u].begin(), adj[u].end(), v, [&](int x, int y) {
            return in[x] < in[y];
        }) - 1;
        return *it;
    }

    int rootedSize(int u, int v) {
        if (u == v) {
            return n;
        }
        if (!isAncester(v, u)) {
            return siz[v];
        }
        return n - siz[rootedParent(u, v)];
    }

    int rootedLca(int a, int b, int c) {
        return lca(a, b) ^ lca(b, c) ^ lca(c, a);
    }
};
```

k维LIS

```
/* k -> (k-1)*log */
struct P {
    int v[K];
    LL f;
    bool d[K];
} o[N << 10];
P* a[K][N << 10];
int k;
void go(int now, int l, int r) {
    if (now == 0) {
        if (l + 1 == r) return;
        int m = (l + r) / 2;
        go(now, l, m);
        FOR (i, l, m) a[now][i]->d[now] = 0;
        FOR (i, m, r) a[now][i]->d[now] = 1;
        copy(a[now] + l, a[now] + r, a[now + 1] + l);
        sort(a[now + 1] + l, a[now + 1] + r, [now](const P * a, const P * b) {
            if (a->v[now] != b->v[now]) return a->v[now] < b->v[now];
            return a->d[now] > b->d[now];
        });
        go(now + 1, l, r);
        go(now, m, r);
    } else {
        if (l + 1 == r) return;
        int m = (l + r) / 2;
        go(now, l, m); go(now, m, r);
        FOR (i, l, m) a[now][i]->d[now] = 0;
        FOR (i, m, r) a[now][i]->d[now] = 1;
        merge(a[now] + l, a[now] + m, a[now] + m, a[now] + r, a[now + 1] + l, [now](const P *
a, const P * b) {
            if (a->v[now] != b->v[now]) return a->v[now] < b->v[now];
            return a->d[now] > b->d[now];
        });
        copy(a[now + 1] + l, a[now + 1] + r, a[now] + l);
        if (now < k - 2) {
            go(now + 1, l, r);
        } else {
            LL sum = 0;
            FOR (i, l, r) {
                dbg(a[now][i]->v[0], a[now][i]->v[1], a[now][i]->f,
                    a[now][i]->d[0], a[now][i]->d[1]);
                int cnt = 0;
                FOR (j, 0, now + 1) cnt += a[now][i]->d[j];
                if (cnt == 0) {
                    sum += a[now][i]->f;
                } else if (cnt == now + 1) {
                    a[now][i]->f = (a[now][i]->f + sum) % MOD;
                }
            }
        }
```

```
    }
}
```

kdtree

```
namespace kd {
const int K = 2, M = 1000005;
const ll inf = 1E16;
extern struct P* null;
struct P {
    ll d[K], l[K], r[K], val;
    ll Max[K], Min[K], sum;
    P *ls, *rs, *fa;
    P* up() {
        rep(i, 0, K - 1) {
            Max[i] = max({d[i], ls->Max[i], rs->Max[i]});
            Min[i] = min({d[i], ls->Min[i], rs->Min[i]});
        }
        sum = val + ls->sum + rs->sum;
        rep(i, 0, K - 1) {
            l[i] = min(d[i], min(ls->l[i], rs->l[i]));
            r[i] = max(d[i], max(ls->r[i], rs->r[i]));
        }
        return ls->fa = rs->fa = this;
    }
} pool[M], *null = new P, *pit = pool;
/*void upd(P* o, int val) {
    o->val = val;
    for (; o != null; o = o->fa)
        o->Max = max(o->Max, val);
}*/
static P *tmp[M], **pt;
void init() {
    null->ls = null->rs = null;
    rep(i, 0, K - 1) null->l[i] = inf, null->r[i] = -inf;
    null->Max[0] = null->Max[1] = -inf;
    null->Min[0] = null->Min[1] = inf;
    null->val = 0;
    null->sum = 0;
}
P* build(P** l, P** r, int d = 0) { // [l, r)
    if (d == K) d = 0;
    if (l >= r) return null;
    P** m = l + (r - l) / 2; assert(l <= m && m < r);
    nth_element(l, m, r, [&](const P * a, const P * b) {
        return a->d[d] < b->d[d];
    });
    P* o = *m;
    o->ls = build(l, m, d + 1); o->rs = build(m + 1, r, d + 1);
    return o->up();
}
```

```
P* Build() {
    pt = tmp; for (auto it = pool; it < pit; it++) *pt++ = it;
    P* ret = build(tmp, pt); ret->fa = null;
    return ret;
}
inline bool inside(int p[], int q[], int l[], int r[]) {
    rep(i, 0, K - 1) if (r[i] < q[i] || p[i] < l[i]) return false;
    return true;
}
/*int query(P* o, int l[], int r[]) {
    if (o == null) return 0;
    rep(i, 0, K - 1) if (o->r[i] < l[i] || r[i] < o->l[i]) return 0;
    if (inside(o->l, o->r, l, r)) return o->Max;
    int ret = 0;
    if (o->val > ret && inside(o->d, o->d, l, r)) ret = max(ret, o->val);
    if (o->ls->Max > ret) ret = max(ret, query(o->ls, l, r));
    if (o->rs->Max > ret) ret = max(ret, query(o->rs, l, r));
    return ret;
}
ll eval(P* o, int d[]) { ... }
ll dist(int d1[], int d2[]) { ... }
ll S;
ll query(P* o, int d[]) {
    if (o == null) return 0;
    S = max(S, dist(o->d, d));
    ll mdl = eval(o->ls, d), mdr = eval(o->rs, d);
    if (mdl < mdr) {
        if (S > mdl) S = max(S, query(o->ls, d));
        if (S > mdr) S = max(S, query(o->rs, d));
    } else {
        if (S > mdr) S = max(S, query(o->rs, d));
        if (S > mdl) S = max(S, query(o->ls, d));
    }
    return S;
}*/
bool check(ll x, ll y, ll a, ll b, ll c) { return a * x + b * y < c; }

ll query(P* o, ll a, ll b, ll c) {
    if (o == null) return 0;
    int chk = 0;
    chk += check(o->Min[0], o->Min[1], a, b, c);
    chk += check(o->Max[0], o->Min[1], a, b, c);
    chk += check(o->Min[0], o->Max[1], a, b, c);
    chk += check(o->Max[0], o->Max[1], a, b, c);
    if (chk == 4) return o->sum;
    if (chk == 0) return 0;
    ll ret = 0;
    if (check(o->d[0], o->d[1], a, b, c)) ret += o->val;
    ret += query(o->ls, a, b, c);
    ret += query(o->rs, a, b, c);
    return ret;
}
```

```
}  // namespace kd

void solve() {
    cin >> n >> m;
    kd::init();
    rep(i, 1, n) {
        int x, y, w;
        cin >> x >> y >> w;
        kd::pit->d[0] = x, kd::pit->d[1] = y, kd::pit->val = w;
        kd::pit++;
    }
    auto rt = kd::Build();
    rep(i, 1, m) {
        int a, b, c;
        cin >> a >> b >> c;
        cout << kd::query(rt, a, b, c) << '\n';
    }
}
```

LCT

```
namespace linkCutTree {

struct node {
    node *child[2], *parent, *max;
    int sum, val, sz, weight, id, rev;
    node(int val, int weight, int id) : child {nullptr, nullptr}, parent(nullptr), max(this),
sum(val), val(val), sz(weight), weight(weight), id(id), rev(false) {}
};

bool isRoot(node *p) {return p->parent == nullptr || p->parent->child[0] != p && p->parent-
>child[1] != p;}

int side(node *p) {return p->parent->child[1] == p;}

int sum(node *p) {return p == nullptr ? 0 : p->sum;}

int sz(node *p) {return p == nullptr ? 0 : p->sz;}

node *max(node *p) {return p == nullptr ? nullptr : p->max;}

node *max(node *p, node *q) {
    if (p == nullptr)
        return q;
    if (q == nullptr)
        return p;
    return p->weight > q->weight ? p : q;
}

void reverse(node *p) {
    if (p == nullptr)
```

```
        return;
    swap(p->child[0], p->child[1]);
    p->rev ^= 1;
}

void push(node *p) {
    if (p->rev == 0)
        return;
    p->rev = 0;
    reverse(p->child[0]);
    reverse(p->child[1]);
}

void pull(node *p) {
    p->sum = sum(p->child[0]) + sum(p->child[1]) + p->val;
    p->max = max(max(max(p->child[0]), max(p->child[1])), p);
    p->sz = p->weight + sz(p->child[0]) + sz(p->child[1]);
}

void connect(node *p, node *q, int side) {
    q->child[side] = p;
    if (p != nullptr)
        p->parent = q;
}

void rotate(node *p) {
    auto q = p->parent;
    int dir = side(p) ^ 1;
    connect(p->child[dir], q, dir ^ 1);
    if (!isRoot(q))
        connect(p, q->parent, side(q));
    else
        p->parent = q->parent;
    connect(q, p, dir);
    pull(q);
}

void splay(node *p) {
    vector<node *> stk;
    for (auto i = p; !isRoot(i); i = i->parent)
        stk.push_back(i->parent);
    while (!stk.empty()) {
        push(stk.back());
        stk.pop_back();
    }
    push(p);
    while (!isRoot(p)) {
        auto q = p->parent;
        if (!isRoot(q))
            rotate(side(p) == side(q) ? q : p);
        rotate(p);
    }
```

```
    pull(p);
}

node *access(node *p) {
    node *j = nullptr;
    for (node *i = p; i != nullptr; j = i, i = i->parent) {
        splay(i);
        i->val -= sum(j);
        i->val += sum(i->child[1]);
        i->child[1] = j;
        pull(i);
    }
    splay(p);
    return j;
}

void makeRoot(node *p) {
    access(p);
    reverse(p);
}

void link(node *p, node *q) {
    makeRoot(p);
    access(q);
    p->parent = q;
    q->val += sum(p);
}

void cut(node *p, node *q) {
    makeRoot(p);
    access(q);
    p->parent = q->child[0] = nullptr;
}

node *pathMax(node *p, node *q) {
    makeRoot(p);
    access(q);
    return max(q);
}

int pathSum(node *p, node *q) {
    makeRoot(p);
    access(q);
    return sz(q);
}

int size(node *p) {
    makeRoot(p);
    return sum(p);
}

bool connected(node *p, node *q) {
```

```
    access(p);
    access(q);
    return p->parent != nullptr;
}

void fix(node *p, ll v) {
    access(p);
    // modify ...
    pull(p);
}

node *lca(node *z,node *x,node *y) {
    makeRoot(z);
    access(x);
    return access(y);
}

}  // namespace linkCutTree
using namespace linkCutTree;
```

Mo

```
#include <bits/stdc++.h>
using namespace std;
#define rep(i,a,n) for (int i=a;i<n;i++)
#define per(i,a,n) for (int i=n-1;i>=a;i--)
#define pb push_back
#define mp make_pair
#define all(x) (x).begin(),(x).end()
#define fi first
#define se second
#define SZ(x) ((int)(x).size())
typedef vector<int> VI;
typedef long long ll;
typedef pair<int,int> PII;
typedef double db;
mt19937 mrand(random_device{}());
const ll mod=1000000007;
int rnd(int x) { return mrand() % x;}
ll powmod(ll a,ll b) {ll res=1;a%=mod; assert(b>=0); for(;b;b>>=1)
{if(b&1)res=res*a%mod;a=a*a%mod;}return res;}
ll gcd(ll a,ll b) { return b?gcd(b,a%b):a;}
// head


const int N=1010000;
int a[N];
namespace Mo {
  int Q,l[N],r[N],f[N],l0,r0,ans[N],n;
  VI ne[N];
  struct point {
```

```
  int x, y, o;
  point(int a, int b, int c): x(a), y(b), o(c) {}
};
inline bool operator<(const point &a, const point &b) {
  if (a.x != b.x) return a.x > b.x;
  else return a.y < b.y;
}
vector<point> p;
struct edge {
  int s, t, d;
  edge(const point &a, const point &b): s(a.o), t(b.o),
    d(abs(a.x - b.x) + abs(a.y - b.y)) {}
};
inline bool operator<(const edge &a, const edge &b) {return a.d < b.d;}
vector<edge> e;
int g[N],z[N];
int cc,cnt[101000];
void addedge() {
  sort(all(p));
    memset(g,0,sizeof(g));
    z[0]=N;
  rep(i,0,SZ(p)) z[i+1]=p[i].x-p[i].y;
  rep(i,0,SZ(p)) {
        int k = 0, t = p[i].x + p[i].y;
        for (int j = t; j; j -= j & -j)
            if (z[g[j]] < z[k]) k = g[j];
        if (k) e.pb(edge(p[i], p[k - 1]));
        k = z[i + 1];
        for (int j = t; j <N; j += j & -j)
            if (k < z[g[j]]) g[j] = i + 1;
    }
}
void updata(int i, bool j,bool k=0) {
  // j=1 insert  j=0 delete
  // k=0 left k=1 right
  if (j==1) {
    cnt[a[i]]++;
    if (cnt[a[i]]%2==0) cc++;
  } else {
    if (cnt[a[i]]%2==0) cc--;
    cnt[a[i]]--;
  }
}
void init(int l,int r) {
  for (int i=l;i<=r;i++) {
    cnt[a[i]]++;
    if (cnt[a[i]]%2==0) cc++;
  }
}
inline int query() {
  return cc;
}
```

```
  int find(int x) { if (f[x] != x) f[x] = find(f[x]); return f[x];}
  void dfs(int i,int p) {
    int l1 = l[i], r1 = r[i];
    per(j,l1,l0) updata(j,1,0);
    rep(j,r0+1,r1+1) updata(j,1,1);
    rep(j,l0,l1) updata(j,0,0);
    per(j,r1+1,r0+1) updata(j,0,1);
    ans[i]=query();l0=l1;r0=r1;
    rep(j,0,SZ(ne[i])) if (ne[i][j]!=p) dfs(ne[i][j],i);
  }
  void solve() {
    p.clear();e.clear();
    rep(i,1,Q+1) ans[i]=0;
    rep(i,1,Q+1) p.pb(point(l[i],r[i],i));
    addedge();
    rep(i,0,SZ(p)) p[i].y =n-p[i].y+1;
    addedge();
    rep(i,0,SZ(p)) {
      int j =n-p[i].x+1;
      p[i].x = p[i].y; p[i].y = j;
    }
    addedge();
    rep(i,0,SZ(p)) p[i].x=n-p[i].x+1;
    addedge();
    sort(all(e));
    rep(i,1,Q+1) ne[i].clear(),f[i]=i;
    rep(i,0,SZ(e)) {
      int j=e[i].s,k=e[i].t;
      if (find(j)!=find(k)) f[f[j]]=f[k],ne[j].pb(k),ne[k].pb(j);
    }
    l0=l[1];r0=r[1];
    init(l0,r0);
    dfs(1,0);
  }
}

int main() {
  scanf("%d",&Mo::n);
  for (int i=1;i<=Mo::n;i++) scanf("%d",a+i);
  scanf("%d",&Mo::Q);
  rep(i,1,Mo::Q+1) scanf("%d%d",&Mo::l[i],&Mo::r[i]);
  Mo::solve();
  rep(i,1,Mo::Q+1) printf("%d\n",Mo::ans[i]);
}
```

moTree

```
void add(int ind, int end) { ... } // add a [ ind ] (end = 0 or 1)
void del(int ind, int end) { ... } // remove a [ ind ]
int calc() { ... } // compute current answer
vi mo(vector<pii> Q) {
```

```
  int L = 0, R = 0, blk = 350; // ~N/sqrt (Q)
  vi s(sz(Q)), res = s;
#define K(x) pii(x.first/blk, x.second ^ -(x.first/blk & 1))
  iota(all(s), 0);
  sort(all(s), [&](int s, int t) { return K(Q[s]) < K(Q[t]); });
  for (int qi : s) {
    pii q = Q[qi];
    while (L > q.first) add(--L, 0);
    while (R < q.second) add(R++, 1);
    while (L < q.first) del(L++, 0);
    while (R > q.second) del(--R, 1);
    res[qi] = calc();
  }
  return res;
}
vi moTree(vector<array<int, 2>> Q, vector<vi>& ed, int root = 0) {
  int N = sz(ed), pos[2] = {}, blk = 350; // ~N/sqrt (Q)
  vi s(sz(Q)), res = s, I(N), L(N), R(N), in(N), par(N);
  add(0, 0), in[0] = 1;
  auto dfs = [&](int x, int p, int dep, auto & f) -> void {
    par[x] = p;
    L[x] = N;
    if (dep) I[x] = N++;
    for (int y : ed[x]) if (y != p) f(y, x, !dep, f);
    if (!dep) I[x] = N++;
    R[x] = N;
  };
  dfs(root, -1, 0, dfs);
#define K(x) pii(I[x[0]] / blk, I[x[1]] ^ -(I[x[0]] / blk & 1))
  iota(all(s), 0);
  sort(all(s), [&](int s, int t) { return K(Q[s]) < K(Q[t]); });
  for (int qi : s) rep(end, 0, 2) {
    int &a = pos[end], b = Q[qi][end], i = 0;
#define step(c) { if (in[c]) { del(a, end); in[a] = 0; } \
else { add(c, end); in[c] = 1; } a = c; }
    while (!(L[b] <= L[a] && R[a] <= R[b]))
      I[i++] = b, b = par[b];
    while (a != b) step(par[a]);
    while (i--) step(I[i]);
    if (end) res[qi] = calc();
  }
  return res;
}
```

MyTreap

```
/**
 *    author:  tourist
 *    created: 07.10.2022 20:32:03
**/
#include <bits/stdc++.h>
```

```cpp
using namespace std;

#define bit(x) (1ll<<(x))

#ifdef LOCAL
#include "algo/debug.h"
#else
#define debug(...) 42
#endif

mt19937_64 rng(chrono::steady_clock::now().time_since_epoch().count());

class node {
public:
  int id;
  node* l;
  node* r;
  node* p;
  bool rev;
  int sz;
  // declare extra variables:
  long long P;
  long long add;
  long long x;

  node(int _id, long long _x) {
    id = _id;
    l = r = p = nullptr;
    rev = false;
    sz = 1;
    // init extra variables:
    P = rng();
    add = 0;
    x = _x;
  }

  // push everything else:
  void push_stuff() {
    if (add != 0) {
      if (l != nullptr) {
        l->unsafe_apply(add);
      }
      if (r != nullptr) {
        r->unsafe_apply(add);
      }
      add = 0;
    }
  }

  void unsafe_reverse() {
    push_stuff();
```

```
    rev ^= 1;
    swap(l, r);
    pull();
  }

  // apply changes:
  void unsafe_apply(long long delta) {
    add += delta;
    x += delta;
  }

  void push() {
    if (rev) {
      if (l != nullptr) {
        l->unsafe_reverse();
      }
      if (r != nullptr) {
        r->unsafe_reverse();
      }
      rev = 0;
    }
    push_stuff();
  }

  void pull() {
    sz = 1;
    if (l != nullptr) {
      l->p = this;
      sz += l->sz;
    }
    if (r != nullptr) {
      r->p = this;
      sz += r->sz;
    }
  }
};

void debug_node(node* v, string pref = "") {
#ifdef LOCAL
  if (v != nullptr) {
    debug_node(v->r, pref + " ");
    cerr << pref << "-" << " " << v->id << '\n';
    debug_node(v->l, pref + " ");
  } else {
    cerr << pref << "-" << " " << "nullptr" << '\n';
  }
#endif
}

namespace treap {

pair<node*, int> find(node* v, const function<int(node*)> &go_to) {
```

```
  // go_to returns: 0 -- found; -1 -- go left; 1 -- go right
  // find returns the last vertex on the descent and its go_to
  if (v == nullptr) {
    return {nullptr, 0};
  }
  int dir;
  while (true) {
    v->push();
    dir = go_to(v);
    if (dir == 0) {
      break;
    }
    node* u = (dir == -1 ? v->l : v->r);
    if (u == nullptr) {
      break;
    }
    v = u;
  }
  return {v, dir};
}

node* get_leftmost(node* v) {
  return find(v, [&](node*) { return -1; }).first;
}

node* get_rightmost(node* v) {
  return find(v, [&](node*) { return 1; }).first;
}

node* get_kth(node* v, int k) { // 0-indexed
  pair<node*, int> p = find(v, [&](node * u) {
    if (u->l != nullptr) {
      if (u->l->sz > k) {
        return -1;
      }
      k -= u->l->sz;
    }
    if (k == 0) {
      return 0;
    }
    k--;
    return 1;
  });
  return (p.second == 0 ? p.first : nullptr);
}

int get_pos(node* v) { // 0-indexed
  int k = (v->l != nullptr ? v->l->sz : 0);
  while (v->p != nullptr) {
    if (v == v->p->r) {
      k++;
      if (v->p->l != nullptr) {
```

```
        k += v->p->l->sz;
      }
    }
    v = v->p;
  }
  return k;
}

node* get_root(node* v) {
  while (v->p != nullptr) {
    v = v->p;
  }
  return v;
}

pair<node*, node*> split(node* v, const function<bool(node*)> &is_right) {
  if (v == nullptr) {
    return {nullptr, nullptr};
  }
  v->push();
  if (is_right(v)) {
    pair<node*, node*> p = split(v->l, is_right);
    if (p.first != nullptr) {
      p.first->p = nullptr;
    }
    v->l = p.second;
    v->pull();
    return {p.first, v};
  } else {
    pair<node*, node*> p = split(v->r, is_right);
    v->r = p.first;
    if (p.second != nullptr) {
      p.second->p = nullptr;
    }
    v->pull();
    return {v, p.second};
  }
}

pair<node*, node*> split_cnt(node* v, int k) {
  if (v == nullptr) {
    return {nullptr, nullptr};
  }
  v->push();
  int left_and_me = (v->l != nullptr ? v->l->sz : 0) + 1;
  if (k < left_and_me) {
    pair<node*, node*> p = split_cnt(v->l, k);
    if (p.first != nullptr) {
      p.first->p = nullptr;
    }
    v->l = p.second;
    v->pull();
```

```
    return {p.first, v};
  } else {
    pair<node*, node*> p = split_cnt(v->r, k - left_and_me);
    v->r = p.first;
    if (p.second != nullptr) {
      p.second->p = nullptr;
    }
    v->pull();
    return {v, p.second};
  }
}

node* merge(node* v, node* u) {
  if (v == nullptr) {
    return u;
  }
  if (u == nullptr) {
    return v;
  }
  if (v->P > u->P) {
//    if (rng() % (v->sz + u->sz) < (unsigned int) v->sz) {
    v->push();
    v->r = merge(v->r, u);
    v->pull();
    return v;
  } else {
    u->push();
    u->l = merge(v, u->l);
    u->pull();
    return u;
  }
}

int count_left(node* v, const function<bool(node*)> &is_right) {
  if (v == nullptr) {
    return 0;
  }
  v->push();
  if (is_right(v)) {
    return count_left(v->l, is_right);
  }
  return (v->l != nullptr ? v->l->sz : 0) + 1 + count_left(v->r, is_right);
}

int count_less(node* v, long long val) {
  int res = 0;
  while (v != nullptr) {
    v->push();
    if (v->x >= val) {
      v = v->l;
    } else {
      res += (v->l != nullptr ? v->l->sz : 0) + 1;
```

```
      v = v->r;
    }
  }
  return res;
}

node* add(node* r, node* v, const function<bool(node*)> &go_left) {
  pair<node*, node*> p = split(r, go_left);
  return merge(p.first, merge(v, p.second));
}

node* remove(node* v) { // returns the new root
  v->push();
  node* x = v->l;
  node* y = v->r;
  node* p = v->p;
  v->l = v->r = v->p = nullptr;
  v->push();
  v->pull(); // now v might be reusable...
  node* z = merge(x, y);
  if (p == nullptr) {
    if (z != nullptr) {
      z->p = nullptr;
    }
    return z;
  }
  if (p->l == v) {
    p->l = z;
  }
  if (p->r == v) {
    p->r = z;
  }
  while (true) {
    p->push();
    p->pull();
    if (p->p == nullptr) {
      break;
    }
    p = p->p;
  }
  return p;
}

node* next(node* v) {
  if (v->r == nullptr) {
    while (v->p != nullptr && v->p->r == v) {
      v = v->p;
    }
    return v->p;
  }
  v->push();
  v = v->r;
```

```
  while (v->l != nullptr) {
    v->push();
    v = v->l;
  }
  return v;
}

node* prev(node* v) {
  if (v->l == nullptr) {
    while (v->p != nullptr && v->p->l == v) {
      v = v->p;
    }
    return v->p;
  }
  v->push();
  v = v->l;
  while (v->r != nullptr) {
    v->push();
    v = v->r;
  }
  return v;
}

int get_size(node* v) {
  return (v != nullptr ? v->sz : 0);
}

template<typename... T>
void Apply(node* v, T... args) {
  v->unsafe_apply(args...);
}

void reverse(node* v) {
  v->unsafe_reverse();
}

// extra of mine
long long lower(node* u, long long x) {
  if (u == nullptr)
    return numeric_limits<long long>::min();
  else if (x <= u->x)
    return lower(u->l, x);
  else
    return max(u->x, lower(u->r, x));
}

long long upper(node* u, long long x) {
  if (u == nullptr)
    return numeric_limits<long long>::max();
  else if (u->x <= x)
    return upper(u->r, x);
  else
```

```
    return min(u->x, upper(u->l, x));
}

}  // namespace treap

using namespace treap;

int n;

int main() {
  ios::sync_with_stdio(false);
  cin.tie(0);
  node* root = nullptr;
  cin >> n;
  for (int i = 1; i <= n; i++) {
    int op;
    long long x;
    cin >> op >> x;
    switch (op) {
      case 1: {
        root = add(root, new node(x, x), [&](node * u) {
          return x < u->x;
        });
        break;
      }
      case 2: {
        auto [pt, w] = find(root, [&](node * u) {
          if (x < u->x) return -1;
          else if (x == u->x) return 0;
          else return 1;
        });
        assert(w == 0);
        root = remove(pt);
        break;
      }
      case 3: {
        cout << count_less(root, x) + 1 << '\n';
        break;
      }
      case 4: {
        cout << get_kth(root, x - 1)->x << '\n';
        break;
      }
      case 5: {
        cout << lower(root, x) << '\n';
        break;
      }
      case 6: {
        cout << upper(root, x) << '\n';
        break;
      }
    }
```

```
  }
}
```

segment tree-binarySearch

```
struct node {
    ll val;
} seg[maxn << 2];

void update(int id) {
    seg[id].val = max(seg[id * 2].val, seg[id * 2 + 1].val);
}

void build(int l, int r, int id) {
    if (l == r) {
        seg[id].val = a[l];
    } else {
        int mid = (l + r) >> 1;
        build(l, mid, id * 2);
        build(mid + 1, r, id * 2 + 1);
        update(id);
    }
}

void modify(int l, int r, int id, int pos, ll d) {
    if (l == r) {
        seg[id].val = d;
    } else {
        int mid = (l + r) >> 1;
        if (pos <= mid)
            modify(l, mid, id * 2, pos, d);
        else
            modify(mid + 1, r, id * 2 + 1, pos, d);
        update(id);
    }
}


ll search(int l, int r, int id, int ql, int qr, int d) {
    if (ql == l && qr == r) {
        int mid = (l + r) / 2;
        // if (l!=r) pushdown(id); ...
        if (seg[id].val < d) return -1;
        else {
            if (l == r) return l;
            else if (seg[id * 2].val >= d)
                return search(l, mid, id * 2, ql, mid, d);
            else
                return search(mid + 1, r, id * 2 + 1, mid + 1, qr, d);
        }
    } else {
```

```
        int mid = (l + r) >> 1;
        // pushdown(id); ...
        if (qr <= mid)
            return search(l, mid, id * 2, ql, qr, d);
        else if (ql > mid)
            return search(mid + 1, r, id * 2 + 1, ql, qr, d);
        else {
            int tmp = search(l, mid, id * 2, ql, mid, d);
            if (tmp != -1)
                return tmp;
            else
                return search(mid + 1, r, id * 2 + 1, mid + 1, qr, d);
        }
    }
}
```

segment tree-tag

```
struct info{
    ll sum;
    int sz;
    friend info operator +(const info &a,const info &b){
        return {(a.sum+b.sum)%mod,a.sz+b.sz};
    }
};

struct tag{
    ll add,mul;
    friend tag operator +(const tag &a,const tag &b){
        tag res= {(a.add*b.mul+b.add)%mod,a.mul*b.mul%mod};
        return res;
    }

};
info operator +(const info &a,const tag &b){
    return {(a.sum*b.mul+a.sz*b.add)%mod,a.sz};
}

struct node{
    info val;
    tag t;
}seg[maxn<<2];

void update(int id){
    seg[id].val=seg[id*2].val+seg[id*2+1].val;
}
void settag(int id,tag t){
    seg[id].val=seg[id].val+t;
    seg[id].t=seg[id].t+t;
}
void pushdown(int id){
```

```
    if(seg[id].t.mul==1 and seg[id].t.add==0) return;
    settag(id*2,seg[id].t);
    settag(id*2+1,seg[id].t);
    seg[id].t.mul=1;
    seg[id].t.add=0;
}
void build(int l,int r,int id){
    seg[id].t={0,1};
    if(l==r){
        seg[id].val={a[l],1};
    }else{
        int mid=(l+r)>>1;
        build(l,mid,id*2);
        build(mid+1,r,id*2+1);
        update(id);
    }
}
void change(int l,int r,int id,int ql,int qr,tag t){
    if(l==ql&&r==qr){
        settag(id,t);
    }else{
        int mid=(l+r)>>1;
        pushdown(id);
        if(qr<=mid){
            change(l,mid,id*2,ql,qr,t);
        }else if(ql>mid){
            change(mid+1,r,id*2+1,ql,qr,t);
        }else{
            change(l,mid,id*2,ql,mid,t);
            change(mid+1,r,id*2+1,mid+1,qr,t);
        }
        update(id);
    }
}
info query(int l,int r,int id,int ql,int qr){
    if(l==ql&&r==qr){
        return seg[id].val;
    }else{
        int mid=(l+r)>>1;
        pushdown(id);
        if(qr<=mid)
            return query(l,mid,id*2,ql,qr);
        else if(ql>mid)
            return query(mid+1,r,id*2+1,ql,qr);
        else
            return query(l,mid,id*2,ql,mid)+
        query(mid+1,r,id*2+1,mid+1,qr);
    }
}
```

segtreefast

```
/**
 * Author: Lucian Bicsi
 * Description: Very fast and quick segment tree.
 * Only useful for easy invariants. 0-indexed.
 * Range queries are half-open.
 */
#pragma once

struct SegmTree {
  vector<int> T; int n;
  SegmTree(int n) : T(2 * n, (int)2e9), n(n) {}

  void Update(int pos, int val) {
    for (T[pos += n] = val; pos > 1; pos /= 2)
      T[pos / 2] = min(T[pos], T[pos ^ 1]);
  }

  int Query(int b, int e) {
    int res = (int)2e9;
    for (b += n, e += n; b < e; b /= 2, e /= 2) {
      if (b % 2) res = min(res, T[b++]);
      if (e % 2) res = min(res, T[--e]);
    }
    return res;
  }
};
```

SparseTable

```
template<class t1>
struct ST {
    int n;
    static const int M = 21;
    t1 p[M][maxn];
    ST() {}
    void build(t1 a[], int sz) {
        n = sz;
        rep(i, 1, n) p[0][i] = a[i];
        rep(i, 1, M - 1)
        rep(j, 1, n) if (j + bit(i) - 1 <= n) {
            p[i][j] = max(p[i - 1][j], p[i - 1][j + bit(i - 1)]);
        }
    }

    t1 query(int l, int r) {
        int len = r - l + 1;
        int k = log2(len);
        return max(p[k][l], p[k][r - bit(k) + 1]);
```

```
    }
};
```

SparseTable2D

```
lg[1] = 0;
rep(i, 2, maxn - 1) {
    lg[i] = lg[i / 2] + 1;
}
int k = log2(r - l + 1);
int k = __lg(r - l + 1);
int k = lg[r - l + 1];
int k = 32 - __builtin_clz(r - l + 1) - 1;

vector<vector<int>> sparse[12];

int query(int x, int y, int d) {
    int k = log2(d);
    int s = d - bit(k);
    return min({sparse[k][x][y], sparse[k][x + s][y], sparse[k][x][y + s]
                , sparse[k][x + s][y + s]
               });
}

void solve() {
    cin >> n >> m;
    rep(i, 0, 11) sparse[i] = vector<vector<int>>(n + 1, vector<int>(m + 1, inf));
    rep(i, 1, n) rep(j, 1, m) cin >> sparse[0][i][j];
    for (int k = 1; k < 12; k++)
        for (int i = 1; i <= n; i++)
            for (int j = 1; j <= m; j++) {
                int d = bit(k - 1);
                if (i + d > n || j + d > m) continue;
                sparse[k][i][j] = min({sparse[k - 1][i][j], sparse[k - 1][i + d][j],
                                       sparse[k - 1][i][j + d], sparse[k - 1][i + d][j + d]
                                      });
            }

}
```

splay

```
/**
 *    author:  tourist
 *    created: 30.07.2021 17:54:21
**/
#include <bits/stdc++.h>

using namespace std;
```

```
int n;

class node {
 public:
  int id;
  node* l;
  node* r;
  node* p;
  bool rev;
  int sz;
  // declare extra variables:
  int fake_sz;
  int w;
  int sum_w;
  int b;
  int add_b;
  int sum_b;

  node(int _id, int _w) {
    id = _id;
    l = r = p = nullptr;
    rev = false;
    sz = 1;
    // init extra variables:
    fake_sz = (id >= n);
    w = _w;
    sum_w = w;
    b = 0;
    add_b = 0;
    sum_b = 0;
  }

  // push everything else:
  void push_stuff() {
    if (add_b != 0) {
      if (l != nullptr) {
        l->unsafe_apply(add_b);
      }
      if (r != nullptr) {
        r->unsafe_apply(add_b);
      }
      add_b = 0;
    }
  }

  void unsafe_reverse() {
    push_stuff();
    rev ^= 1;
    swap(l, r);
    pull();
  }
```

```
  // apply changes:
  void unsafe_apply(int bb) {
    if (id >= n) {
      b += bb;
    }
    add_b += bb;
    sum_b += bb * fake_sz;
  }

  void push() {
    if (rev) {
      if (l != nullptr) {
        l->unsafe_reverse();
      }
      if (r != nullptr) {
        r->unsafe_reverse();
      }
      rev = 0;
    }
    push_stuff();
  }

  void pull() {
    sz = 1;
    // now init from self:
    sum_w = w;
    sum_b = b;
    fake_sz = (id >= n);
    if (l != nullptr) {
      l->p = this;
      sz += l->sz;
      // now pull from l:
      sum_w += l->sum_w;
      sum_b += l->sum_b;
      fake_sz += l->fake_sz;
    }
    if (r != nullptr) {
      r->p = this;
      sz += r->sz;
      // now pull from r:
      sum_w += r->sum_w;
      sum_b += r->sum_b;
      fake_sz += r->fake_sz;
    }
  }
};

void debug_node(node* v, string pref = "") {
  #ifdef LOCAL
    if (v != nullptr) {
      debug_node(v->r, pref + " ");
      cerr << pref << "-" << " " << v->id << '\n';
```

```
      debug_node(v->l, pref + " ");
    } else {
      cerr << pref << "-" << " " << "nullptr" << '\n';
    }
  #endif
}

namespace splay_tree {

bool is_bst_root(node* v) {
  if (v == nullptr) {
    return false;
  }
  return (v->p == nullptr || (v->p->l != v && v->p->r != v));
}

void rotate(node* v) {
  node* u = v->p;
  assert(u != nullptr);
  u->push();
  v->push();
  v->p = u->p;
  if (v->p != nullptr) {
    if (v->p->l == u) {
      v->p->l = v;
    }
    if (v->p->r == u) {
      v->p->r = v;
    }
  }
  if (v == u->l) {
    u->l = v->r;
    v->r = u;
  } else {
    u->r = v->l;
    v->l = u;
  }
  u->pull();
  v->pull();
}

void splay(node* v) {
  if (v == nullptr) {
    return;
  }
  while (!is_bst_root(v)) {
    node* u = v->p;
    if (!is_bst_root(u)) {
      if ((u->l == v) ^ (u->p->l == u)) {
        rotate(v);
      } else {
        rotate(u);
```

```
      }
    }
    rotate(v);
  }
}

pair<node*, int> find(node* v, const function<int(node*)> &go_to) {
  // go_to returns: 0 -- found; -1 -- go left; 1 -- go right
  // find returns the last vertex on the descent and its go_to
  if (v == nullptr) {
    return {nullptr, 0};
  }
  splay(v);
  int dir;
  while (true) {
    v->push();
    dir = go_to(v);
    if (dir == 0) {
      break;
    }
    node* u = (dir == -1 ? v->l : v->r);
    if (u == nullptr) {
      break;
    }
    v = u;
  }
  splay(v);
  return {v, dir};
}

node* get_leftmost(node* v) {
  return find(v, [&](node*) { return -1; }).first;
}

node* get_rightmost(node* v) {
  return find(v, [&](node*) { return 1; }).first;
}

node* get_kth(node* v, int k) { // 0-indexed
  pair<node*, int> p = find(v, [&](node* u) {
    if (u->l != nullptr) {
      if (u->l->sz > k) {
        return -1;
      }
      k -= u->l->sz;
    }
    if (k == 0) {
      return 0;
    }
    k--;
    return 1;
  });
```

```
  return (p.second == 0 ? p.first : nullptr);
}

int get_position(node* v) { // 0-indexed
  splay(v);
  return (v->l != nullptr ? v->l->sz : 0);
}

node* get_bst_root(node* v) {
  splay(v);
  return v;
}

pair<node*, node*> split(node* v, const function<bool(node*)> &is_right) {
  if (v == nullptr) {
    return {nullptr, nullptr};
  }
  pair<node*, int> p = find(v, [&](node* u) { return is_right(u) ? -1 : 1; });
  v = p.first;
  v->push();
  if (p.second == -1) {
    node* u = v->l;
    if (u == nullptr) {
      return {nullptr, v};
    }
    v->l = nullptr;
    u->p = v->p;
    u = get_rightmost(u);
    v->p = u;
    v->pull();
    return {u, v};
  } else {
    node* u = v->r;
    if (u == nullptr) {
      return {v, nullptr};
    }
    v->r = nullptr;
    v->pull();
    return {v, u};
  }
}

pair<node*, node*> split_leftmost_k(node* v, int k) {
  return split(v, [&](node* u) {
    int left_and_me = (u->l != nullptr ? u->l->sz : 0) + 1;
    if (k >= left_and_me) {
      k -= left_and_me;
      return false;
    }
    return true;
  });
}
```

```
node* merge(node* v, node* u) {
  if (v == nullptr) {
    return u;
  }
  if (u == nullptr) {
    return v;
  }
  v = get_rightmost(v);
  assert(v->r == nullptr);
  splay(u);
  v->push();
  v->r = u;
  v->pull();
  return v;
}

int count_left(node* v, const function<bool(node*)> &is_right) {
  if (v == nullptr) {
    return 0;
  }
  pair<node*, int> p = find(v, [&](node* u) { return is_right(u) ? -1 : 1; });
  node* u = p.first;
  return (u->l != nullptr ? u->l->sz : 0) + (p.second == 1);
}

node* add(node* r, node* v, const function<bool(node*)> &go_left) {
  pair<node*, node*> p = split(r, go_left);
  return merge(p.first, merge(v, p.second));
}

node* remove(node* v) { // returns the new root
  splay(v);
  v->push();
  node* x = v->l;
  node* y = v->r;
  v->l = v->r = nullptr;
  node* z = merge(x, y);
  if (z != nullptr) {
    z->p = v->p;
  }
  v->p = nullptr;
  v->push();
  v->pull(); // now v might be reusable...
  return z;
}

node* next(node* v) {
  splay(v);
  v->push();
  if (v->r == nullptr) {
    return nullptr;
```

```
  }
  v = v->r;
  while (v->l != nullptr) {
    v->push();
    v = v->l;
  }
  splay(v);
  return v;
}

node* prev(node* v) {
  splay(v);
  v->push();
  if (v->l == nullptr) {
    return nullptr;
  }
  v = v->l;
  while (v->r != nullptr) {
    v->push();
    v = v->r;
  }
  splay(v);
  return v;
}

int get_size(node* v) {
  splay(v);
  return (v != nullptr ? v->sz : 0);
}

template<typename... T>
void my_apply(node* v, T... args) {
  splay(v);
  v->unsafe_apply(args...);
}

void reverse(node* v) {
  splay(v);
  v->unsafe_reverse();
}

}  // namespace splay_tree

using namespace splay_tree;

template <bool rooted>
class link_cut_tree {
 public:
  int n;
  vector<node*> nodes;

  link_cut_tree(int _n) : n(_n) {
```

```
  nodes.resize(n);
  for (int i = 0; i < n; i++) {
    nodes[i] = new node(i, 0);
  }
}

int add_node(int x) {
  int id = (int) nodes.size();
  nodes.push_back(new node(id, x));
  return id;
}

void expose(node* v) {
  node* r = nullptr;
  node* u = v;
  while (u != nullptr) {
    splay(u);
    u->push();
    u->r = r;
    u->pull();
    r = u;
    u = u->p;
  }
  splay(v);
  assert(v->p == nullptr);
}

int get_root(int i) {
  node* v = nodes[i];
  expose(v);
  return get_leftmost(v)->id;
}

bool link(int i, int j) { // for rooted: (x, parent[x])
  if (i == j) {
    return false;
  }
  node* v = nodes[i];
  node* u = nodes[j];
  if (rooted) {
    splay(v);
    if (v->p != nullptr || v->l != nullptr) {
      return false; // not a root
    }
  } else {
    make_root(i);
  }
  expose(u);
  if (v->p != nullptr) {
    return false;
  }
  v->p = u;
```

```
  return true;
}

bool cut(int i, int j) { // for rooted: (x, parent[x])
  if (i == j) {
    return false;
  }
  node* v = nodes[i];
  node* u = nodes[j];
  expose(u);
  splay(v);
  if (v->p != u) {
    if (rooted) {
      return false;
    }
    swap(u, v);
    expose(u);
    splay(v);
    if (v->p != u) {
      return false;
    }
  }
  v->p = nullptr;
  return true;
}

bool cut(int i) { // only for rooted
  assert(rooted);
  node* v = nodes[i];
  expose(v);
  v->push();
  if (v->l == nullptr) {
    return false; // already a root
  }
  v->l->p = nullptr;
  v->l = nullptr;
  v->pull();
  return true;
}

bool connected(int i, int j) {
  if (i == j) {
    return true;
  }
  node* v = nodes[i];
  node* u = nodes[j];
  expose(v);
  expose(u);
  return v->p != nullptr;
}

int lca(int i, int j) {
```

```
    if (i == j) {
      return i;
    }
    node* v = nodes[i];
    node* u = nodes[j];
    expose(v);
    expose(u);
    if (v->p == nullptr) {
      return -1;
    }
    splay(v);
    if (v->p == nullptr) {
      return v->id;
    }
    return v->p->id;
  }

  bool is_ancestor(int i, int j) {
    if (i == j) {
      return true;
    }
    node* v = nodes[i];
    node* u = nodes[j];
    expose(u);
    splay(v);
    return v->p == nullptr && u->p != nullptr;
  }

  void make_root(int i) {
    assert(!rooted);
    node* v = nodes[i];
    expose(v);
    reverse(v);
  }

  node* get_path_from_root(int i) {
    node* v = nodes[i];
    expose(v);
    return v;
  }

  template <typename... T>
  void my_apply(int i, T... args) {
    node* v = nodes[i];
    splay_tree::my_apply(v, args...);
  }
};

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

```
  cin >> n >> tt;
  link_cut_tree<false> lct(n);
  while (tt--) {
    int u, v, x;
    cin >> u >> v >> x;
    --u; --v;
    if (lct.connected(u, v)) {
      lct.make_root(u);
      auto t = lct.get_path_from_root(v);
      if (t->sum_b == 0 && (t->sum_w + x) % 2 == 1) {
        cout << "YES" << '\n';
        my_apply(t, 1);
      } else {
        cout << "NO" << '\n';
      }
    } else {
      cout << "YES" << '\n';
      int tmp = lct.add_node(x);
      lct.link(u, tmp);
      lct.link(v, tmp);
    }
  }
  return 0;
}
```

ST-Class

```
template <typename T, class F = function<T(const T&, const T&)>>
class SparseTable {
 public:
  int n;
  vector<vector<T>> mat;
  F func;

  SparseTable(const vector<T>& a, const F& f) : func(f) {
    n = static_cast<int>(a.size());
    int max_log = 32 - __builtin_clz(n);
    mat.resize(max_log);
    mat[0] = a;
    for (int j = 1; j < max_log; j++) {
      mat[j].resize(n - (1 << j) + 1);
      for (int i = 0; i <= n - (1 << j); i++) {
        mat[j][i] = func(mat[j - 1][i], mat[j - 1][i + (1 << (j - 1))]);
      }
    }
  }

  T get(int from, int to) const {
    assert(0 <= from && from <= to && to <= n - 1);
    int lg = 32 - __builtin_clz(to - from + 1) - 1;
    return func(mat[lg][from], mat[lg][to - (1 << lg) + 1]);
```

```
  }
};
```

sweepline Mo-rollback

```
int n, q, k, block;
int cnt[maxn], ans[maxn], a[maxn], vis[maxn];
vector<array<int, 4>> que;

int getb(int x) {
    return (x - 1) / block + 1;
}

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

    cin >> n;
    block = sqrt(n);

    rep(i, 1, n) cin >> a[i];
    cin >> q;
    rep(i, 1, q) {
        int l, r;
        cin >> l >> r >> k;
        que.pb({l, r, i, k});
    }
    sort(ALL(que), [&](array<int, 4> a, array<int, 4> b)->bool{
        if (getb(a[0]) != getb(b[0]))
            return getb(a[0]) < getb(b[0]);
        else
            return a[1] < b[1];
    });

    int len = que.size();
    int l, r;
    auto add = [&](int x, int t) {
        cnt[vis[a[x]]]--;
        vis[a[x]]++;
        cnt[vis[a[x]]]++;
    };
    auto del = [&](int x) {
        cnt[vis[a[x]]]--;
        vis[a[x]]--;
        cnt[vis[a[x]]]++;
    };

    for (int x = 0; x < len;) {
        int y = x;
        while (y < len && getb(que[y][0]) == getb(que[x][0])) y++;
        //暴力块内
```

```
        while (x < y && que[x][1] <= getb(que[x][0])*block) {
            for (int j = que[x][0]; j <= que[x][1]; j++)
                add(j, que[x][3]);
            ans[que[x][2]] = cnt[que[x][3]];
            for (int j = que[x][0]; j <= que[x][1]; j++)
                del(j);
            x++;
        }
        //块外
        r = getb(que[x][0]) * block;
        while (x < y) {
            l = getb(que[x][0]) * block + 1;
            while (r < que[x][1]) r++, add(r, que[x][3]);
            while (l > que[x][0]) l--, add(l, que[x][3]);
            ans[que[x][2]] = cnt[que[x][3]];
            for (int j = que[x][0]; j <= getb(que[x][0])*block; j++)
                del(j);
            x++;
        }
        for (int j = getb(que[x - 1][0]) * block + 1; j <= que[x - 1][1]; j++)
            del(j);
    }
    rep(i, 1, q) cout << ans[i] << '\n';
}
```

sweepline Mo

```
int main() {
    std::ios::sync_with_stdio(false);
    cin.tie(0); cout.tie(0);

    for (int i = 1; i <= m; i++) {
        int x, y;
        cin >> x >> y;
        q.pb({x, y, i});
        rej[i] = (y - x + 1LL) * (y - x) / 2LL;
    }
    sort(q.begin(), q.end(), [&](array<int, 3> a, array<int, 3> b)->bool{
        if (getb(a[0]) == getb(b[0]))
            if (getb(a[0]) & 1)
                return a[1] < b[1];
            else
                return a[1] > b[1];
        else return getb(a[0]) < getb(b[0]);
    });

    int L = 1, R = 0;
    for (int i = 0; i < m; i++) {
        while (R < q[i][1]) R++, add(R);
        while (L > q[i][0]) L--, add(L);
        while (L < q[i][0]) del(L), L++;
```

```
        while (R > q[i][1]) del(R), R--;
        ans[q[i][2]] = tmp;
    }
}
```

treap(shared_ptr)

```
struct Tree {
    std::shared_ptr<Tree> l;
    std::shared_ptr<Tree> r;
    // Tree *l, *r;
    int v;
    int s;
    Tree(int v = -1) : l(nullptr), r(nullptr), v(v), s(1) {}
    void pull() {
        s = 1;
        if (l != nullptr) {
            s += l->s;
        }
        if (r != nullptr) {
            s += r->s;
        }
    }
};

using pTree = std::shared_ptr<Tree>;
// using pTree = Tree*;

std::pair<pTree, pTree> split(pTree t, int k) {
    if (k == 0) {
        return {nullptr, t};
    }
    if (k == t->s) {
        return {t, nullptr};
    }
    pTree nt = std::make_shared<Tree>();
    // pTree nt = new Tree();
    *nt = *t;
    if (t->l != nullptr && k <= t->l->s) {
        auto [a, b] = split(t->l, k);
        nt->l = b;
        nt->pull();
        return {a, nt};
    } else {
        auto [a, b] = split(t->r, k - 1 - (t->l == nullptr ? 0 : t->l->s));
        nt->r = a;
        nt->pull();
        return {nt, b};
    }
}
```

```
std::tuple<pTree, pTree, pTree> split3(pTree t, int l, int r) {
    auto [LM, R] = split(t, r);
    auto [L, M] = split(LM, l);
    return {L, M, R};
}

std::mt19937 rnd(std::chrono::steady_clock::now().time_since_epoch().count());

pTree merge(pTree a, pTree b) {
    if (a == nullptr) {
        return b;
    }
    if (b == nullptr) {
        return a;
    }

    pTree t = std::make_shared<Tree>();
    // pTree t = new Tree();

    if (int(rnd() % (a->s + b->s)) < a->s) {
        *t = *a;
        t->r = merge(a->r, b);
    } else {
        *t = *b;
        t->l = merge(a, t->l);
    }
    t->pull();

    return t;
}

pTree build(const std::vector<int> &v, int l, int r) {
    if (l == r) {
        return nullptr;
    }
    int m = (l + r) / 2;
    auto t = std::make_shared<Tree>(v[m]);
    // auto t = new Tree(v[m]);

    t->l = build(v, l, m);
    t->r = build(v, m + 1, r);
    t->pull();
    return t;
}

void rec(pTree t, std::vector<int> &v, int &cnt) {
    if (t == nullptr) {
        return;
    }
    rec(t->l, v, cnt);
    v[cnt++] = t->v;
    rec(t->r, v, cnt);
```

```
}
```

Tree Decomposition

```
void solve(int u, int s) {
    int root = -1, cnt = s + 1;
    function<void(int, int)> center = [&](int u, int f) {
        sz[u] = 1, maxs[u] = 0;
        for (auto [v, w] : e[u]) if (v != f && !del[v]) {
                center(v, u);
                sz[u] += sz[v];
                maxs[u] = max(maxs[u], sz[v]);
            }
        maxs[u] = max(maxs[u], s - sz[u]);
        if (maxs[u] < cnt) cnt = maxs[u], root = u;
    };  // using lambda(const auto &self) => faster
    center(u, 0);

    // calc
    vector<pair<int, bool>> d;
    cur[s] = 1;
    function<void(int, int, int)> dfs = [&](int u, int f, int dep) {
        d.pb({dep, cur[s + dep] != 0});
        if (dep == 0 && cur[s] > 1) ans++;
        cur[s + dep]++;
        for (auto [v, w] : e[u]) if (v != f && !del[v]) {
                dfs(v, u, dep + w);
            }
        cur[s + dep]--;
    };  // using lambda(const auto &self) => faster

    for (auto [v, w] : e[root]) if (!del[v]) {
            dfs(v, root, w);
            for (auto [d1, d2] : d) {
                if (d2)
                    ans += c[s - d1][0] + c[s - d1][1];
                else
                    ans += c[s - d1][1];
            }
            for (auto [d1, d2] : d) {
                c[s + d1][d2]++;
            }
            d.clear();
        }
    cur[s]--;
    rep(i, 0, 2 * s) c[i][0] = c[i][1] = 0;

    del[root] = 1;
    for (auto [v, w] : e[root])
        if (!del[v]) solve(v, sz[v]);
}
```

union find

```
ll fa[maxn], d[maxn];
void init() {
    rep(i, 1, n) fa[i] = i, d[i] = 0;
}
int find(int x) {
    if (fa[x] == x) return fa[x];
    int p = fa[x];
    fa[x] = find(fa[x]);
    d[x] = d[x] + d[p];
    return fa[x];
}
void unite(int l, int r, ll x) {
    int fl = find(l);
    int fr = find(r);
    fa[fr] = fl;
    d[fr] = d[l] - d[r] + x;
}
```

VirtualTree

```
namespace compact {
const int LOGN=18;
int l[N],r[N],tot,p[N][20],n;
map<int,int> cv;
int lca(int u,int v) {
    if (dep[u]>dep[v]) swap(u,v);
    per(i,LOGN-1,0) if (dep[p[v][i]]>=dep[u]) v=p[v][i];
    if (u==v) return u;
    per(i,LOGN-1,0) if (p[v][i]!=p[u][i]) u=p[u][i],v=p[v][i];
    return p[u][0];
}
void dfs(int u,int f) {
    l[u]=++tot; dep[u]=dep[f]+1; p[u][0]=f;
    vec[dep[u]].pb(u);
    for (auto v:vE[u]) {
        if (v==f) continue;
        dfs(v,u);
    }
    r[u]=tot;
}
void build(int _n) {
    n=_n; tot=0;
    dfs(1,0);
    rep(j,1,LOGN-1) rep(i,1,n) p[i][j]=p[p[i][j-1]][j-1];
}

bool cmp(int u,int v) { return l[u]<l[v]; }
vector<PII> compact(VI v) {
```

```
    int m=SZ(v);
    vector<PII> E;
    sort(all(v),cmp);
    rep(i,0,m-2) {
        int w=lca(v[i],v[i+1]);
        v.pb(w);
    }
    v.pb(0);
    v.pb(1);
    sort(all(v),cmp);
    v.erase(unique(all(v)),v.end());
    cv.clear();
    per(i,SZ(v)-1,1) {
        int u=v[i];
        while (1) {
            auto it=cv.lower_bound(l[u]);
            if (it==cv.end()||it->fi>r[u]) break;
            E.pb(mp(u,v[it->se]));
            cv.erase(it);
        }
        cv[l[u]]=i;
    }
    return E;
}
};
```

DP

有依赖决策单调

```
pair<int, int> stk[N];
auto calc = [&](int i, int j) { ... } // dp[j] -> dp[i]
int h = 0, t = 0;
stk[t++] = {1, 0}; // {left, opt}

for (int i = 1; i <= n; i++) {
    if (h < t && stk[h].first < i) stk[h].first++;
    if (h + 1 < t && stk[h].first >= stk[h + 1].first) ++h;
    dp[i] = calc(i, stk[h].second);
    while (h < t && calc(stk[t - 1].first, stk[t - 1].second) >= calc(stk[t - 1].first, i))
        --t;
    if (h < t) {
        int l = stk[t - 1].first, r = n + 1;
        while (l + 1 < r) {
            int md = (l + r) >> 1;
            if (calc(md, stk[t - 1].second) < calc(md, i)) l = md; else r = md;
        }
        if (r <= n) stk[t++] = {r, i};
```

```
    } else stk[t++] = {i, i};
}
```

Convex hull optimization

```
array<ll, 3> a[maxn];
int q[maxn];
ll ans[maxn];

ll X(int p) {
    return 2ll * a[p][0];
}
ll Y(int p) {
    return a[p][0] * a[p][0] + a[p][1];
}
ldb slope(int x, int y) {
    return (ldb)(Y(y) - Y(x)) / (X(y) - X(x));
}
void solve() {
    cin >> n;
    int head = 1, rear = 0;
    rep(i, 1, n) {
        cin >> a[i][0] >> a[i][1];
        a[i][2] = i;
    }
    sort(a + 1, a + n + 1);

    rep(i, 1, n) {
        while (head < rear && slope(q[rear], i) <= slope(q[rear], q[rear - 1])) rear--;
        q[++rear] = i;
    }
    rep(i, 1, n) {
        ll k = -a[i][0];
        while (head < rear && slope(q[head], q[head + 1]) <= k) head++;
        ans[a[i][2]] = (a[i][0] + a[q[head]][0]) * (a[i][0] + a[q[head]][0]) + a[i][1] +
a[q[head]][1];
    }
    rep(i, 1, n) cout << ans[i] << '\n';
}
```

DivideAndConquerDP

```
ll w[N][N],sum[N][N],opt[N],dp[805][N];

ll calc(int i,int j) { return sum[j][j]-sum[j][i]-sum[i][j]+sum[i][i]; }

void rec(int d,int l,int r,int optl,int optr) {
    if (l>r) return;
    int md=(l+r)>>1;
    rep(i,optl,optr) if (dp[d-1][i]+calc(i,md)<dp[d][md]) {
```

```
        dp[d][md]=dp[d-1][i]+calc(i,md);
        opt[md]=i;
    }
    rec(d,l,md-1,optl,opt[md]);
    rec(d,md+1,r,opt[md],optr);
}
```

Math

扩展欧拉定理

```
// mod [min(b, b % phi + phi)]
ll calc(ll p) {
    if (p == 1) return 0;
    int phi = p, q = p;
    for (int i = 2; i * i <= p; i++) {
        if (q % i == 0) {
            phi = phi / i * (i - 1);
            while (q % i == 0) q /= i;
        }
    }
    if (q != 1) phi = phi / q * (q - 1);
    return powmod(2, calc(phi) + phi, p);
}
```

拉格朗日插值

```
/*
    k阶多项式(需要k+1个点)
    求在点n上的值
    O(k)
*/
ll lagrange(ll n,int k) {
    vector<ll> x(k+5),y(k+5);
    rep(i,1,k+1) {
        x[i]=i;
        // y[i]=(y[i-1]+powmod(i,k-1,mod))%mod;
    }
    if (n<=k+1) return y[n];

    vector<ll> fac(k+5); fac[0]=1;
    ll coe=1;
    rep(i,1,k+4) fac[i]=fac[i-1]*i%mod;
    rep(i,1,k+1) coe=coe*(n-i+mod)%mod;
    ll ans=0;
    rep(i,1,k+1) {
        ll sgn=(((k+1-i)%2)?-1:1);
```

```
        ll f1=powmod(fac[i-1]*fac[k+1-i]%mod,mod-2,mod);
        ll f2=powmod(n-i,mod-2,mod);
        ans+=sgn*coe*f1%mod*f2%mod*y[i]%mod;
        ans=(ans+mod)%mod;
    }
    return ans;
}
```

枚举超集/子集

```
void solve() {
    for (int i = 1; i < (1ll << n); i++) {
        int t = i;
        while (true) {
            t = (t + 1) | i;
            if (t == bit(n) - 1) break;
        }
    }
}

void solve() {
    cin >> n;
    f[0] = 1;
    for (int i = 1; i < (1ll << n); i++) {
        int t = i;
        ll res = 0;
        while (true) {
            if (t == 0) break;
            t = (t - 1)&i;
            res = (res + f[t]) % mod;
        }
        f[i] = res * i;
    }
}
```

幂转下降幂(求幂和)

```
ll comb[N][N];
ll s[maxn],inv[maxn],p;
/* 1^k+2^k+...+n^k */
void solve() {
    cin>>k>>n>>p;
    rep(i,0,k+1) {
        comb[i][0]=comb[i][i]=1;
        rep(j,1,i-1) {
            comb[i][j]=(comb[i-1][j-1]+comb[i-1][j])%p;
        }
    }
    inv[1]=1;
    rep(i,2,k+1) inv[i]=(p-p/i)*inv[p%i]%p;
```

```
    assert(inv[k]*k%p==1);

    ll pw=1;
    // (k+1)*S[k]=(n+1)^(k+1)-[0-k-1](k+1,j)*S[j]-1
    rep(i,0,k) {
        pw=pw*(n+1)%p;
        s[i]=(pw-1+p)%p;
        rep(j,0,i-1) {
            s[i]=(s[i]-comb[i+1][j]*s[j]%p+p)%p;
        }
        s[i]=s[i]*inv[i+1]%p;
    }
    cout<<s[k]<<'\n';
}
```

莫比乌斯反演

```
uint pr[maxn], p[maxn], pe[maxn], u[maxn], tot;
uint g[maxn], f[maxn];
int main() {
    p[1] = 1;
    for (int i = 2; i <= n; i++) {
        if (!p[i]) pe[i] = i, p[i] = i, pr[++tot] = i;
        for (uint j = 1; j <= tot && pr[j]*i <= n; j++) {
            p[pr[j]*i] = pr[j];
            if (pr[j] == p[i]) {
                pe[pr[j]*i] = pe[i] * p[i];
                break;
            } else {
                pe[pr[j]*i] = pr[j];
            }
        }
    }
    u[1] = 1;
    for (uint i = 2; i <= n; i++) {
        if (i == pe[i]) {
            if (i == p[i]) u[i] = (uint) - 1;
            else u[i] = (uint)0;
        } else {
            u[i] = u[pe[i]] * u[i / pe[i]];
        }
    }
    for (int i = 1; i <= n; i++)
        for (int j = 1; i * j <= n; j++) {
            g[i * j] += f[i] * u[j];
        }
}
```

莫比乌斯反演&gcd常见结论

```
int n = 1e7 + 15, m1, m2;
int pr[maxn], p[maxn], pe[maxn], u[maxn], tot;
int su[maxn];
/*  u * 1 = e, phi * 1 = id, phi = id * u  */
int main() {
    p[1] = 1;
    for (int i = 2; i <= n; i++) {
        if (!p[i]) pe[i] = i, p[i] = i, pr[++tot] = i;
        for (int j = 1; j <= tot && pr[j]*i <= n; j++) {
            p[pr[j]*i] = pr[j];
            if (pr[j] == p[i]) {
                pe[pr[j]*i] = pe[i] * p[i];
                break;
            } else {
                pe[pr[j]*i] = pr[j];
            }
        }
    }
    u[1] = 1;
    for (int i = 2; i <= n; i++) {
        if (i == pe[i]) {
            if (i == p[i]) u[i] = -1;
            else u[i] = 0;
        } else {
            u[i] = u[pe[i]] * u[i / pe[i]];
        }
    }
    rep(i, 1, n) su[i] = su[i - 1] + u[i];

    cin >> m1 >> m2;
    ll ans = 0;
    for (int l = 1; l <= m1 && l <= m2; l++) {
        int d1 = m1 / l, d2 = m2 / l;
        int r = min(m1 / d1, m2 / d2);
        ans += 1ll * (m1 / l) * (m2 / l) * (su[r] - su[l - 1]);
        l = r;
    }
    cout << ans << '\n';
}
```

区间互质

```
int p[100], num;
void prime(int n) {
    num = 0;
    for (int i = 2; i * i <= n; i++) {
        if ((n % i) == 0) {
            p[++num] = i;
```

```
            while ((n % i) == 0) n /= i;
        }
    }
    if (n > 1) p[++num] = n;
}
ll solve(ll r, int k) {
    prime(k); ll res = 0;
    for (int i = 1; i < (1 << num); i++) {
        int k = 0; ll div = 1;
        for (int j = 1; j <= num; j++) {
            if (i & (1 << (j - 1))) {
                k++; div *= p[j];
            }
        }
        if (k % 2) res += r / div;
        else res -= r / div;
    }
    return r - res;
}
ll que(ll L, ll R, ll k) {
    return solve(R, k) - solve(L - 1, k);
}
```

线性筛&常见积性函数

```
uint p[maxn], pe[maxn], prime[maxn];
// 因子个数  因子和  欧拉函数  莫比乌斯函数
uint d[maxn], f[maxn], phip[maxn], u[maxn];
uint tot;
void solve() {
    p[1] = 1;
    for (uint i = 2; i <= n; i++) {
        if (!p[i]) p[i] = i, pe[i] = i, prime[++tot] = i;
        for (uint j = 1; j <= tot && prime[j]*i <= n; j++) {
            p[prime[j]*i] = prime[j];
            if (prime[j] == p[i]) {
                pe[prime[j]*i] = pe[i] * p[i];
                break;
            } else {
                pe[prime[j]*i] = prime[j];
            }
        }
    }

    d[1] = 1;
    for (uint i = 2; i <= n; i++) {
        if (i == pe[i])
            d[i] = d[i / p[i]] + 1;
        else
            d[i] = d[i / pe[i]] * d[pe[i]];
    }
```

```
    f[1] = 1;
    for (uint i = 2; i <= n; i++) {
        if (i == pe[i])
            f[i] = f[i / p[i]] + i;
        else
            f[i] = f[i / pe[i]] * f[pe[i]];
    }

    phip[1] = 1;
    for (uint i = 2; i <= n; i++) {
        if (i == pe[i])
            phip[i] = i / p[i] * (p[i] - 1);
        else
            phip[i] = phip[i / pe[i]] * phip[pe[i]];
    }

    u[1] = (uint)1;
    for (uint i = 2; i <= n; i++) {
        if (i == pe[i])
            if (i == p[i]) u[i] = (uint) - 1;
            else u[i] = (uint)0;
        else
            u[i] = u[i / pe[i]] * u[pe[i]];
    }
}
```

整除分块

```
void solve() {
    u64 ans = 0;
    cin >> n;
    for (ll l = 1; l <= n; l++) {
        ll d = n / l, r = n / d;
        ans += (l + r) * (r - l + 1) / 2 * d;
        l = r;
    }
}
```

binom分段打表

```
ll fact[] = {};
ll fct(ll x){
    ll res=fact[x/1000000];
    for(ll i=x/1000000*1000000+1;i<=x;i++){
        res=res*i%mod;
    }
    return res;
}

ll binom(ll a,ll b){
    if(b<0||b>a) return 0;
    return fct(a)*powmod(fct(b)*fct(a-b)%mod,mod-2,mod)%mod;
}
```

BM

```
namespace linear_seq {
    const int N=10010;
    ll res[N],base[N],_c[N],_md[N];

    vector<int> Md;
    void mul(ll *a,ll *b,int k) {
        rep(i,0,k+k) _c[i]=0;
        rep(i,0,k) if (a[i]) rep(j,0,k) _c[i+j]=(_c[i+j]+a[i]*b[j])%mod;
        for (int i=k+k-1;i>=k;i--) if (_c[i])
            rep(j,0,SZ(Md)) _c[i-k+Md[j]]=(_c[i-k+Md[j]]-_c[i]*_md[Md[j]])%mod;
        rep(i,0,k) a[i]=_c[i];
    }
    int solve(ll n,VI a,VI b) { // a 系数 b 初值 b[n+1]=a[0]*b[n]+...
        ll ans=0,pnt=0;
        int k=SZ(a);
        assert(SZ(a)==SZ(b));
        rep(i,0,k) _md[k-1-i]=-a[i];_md[k]=1;
        Md.clear();
        rep(i,0,k) if (_md[i]!=0) Md.push_back(i);
        rep(i,0,k) res[i]=base[i]=0;
        res[0]=1;
        while ((1ll<<pnt)<=n) pnt++;
        for (int p=pnt;p>=0;p--) {
            mul(res,res,k);
            if ((n>>p)&1) {
                for (int i=k-1;i>=0;i--) res[i+1]=res[i];res[0]=0;
                rep(j,0,SZ(Md)) res[Md[j]]=(res[Md[j]]-res[k]*_md[Md[j]])%mod;
            }
        }
        rep(i,0,k) ans=(ans+res[i]*b[i])%mod;
        if (ans<0) ans+=mod;
        return ans;
    }
    VI BM(VI s) {
        VI C(1,1),B(1,1);
```

```
        int L=0,m=1,b=1;
        rep(n,0,SZ(s)) {
            ll d=0;
            rep(i,0,L+1) d=(d+(ll)C[i]*s[n-i])%mod;
            if (d==0) ++m;
            else if (2*L<=n) {
                VI T=C;
                ll c=mod-d*powmod(b,mod-2)%mod;
                while (SZ(C)<SZ(B)+m) C.pb(0);
                rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
                L=n+1-L; B=T; b=d; m=1;
            } else {
                ll c=mod-d*powmod(b,mod-2)%mod;
                while (SZ(C)<SZ(B)+m) C.pb(0);
                rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
                ++m;
            }
        }
        return C;
    }
    int gao(VI a,ll n) {
        VI c=BM(a);
        c.erase(c.begin());
        rep(i,0,SZ(c)) c[i]=(mod-c[i])%mod;
        return solve(n,c,VI(a.begin(),a.begin()+SZ(c)));
    }
};
```

bsgs

```
struct Hash_table {
    static const int V = 1000003;
    int fst[V], nxt[V];
    int ctm, ptm[V], T;
    int val[V];
    ll key[V];
    void init() {T = 0, ctm++;}
    void insert(ll k, int v) {
        int s = k % V;
        if (ptm[s] != ctm) ptm[s] = ctm, fst[s] = -1;
        for (int i = fst[s]; i != -1; i = nxt[i]) if (key[i] == k) {
                return;
            }
        nxt[T] = fst[s], fst[s] = T, key[T] = k, val[T] = v;
        T++;
    }
    int query(ll k) {
        int s = k % V;
        if (ptm[s] != ctm) return -1;
        for (int i = fst[s]; i != -1; i = nxt[i]) {
            if (key[i] == k) return val[i];
```

```
        }
        return -1;
    }
} hs;

int bsgs(int a, int b, int m) { // a^x=b(mod m)
    int res = m + 1;
    int t = sqrt(m) + 2;
    ll d = powmod(a, t, m);
    ll cnt = 1;
    //map<int,int> p;
    hs.init();
    for (int i = 1; i <= t; i++) {
        cnt = cnt * d % m;
        //if (!p.count(cnt)) p[cnt] = i;
        if (hs.query(cnt) == -1) hs.insert(cnt, i);
    }
    cnt = b;
    for (int i = 1; i <= t; i++) {
        cnt = cnt * a % m;
        //if (p.count(cnt)) res = min(res, p[cnt] * t - i);
        int tmp = hs.query(cnt);
        if (tmp != -1) res = min(res, tmp * t - i);
    }
    if (res >= m) res = -1;
    return res;
}
```

cantor

```
ll fac[maxn], A[maxn], w[maxn];
void init(int n) {
    fac[0] = 1;
    rep(i, 1, n) fac[i] = fac[i - 1] * i % mod;
}
ll cantor(int w[], int n) {
    ll ans = 1;
    for (int i = 1; i <= n; i++) { // can optimize by BIT
        for (int j = i + 1; j <= n; j++) {
            if (w[i] > w[j]) A[i]++;
        }
    }
    for (int i = 1; i < n; i++) {
        ans += A[i] * fac[n - i];
    }
    return ans;
}

void decanter(ll x, int n) { // x->rank n->length
    x--;
    vector<int> rest(n, 0);
```

```
    iota(rest.begin(), rest.end(), 1); // rest->1,2,3,4...
    for (int i = 1; i <= n; i++) {
        A[i] = x / fac[n - i];
        x %= fac[n - i];
    }
    for (int i = 1; i <= n; i++) {
        w[i] = rest[A[i]];
        rest.erase(lower_bound(rest.begin(), rest.end(), w[i]));
    }
}
```

EXCRT&modequ&exgcd

```
ll exgcd(ll a, ll b, ll &x, ll &y) {
    if (b == 0) {
        x = 1, y = 0;
        return a;
    }
    ll d = exgcd(b, a % b, y, x);
    y -= (a / b) * x;
    return d;
}

// 求 a * x = b (mod m) 的解
ll modequ(ll a, ll b, ll m) {
    ll x, y;
    ll d = exgcd(a, m, x, y);
    if (b % d != 0) return -1;
    m /= d; a /= d; b /= d;
    x = x * b % m;
    if (x < 0) x += m;
    return x;
}

void merge(ll &a, ll &b, ll c, ll d) {
    if (a == -1 || b == -1) return;
    ll x, y;
    ll g = exgcd(b, d, x, y);
    if ((c - a) % g != 0) {
        a = -1, b = -1;
        return;
    }
    d /= g;
    ll t = ((c - a) / g) % d * x % d;
    if (t < 0) t += d;
    a = b * t + a;
    b = b * d;
}
```

exgcd&(floor/ceil)div

```
ll exgcd(ll a, ll b, ll &x, ll &y) { // ax+by=(a,b)
    if (b == 0) {
        x = 1, y = 0;
        return a;
    }
    ll d = exgcd(b, a % b, y, x);
    y -= (a / b) * x;
    return d;
}

ll floordiv(ll a, ll b) {
    if (a % b == 0) return a / b;
    else if ((a > 0 && b > 0) || (a < 0 && b < 0)) return a / b;
    else return a / b - 1;
}

ll ceildiv(ll a, ll b) {
    if (a % b == 0) return a / b;
    else if ((a > 0 && b > 0) || (a < 0 && b < 0)) return a / b + 1;
    else return a / b;
}

void solve() {
    ll a, b, d, l1, l2, r1, r2;
    cin >> a >> b >> d >> l1 >> r1 >> l2 >> r2;
    ll x, y;
    ll g = exgcd(a, b, x, y);
    a /= g, b /= g, d /= g;
    x = d % b * x % b;
    y = (d - a * x) / b;
    /*
    l1<=x=x0+b*t<=r1
    l2<=y=y0-a*t<=r2
    */
    ll L = max(ceildiv(l1 - x, b), ceildiv(r2 - y, -a));
    ll R = min(floordiv(r1 - x, b), floordiv(l2 - y, -a));

    if (L > R) cout << 0 << '\n';
    else cout << R - L + 1 << '\n';
}
```

factor

```
namespace Factor {
    const int N=1010000;
    ll C,fac[10010],n,mut,a[1001000];
    int T,cnt,i,l,prime[N],p[N],psize,_cnt;
    ll _e[100],_pr[100];
```

```
vector<ll> d;
inline ll mul(ll a,ll b,ll p) {
    if (p<=1000000000) return a*b%p;
    else if (p<=100000000000000ll) return (((a*(b>>20)%p)<<20)+(a*(b&((1<<20)-1))))%p;
    else {
        ll d=(ll)floor(a*(long double)b/p+0.5);
        ll ret=(a*b-d*p)%p;
        if (ret<0) ret+=p;
        return ret;
    }
}
void prime_table(){
    int i,j,tot,t1;
    for (i=1;i<=psize;i++) p[i]=i;
    for (i=2,tot=0;i<=psize;i++){
        if (p[i]==i) prime[++tot]=i;
        for (j=1;j<=tot && (t1=prime[j]*i)<=psize;j++){
            p[t1]=prime[j];
            if (i%prime[j]==0) break;
        }
    }
}
void init(int ps) {
    psize=ps;
    prime_table();
}
ll powl(ll a,ll n,ll p) {
    ll ans=1;
    for (;n;n>>=1) {
        if (n&1) ans=mul(ans,a,p);
        a=mul(a,a,p);
    }
    return ans;
}
bool witness(ll a,ll n) {
    int t=0;
    ll u=n-1;
    for (;~u&1;u>>=1) t++;
    ll x=powl(a,u,n),_x=0;
    for (;t;t--) {
        _x=mul(x,x,n);
        if (_x==1 && x!=1 && x!=n-1) return 1;
        x=_x;
    }
    return _x!=1;
}
bool miller(ll n) {
    if (n<2) return 0;
    if (n<=psize) return p[n]==n;
    if (~n&1) return 0;
    for (int j=0;j<=7;j++) if (witness(rng()%(n-1)+1,n)) return 0;
    return 1;
```

```
}
ll gcd(ll a,ll b) {
    ll ret=1;
    while (a!=0) {
        if ((~a&1) && (~b&1)) ret<<=1,a>>=1,b>>=1;
        else if (~a&1) a>>=1; else if (~b&1) b>>=1;
        else {
            if (a<b) swap(a,b);
            a-=b;
        }
    }
    return ret*b;
}
ll rho(ll n) {
    while (1) {
        ll X=rng()%n,Y,Z,T=1,*lY=a,*lX=lY;
        int tmp=20;
        C=rng()%10+3;
        X=mul(X,X,n)+C;*(lY++)=X;lX++;
        Y=mul(X,X,n)+C;*(lY++)=Y;
        for(;X!=Y;) {
            ll t=X-Y+n;
            Z=mul(T,t,n);
            if(Z==0) return gcd(T,n);
            tmp--;
            if (tmp==0) {
                tmp=20;
                Z=gcd(Z,n);
                if (Z!=1 && Z!=n) return Z;
            }
            T=Z;
            Y=*(lY++)=mul(Y,Y,n)+C;
            Y=*(lY++)=mul(Y,Y,n)+C;
            X=*(lX++);
        }
    }
}
void _factor(ll n) {
    for (int i=0;i<cnt;i++) {
        if (n%fac[i]==0) n/=fac[i],fac[cnt++]=fac[i];}
    if (n<=psize) {
        for (;n!=1;n/=p[n]) fac[cnt++]=p[n];
        return;
    }
    if (miller(n)) fac[cnt++]=n;
    else {
        ll x=rho(n);
        _factor(x);_factor(n/x);
    }
}
void dfs(ll x,int dep) {
    if (dep==_cnt) d.pb(x);
```

```
        else {
            dfs(x,dep+1);
            for (int i=1;i<=_e[dep];i++) dfs(x*=_pr[dep],dep+1);
        }
    }
    void norm() {
        sort(fac,fac+cnt);
        _cnt=0;
        rep(i,0,cnt-1) if (i==0||fac[i]!=fac[i-1]) _pr[_cnt]=fac[i],_e[_cnt++]=1;
            else _e[_cnt-1]++;
    }
    vector<ll> getd() {
        d.clear();
        dfs(1,0);
        return d;
    }
    vector<ll> factor(ll n) {
        cnt=0;
        _factor(n);
        norm();
        return getd();
    }
    vector<PLL> factorG(ll n) {
        cnt=0;
        _factor(n);
        norm();
        vector<PLL> d;
        rep(i,0,_cnt-1) d.pb(mp(_pr[i],_e[i]));
        return d;
    }
    bool is_primitive(ll a,ll p) {
        assert(miller(p));
        vector<PLL> D=factorG(p-1);
        rep(i,0,SZ(D)-1) if (powl(a,(p-1)/D[i].fi,p)==1) return 0;
        return 1;
    }
    ll phi(ll n) {
        auto d=factorG(n);
        for (auto p:d) n=n/p.fi*(p.fi-1);
        return n;
    }
}
```

fft

```
namespace fft {
  typedef double dbl;

  struct num {
    dbl x, y;
    num() { x = y = 0; }
```

```
    num(dbl x, dbl y) : x(x), y(y) { }
  };

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

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

  const dbl PI = acosl(-1.0);

  void ensure_base(int nbase) {
    if (nbase <= base) {
      return;
    }
    rev.resize(1 << nbase);
    for (int i = 0; i < (1 << nbase); i++) {
      rev[i] = (rev[i >> 1] >> 1) + ((i & 1) << (nbase - 1));
    }
    roots.resize(1 << nbase);
    while (base < nbase) {
      dbl angle = 2 * PI / (1 << (base + 1));
//      num z(cos(angle), sin(angle));
      for (int i = 1 << (base - 1); i < (1 << base); i++) {
        roots[i << 1] = roots[i];
//        roots[(i << 1) + 1] = roots[i] * z;
        dbl angle_i = angle * (2 * i + 1 - (1 << base));
        roots[(i << 1) + 1] = num(cos(angle_i), sin(angle_i));
      }
      base++;
    }
  }

  void fft(vector<num> &a, int n = -1) {
    if (n == -1) {
      n = a.size();
    }
    assert((n & (n - 1)) == 0);
    int zeros = __builtin_ctz(n);
    ensure_base(zeros);
    int shift = base - zeros;
    for (int i = 0; i < n; i++) {
      if (i < (rev[i] >> shift)) {
        swap(a[i], a[rev[i] >> shift]);
      }
    }
/*    for (int k = 1; k < n; k <<= 1) {
      for (int i = 0; i < n; i += 2 * k) {
```

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

vector<num> fa, fb;

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

```
vector<int> multiply_mod(vector<int> &a, vector<int> &b, int m, int eq = 0) {
  int need = a.size() + b.size() - 1;
  int nbase = 0;
  while ((1 << nbase) < need) nbase++;
  ensure_base(nbase);
  int sz = 1 << nbase;
  if (sz > (int) fa.size()) {
    fa.resize(sz);
  }
  for (int i = 0; i < (int) a.size(); i++) {
    int x = (a[i] % m + m) % m;
    fa[i] = num(x & ((1 << 15) - 1), x >> 15);
  }
  fill(fa.begin() + a.size(), fa.begin() + sz, num {0, 0});
  fft(fa, sz);
  if (eq) {
    copy(fa.begin(), fa.begin() + sz, fb.begin());
  } else {
    if (sz > (int) fb.size()) {
      fb.resize(sz);
    }
    for (int i = 0; i < (int) b.size(); i++) {
      int x = (b[i] % m + m) % m;
      fb[i] = num(x & ((1 << 15) - 1), x >> 15);
    }
    fill(fb.begin() + b.size(), fb.begin() + sz, num {0, 0});
    fft(fb, sz);
  }
  dbl ratio = 0.25 / sz;
  num r2(0, -1);
  num r3(ratio, 0);
  num r4(0, -ratio);
  num r5(0, 1);
  for (int i = 0; i <= (sz >> 1); i++) {
    int j = (sz - i) & (sz - 1);
    num a1 = (fa[i] + conj(fa[j]));
    num a2 = (fa[i] - conj(fa[j])) * r2;
    num b1 = (fb[i] + conj(fb[j])) * r3;
    num b2 = (fb[i] - conj(fb[j])) * r4;
    if (i != j) {
      num c1 = (fa[j] + conj(fa[i]));
      num c2 = (fa[j] - conj(fa[i])) * r2;
      num d1 = (fb[j] + conj(fb[i])) * r3;
      num d2 = (fb[j] - conj(fb[i])) * r4;
      fa[i] = c1 * d1 + c2 * d2 * r5;
      fb[i] = c1 * d2 + c2 * d1;
    }
    fa[j] = a1 * b1 + a2 * b2 * r5;
    fb[j] = a1 * b2 + a2 * b1;
  }
  fft(fa, sz);
  fft(fb, sz);
```

```
    vector<int> res(need);
    for (int i = 0; i < need; i++) {
      long long aa = fa[i].x + 0.5;
      long long bb = fb[i].x + 0.5;
      long long cc = fa[i].y + 0.5;
      res[i] = (aa + ((bb % m) << 15) + ((cc % m) << 30)) % m;
    }
    return res;
  }

  vector<int> square_mod(vector<int> &a, int m) {
    return multiply_mod(a, a, m, 1);
  }
  // fft::multiply uses dbl, outputs vector<long long> of rounded values
  // fft::multiply_mod might work for res.size() up to 2^21
  // typedef long double dbl;          =>        up to 2^25 (but takes a lot of memory)
};
```

fftfast

```
// FFT_MAXN = 2^k (2^18)
// fft_init() to precalc FFT_MAXN-th roots
// 0 Base !!!
#include<bits/stdc++.h>
using namespace std;
#define ll long long
#define db double
#define ldb long double
#define rep(i,a,b) for(int i=a;i<=b;i++)
#define per(i,b,a) for(int i=b;i>=a;i--)

const int maxn=3e5+105;
const int N = 3e5+5;
const ldb pi=acosl(-1.0);
const int mod=998244353;

const int FFT_MAXN=262144;
struct cp{
    ldb a,b;
    cp operator+(const cp &y) const{
        return (cp){a+y.a,b+y.b};
    }
    cp operator-(const cp &y) const{
        return (cp){a-y.a,b-y.b};
    }
    cp operator*(const cp &y) const{
        return (cp){a*y.a-b*y.b,a*y.b+b*y.a};
    }
    cp operator!() const{
        return (cp){a,-b};
    }
```

```
}nw[FFT_MAXN+1];

int bitrev[FFT_MAXN+1];

void dft(cp*a,int n,int flag=1){
    int d=0;
    while((1<<d)*n!=FFT_MAXN) d++;
    rep(i,0,n-1)
        if(i<(bitrev[i]>>d))
            swap(a[i],a[bitrev[i]>>d]);
    for(int l=2;l<=n;l<<=1){
        int del=FFT_MAXN/l*flag;
        for(int i=0;i<n;i+=l){
            cp *le=a+i,*ri=a+i+(l>>1),*w=flag==1?nw:nw+FFT_MAXN;
            rep(k,0,l/2-1){
                cp ne=*ri**w;
                *ri=*le-ne,*le=*le+ne;
                le++,ri++,w+=del;
            }
        }
    }
    if(flag!=1)
        rep(i,0,n-1) a[i].a/=n,a[i].b/=n;
}
void fft_init(){
    int L=0;
    while((1<<L)!=FFT_MAXN) L++;
    bitrev[0]=0;
    rep(i,1,FFT_MAXN)
        bitrev[i]=bitrev[i>>1]>>1|((i&1)<<(L-1));

    nw[0]=nw[FFT_MAXN]=(cp){1,0};
    rep(i,0,FFT_MAXN)
        nw[i]=(cp){cosl(2*pi/FFT_MAXN*i),sinl(2*pi/FFT_MAXN*i)};  //very slow
}

void convo(ldb*a,int n,ldb*b,int m,ldb*c){
    static cp f[FFT_MAXN>>1],g[FFT_MAXN>>1],t[FFT_MAXN>>1];
    int N=2;
    while(N<=n+m) N<<=1;
    rep(i,0,N-1)
        if(i&1){
            f[i>>1].b=(i<=n)?a[i]:0.0;
            g[i>>1].b=(i<=m)?b[i]:0.0;
        }else{
            f[i>>1].a=(i<=n)?a[i]:0.0;
            g[i>>1].a=(i<=m)?b[i]:0.0;
        }
    dft(f,N>>1);dft(g,N>>1);
    int del=FFT_MAXN/(N>>1);
    cp qua=(cp){0,0.25},one=(cp){1,0},four=(cp){4,0},*w=nw;
    rep(i,0,N/2-1){
```

```
        int j=i?(N>>1)-i:0;
        t[i]=(four*!(f[j]*g[j])-(!f[j]-f[i])*(!g[j]-g[i])*(one+*w))*qua;
        w+=del;
    }
    dft(t,N>>1,-1);
    rep(i,0,n+m)
        c[i]=(i&1)?t[i>>1].a:t[i>>1].b;
}
/*
    void mul(int *a,int *b,int n){// n<=N, 0<=a[i],b[i]<mo
     static cp f[N],g[N],t[N],r[N];
     int nn=2;while(nn<=n+n)nn<<=1;
     rep(i,0,nn){
         f[i]=(i<=n)?(cp){(db)(a[i]>>15),(db)(a[i]&32767)}:(cp){0,0};
         g[i]=(i<=n)?(cp){(db)(b[i]>>15),(db)(b[i]&32767)}:(cp){0,0};
     }
     swap(n,nn);
     dft(f,n,1);dft(g,n,1);
     rep(i,0,n){
         int j=i?n-i:0;
         t[i]=( (f[i]+!f[j])*(!g[j]-g[i]) + (!f[j]-f[i])*(g[i]+!g[j]) )*(cp){0,0.25};
         r[i]=(!f[j]-f[i])*(!g[j]-g[i])*(cp){-0.25,0} + (cp){0,0.25}*(f[i]+!f[j])*
(g[i]+!g[j]);
     }
     dft(t,n,-1); dft(r,n,-1);
     rep(i,0,n)a[i]=((ll(t[i].a+0.5)%mo<<15) + ll(r[i].a+0.5) + (ll(r[i].b+0.5)%mo<<30))%mo;
    }
*/
int n,m;
ldb f[maxn],g[maxn],h[maxn];
int main(){
    fft_init();
    cin>>n>>m;
    rep(i,0,n) cin>>f[i];
    rep(i,0,m) cin>>g[i];
    convo(f,n,g,m,h);
    rep(i,0,n+m) cout<<(ll)(h[i]+0.5)<<" \n"[i==n+m];
}
```

fftfast(original)

```
// FFT_MAXN = 2^k
// fft_init() to precalc FFT_MAXN-th roots

typedef long double db;
const int FFT_MAXN=262144;
const db pi=acos(-1.);
struct cp{
  db a,b;
  cp operator+(const cp&y)const{return (cp){a+y.a,b+y.b};}
  cp operator-(const cp&y)const{return (cp){a-y.a,b-y.b};}
```

```
  cp operator*(const cp&y)const{return (cp){a*y.a-b*y.b,a*y.b+b*y.a};}
  cp operator!()const{return (cp){a,-b};};
}nw[FFT_MAXN+1];int bitrev[FFT_MAXN];
void dft(cp*a,int n,int flag=1){
  int d=0;while((1<<d)*n!=FFT_MAXN)d++;
  rep(i,0,n)if(i<(bitrev[i]>>d))swap(a[i],a[bitrev[i]>>d]);
  for (int l=2;l<=n;l<<=1){
    int del=FFT_MAXN/l*flag;
    for (int i=0;i<n;i+=l){
      cp *le=a+i,*ri=a+i+(l>>1),*w=flag==1?nw:nw+FFT_MAXN;
      rep(k,0,l>>1){
        cp ne=*ri**w;
        *ri=*le-ne,*le=*le+ne;
        le++,ri++,w+=del;
      }
    }
  }
  if(flag!=1)rep(i,0,n)a[i].a/=n,a[i].b/=n;
}
void fft_init(){
  int L=0;while((1<<L)!=FFT_MAXN)L++;
  bitrev[0]=0;rep(i,1,FFT_MAXN)bitrev[i]=bitrev[i>>1]>>1|((i&1)<<(L-1));
  nw[0]=nw[FFT_MAXN]=(cp){1,0};
  rep(i,0,FFT_MAXN+1)nw[i]=(cp){cosl(2*pi/FFT_MAXN*i),sinl(2*pi/FFT_MAXN*i)}; //very slow
}

void convo(db*a,int n,db*b,int m,db*c){
  static cp f[FFT_MAXN>>1],g[FFT_MAXN>>1],t[FFT_MAXN>>1];
  int N=2;while(N<=n+m)N<<=1;
  rep(i,0,N)
    if(i&1){
      f[i>>1].b=(i<=n)?a[i]:0.0;
      g[i>>1].b=(i<=m)?b[i]:0.0;
    }else{
      f[i>>1].a=(i<=n)?a[i]:0.0;
      g[i>>1].a=(i<=m)?b[i]:0.0;
    }
  dft(f,N>>1);dft(g,N>>1);
  int del=FFT_MAXN/(N>>1);
  cp qua=(cp){0,0.25},one=(cp){1,0},four=(cp){4,0},*w=nw;
  rep(i,0,N>>1){
    int j=i?(N>>1)-i:0;
    t[i]=(four*!(f[j]*g[j])-(!f[j]-f[i])*(!g[j]-g[i])*(one+*w))*qua;
    w+=del;
  }
  dft(t,N>>1,-1);
  rep(i,0,n+m+1)c[i]=(i&1)?t[i>>1].a:t[i>>1].b;
}

void mul(int *a,int *b,int n){// n<=N, 0<=a[i],b[i]<mo
  static cp f[N],g[N],t[N],r[N];
  int nn=2;while(nn<=n+n)nn<<=1;
```

```
  rep(i,0,nn){
    f[i]=(i<=n)?(cp){(db)(a[i]>>15),(db)(a[i]&32767)}:(cp){0,0};
    g[i]=(i<=n)?(cp){(db)(b[i]>>15),(db)(b[i]&32767)}:(cp){0,0};
  }
  swap(n,nn);
  dft(f,n,1);dft(g,n,1);
  rep(i,0,n){
    int j=i?n-i:0;
    t[i]=( (f[i]+!f[j])*(!g[j]-g[i]) + (!f[j]-f[i])*(g[i]+!g[j]) )*(cp){0,0.25};
    r[i]=(!f[j]-f[i])*(!g[j]-g[i])*(cp){-0.25,0} + (cp){0,0.25}*(f[i]+!f[j])*(g[i]+!g[j]);
  }
  dft(t,n,-1); dft(r,n,-1);
  rep(i,0,n)a[i]=( (ll(t[i].a+0.5)%mo<<15) + ll(r[i].a+0.5) + (ll(r[i].b+0.5)%mo<<30) )%mo;
}
```

fftnew

```
namespace fft {

typedef double dbl;

struct num {
  dbl x, y;
  num() { x = y = 0; }
  num(dbl x_, dbl y_) : x(x_), y(y_) {}
};

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

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

const dbl PI = static_cast<dbl>(acosl(-1.0));

void ensure_base(int nbase) {
  if (nbase <= base) {
    return;
  }
  rev.resize(1 << nbase);
  for (int i = 0; i < (1 << nbase); i++) {
    rev[i] = (rev[i >> 1] >> 1) + ((i & 1) << (nbase - 1));
  }
  roots.resize(1 << nbase);
  while (base < nbase) {
    dbl angle = 2 * PI / (1 << (base + 1));
//      num z(cos(angle), sin(angle));
```

```
    for (int i = 1 << (base - 1); i < (1 << base); i++) {
      roots[i << 1] = roots[i];
//        roots[(i << 1) + 1] = roots[i] * z;
      dbl angle_i = angle * (2 * i + 1 - (1 << base));
      roots[(i << 1) + 1] = num(cos(angle_i), sin(angle_i));
    }
    base++;
  }
}

void fft(vector<num>& a, int n = -1) {
  if (n == -1) {
    n = (int) a.size();
  }
  assert((n & (n - 1)) == 0);
  int zeros = __builtin_ctz(n);
  ensure_base(zeros);
  int shift = base - zeros;
  for (int i = 0; i < n; i++) {
    if (i < (rev[i] >> shift)) {
      swap(a[i], a[rev[i] >> shift]);
    }
  }
  for (int k = 1; k < n; k <<= 1) {
    for (int i = 0; i < n; i += 2 * k) {
      for (int j = 0; j < k; j++) {
        num z = a[i + j + k] * roots[j + k];
        a[i + j + k] = a[i + j] - z;
        a[i + j] = a[i + j] + z;
      }
    }
  }
}

vector<num> fa, fb;

vector<int64_t> square(const vector<int>& a) {
  if (a.empty()) {
    return {};
  }
  int need = (int) a.size() + (int) a.size() - 1;
  int nbase = 1;
  while ((1 << nbase) < need) nbase++;
  ensure_base(nbase);
  int sz = 1 << nbase;
  if ((sz >> 1) > (int) fa.size()) {
    fa.resize(sz >> 1);
  }
  for (int i = 0; i < (sz >> 1); i++) {
    int x = (2 * i < (int) a.size() ? a[2 * i] : 0);
    int y = (2 * i + 1 < (int) a.size() ? a[2 * i + 1] : 0);
    fa[i] = num(x, y);
```

```
  }
  fft(fa, sz >> 1);
  num r(1.0 / (sz >> 1), 0.0);
  for (int i = 0; i <= (sz >> 2); i++) {
    int j = ((sz >> 1) - i) & ((sz >> 1) - 1);
    num fe = (fa[i] + conj(fa[j])) * num(0.5, 0);
    num fo = (fa[i] - conj(fa[j])) * num(0, -0.5);
    num aux = fe * fe + fo * fo * roots[(sz >> 1) + i] * roots[(sz >> 1) + i];
    num tmp = fe * fo;
    fa[i] = r * (conj(aux) + num(0, 2) * conj(tmp));
    fa[j] = r * (aux + num(0, 2) * tmp);
  }
  fft(fa, sz >> 1);
  vector<int64_t> res(need);
  for (int i = 0; i < need; i++) {
    res[i] = llround(i % 2 == 0 ? fa[i >> 1].x : fa[i >> 1].y);
  }
  return res;
}

vector<int64_t> multiply(const vector<int>& a, const vector<int>& b) {
  if (a.empty() || b.empty()) {
    return {};
  }
  if (a == b) {
    return square(a);
  }
  int need = (int) a.size() + (int) b.size() - 1;
  int nbase = 1;
  while ((1 << nbase) < need) nbase++;
  ensure_base(nbase);
  int sz = 1 << nbase;
  if (sz > (int) fa.size()) {
    fa.resize(sz);
  }
  for (int i = 0; i < sz; i++) {
    int x = (i < (int) a.size() ? a[i] : 0);
    int y = (i < (int) b.size() ? b[i] : 0);
    fa[i] = num(x, y);
  }
  fft(fa, sz);
  num r(0, -0.25 / (sz >> 1));
  for (int i = 0; i <= (sz >> 1); i++) {
    int j = (sz - i) & (sz - 1);
    num z = (fa[j] * fa[j] - conj(fa[i] * fa[i])) * r;
    fa[j] = (fa[i] * fa[i] - conj(fa[j] * fa[j])) * r;
    fa[i] = z;
  }
  for (int i = 0; i < (sz >> 1); i++) {
    num A0 = (fa[i] + fa[i + (sz >> 1)]) * num(0.5, 0);
    num A1 = (fa[i] - fa[i + (sz >> 1)]) * num(0.5, 0) * roots[(sz >> 1) + i];
    fa[i] = A0 + A1 * num(0, 1);
```

```
  }
  fft(fa, sz >> 1);
  vector<int64_t> res(need);
  for (int i = 0; i < need; i++) {
    res[i] = llround(i % 2 == 0 ? fa[i >> 1].x : fa[i >> 1].y);
  }
  return res;
}

vector<int> multiply_mod(const vector<int>& a, const vector<int>& b, int m) {
  if (a.empty() || b.empty()) {
    return {};
  }
  int eq = (a.size() == b.size() && a == b);
  int need = (int) a.size() + (int) b.size() - 1;
  int nbase = 0;
  while ((1 << nbase) < need) nbase++;
  ensure_base(nbase);
  int sz = 1 << nbase;
  if (sz > (int) fa.size()) {
    fa.resize(sz);
  }
  for (int i = 0; i < (int) a.size(); i++) {
    int x = (a[i] % m + m) % m;
    fa[i] = num(x & ((1 << 15) - 1), x >> 15);
  }
  fill(fa.begin() + a.size(), fa.begin() + sz, num {0, 0});
  fft(fa, sz);
  if (sz > (int) fb.size()) {
    fb.resize(sz);
  }
  if (eq) {
    copy(fa.begin(), fa.begin() + sz, fb.begin());
  } else {
    for (int i = 0; i < (int) b.size(); i++) {
      int x = (b[i] % m + m) % m;
      fb[i] = num(x & ((1 << 15) - 1), x >> 15);
    }
    fill(fb.begin() + b.size(), fb.begin() + sz, num {0, 0});
    fft(fb, sz);
  }
  dbl ratio = 0.25 / sz;
  num r2(0, -1);
  num r3(ratio, 0);
  num r4(0, -ratio);
  num r5(0, 1);
  for (int i = 0; i <= (sz >> 1); i++) {
    int j = (sz - i) & (sz - 1);
    num a1 = (fa[i] + conj(fa[j]));
    num a2 = (fa[i] - conj(fa[j])) * r2;
    num b1 = (fb[i] + conj(fb[j])) * r3;
    num b2 = (fb[i] - conj(fb[j])) * r4;
```

```
    if (i != j) {
      num c1 = (fa[j] + conj(fa[i]));
      num c2 = (fa[j] - conj(fa[i])) * r2;
      num d1 = (fb[j] + conj(fb[i])) * r3;
      num d2 = (fb[j] - conj(fb[i])) * r4;
      fa[i] = c1 * d1 + c2 * d2 * r5;
      fb[i] = c1 * d2 + c2 * d1;
    }
    fa[j] = a1 * b1 + a2 * b2 * r5;
    fb[j] = a1 * b2 + a2 * b1;
  }
  fft(fa, sz);
  fft(fb, sz);
  vector<int> res(need);
  for (int i = 0; i < need; i++) {
    int64_t aa = llround(fa[i].x);
    int64_t bb = llround(fb[i].x);
    int64_t cc = llround(fa[i].y);
    res[i] = static_cast<int>((aa + ((bb % m) << 15) + ((cc % m) << 30)) % m);
  }
  return res;
}

}  // namespace fft
```

FST

```
void fst(VI &a,bool inv) {
    for (int n=SZ(a),step=1;step<n;step*=2) {
        for (int i=0;i<n;i+=2*step) rep(j,i,i+step-1) {
            int &u=a[j],&v=a[j+step];
            tie(u,v)=
            inv?PII(v-u,u):PII(v,u+v); // AND
            inv?PII(v,u-v):PII(u+v,u); // OR
            PII(u+v,u-v); // XOR
        }
    }
    if (inv) for (auto &x : a) x/=SZ(a); // XOR only
}
VI conv(VI a,VI b) {
    fst(a,0),fst(b,0);
    rep(i,0,SZ(a)-1) a[i]=a[i]*b[i];
    fst(a,1); return a;
}
```

FWT

```
ll f[maxn],g[maxn],h[maxn];
int main() {
    for(int i=0;i<n;i++){
        for(int j=0;j<bit(n);j++){
            if((j&bit(i))==0){
                f[j]+=f[j+bit(i)];
                g[j]+=g[j+bit(i)];
            }
        }
    }
    for(int i=0;i<bit(n);i++){
        f[i]%=mod;
        g[i]%=mod;
        h[i]=f[i]*g[i]%mod;
    }
    for(int i=0;i<n;i++){
        for(int j=0;j<bit(n);j++){
            if((j&bit(i))==0)
                h[j]-=h[j+bit(i)];
        }
    }
    for(int i=0;i<bit(n);i++){
        h[i]%=mod;
        if(h[i]<0) h[i]+=mod;
    }

    ll ans=0;
    rep(i,0,bit(n)-1) ans^=h[i];
    cout<<ans<<'\n';
}
```

gauss

```
ll f[N][N];
ll v[N], a[N];
void gauss() {
    for (int i = 1; i <= n; i++) {
        for (int j = i; j <= n; j++) {
            if (f[j][i] > f[i][i]) {
                swap(v[i], v[j]);
                for (int k = 1; k <= n; k++)
                    swap(f[j][k], f[i][k]);
            }
        }
        for (int j = i + 1; j <= n; j++) {
            if (f[j][i]) {
                int delta = f[j][i] * fpow(f[i][i], mod - 2) % mod;
                for (int k = i; k <= n; k++) {
```

```
                    f[j][k] -= f[i][k] * delta % mod;
                    if (f[j][k] < 0)
                        f[j][k] += mod;
                }
                v[j] -= v[i] * delta % mod;
                if (v[j] < mod)
                    v[j] += mod;
            }
        }
    }
    for (int j = n; j > 0; j--) {
        for (int k = j + 1; k <= n; k++) {
            v[j] -= f[j][k] * a[k] % mod;
            if (v[j] < 0)
                v[j] += mod;
        }
        a[j] = v[j] * fpow(f[j][j], mod - 2) % mod;
    }
}
```

gauss(合数)

```
void gauss(int n) {
    int ans = 1;
    //rep(i,1,n) rep(j,1,n) p[i][j]%=mod;
    for (int i = 1; i <= n; i++) {
        for (int j = i + 1; j <= n; j++) {
            int x = i, y = j;
            while (p[x][i]) {
                int t = p[y][i] / p[x][i];
                for (int k = i; k <= n; k++)
                    p[y][k] = (p[y][k] - p[x][k] * t) % mod;
                swap(x, y);
            }
            if (x == i) {
                for (int k = i; k <= n; k++) swap(p[i][k], p[j][k]);
                ans = -ans;
            }
        }
    }
}
```

linear basis

```
struct linear_base {
    ll w[64];
    ll zero = 0;
    ll tot = -1;
    void clear() {
        rep(i, 0, 63) w[i] = 0;
```

```
        zero = 0;
        tot = -1;
    }
    void insert(ll x) {
        for (int i = 62; i >= 0; i--) {
            if (x & bit(i))
                if (!w[i]) {w[i] = x; return;}
                else x ^= w[i];
        }
        zero++;
    }
    void build() {
        rep(i, 0, 63) rep(j, 0, i - 1) {
            if (w[i]&bit(j)) w[i] ^= w[j];
        }
        for (int i = 0; i <= 62; i++) {
            if (w[i] != 0) w[++tot] = w[i];
        }
    }
    ll qmax() {
        ll res = 0;
        for (int i = 62; i >= 0; i--) {
            res = max(res, res ^ w[i]);
        }
        return res;
    }
    bool check(ll x) {
        for (int i = 62; i >= 0; i--) {
            if (x & bit(i))
                if (!w[i]) return false;
                else x ^= w[i];
        }
        return true;
    }
    ll query(ll k) {
        ll res = 0;
        // if(zero) k-=1;
        // if(k>=bit(tot)) return -1;
        for (int i = tot; i >= 0; i--) {
            if (k & bit(i)) {
                res = max(res, res ^ w[i]);
            } else {
                res = min(res, res ^ w[i]);
            }
        }
        return res;
    }
};
```

lucas

```
ll fac[maxn], fnv[maxn];

ll binom(ll a, ll b) {
    if (b > a || b < 0) return 0;
    return fac[a] * fnv[a - b] % p * fnv[b] % p;
}

ll lucas(ll a, ll b, ll p) {
    ll ans = 1;
    while (a > 0 || b > 0) {
        ans = (ans * binom(a % p, b % p)) % p;
        a /= p, b /= p;
    }
    return ans;
}

int main() {
    cin >> p >> T;
    fac[0] = 1;
    rep(i, 1, p - 1) fac[i] = fac[i - 1] * i % p;
    fnv[p - 1] = powmod(fac[p - 1], p - 2, p);
    per(i, p - 2, 0) fnv[i] = fnv[i + 1] * (i + 1) % p;
    assert(fnv[0] == 1);
}
```

matrix

```
struct matrix {
    int r, c;
    vector<vector<ll>> a;
    matrix(int x, int y): r(x), c(y) {
        a = vector<vector<ll>>(r + 1, vector<ll>(c + 1));
    }
    matrix friend operator *(const matrix &x, const matrix &y) {
        matrix res(x.r, y.c);
        assert(x.c == y.r);
        for (int i = 1; i <= res.r; i++)
            for (int j = 1; j <= res.c; j++)
                for (int k = 1; k <= x.c; k++) {
                    res.a[i][j] += x.a[i][k] * y.a[k][j] % mod;
                    if (res.a[i][j] >= mod)
                        res.a[i][j] -= mod;
                }
        return res;
    }
    matrix friend matrixpow(matrix x, ll b) {
        matrix res(x.r, x.c);
        assert(x.r == x.c);
```

```
        rep(i, 1, x.r) res.a[i][i] = 1;
        while (b) {
            if (b & 1) res = res * x;
            b >>= 1;
            x = x * x;
        }
        return res;
    }
};
```

matrixfast

```
Description: Basic operations on square matrices.
Usage: Matrix<int, 3> A;
A.d = {{{{1, 2, 3}}, {{4, 5, 6}}, {{7, 8, 9}}}};
vector<int> vec = {1, 2, 3};
vec = (A^N) * vec;

template<class T, int N> struct Matrix {
    typedef Matrix M;
    array<array<T, N>, N> d{};
    M operator*(const M& m) const {
        M a;
        rep(i, 0, N) rep(j, 0, N)
        rep(k, 0, N) a.d[i][j] += d[i][k] * m.d[k][j];
        return a;
    }
    vector<T> operator*(const vector<T>& vec) const {
        vector<T> ret(N);
        rep(i, 0, N) rep(j, 0, N) ret[i] += d[i][j] * vec[j];
        return ret;
    }
    M operator^(ll p) const {
        assert(p >= 0);
        M a, b(*this);
        rep(i, 0, N) a.d[i][i] = 1;
        while (p) {
            if (p & 1) a = a * b;
            b = b * b;
            p >>= 1;
        }
        return a;
    }
};
```

MillerRabbin&pollard&modmul

```
/*ModMulLL.h
Description: Calculate a·b mod c (or a
b mod c) for 0 ≤ a, b ≤ c ≤ 7.2·10^18
Time: O (1) for modmul, O (log b) for modpow*/
/*ull modmul(ull a, ull b, ull M) {
    ll ret = a * b - M * ull(1.L / M * a * b);
    return ret + M * (ret < 0) - M * (ret >= (ll)M);
}
ull modpow(ull b, ull e, ull mod) {
    ull ans = 1;
    for (; e; b = modmul(b, b, mod), e /= 2)
        if (e & 1) ans = modmul(ans, b, mod);
    return ans;
}*/
ll modmul(ll a, ll b, ll m) {
    a %= m, b %= m;
    ll d = ((ldb)a * b / m);
    d = a * b - d * m;
    if (d >= m) d -= m;
    if (d < 0) d += m;
    return d;
}
ll modpow(ll a, ll b, ll p) {
    ll ans = 1;
    while (b) {
        if (b & 1) ans = modmul(ans, a, p);
        a = modmul(a, a, p); b >>= 1;
    } return ans;
}
/*MillerRabin.h
Description: Deterministic Miller-Rabin primality test. Guaranteed to
work for numbers up to 7 · 1018; for larger numbers, use Python and ex•tend A randomly.
Time: 7 times the complexity of a^b mod c.*/
bool isPrime(ll n) {
    if (n < 2 || n % 6 % 4 != 1) return (n | 1) == 3;
    ll A[] = {2, 325, 9375, 28178, 450775, 9780504, 1795265022},
              s = __builtin_ctzll(n - 1), d = n >> s;
    for (ll a : A) { // ^ count trailing zeroes
        ll p = modpow(a % n, d, n), i = s;
        while (p != 1 && p != n - 1 && a % n && i--)
            p = modmul(p, p, n);
        if (p != n - 1 && i != s) return 0;
    }
    return 1;
}
/*Factor.h
Description: Pollard-rho randomized factorization algorithm. Returns
prime factors of a number, in arbitrary order (e.g. 2299 -> {11, 19, 11}).
Time: O(n^1/4), less for numbers with small factors.*/
ll pollard(ll n) {
```

```
    auto f = [n](ll x) { return modmul(x, x, n) + 1; };
    ll x = 0, y = 0, t = 30, prd = 2, i = 1, q;
    while (t++ % 40 || __gcd(prd, n) == 1) {
        if (x == y) x = ++i, y = f(x);
        if ((q = modmul(prd, max(x, y) - min(x, y), n))) prd = q;
        x = f(x), y = f(f(y));
    }
    return __gcd(prd, n);
}
vector<ll> factor(ll n) {
    if (n == 1) return {};
    if (isPrime(n)) return {n};
    ll x = pollard(n);
    auto l = factor(x), r = factor(n / x);
    l.insert(l.end(), all(r));
    return l;
}
```

ntt(polynomial)

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

const int mod = 998244353;

inline void add(int &x, int y) {
  x += y;
  if (x >= mod) {
    x -= mod;
  }
}

inline void sub(int &x, int y) {
  x -= y;
  if (x < 0) {
    x += mod;
  }
}

inline int mul(int x, int y) {
  return (long long) x * y % mod;
}

inline int power(int x, int y) {
  int res = 1;
  for (; y; y >>= 1, x = mul(x, x)) {
    if (y & 1) {
      res = mul(res, x);
    }
  }
  return res;
```

```
}

inline int inv(int a) {
  a %= mod;
  if (a < 0) {
    a += mod;
  }
  int b = mod, u = 0, v = 1;
  while (a) {
    int t = b / a;
    b -= t * a;
    swap(a, b);
    u -= t * v;
    swap(u, v);
  }
  if (u < 0) {
    u += mod;
  }
  return u;
}

namespace ntt {
int base = 1, root = -1, max_base = -1;
vector<int> rev = {0, 1}, roots = {0, 1};

void init() {
  int temp = mod - 1;
  max_base = 0;
  while (temp % 2 == 0) {
    temp >>= 1;
    ++max_base;
  }
  root = 2;
  while (true) {
    if (power(root, 1 << max_base) == 1 && power(root, 1 << (max_base - 1)) != 1) {
      break;
    }
    ++root;
  }
}

void ensure_base(int nbase) {
  if (max_base == -1) {
    init();
  }
  if (nbase <= base) {
    return;
  }
  assert(nbase <= max_base);
  rev.resize(1 << nbase);
  for (int i = 0; i < 1 << nbase; ++i) {
    rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (nbase - 1));
```

```
  }
  roots.resize(1 << nbase);
  while (base < nbase) {
    int z = power(root, 1 << (max_base - 1 - base));
    for (int i = 1 << (base - 1); i < 1 << base; ++i) {
      roots[i << 1] = roots[i];
      roots[i << 1 | 1] = mul(roots[i], z);
    }
    ++base;
  }
}

void dft(vector<int> &a) {
  int n = a.size(), zeros = __builtin_ctz(n);
  ensure_base(zeros);
  int shift = base - zeros;
  for (int i = 0; i < n; ++i) {
    if (i < rev[i] >> shift) {
      swap(a[i], a[rev[i] >> shift]);
    }
  }
  for (int i = 1; i < n; i <<= 1) {
    for (int j = 0; j < n; j += i << 1) {
      for (int k = 0; k < i; ++k) {
        int x = a[j + k], y = mul(a[j + k + i], roots[i + k]);
        a[j + k] = (x + y) % mod;
        a[j + k + i] = (x + mod - y) % mod;
      }
    }
  }
}

vector<int> multiply(vector<int> a, vector<int> b) {
  int need = a.size() + b.size() - 1, nbase = 0;
  while (1 << nbase < need) {
    ++nbase;
  }
  ensure_base(nbase);
  int sz = 1 << nbase;
  a.resize(sz);
  b.resize(sz);
  bool equal = a == b;
  dft(a);
  if (equal) {
    b = a;
  } else {
    dft(b);
  }
  int inv_sz = inv(sz);
  for (int i = 0; i < sz; ++i) {
    a[i] = mul(mul(a[i], b[i]), inv_sz);
  }
```

```
  reverse(a.begin() + 1, a.end());
  dft(a);
  a.resize(need);
  return a;
}

vector<int> inverse_new(const vector<int> &a) {
  assert(!a.empty());
  int n = (int) a.size();
  vector<int> b = {inv(a[0])};
  while ((int) b.size() < n) {
    vector<int> x(a.begin(), a.begin() + min(a.size(), b.size() << 1));
    x.resize(b.size() << 1);
    b.resize(b.size() << 1);
    vector<int> c = b;
    // NTT<T>::fft(c);
    // NTT<T>::fft(x);
    dft(c);
    dft(x);
    // Modular<T> inv = 1 / static_cast<Modular<T>>((int) x.size());
    int inv_sz = inv((int)x.size());
    for (int i = 0; i < (int) x.size(); i++) {
      // x[i] *= c[i] * inv;
      x[i] = mul(x[i], mul(c[i], inv_sz));
    }
    reverse(x.begin() + 1, x.end());
    // NTT<T>::fft(x);
    dft(x);
    rotate(x.begin(), x.begin() + (x.size() >> 1), x.end());
    fill(x.begin() + (x.size() >> 1), x.end(), 0);
    // NTT<T>::fft(x);
    dft(x);
    for (int i = 0; i < (int) x.size(); i++) {
      // x[i] *= c[i] * inv;
      x[i] = mul(x[i], mul(c[i], inv_sz));
    }
    reverse(x.begin() + 1, x.end());
    // NTT<T>::fft(x);
    dft(x);
    for (int i = 0; i < ((int) x.size() >> 1); i++) {
      // b[i + ((int) x.size() >> 1)] = -x[i];
      int t = 0; sub(t, x[i]);
      b[i + ((int) x.size() >> 1)] = t;
    }
  }
  b.resize(n);
  return b;
}

vector<int> inverse(vector<int> a) {
  int n = a.size(), m = (n + 1) >> 1;
  if (n == 1) {
```

```
    return vector<int>(1, inv(a[0]));
  } else {
    vector<int> b = inverse(vector<int>(a.begin(), a.begin() + m));
    int need = n << 1, nbase = 0;
    while (1 << nbase < need) {
      ++nbase;
    }
    ensure_base(nbase);
    int sz = 1 << nbase;
    a.resize(sz);
    b.resize(sz);
    dft(a);
    dft(b);
    int inv_sz = inv(sz);
    for (int i = 0; i < sz; ++i) {
      a[i] = mul(mul(mod + 2 - mul(a[i], b[i]), b[i]), inv_sz);
    }
    reverse(a.begin() + 1, a.end());
    dft(a);
    a.resize(n);
    return a;
  }
}
}

using ntt::multiply;
using ntt::inverse;

vector<int>& operator += (vector<int> &a, const vector<int> &b) {
  if (a.size() < b.size()) {
    a.resize(b.size());
  }
  for (int i = 0; i < b.size(); ++i) {
    add(a[i], b[i]);
  }
  return a;
}

vector<int> operator + (const vector<int> &a, const vector<int> &b) {
  vector<int> c = a;
  return c += b;
}

vector<int>& operator -= (vector<int> &a, const vector<int> &b) {
  if (a.size() < b.size()) {
    a.resize(b.size());
  }
  for (int i = 0; i < b.size(); ++i) {
    sub(a[i], b[i]);
  }
  return a;
}
```

```
vector<int> operator - (const vector<int> &a, const vector<int> &b) {
  vector<int> c = a;
  return c -= b;
}

vector<int>& operator *= (vector<int> &a, const vector<int> &b) {
  if (min(a.size(), b.size()) < 128) {
    vector<int> c = a;
    a.assign(a.size() + b.size() - 1, 0);
    for (int i = 0; i < c.size(); ++i) {
      for (int j = 0; j < b.size(); ++j) {
        add(a[i + j], mul(c[i], b[j]));
      }
    }
  } else {
    a = multiply(a, b);
  }
  return a;
}

vector<int> operator * (const vector<int> &a, const vector<int> &b) {
  vector<int> c = a;
  return c *= b;
}

vector<int>& operator /= (vector<int> &a, const vector<int> &b) {
  int n = a.size(), m = b.size();
  if (n < m) {
    a.clear();
  } else {
    vector<int> c = b;
    reverse(a.begin(), a.end());
    reverse(c.begin(), c.end());
    c.resize(n - m + 1);
    a *= inverse(c);
    a.erase(a.begin() + n - m + 1, a.end());
    reverse(a.begin(), a.end());
  }
  return a;
}

vector<int> operator / (const vector<int> &a, const vector<int> &b) {
  vector<int> c = a;
  return c /= b;
}

vector<int>& operator %= (vector<int> &a, const vector<int> &b) {
  int n = a.size(), m = b.size();
  if (n >= m) {
    vector<int> c = (a / b) * b;
    a.resize(m - 1);
```

```
    for (int i = 0; i < m - 1; ++i) {
      sub(a[i], c[i]);
    }
  }
  return a;
}

vector<int> operator % (const vector<int> &a, const vector<int> &b) {
  vector<int> c = a;
  return c %= b;
}

vector<int> derivative(const vector<int> &a) {
  int n = a.size();
  vector<int> b(n - 1);
  for (int i = 1; i < n; ++i) {
    b[i - 1] = mul(a[i], i);
  }
  return b;
}

vector<int> primitive(const vector<int> &a) {
  int n = a.size();
  vector<int> b(n + 1), invs(n + 1);
  for (int i = 1; i <= n; ++i) {
    invs[i] = i == 1 ? 1 : mul(mod - mod / i, invs[mod % i]);
    b[i] = mul(a[i - 1], invs[i]);
  }
  return b;
}

vector<int> logarithm(const vector<int> &a) {
  vector<int> b = primitive(derivative(a) * inverse(a));
  b.resize(a.size());
  return b;
}

vector<int> exponent(const vector<int> &a) {
  vector<int> b(1, 1);
  while (b.size() < a.size()) {
    vector<int> c(a.begin(), a.begin() + min(a.size(), b.size() << 1));
    add(c[0], 1);
    vector<int> old_b = b;
    b.resize(b.size() << 1);
    c -= logarithm(b);
    c *= old_b;
    for (int i = b.size() >> 1; i < b.size(); ++i) {
      b[i] = c[i];
    }
  }
  b.resize(a.size());
  return b;
```

```
}

vector<int> power(vector<int> a, int m) {
  int n = a.size(), p = -1;
  vector<int> b(n);
  for (int i = 0; i < n; ++i) {
    if (a[i]) {
      p = i;
      break;
    }
  }
  if (p == -1) {
    b[0] = !m;
    return b;
  }
  if ((long long) m * p >= n) {
    return b;
  }
  int mu = power(a[p], m), di = inv(a[p]);
  vector<int> c(n - m * p);
  for (int i = 0; i < n - m * p; ++i) {
    c[i] = mul(a[i + p], di);
  }
  c = logarithm(c);
  for (int i = 0; i < n - m * p; ++i) {
    c[i] = mul(c[i], m);
  }
  c = exponent(c);
  for (int i = 0; i < n - m * p; ++i) {
    b[i + m * p] = mul(c[i], mu);
  }
  return b;
}

vector<int> sqrt(const vector<int> &a) {
  vector<int> b(1, 1);
  while (b.size() < a.size()) {
    vector<int> c(a.begin(), a.begin() + min(a.size(), b.size() << 1));
    vector<int> old_b = b;
    b.resize(b.size() << 1);
    c *= inverse(b);
    for (int i = b.size() >> 1; i < b.size(); ++i) {
      b[i] = mul(c[i], (mod + 1) >> 1);
    }
  }
  b.resize(a.size());
  return b;
}

vector<int> multiply_all(int l, int r, vector<vector<int>> &all) {
  if (l > r) {
    return vector<int>();
```

```
  } else if (l == r) {
    return all[l];
  } else {
    int y = (l + r) >> 1;
    return multiply_all(l, y, all) * multiply_all(y + 1, r, all);
  }
}

vector<int> evaluate(const vector<int> &f, const vector<int> &x) {
  int n = x.size();
  if (!n) {
    return vector<int>();
  }
  vector<vector<int>> up(n * 2);
  for (int i = 0; i < n; ++i) {
    up[i + n] = vector<int> {(mod - x[i]) % mod, 1};
  }
  for (int i = n - 1; i; --i) {
    up[i] = up[i << 1] * up[i << 1 | 1];
  }
  vector<vector<int>> down(n * 2);
  down[1] = f % up[1];
  for (int i = 2; i < n * 2; ++i) {
    down[i] = down[i >> 1] % up[i];
  }
  vector<int> y(n);
  for (int i = 0; i < n; ++i) {
    y[i] = down[i + n][0];
  }
  return y;
}

vector<int> interpolate(const vector<int> &x, const vector<int> &y) {
  int n = x.size();
  vector<vector<int>> up(n * 2);
  for (int i = 0; i < n; ++i) {
    up[i + n] = vector<int> {(mod - x[i]) % mod, 1};
  }
  for (int i = n - 1; i; --i) {
    up[i] = up[i << 1] * up[i << 1 | 1];
  }
  vector<int> a = evaluate(derivative(up[1]), x);
  for (int i = 0; i < n; ++i) {
    a[i] = mul(y[i], inv(a[i]));
  }
  vector<vector<int>> down(n * 2);
  for (int i = 0; i < n; ++i) {
    down[i + n] = vector<int>(1, a[i]);
  }
  for (int i = n - 1; i; --i) {
    down[i] = down[i << 1] * up[i << 1 | 1] + down[i << 1 | 1] * up[i << 1];
  }
```

```
  return down[1];
}

int main() {

}
```

Poly

```
mt19937_64 rng(chrono::steady_clock::now().time_since_epoch().count());

using uint = unsigned int;
ll myRand(ll B) { return (ull)rng() % B; }
const uint MOD = 998244353;
template<uint mod = MOD> struct mint { // 1000000007  1000000009
    uint x;

    mint() : x(0) {}
    mint(ll _x) {
        _x %= mod;
        if (_x < 0) _x += mod;
        x = _x;
    }

    mint& operator += (const mint &a) {
        x += a.x;
        if (x >= mod) x -= mod;
        return *this;
    }
    mint& operator -= (const mint &a) {
        x += mod - a.x;
        if (x >= mod) x -= mod;
        return *this;
    }
    mint& operator *= (const mint &a) {
        x = (ull)x * a.x % mod;
        return *this;
    }
    mint pow(ll pw) const {
        mint res = 1;
        mint cur = *this;
        while (pw) {
            if (pw & 1) res *= cur;
            cur *= cur;
            pw >>= 1;
        }
        return res;
    }
    mint inv() const {
        assert(x != 0);
        uint t = x;
```

```
    uint res = 1;
    while (t != 1) {
        uint z = mod / t;
        res = (ull)res * (mod - z) % mod;
        t = mod - t * z;
    }
    return res;
}
mint& operator /= (const mint &a) {
    return *this *= a.inv();
}
mint operator + (const mint &a) const {
    return mint(*this) += a;
}
mint operator - (const mint &a) const {
    return mint(*this) -= a;
}
mint operator * (const mint &a) const {
    return mint(*this) *= a;
}
mint operator / (const mint &a) const {
    return mint(*this) /= a;
}

bool sqrt(mint &res) const {
    if (mod == 2 || x == 0) {
        res = *this;
        return true;
    }
    if (pow((mod - 1) / 2) != 1) return false;
    if (mod % 4 == 3) {
        res = pow((mod + 1) / 4);
        return true;
    }
    int pw = (mod - 1) / 2;
    int K = 30;
    while ((1 << K) > pw) K--;
    while (true) {
        mint t = myRand(mod);
        mint a = 0, b = 0, c = 1;
        for (int k = K; k >= 0; k--) {
            a = b * b;
            b = b * c * 2;
            c = c * c + a * *this;
            if (((pw >> k) & 1) == 0) continue;
            a = b;
            b = b * t + c;
            c = c * t + a * *this;
        }
        if (b == 0) continue;
        c -= 1;
        c *= mint() - b.inv();
```

```
            if (c * c == *this) {
                res = c;
                return true;
            }
        }
        assert(false);
    }

    bool operator == (const mint &a) const {
        return x == a.x;
    }
    bool operator != (const mint &a) const {
        return x != a.x;
    }
    bool operator < (const mint &a) const {
        return x < a.x;
    }
};
template<uint mod = MOD> struct Factorials {
    using Mint = mint<mod>;
    vector<Mint> f, fi;

    Factorials() : f(), fi() {}
    Factorials(int n) {
        n += 10;
        f = vector<Mint>(n);
        fi = vector<Mint>(n);
        f[0] = 1;
        for (int i = 1; i < n; i++)
            f[i] = f[i - 1] * i;
        fi[n - 1] = f[n - 1].inv();
        for (int i = n - 1; i > 0; i--)
            fi[i - 1] = fi[i] * i;
    }

    Mint C(int n, int k) {
        if (k < 0 || k > n) return 0;
        return f[n] * fi[k] * fi[n - k];
    }
};
template<uint mod = MOD> struct Powers {
    using Mint = mint<mod>;
    vector<Mint> p, pi;

    Powers() : p(), pi() {}
    Powers(int n, Mint x) {
        n += 10;
        if (x == 0) {
            p = vector<Mint>(n);
            p[0] = 1;
        } else {
            p = vector<Mint>(n);
```

```
            pi = vector<Mint>(n);
            p[0] = pi[0] = 1;
            Mint xi = x.inv();
            for (int i = 1; i < n; i++) {
                p[i] = p[i - 1] * x;
                pi[i] = pi[i - 1] * xi;
            }
        }
    }

    Mint pow(int n) {
        if (n >= 0)
            return p[n];
        else
            return pi[-n];
    }
};
template<uint mod = MOD> struct Inverses {
    using Mint = mint<mod>;
    vector<Mint> ii;

    Inverses() : ii() {}
    Inverses(int n) {
        n += 10;
        ii = vector<Mint>(n);
        ii[1] = 1;
        for (int x = 2; x < n; x++)
            ii[x] = Mint() - ii[mod % x] * (mod / x);
    }

    Mint inv(Mint x) {
        assert(x != 0);
        uint t = x.x;
        uint res = 1;
        while (t >= (int)ii.size()) {
            uint z = mod / t;
            res = (ull)res * (mod - z) % mod;
            t = mod - t * z;
        }
        return ii[t] * res;
    }
};
using Mint = mint<>;

const int LOG = 20; // CHECK!!!!
Powers W;
vector<int> binRev;
void initFFT() {
    binRev = vector<int>((1 << LOG) + 3, 0);
    Mint w = 2;
    while (true) {
        Mint x = w;
```

```
        for (int i = 1; i < LOG; i++)
            x *= x;
        if (x == -1) break;
        w += 1;
    }
    W = Powers(1 << LOG, w);
    for (int mask = 1; mask < (1 << LOG); mask++) {
        binRev[mask] = (binRev[mask >> 1] >> 1) ^ ((mask & 1) << (LOG - 1));
    }
}

template<unsigned mod = MOD> struct Poly {
    using Mint = mint<mod>;
    vector<Mint> a;

    Poly() : a() {}
    Poly(vector<Mint> _a) {
        a = _a;
        while (!a.empty() && a.back() == 0) a.pop_back();
    }

    void print(int n = -1) {
        if (n == -1) n = (int)a.size();
        for (int i = 0; i < n; i++)
            printf("%u ", at(i).x);
        printf("\n");
    }
    /*void eprint() {
        eprintf("[");
        for (int i = 0; i < (int)a.size(); i++)
            eprintf("%u ", a[i].x);
        eprintf("]\n");
    }*/

    static void fft(vector<Mint> &A) {
        int L = (int)A.size();
        assert((L & (L - 1)) == 0);
        int k = 0;
        while ((1 << k) < L) k++;
        for (int i = 0; i < L; i++) {
            int x = binRev[i] >> (LOG - k);
            if (i < x) swap(A[i], A[x]);
        }
        for (int lvl = 0; lvl < k; lvl++) {
            int len = 1 << lvl;
            for (int st = 0; st < L; st += (len << 1))
                for (int i = 0; i < len; i++) {
                    Mint x = A[st + i], y = A[st + len + i] * W.pow(i << (LOG - 1 - lvl));
                    A[st + i] = x + y;
                    A[st + len + i] = x - y;
                }
        }
```

```
    }

    Mint& operator [] (const int i) {
        assert(0 <= i && i <= deg());
        return a[i];
    }
    Mint at(const int i) const {
        if (i < 0 || i > deg()) return 0;
        return a[i];
    }
    int deg() const { // deg of polynomial 0 is -1
        return (int)a.size() - 1;
    }
    Mint eval(const Mint &x) const {
        Mint res = 0;
        for (int i = deg(); i >= 0; i--)
            res = res * x + a[i];
        return res;
    }
    Poly substr(const int &l, const int &r) const {
        vector<Mint> f(r - l);
        for (int i = l; i < r; i++)
            f[i - l] = at(i);
        return f;
    }

    Poly& operator += (const Poly &A) {
        if (deg() < A.deg()) a.resize(A.a.size());
        for (int i = 0; i <= A.deg(); i++)
            a[i] += A.a[i];
        while (!a.empty() && a.back() == 0) a.pop_back();
        return *this;
    }
    Poly& operator -= (const Poly &A) {
        if (deg() < A.deg()) a.resize(A.a.size());
        for (int i = 0; i <= A.deg(); i++)
            a[i] -= A.a[i];
        while (!a.empty() && a.back() == 0) a.pop_back();
        return *this;
    }
    Poly& operator *= (const Mint &k) {
        if (k == 0) a.clear();
        for (Mint &x : a) x *= k;
        return *this;
    }
    Poly& operator /= (const Mint &k) {
        Mint ki = k.inv();
        for (Mint &x : a) x *= ki;
        return *this;
    }
    Poly operator + (const Poly &A) const {
        return Poly(*this) += A;
```

```
    }
    Poly operator - (const Poly &A) const {
        return Poly(*this) -= A;
    }
    Poly operator * (const Mint &k) const {
        return Poly(*this) *= k;
    }
    Poly operator / (const Mint &k) const {
        return Poly(*this) /= k;
    }

    Poly& operator *= (const Poly &A) {
        if (a.empty() || A.a.empty()) {
            a.clear();
            return *this;
        }
        int nd = deg() + A.deg();

        if (deg() < LOG || A.deg() < LOG) {
            vector<Mint> res(nd + 1, 0);
            for (int i = 0; i <= deg(); i++)
                for (int j = 0; j <= A.deg(); j++)
                    res[i + j] += a[i] * A.a[j];
            return *this = Poly(res);
        }

        int k = 0;
        while ((1 << k) <= nd) k++;
        int L = 1 << k;
        vector<Mint> f = a, g = A.a;
        f.resize(L, 0);
        g.resize(L, 0);
        fft(f);
        fft(g);
        for (int i = 0; i < L; i++)
            f[i] *= g[i];
        fft(f);
        reverse(f.begin() + 1, f.end());
        return *this = (Poly(f) / L);
    }
    Poly operator * (const Poly &A) const {
        return Poly(*this) *= A;
    }

    Poly inv(int n) const {
        assert(deg() >= 0 && at(0) != 0);
        if (n <= 0) return Poly();
        vector<Mint> res(n);
        res[0] = a[0].inv();
        vector<Mint> f, g;
        for (int L = 1; L < n; L <<= 1) {
            f = vector<Mint>(2 * L);
```

```
            g = vector<Mint>(2 * L);
            for (int i = 0; i < 2 * L && i <= deg(); i++)
                f[i] = a[i];
            for (int i = 0; i < L; i++)
                g[i] = res[i];
            fft(f);
            fft(g);
            for (int i = 0; i < 2 * L; i++)
                f[i] *= g[i];
            fft(f);
            reverse(f.begin() + 1, f.end());
            for (int i = 0; i < L; i++)
                f[i] = 0;
            for (int i = L; i < 2 * L; i++)
                f[i] = Mint() - f[i];
            fft(f);
            for (int i = 0; i < 2 * L; i++)
                f[i] *= g[i];
            fft(f);
            reverse(f.begin() + 1, f.end());
            Mint Li = Mint(2 * L).inv();
            Li *= Li;
            for (int i = L; i < 2 * L && i < n; i++)
                res[i] = f[i] * Li;
        }
        return res;
    }

    static vector<Mint> div_stupid(vector<Mint> A, vector<Mint> B) {
        int n = (int)A.size(), m = (int)B.size();
        Mint Bi = B.back().inv();
        for (auto &x : B) x *= Bi;
        vector<Mint> C(n - m + 1);
        for (int i = n; i >= m; i--) {
            C[i - m] = A[i - 1] * Bi;
            for (int j = 0; j < m; j++)
                A[i - m + j] -= B[j] * A[i - 1];
        }
        return C;
    }
    Poly& operator /= (const Poly &A) {
        int d1 = deg(), d2 = A.deg();
        assert(d2 >= 0);
        if (d1 < d2) return *this = Poly();
        if (d2 < 4 * LOG || d1 - d2 < 4 * LOG)
            return *this = div_stupid(a, A.a);
        vector<Mint> f = a, g = A.a;
        reverse(all(f));
        reverse(all(g));
        Poly H = Poly(vector<Mint>(f.begin(), f.begin() + d1 - d2 + 1)) * Poly(g).inv(d1 - d2
+ 1);
        vector<Mint> t = vector<Mint>(H.a.begin(), H.a.begin() + d1 - d2 + 1);
```

```
        reverse(all(t));
        return *this = t;
    }
    Poly operator / (const Poly &A) const {
        return Poly(*this) /= A;
    }
    Poly& operator %= (const Poly &A) {
        assert(A.deg() >= 0);
        if (deg() < A.deg()) return *this;
        return *this -= A * (*this / A);
    }
    Poly operator % (const Poly &A) const {
        return Poly(*this) %= A;
    }

    Poly derivate() const {
        int n = deg();
        if (n <= 0) return Poly();
        vector<Mint> f(n);
        for (int i = 0; i < n; i++)
            f[i] = a[i + 1] * (i + 1);
        return f;
    }
    Poly integrate() const {
        int n = deg();
        if (n < 0) return Poly();
        n += 2;
        vector<Mint> f(n);
        Inverses I = Inverses(n);
        for (int i = 1; i < n; i++)
            f[i] = a[i - 1] * I.inv(i);
        return f;
    }
    Poly log(int n) const {
        if (n <= 1) return Poly();
        assert(deg() >= 0 && at(0) == 1);
        return (derivate() * inv(n)).substr(0, n - 1).integrate();
    }
    Poly exp(int n) const {
        if (n <= 0) return Poly();
        if (deg() < 0) return Poly({1});
        assert(at(0) == 0);
        vector<Mint> res(n);
        res[0] = 1;
        vector<Mint> f, g;
        for (int L = 1; L < n; L <<= 1) {
            f = vector<Mint>(2 * L);
            g = vector<Mint>(2 * L);
            Poly LG = Poly(vector<Mint>(res.begin(), res.begin() + L)).log(2 * L);
            for (int i = 0; i < L; i++)
                assert(at(i) == LG.at(i));
            for (int i = 0; i < L; i++) {
```

```
                f[i] = res[i];
                g[i] = at(L + i) - LG.at(L + i);
            }
            fft(f);
            fft(g);
            for (int i = 0; i < 2 * L; i++)
                f[i] *= g[i];
            fft(f);
            reverse(f.begin() + 1, f.end());
            Mint Li = Mint(2 * L).inv();
            for (int i = L; i < 2 * L && i < n; i++)
                res[i] = f[i - L] * Li;
        }
        return res;
    }
    Poly sqr(int n) const {
        return (*this * *this).substr(0, n);
    }
    Poly pow_(Mint k, int n) const { // k can be non-negative rational (k = 1/2 is sqrt), but
assert(a[0] == 1);
        if (deg() < 0 || n <= 0) return Poly();
        return (log(n) * k).exp(n);
    }
    Poly pow(ll k, int n) const { // k is non-negative integer
        if (n <= 0) return Poly();
        if (k == 0) return Poly({1});
        if (k == 1) return substr(0, n);
        if (k == 2) return sqr(n);
        if (k < LOG) {
            Poly cur = substr(0, n);
            Poly res = Poly({1});
            while (k) {
                if (k & 1) res = (res * cur).substr(0, n);
                cur = cur.sqr(n);
                k >>= 1;
            }
            return res;
        }
        int z = 0;
        while (z * k < n && at(z) == 0) z++;
        if (z * k >= n) return Poly();
        Poly A = substr(z, z + n - z * k);
        Mint cf = A[0].pow(k);
        A /= A[0];
        A = A.pow_(k, n - z * k) * cf;
        return A.substr(-z * k, n - z * k);
    }
    Poly sqrt_(int n) const {
        if (deg() < 0 || n <= 0) return Poly();
        assert(at(0) == 1);
//    return pow_(Mint(2).inv(), n);
        vector<Mint> res(n);
```

```
        res[0] = 1;
        vector<Mint> f, g;
        for (int L = 1; L < n; L <<= 1) {
            f = vector<Mint>(2 * L);
            g = vector<Mint>(2 * L);
            for (int i = 0; i < L; i++)
                f[i] = res[i];
            fft(f);
            for (int i = 0; i < 2 * L; i++)
                f[i] *= f[i];
            fft(f);
            reverse(f.begin() + 1, f.end());
            Mint Li = Mint(2 * L).inv();
            for (int i = 0; i < 2 * L; i++)
                f[i] *= Li;
            for (int i = 0; i < 2 * L; i++)
                f[i] = at(i) - f[i];
            for (int i = 0; i < L; i++)
                assert(f[i] == 0);
            for (int i = 0; i < L; i++) {
                f[i] = f[i + L];
                f[i + L] = 0;
            }
            Poly Q = Poly(vector<Mint>(res.begin(), res.begin() + L)).inv(L);
            for (int i = 0; i < L; i++)
                g[i] = Q.at(i);
            fft(f);
            fft(g);
            for (int i = 0; i < 2 * L; i++)
                f[i] *= g[i];
            fft(f);
            reverse(f.begin() + 1, f.end());
            Li /= 2;
            for (int i = L; i < 2 * L && i < n; i++)
                res[i] = f[i - L] * Li;
        }
        return res;
    }
    bool sqrt(int n, Poly &R) const {
        if (deg() < 0) {
            R = Poly();
            return true;
        }
        if (at(0) == 1) {
            R = sqrt_(n);
            return true;
        }
        int z = 0;
        while (at(z) == 0) z++;
        if (z & 1) return false;
        Poly A = substr(z, n + z / 2);
        Mint cf;
```

```
    if (!A[0].sqrt(cf)) return false;
    A /= A[0];
    A = A.sqrt_(n - z / 2) * cf;
    R = A.substr(-z / 2, n - z / 2);
    return true;
}

static Poly multiply_all(vector<Poly> polys) {
    if (polys.empty()) return Poly({1});
    set<PII> setik;
    for (int i = 0; i < (int)polys.size(); i++)
        setik.insert(mp(polys[i].deg(), i));
    while ((int)setik.size() > 1) {
        int p = setik.begin()->se;
        setik.erase(setik.begin());
        int q = setik.begin()->se;
        setik.erase(setik.begin());
        polys[p] *= polys[q];
        setik.insert(mp(polys[p].deg(), p));
    }
    return polys[setik.begin()->se];
}
static Poly given_roots(const vector<Mint> &xs) {
    int n = (int)xs.size();
    vector<Poly> polys(n);
    for (int i = 0; i < n; i++)
        polys[i] = Poly({Mint() - xs[i], 1});
    return multiply_all(polys);
}

vector<Mint> multipoint(const vector<Mint> &xs) const {
    int n = (int)xs.size();
    if (n == 0) return {};
    if (n == 1) return {eval(xs[0])};
    int L = n;
    while (L & (L - 1)) L++;
    vector<Poly> tree(2 * L);
    for (int i = 0; i < n; i++)
        tree[L + i] = Poly({Mint() - xs[i], 1});
    for (int i = n; i < L; i++)
        tree[L + i] = Poly({1});
    for (int i = L - 1; i > 0; i--)
        tree[i] = tree[2 * i] * tree[2 * i + 1];
    tree[1] = *this % tree[1];
    for (int i = 2; i < L + n; i++)
        tree[i] = tree[i / 2] % tree[i];
    vector<Mint> res(n);
    for (int i = 0; i < n; i++)
        res[i] = tree[L + i].at(0);
    return res;
}
static pair<Poly, Poly> interpolate_(const vector<pair<Mint, Mint>> &vals, int l, int r) {
```

```
        if (r - l == 1) return mp(Poly({vals[l].se}), Poly({Mint() - vals[l].first, 1}));
        int m = (l + r) / 2;
        auto L = interpolate_(vals, l, m), R = interpolate_(vals, m, r);
        return mp(L.first * R.se + R.first * L.se, L.se * R.se);

    }
    static Poly interpolate(vector<pair<Mint, Mint>> vals) {
        if (vals.empty()) return Poly();
        int n = (int)vals.size();
        vector<Mint> xs(n);
        for (int i = 0; i < n; i++)
            xs[i] = vals[i].first;
        Poly P = given_roots(xs);
        P = P.derivate();
        vector<Mint> cf = P.multipoint(xs);
        for (int i = 0; i < n; i++)
            vals[i].se /= cf[i];
        return interpolate_(vals, 0, (int)vals.size()).first;
    }
    Poly x_k_mod_this(ll k) const { // x^k % P
        Poly res = Poly({1});
        int t = 0;
        while ((1LL << t) <= k) t++;
        for (int i = t - 1; i >= 0; i--) {
            res *= res;
            if ((k >> i) & 1) res = res.substr(-1, res.deg() + 1);
            res %= *this;
        }
        return res;
    }

    vector<Mint> chirp_z(Mint z, int n) const { // eval at [z^0, z^1, ..., z^(n-1)]
        int m = deg();
        if (m < 0 || n == 0) return vector<Mint>(n);
        if (z == 0) {
            vector<Mint> res(n, at(0));
            res[0] = eval(1);
            return res;
        }
        Mint zi = z.inv();
        vector<Mint> Z(n + m, 1), Zi(max(m + 1, n), 1);
        Mint w = 1, wi = 1;
        for (int i = 1; i < (int)Z.size(); i++) {
            Z[i] = Z[i - 1] * w;
            w *= z;
        }
        for (int i = 1; i < (int)Zi.size(); i++) {
            Zi[i] = Zi[i - 1] * wi;
            wi *= zi;
        }
        vector<Mint> f(m + 1);
        for (int i = 0; i <= m; i++)
```

```
            f[i] = at(i) * Zi[i];
        reverse(all(Z));
        Poly C = Poly(f) * Z;
        vector<Mint> res(n);
        for (int k = 0; k < n; k++)
            res[k] = C.at(n + m - 1 - k) * Zi[k];
        return res;
    }

    Poly shift_c(Mint c) const { // P(x + c)
        int n = deg();
        if (n < 0) return Poly();
        Factorials F(n);
        Powers P(n, c);
        vector<Mint> f(n + 1), g(n + 1);
        for (int i = 0; i <= n; i++) {
            f[i] = at(i) * F.f[i];
            g[i] = P.pow(i) * F.fi[i];
        }
        reverse(all(g));
        Poly C = Poly(f) * g;
        for (int i = 0; i <= n; i++)
            f[i] = C.at(n + i) * F.fi[i];
        return f;
    }

    static pair<Poly, Poly> _sub_exp(const vector<Mint> &a, int l, int r) {
        if (r - l == 1) return mp(Poly({a[l]}), Poly({1, -l}));
        int m = (l + r) / 2;
        auto L = _sub_exp(a, l, m), R = _sub_exp(a, m, r);
        return mp(L.first * R.se + R.first * L.se, L.se * R.se);
    }
    Poly substitute_exp(int m) const { // P(e^x)
        auto t = _sub_exp(a, 0, deg() + 1);
        auto A = (t.first * t.se.inv(m)).substr(0, m);
        Factorials<mod> F(m);
        vector<Mint> b(m, 0);
        for (int i = 0; i < m; i++)
            b[i] = A.at(i) * F.fi[i];
        return b;
    }
};

template<uint mod = MOD>
vector<mint<mod>> BerlekampMassey(vector<mint<mod>> x) {
    using Mint = mint<mod>;
    vector<Mint> ls, cur;
    int lf;
    Mint ld;
    for (int i = 0; i < (int)x.size(); i++) {
        Mint t = 0;
        for (int j = 0; j < (int)cur.size(); j++)
```

```
            t += cur[j] * x[i - j - 1];
        if (t == x[i]) continue;
        if (cur.empty()) {
            cur.resize(i + 1);
            lf = i;
            ld = t - x[i];
            continue;
        }
        Mint k = (t - x[i]) / ld;
        vector<Mint> c(i - lf - 1);
        c.push_back(k);
        for (auto t : ls) {
            c.push_back(Mint() - t * k);
        }
        if (c.size() < cur.size()) c.resize(cur.size());
        for (int j = 0; j < (int)cur.size(); j++)
            c[j] += cur[j];
        if (i - lf + (int)ls.size() >= (int)cur.size()) {
            ls = cur;
            lf = i;
            ld = t - x[i];
        }
        cur = c;
    }
    return cur;
}

template<uint mod = MOD>
vector<mint<mod>> MomentsOfDivisionOfPolynomials(const Poly<mod> &P, const Poly<mod> &Q, int
m) {
    using Mint = mint<mod>;
    m++;
    auto T = P.substitute_exp(m) * Q.substitute_exp(m).inv(m);
    vector<Mint> res(m, 0);
    Mint F = 1;
    for (int k = 0; k < m; k++) {
        res[k] = T.at(k) * F;
        F *= k + 1;
    }
    return res;
}

// CALL initFFT() and CHECK LOG
```

polysum

```
namespace polysum {
    const int D=101000;
    ll a[D],f[D],g[D],p[D],p1[D],p2[D],b[D],h[D][2],C[D];
    ll calcn(int d,ll *a,ll n) {//d次多项式(a[0-d])求第n项
        if (n<=d) return a[n];
```

```
        p1[0]=p2[0]=1;
        rep(i,0,d+1) {
            ll t=(n-i+mod)%mod;
            p1[i+1]=p1[i]*t%mod;
        }
        rep(i,0,d+1) {
            ll t=(n-d+i+mod)%mod;
            p2[i+1]=p2[i]*t%mod;
        }
        ll ans=0;
        rep(i,0,d+1) {
            ll t=g[i]*g[d-i]%mod*p1[i]%mod*p2[d-i]%mod*a[i]%mod;
            if ((d-i)&1) ans=(ans-t+mod)%mod;
            else ans=(ans+t)%mod;
        }
        return ans;
    }
    void init(int M) {//初始化预处理阶乘和逆元(取模乘法)
        f[0]=f[1]=g[0]=g[1]=1;
        rep(i,2,M+5) f[i]=f[i-1]*i%mod;
        g[M+4]=powmod(f[M+4],mod-2);
        per(i,1,M+4) g[i]=g[i+1]*(i+1)%mod;
    }
    ll polysum(ll n,ll *a,ll m) { // a[0].. a[m] \sum_{i=0}^{n-1} a[i]
                                  // m次多项式求第n项前缀和
        a[m+1]=calcn(m,a,m+1);
        rep(i,1,m+2) a[i]=(a[i-1]+a[i])%mod;
        return calcn(m+1,a,n-1);
    }
    ll qpolysum(ll R,ll n,ll *a,ll m) { // a[0].. a[m] \sum_{i=0}^{n-1} a[i]*R^i
        if (R==1) return polysum(n,a,m);
        a[m+1]=calcn(m,a,m+1);
        ll r=powmod(R,mod-2),p3=0,p4=0,c,ans;
        h[0][0]=0;h[0][1]=1;
        rep(i,1,m+2) {
            h[i][0]=(h[i-1][0]+a[i-1])*r%mod;
            h[i][1]=h[i-1][1]*r%mod;
        }
        rep(i,0,m+2) {
            ll t=g[i]*g[m+1-i]%mod;
            if (i&1) p3=((p3-h[i][0]*t)%mod+mod)%mod,p4=((p4-h[i][1]*t)%mod+mod)%mod;
            else p3=(p3+h[i][0]*t)%mod,p4=(p4+h[i][1]*t)%mod;
        }
        c=powmod(p4,mod-2)*(mod-p3)%mod;
        rep(i,0,m+2) h[i][0]=(h[i][0]+h[i][1]*c)%mod;
        rep(i,0,m+2) C[i]=h[i][0];
        ans=(calcn(m,C,n)*powmod(R,n)-c)%mod;
        if (ans<0) ans+=mod;
        return ans;
    }
}
```

ACAM-ptr

```
const int M = 26, N = 210000;
struct node {
    node *son[M], *go[M], *fail;
    int cnt;
} pool[N], *cur = pool, *d[N], *q[N], *root;

node *newnode() {
    return cur++;
}
int t, n;
char tt[N], s[N * 10];

void build() {
    t = 0;
    q[t++] = root;
    for (int i = 0; i < t; i++) {
        node *u = q[i];
        for (int j = 0; j < M; j++) {
            if (u->son[j]) {
                u->go[j] = u->son[j];
                if (u != root) u->go[j]->fail = u->fail->go[j];
                else u->go[j]->fail = root;
                q[t++] = u->son[j];
            }
            else {
                if (u != root) u->go[j] = u->fail->go[j];
                else u->go[j] = root;
            }
        }
    }
}

int main() {
    scanf("%d", &n);
    root = newnode();
    for (int i = 0; i < n; i++) {
        scanf("%s", tt);
        int m = strlen(tt);
        node *p = root;
        for (int j = 0; j < m; j++) {
            int w = tt[j] - 'a';
            if (!p->son[w]) p->son[w] = newnode();
            p = p->son[w];
        }
        d[i] = p;
    }
    build();
```

```
    scanf("%s", s);
    node *p = root;
    int l = strlen(s);
    for (int i = 0; i < l; i++) {
        p = p->go[s[i] - 'a'];
        p->cnt++;
    }
    for (int i = t - 1; i; i--) {
        q[i]->fail->cnt += q[i]->cnt;
    }
    for (int i = 0; i < n; i++) {
        printf("%d\n", d[i]->cnt);
    }
}
```

ACAM

```
struct node {
    int son[26], go[26];
    int fail, end, cnt;
    node() {
        rep(i, 0, 25) { son[i] = 0; go[i] = 0; }
        fail = 0; end = 0; cnt = 0;
    }
} ac[maxn];
int tot, d[maxn];

void insert(string s, int id) {
    int u = 0;
    for (int i = 0; i < s.size(); i++) {
        int w = s[i] - 'a';
        if (!ac[u].son[w])
            ac[u].son[w] = ++tot;
        u = ac[u].son[w];
    }
    // ac[u].end++;
    d[id] = u;
}

void get_fail() {
    queue<int> q;
    for (int i = 0; i < 26; i++) {
        if (ac[0].son[i]) {
            ac[0].go[i] = ac[0].son[i];
            ac[ac[0].son[i]].fail = 0;
            q.push(ac[0].son[i]);
        } else {
            ac[0].go[i] = 0;
        }
    }
    while (q.size()) {
```

```
        int u = q.front(); q.pop();
        ans.pb(u);
        for (int i = 0; i < 26; i++) {
            if (ac[u].son[i]) {
                ac[u].go[i] = ac[u].son[i];
                ac[ac[u].son[i]].fail = ac[ac[u].fail].go[i];
                q.push(ac[u].son[i]);
            } else {
                ac[u].go[i] = ac[ac[u].fail].go[i];
            }
        }
    }
}

void query(string s) {
    int u = 0;
    for (int i = 0; i < s.size(); i++) {
        if (s[i] == ' ') u = 0;
        else u = ac[u].go[s[i] - 'a'];
        ac[u].cnt++;
    }
}

void solve() {
    cin >> n;
    string s, t;
    rep(i, 1, n) {
        cin >> t;
        insert(t, i);
        s += t;
        if (i != n) s.pb(' ');
    }
    get_fail();
    query(s);
    reverse(all(ans));
    for (auto y : ans) {
        ac[ac[y].fail].cnt += ac[y].cnt;
    }
    rep(i, 1, n) cout << ac[d[i]].cnt << '\n';
}
```

exkmp

```
char s[maxn], a[maxn];
int z[maxn], p[maxn];

void exkmp(char s[], int len) {
    int L = 1, R = 0;
    z[1] = 0;
    rep(i, 2, len) {
        if (i > R) z[i] = 0;
```

```
        else {
            int k = i - L + 1;
            z[i] = min(z[k], R - i + 1);
        }
        while (i + z[i] <= len && s[z[i] + 1] == s[i + z[i]])
            z[i]++;
        if (i + z[i] - 1 > R)
            L = i, R = i + z[i] - 1;
    }
}

void match(char a[], char s[], int m, int n) {
    int L, R = 0;
    rep(i, 1, m) {
        if (i > R) p[i] = 0;
        else {
            int k = i - L + 1;
            p[i] = min(z[k], R - i + 1);
        }
        while (p[i] + 1 <= n && i + p[i] <= m && s[p[i] + 1] == a[p[i] + i])
            p[i]++;
        if (i + p[i] - 1 > R)
            L = i, R = i + p[i] - 1;
    }
}
```

hash_string

```
struct Hash {
    string s;
    vector<long long> p1, h1;
    vector<long long> p2, h2;
    static const int M1 = 1e9 + 7, w1 = 101, M2 = 998244353, w2 = 91;
    void build(string _s) {
        h1.clear();
        h2.clear();
        s = _s;
        h1.push_back(0);
        h2.push_back(0);
        p1.push_back(1);
        p2.push_back(1);

        for (int i = 0; i < (int)s.size(); i++) {
            h1.push_back((h1.back() * w1 % M1 + s[i] - 'a' + 1) % M1);
            h2.push_back((h2.back() * w2 % M2 + s[i] - 'a' + 1) % M2);
            p1.push_back(p1.back() * w1 % M1);
            p2.push_back(p2.back() * w2 % M2);
        }
    }
    pair<long long, long long> hash(int l, int r) {
        long long res1 = (h1[r] - h1[l - 1] * p1[r - l + 1] % M1) % M1;
```

```
        long long res2 = (h2[r] - h2[l - 1] * p2[r - l + 1] % M2) % M2;
        return {(res1 + M1) % M1, (res2 + M2) % M2};
    }
};
```

kmp

```
int nxt[maxn];
char a[maxn], b[maxn];
vector<int> pos;
void get_next(char s[], int len) {
    nxt[1] = 0;
    int x = 0;
    for (int i = 2; i <= len; i++) {
        while (x > 0 && s[x + 1] != s[i]) x = nxt[x];
        if (s[x + 1] == s[i])
            nxt[i] = x + 1, x++;
        else nxt[i] = x;
    }
}
int match(char a[], char s[], int n, int m) {
    int x = 0, ans = 0;
    for (int i = 1; i <= n; i++) {
        while (x > 0 && a[i] != s[x + 1]) x = nxt[x];
        if (a[i] == s[x + 1]) x++;
        if (x == m) ans++, x = nxt[x], pos.pb(i - m + 1);
    }
    return ans;
}
```

manacherfast

```
template <typename T>
vector<int> manacher(int n, const T &s) {
  if (n == 0) {
    return vector<int>();
  }
  vector<int> res(2 * n - 1, 0);
  int l = -1, r = -1;
  for (int z = 0; z < 2 * n - 1; z++) {
    int i = (z + 1) >> 1;
    int j = z >> 1;
    int p = (i >= r ? 0 : min(r - i, res[2 * (l + r) - z]));
    while (j + p + 1 < n && i - p - 1 >= 0) {
      if (!(s[j + p + 1] == s[i - p - 1])) {
        break;
      }
      p++;
    }
    if (j + p > r) {
```

```
      l = i - p;
      r = j + p;
    }
    res[z] = p;
  }
  return res;
}

template <typename T>
vector<int> manacher(const T &s) {
  return manacher((int) s.size(), s);
}
```

MinRotation

```
Description: Finds the lexicographically smallest rotation of a string.
Usage: rotate(v.begin(), v.begin() + minRotation(v), v.end());
Time: O (N)

int minRotation(string s) {
    int a = 0, N = sz(s); s += s;
    rep(b, 0, N) rep(k, 0, N) {
        if (a + k == b || s[a + k] < s[b + k]) {b += max(0, k - 1); break;}
        if (s[a + k] > s[b + k]) { a = b; break; }
    }
    return a;
}
```

rollingHash

```
typedef pair<int,int> hashv;
const ll mod1=1000000007;
const ll mod2=1000000009;

// prefixSum trick for high dimensions

hashv operator + (hashv a,hashv b) {
    int c1=a.fi+b.fi,c2=a.se+b.se;
    if (c1>=mod1) c1-=mod1;
    if (c2>=mod2) c2-=mod2;
    return mp(c1,c2);
}

hashv operator - (hashv a,hashv b) {
    int c1=a.fi-b.fi,c2=a.se-b.se;
    if (c1<0) c1+=mod1;
    if (c2<0) c2+=mod2;
    return mp(c1,c2);
}
```

```
hashv operator * (hashv a,hashv b) {
    return mp(1ll*a.fi*b.fi%mod1,1ll*a.se*b.se%mod2);
}
```

SA_IS

```
/*
 * Time Complexity: Suffix Array: O(N + Character_Set_Size) time and space //
 128 --- ASCII
 *                  LCP: O(N) time and space
 * Usage:
 *       1. Suffix Array (returns s.size() elements, NOT considering
 0-length/empty suffix)
 *             auto sa = suffix_array(s); // s is the input string with ASCII
 characters
 *             auto sa_wide_char = suffix_array(s, LIM); // LIM = max(s[i]) + 2,
 s is the string with arbitary big characters.
 *       2. LCP:
 *            auto lcp = LCP(s, suffix_array(s)); // returns s.size() elements,
 where lcp[i]=LCP(sa[i], sa[i+1])
 * Status: Tested (DMOJ: ccc03s4, SPOJ: SARRAY (100pts), Yosupo's: Suffix Array
 & Number of Substrings, CodeForces EDU
 */
// Based on: Rickypon, https://judge.yosupo.jp/submission/10105
void induced_sort(const std::vector<int>& vec, int val_range,
                  std::vector<int>& SA, const std::vector<bool>& sl,
                  const std::vector<int>& lms_idx) {
    std::vector<int> l(val_range, 0), r(val_range, 0);
    for (int c : vec) {
        if (c + 1 < val_range) ++l[c + 1];
        ++r[c];
    }
    std::partial_sum(l.begin(), l.end(), l.begin());
    std::partial_sum(r.begin(), r.end(), r.begin());
    std::fill(SA.begin(), SA.end(), -1);
    for (int i = (int)lms_idx.size() - 1; i >= 0; --i)
        SA[--r[vec[lms_idx[i]]]] = lms_idx[i];
    for (int i : SA)
        if (i >= 1 && sl[i - 1]) SA[l[vec[i - 1]]++] = i - 1;
    std::fill(r.begin(), r.end(), 0);
    for (int c : vec) ++r[c];
    std::partial_sum(r.begin(), r.end(), r.begin());
    for (int k = (int)SA.size() - 1, i = SA[k]; k >= 1; --k, i = SA[k])
        if (i >= 1 && !sl[i - 1]) {
            SA[--r[vec[i - 1]]] = i - 1;
        }
}

std::vector<int> SA_IS(const std::vector<int>& vec, int val_range) {
    const int n = vec.size();
    std::vector<int> SA(n), lms_idx;
```

```
    std::vector<bool> sl(n);
    sl[n - 1] = false;
    for (int i = n - 2; i >= 0; --i) {
        sl[i] = (vec[i] > vec[i + 1] || (vec[i] == vec[i + 1] && sl[i + 1]));
        if (sl[i] && !sl[i + 1]) lms_idx.push_back(i + 1);
    }
    std::reverse(lms_idx.begin(), lms_idx.end());
    induced_sort(vec, val_range, SA, sl, lms_idx);
    std::vector<int> new_lms_idx(lms_idx.size()), lms_vec(lms_idx.size());
    for (int i = 0, k = 0; i < n; ++i)
        if (!sl[SA[i]] && SA[i] >= 1 && sl[SA[i] - 1]) {
            new_lms_idx[k++] = SA[i];
        }
    int cur = 0;
    SA[n - 1] = cur;
    for (size_t k = 1; k < new_lms_idx.size(); ++k) {
        int i = new_lms_idx[k - 1], j = new_lms_idx[k];
        if (vec[i] != vec[j]) {
            SA[j] = ++cur;
            continue;
        }
        bool flag = false;
        for (int a = i + 1, b = j + 1;; ++a, ++b) {
            if (vec[a] != vec[b]) {
                flag = true;
                break;
            }
            if ((!sl[a] && sl[a - 1]) || (!sl[b] && sl[b - 1])) {
                flag = !((!sl[a] && sl[a - 1]) && (!sl[b] && sl[b - 1]));
                break;
            }
        }
        SA[j] = (flag ? ++cur : cur);
    }
    for (size_t i = 0; i < lms_idx.size(); ++i) lms_vec[i] = SA[lms_idx[i]];
    if (cur + 1 < (int)lms_idx.size()) {
        auto lms_SA = SA_IS(lms_vec, cur + 1);
        for (size_t i = 0; i < lms_idx.size(); ++i) {
            new_lms_idx[i] = lms_idx[lms_SA[i]];
        }
    }
    induced_sort(vec, val_range, SA, sl, new_lms_idx);
    return SA;
}

std::vector<int> suffix_array(const std::string& s, const char first = 'a',
                              const char last = 'z') {
    std::vector<int> vec(s.size() + 1);
    std::copy(std::begin(s), std::end(s), std::begin(vec));
    for (auto& x : vec) x -= (int)first - 1;
    vec.back() = 0;
    auto ret = SA_IS(vec, (int)last - (int)first + 2);
```

```
    ret.erase(ret.begin());
    return ret;
}
// Author: https://codeforces.com/blog/entry/12796?#comment-175287
// Uses kasai's algorithm linear in time and space
std::vector<int> LCP(const std::string& s, const std::vector<int>& sa) {
    int n = s.size(), k = 0;
    std::vector<int> lcp(n), rank(n);
    for (int i = 0; i < n; i++) rank[sa[i]] = i;
    for (int i = 0; i < n; i++, k ? k-- : 0) {
        if (rank[i] == n - 1) {
            k = 0;
            continue;
        }
        int j = sa[rank[i] + 1];
        while (i + k < n && j + k < n && s[i + k] == s[j + k]) k++;
        lcp[rank[i]] = k;
    }
    lcp[n - 1] = 0;
    return lcp;
}

template <typename T, class F = function<T(const T&, const T&)>>
class SparseTable {
 public:
  int n;
  vector<vector<T>> mat;
  F func;

  SparseTable(const vector<T>& a, const F& f) : func(f) {
    n = static_cast<int>(a.size());
    int max_log = 32 - __builtin_clz(n);
    mat.resize(max_log);
    mat[0] = a;
    for (int j = 1; j < max_log; j++) {
      mat[j].resize(n - (1 << j) + 1);
      for (int i = 0; i <= n - (1 << j); i++) {
        mat[j][i] = func(mat[j - 1][i], mat[j - 1][i + (1 << (j - 1))]);
      }
    }
  }

  T get(int from, int to) const {
    assert(0 <= from && from <= to && to <= n - 1);
    int lg = 32 - __builtin_clz(to - from + 1) - 1;
    return func(mat[lg][from], mat[lg][to - (1 << lg) + 1]);
  }
};
```

SA

```
struct SA {
    int n, m;
    char s[maxn];
    int sa[maxn], rk[maxn], ht[maxn], x[maxn], y[maxn], c[maxn];
    void init() {
        cin >> (s + 1);
        n = strlen(s + 1), m = 128;
    }
    void get_sa() {
        rep(i, 1, n) c[x[i] = s[i]]++;
        rep(i, 2, m) c[i] += c[i - 1];
        per(i, n, 1) sa[c[x[i]]--] = i;
        // for (int k = 1; k <= n; k <<= 1) {
        for (int k = 1; k < n; k <<= 1) {
            int num = 0;
            rep(i, n - k + 1, n) y[++num] = i;
            rep(i, 1, n) if (sa[i] > k) y[++num] = sa[i] - k;
            rep(i, 1, m) c[i] = 0;
            rep(i, 1, n) c[x[i]]++;
            rep(i, 2, m) c[i] += c[i - 1];
            per(i, n, 1) sa[c[x[y[i]]]--] = y[i], y[i] = 0;
            swap(x, y);
            x[sa[1]] = 1, num = 1;
            rep(i, 2, n)
            x[sa[i]] = (y[sa[i]] == y[sa[i - 1]] && y[sa[i] + k] == y[sa[i - 1] + k]) ? num :
++num;
            if (num == n) break;
            m = num;
        }
    }
    void get_height() {
        rep(i, 1, n) rk[sa[i]] = i;
        for (int i = 1, k = 0; i <= n; i++) {
            if (rk[i] == 1) continue;
            if (k) k--;
            int j = sa[rk[i] - 1];
            while (i + k <= n && j + k <= n && s[i + k] == s[j + k]) k++;
            ht[rk[i]] = k;
        }
    }
};
SA f;
ll fa[maxn], sz[maxn];
vector<array<ll, 3>> seg[maxn];
vector<ll> vec[maxn];
ll len[maxn];

int find(int x) {
    if (fa[x] == x) return fa[x];
    else return fa[x] = find(fa[x]);
```

```
}
void init() {
    rep(i, 1, n) fa[i] = i, sz[i] = 1;
}

void answer(int l, int r) {
    rep(i, l, r) cout << f.s[i];
    cout << '\n';
}

void solve() {
    f.init(); f.get_sa(); f.get_height();
    n = f.n;
    init();
    int tp;
    cin >> tp >> k;
    // tp==0 -> 不同位置相同子串算一个
    // tp==1 -> 不同位置相同子串算多个
    rep(i, 1, n) len[i] = n - f.sa[i] + 1;
    if (tp == 0) {
        rep(i, 1, n) {
            if (k > (n - f.sa[i] + 1) - f.ht[i]) k -= (n - f.sa[i] + 1) - f.ht[i];
            else {
                answer(f.sa[i], f.sa[i] + k - 1 + f.ht[i]);
                return;
            }
        }
        cout << -1 << '\n';
    } else {
        rep(i, 2, n) vec[f.ht[i]].pb(i);
        for (int l = n - 1; l >= 0; l--) {
            for (auto y : vec[l]) {
                int u = find(y - 1), v = find(y);
                if (l < len[u])
                    seg[u].pb({l + 1, len[u], sz[u]});
                if (l < len[v])
                    seg[v].pb({l + 1, len[v], sz[v]});
                fa[v] = u;
                sz[u] += sz[v];
                len[u] = l;
            }
        }
        if (len[1] > 0)
            seg[1].pb({1, len[1], sz[1]});
        rep(i, 1, n) reverse(ALL(seg[i]));
        for (int i = 1; i <= n; i++)
            for (auto [l, r, w] : seg[i]) {
                if (k > (r - l + 1)*w) k -= (r - l + 1) * w;
                else {
                    answer(f.sa[i], f.sa[i] + l - 1 + (k - 1) / w);
                    return;
                }
```

```
            }
        cout << -1 << '\n';
    }
}
```

SAfast

```
template <typename T>
vector<int> suffix_array(int n, const T &s, int char_bound) {
  vector<int> a(n);
  if (n == 0) {
    return a;
  }
  if (char_bound != -1) {
    vector<int> aux(char_bound, 0);
    for (int i = 0; i < n; i++) {
      aux[s[i]]++;
    }
    int sum = 0;
    for (int i = 0; i < char_bound; i++) {
      int add = aux[i];
      aux[i] = sum;
      sum += add;
    }
    for (int i = 0; i < n; i++) {
      a[aux[s[i]]++] = i;
    }
  } else {
    iota(a.begin(), a.end(), 0);
    sort(a.begin(), a.end(), [&s](int i, int j) { return s[i] < s[j]; });
  }
  vector<int> sorted_by_second(n);
  vector<int> ptr_group(n);
  vector<int> new_group(n);
  vector<int> group(n);
  group[a[0]] = 0;
  for (int i = 1; i < n; i++) {
    group[a[i]] = group[a[i - 1]] + (!(s[a[i]] == s[a[i - 1]]));
  }
  int cnt = group[a[n - 1]] + 1;
  int step = 1;
  while (cnt < n) {
    int at = 0;
    for (int i = n - step; i < n; i++) {
      sorted_by_second[at++] = i;
    }
    for (int i = 0; i < n; i++) {
      if (a[i] - step >= 0) {
        sorted_by_second[at++] = a[i] - step;
      }
    }
```

```
    for (int i = n - 1; i >= 0; i--) {
      ptr_group[group[a[i]]] = i;
    }
    for (int i = 0; i < n; i++) {
      int x = sorted_by_second[i];
      a[ptr_group[group[x]]++] = x;
    }
    new_group[a[0]] = 0;
    for (int i = 1; i < n; i++) {
      if (group[a[i]] != group[a[i - 1]]) {
        new_group[a[i]] = new_group[a[i - 1]] + 1;
      } else {
        int pre = (a[i - 1] + step >= n ? -1 : group[a[i - 1] + step]);
        int cur = (a[i] + step >= n ? -1 : group[a[i] + step]);
        new_group[a[i]] = new_group[a[i - 1]] + (pre != cur);
      }
    }
    swap(group, new_group);
    cnt = group[a[n - 1]] + 1;
    step <<= 1;
  }
  return a;
}

template <typename T>
vector<int> suffix_array(const T &s, int char_bound) {
  return suffix_array((int) s.size(), s, char_bound);
}

template <typename T>
vector<int> build_lcp(int n, const T &s, const vector<int> &sa) {
  assert((int) sa.size() == n);
  vector<int> pos(n);
  for (int i = 0; i < n; i++) {
    pos[sa[i]] = i;
  }
  vector<int> lcp(max(n - 1, 0));
  int k = 0;
  for (int i = 0; i < n; i++) {
    k = max(k - 1, 0);
    if (pos[i] == n - 1) {
      k = 0;
    } else {
      int j = sa[pos[i] + 1];
      while (i + k < n && j + k < n && s[i + k] == s[j + k]) {
        k++;
      }
      lcp[pos[i]] = k;
    }
  }
  return lcp;
}
```

```cpp
template <typename T>
vector<int> build_lcp(const T &s, const vector<int> &sa) {
  return build_lcp((int) s.size(), s, sa);
}
```

SAM

```cpp
struct SAM {
    static constexpr int ALPHABET_SIZE = 26;
    struct Node {
        int len;
        int link;
        std::array<int, ALPHABET_SIZE> next;
        Node() : len{}, link{}, next{} {}
    };
    std::vector<Node> t;
    SAM() {
        init();
    }
    void init() {
        t.assign(2, Node());
        t[0].next.fill(1);
        t[0].len = -1;
    }
    int newNode() {
        t.emplace_back();
        return t.size() - 1;
    }
    int extend(int p, int c) {
        if (t[p].next[c]) {
            int q = t[p].next[c];
            if (t[q].len == t[p].len + 1) {
                return q;
            }
            int r = newNode();
            t[r].len = t[p].len + 1;
            t[r].link = t[q].link;
            t[r].next = t[q].next;
            t[q].link = r;
            while (t[p].next[c] == q) {
                t[p].next[c] = r;
                p = t[p].link;
            }
            return r;
        }
        int cur = newNode();
        t[cur].len = t[p].len + 1;
        while (!t[p].next[c]) {
            t[p].next[c] = cur;
            p = t[p].link;
```

```
        }
        t[cur].link = extend(p, c);
        return cur;
    }
};
```

Z

```
template <typename T>
vector<int> z_function(int n, const T &s) {
  vector<int> z(n, n);
  int l = 0, r = 0;
  for (int i = 1; i < n; i++) {
    z[i] = (i > r ? 0 : 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;
}

template <typename T>
vector<int> z_function(const T &s) {
  return z_function((int) s.size(), s);
}
```

Graph theory

2-SAT

```
void dfs(int x) {
    low[x] = dfn[x] = ++idx;
    stk.push(x);
    ins[x] = 1;
    for (auto y : g[x]) {
        if (!dfn[y]) {
            dfs(y);
            low[x] = min(low[x], low[y]);
        } else {
            if (ins[y]) low[x] = min(low[x], dfn[y]);
        }
    }
    if (low[x] >= dfn[x]) {
```

```
        ++tot;
        while (true) {
            int cnt = stk.top();
            stk.pop();
            ins[cnt] = 0;
            belong[cnt] = tot;
            if (cnt == x) break;
        }
    }
}
int main() {
    cin >> n >> m;
    char a[10], b[10];
    rep(i, 1, m) {
        cin >> a >> b;
        int t1, t2;
        t1 = a[1] - '0';
        t2 = b[1] - '0';
        int ida0 = (a[0] == 'm' ? t1 : t1 + n);
        int idb0 = (b[0] == 'm' ? t2 : t2 + n);
        int ida1 = (ida0 > n ? ida0 - n : ida0 + n);
        int idb1 = (idb0 > n ? idb0 - n : idb0 + n);
        // a0->b1 b0->a1
        g[idb1].pb(ida0);
        g[ida1].pb(idb0);
    }
    rep(i, 1, 2 * n) if (!dfn[i]) dfs(i);
    bool ok = 1;
    rep(i, 1, n) if (belong[i] == belong[i + n]) {ok = 0; break;}
    cout << (ok ? "GOOD" : "BAD") << '\n';
}
```

差分约束系统

```
vector<PII> g[maxn];
int dist[maxn];

int main() {
    cin >> n >> m;
    rep(i, 1, m) {
        int u, v, w;
        cin >> u >> v >> w;
        // xu-xv<=w -> xu<=xv+w
        // xv-xu<=w -> xv<=xu+w
        g[u].pb({v, w});
    }
    fill(dist + 1, dist + 1 + n, 0);
    while (true) {
        bool ok = 0;
        for (int i = 1; i <= n; i++)
            for (auto [v, w] : g[i]) {
```

```
                if (dist[v] > dist[i] + w) ok = 1;
                dist[v] = min(dist[v], dist[i] + w);
            }
        if (!ok) break;
    }
    rep(i, 1, n) cout << -dist[i] << '\n';
}
```

bellmanford

```
vector<PII> g[maxn];
int dist[maxn];
bool ins[maxn];
void bellman_ford(int st) {
    memset(dist, 0x3f, sizeof dist);
    memset(ins, 0, sizeof ins);
    dist[st] = 0;
    int cnt = 0;
    while (true) {
        cnt++;
        bool ok = 0;
        for (int i = 1; i <= n; i++)
            for (auto [v, w] : g[i]) {
                if (dist[v] > dist[i] + w) {
                    dist[v] = dist[i] + w;
                    ok = 1;
                }
            }
        if (!ok) break;
        if (cnt == n) break;
    }
}
```

BlockCutTree

```
struct BlockCutTree {
    int n;
    std::vector<std::vector<int>> adj;
    std::vector<int> dfn, low, stk;
    int cnt, cur;
    std::vector<std::pair<int, int>> edges;

    BlockCutTree() {}
    BlockCutTree(int n) {
        init(n);
    }

    void init(int n) {
        this->n = n;
        adj.assign(n, {});
```

```
        dfn.assign(n, -1);
        low.resize(n);
        stk.clear();
        cnt = cur = 0;
        edges.clear();
    }

    void addEdge(int u, int v) {
        adj[u].push_back(v);
        adj[v].push_back(u);
    }

    void dfs(int x) {
        stk.push_back(x);
        dfn[x] = low[x] = cur++;

        for (auto y : adj[x]) {
            if (dfn[y] == -1) {
                dfs(y);
                low[x] = std::min(low[x], low[y]);
                if (low[y] == dfn[x]) {
                    int v;
                    do {
                        v = stk.back();
                        stk.pop_back();
                        edges.emplace_back(n + cnt, v);
                    } while (v != y);
                    edges.emplace_back(x, n + cnt);
                    cnt++;
                }
            } else {
                low[x] = std::min(low[x], dfn[y]);
            }
        }
    }

    std::pair<int, std::vector<std::pair<int, int>>> work() {
        for (int i = 0; i < n; i++) {
            if (dfn[i] == -1) {
                stk.clear();
                dfs(i);
            }
        }
        return {cnt, edges};
    }
};
```

dijkstra

```
int n, m, k;
vector<PII> g[maxn];
int dist[maxn];
bool ins[maxn];
void dijkstra(int st, int end) {
    fill(dist + 1, dist + n + 1, bit(29));
    fill(ins + 1, ins + n + 1, 0);
    dist[st] = 0;
    while (true) {
        int cnt = -1;
        bool ok = 0;
        for (int i = 1; i <= n; i++) {
            if (dist[i] < bit(28) && !ins[i])
                if (cnt == -1 || dist[i] < dist[cnt])
                    cnt = i, ok = 1;
        }
        if (!ok) break;
        ins[cnt] = 1;
        for (auto [id, w] : g[cnt]) {
            dist[id] = min(dist[id], dist[cnt] + w);
        }
    }
}
```

dijkstra&heap

```
int n, m, k;
vector<PII> g[maxn];
int dist[maxn];
bool use[maxn];
void dijkstra(int st, int end) {
    fill(dist + 1, dist + n + 1, bit(29));
    fill(use + 1, use + n + 1, 0);
    dist[st] = 0;
    priority_queue<PII, vector<PII>, greater<PII>> heap;
    heap.push({0, st});
    while (heap.size()) {
        auto [w, id] = heap.top();
        heap.pop();
        if (id == end) break;
        if (use[id]) continue;
        use[id] = 1;
        for (auto [v, w1] : g[id]) {
            if (dist[v] > dist[id] + w1) {
                dist[v] = dist[id] + w1;
                heap.push({dist[v], v});
            }
        }
```

```
    }
}
```

dinic

```
template<typename T>
struct FlowGraph {
    static const int V = 1015;
    static const int E = 100015;
    int s, t, vtot;
    int head[V], etot;
    int dis[V], cur[V];
    struct edge {
        int v, nxt;
        T f;
    } e[E * 2];
    void addedge(int u, int v, T f) {
        e[etot] = {v, head[u], f};
        head[u] = etot++;
        e[etot] = {u, head[v], 0};
        head[v] = etot++;
    }
    bool bfs() {
        for (int i = 1; i <= vtot; i++) {
            dis[i] = 0;
            cur[i] = head[i];
        }
        queue<int> q;
        q.push(s); dis[s] = 1;
        while (!q.empty()) {
            int u = q.front(); q.pop();
            for (int i = head[u]; i != -1; i = e[i].nxt) {
                if (e[i].f && !dis[e[i].v]) {
                    int v = e[i].v;
                    dis[v] = dis[u] + 1;
                    if (v == t) return true;
                    q.push(v);
                }
            }
        }
        return false;
    }
    T dfs(int u, T m) {
        if (u == t) return m;
        T flow = 0;
        for (int i = cur[u]; i != -1; cur[u] = i = e[i].nxt) {
            if (e[i].f && dis[e[i].v] == dis[u] + 1) {
                T f = dfs(e[i].v, min(m, e[i].f));
                e[i].f -= f;
                e[i ^ 1].f += f;
                m -= f;
```

```
                flow += f;
                if (!m) break;
            }
        }
        if (!flow) dis[u] = -1;
        return flow;
    }
    T dinic() {
        T flow = 0;
        while (bfs()) flow += dfs(s, numeric_limits<T>::max());
        return flow;
    }
    void init(int _s, int _t, int _vtot) {
        s = _s;
        t = _t;
        vtot = _vtot;
        etot = 0;
        for (int i = 1; i <= vtot; i++) head[i] = -1;
    }
};
```

dinicClass

```
template <typename T>
class flow_graph {
public:
    static constexpr T eps = (T) 1e-9;

    struct edge {
        int from;
        int to;
        T c;
        T f;
    };

    vector<vector<int>> g;
    vector<edge> edges;
    int n;
    int st;
    int fin;
    T flow;

    flow_graph(int _n, int _st, int _fin) : n(_n), st(_st), fin(_fin) {
        assert(0 <= st && st < n && 0 <= fin && fin < n && st != fin);
        g.resize(n);
        flow = 0;
    }

    void clear_flow() {
        for (const edge &e : edges) {
            e.f = 0;
```

```
        }
        flow = 0;
    }

    int add(int from, int to, T forward_cap, T backward_cap) {
        assert(0 <= from && from < n && 0 <= to && to < n);
        int id = (int) edges.size();
        g[from].push_back(id);
        edges.push_back({from, to, forward_cap, 0});
        g[to].push_back(id + 1);
        edges.push_back({to, from, backward_cap, 0});
        return id;
    }
};

template <typename T>
class dinic {
public:
    flow_graph<T> &g;

    vector<int> ptr;
    vector<int> d;
    vector<int> q;

    dinic(flow_graph<T> &_g) : g(_g) {
        ptr.resize(g.n);
        d.resize(g.n);
        q.resize(g.n);
    }

    bool expath() {
        fill(d.begin(), d.end(), -1);
        q[0] = g.fin;
        d[g.fin] = 0;
        int beg = 0, end = 1;
        while (beg < end) {
            int i = q[beg++];
            for (int id : g.g[i]) {
                const auto &e = g.edges[id];
                const auto &back = g.edges[id ^ 1];
                if (back.c - back.f > g.eps && d[e.to] == -1) {
                    d[e.to] = d[i] + 1;
                    if (e.to == g.st) {
                        return true;
                    }
                    q[end++] = e.to;
                }
            }
        }
        return false;
    }
```

```
    T dfs(int v, T w) {
        if (v == g.fin) {
            return w;
        }
        int &j = ptr[v];
        while (j >= 0) {
            int id = g.g[v][j];
            const auto &e = g.edges[id];
            if (e.c - e.f > g.eps && d[e.to] == d[v] - 1) {
                T t = dfs(e.to, min(e.c - e.f, w));
                if (t > g.eps) {
                    g.edges[id].f += t;
                    g.edges[id ^ 1].f -= t;
                    return t;
                }
            }
            j--;
        }
        return 0;
    }

    T max_flow() {
        while (expath()) {
            for (int i = 0; i < g.n; i++) {
                ptr[i] = (int) g.g[i].size() - 1;
            }
            T big_add = 0;
            while (true) {
                T add = dfs(g.st, numeric_limits<T>::max());
                if (add <= g.eps) {
                    break;
                }
                big_add += add;
            }
            if (big_add <= g.eps) {
                break;
            }
            g.flow += big_add;
        }
        return g.flow;
    }

    vector<bool> min_cut() {
        max_flow();
        vector<bool> ret(g.n);
        for (int i = 0; i < g.n; i++) {
            ret[i] = (d[i] != -1);
        }
        return ret;
    }
};
```

eulerPath(directed)

```
vector<int> g[maxn];
int in[maxn], out[maxn], f[maxn], vis[maxn];
char s[maxn];
vector<int> vec;

void dfs(int x) {
    while (f[x] < (int)g[x].size()) {
        int y = g[x][f[x]];
        f[x]++;
        dfs(y);
    }
    vec.pb(x);
}

bool euler() {
    int st = -1, dif = 0, stn = 0;
    rep(i, 1, n) {
        if (in[i] + 1 == out[i]) stn++, st = i;
        if (in[i] != out[i]) dif++;
    }
    if (!(dif == 0 || (dif == 2 && stn == 1))) return false;
    if (st == -1)
        rep(i, 1, n) if (in[i]) { st = i; break; }
    dfs(st);
    // vec.pb(st);
    // reverse(all(vec));
    if ((int)vec.size() != m + 1) return false;
    return true;
}

int main() {
    cin >> m;
    n = 26;
    rep(i, 1, m) {
        cin >> (s + 1);
        int len = strlen(s + 1);
        int u = s[1] - 'a' + 1, v = s[len] - 'a' + 1;
        g[u].pb(v);
        in[v]++, out[u]++;
    }
    cout << (euler() ? "Yes" : "No") << '\n';
}
```

eulerPath(undirected)

```
vector<PII> g[maxn];
int d[maxn], f[maxn], vis[maxn], idx;
vector<int> vec;

void dfs(int x) {
    while (f[x] < (int)g[x].size()) {
        auto [v, id] = g[x][f[x]];
        f[x]++;
        if (vis[id]) continue;
        vis[id] = 1;
        dfs(v);
    }
    vec.pb(x);
}

bool euler() {
    int st = -1, stn = 0;
    rep(i, 1, n) {
        if (d[i] & 1) stn++, st = i;
    }
    if (!(stn == 0 || (stn == 2 && st != -1))) return false;
    if (st == -1)
        rep(i, 1, n) if (d[i]) { st = i; break; }
    dfs(st);
    // vec.pb(st);
    // reverse(all(vec));
    if ((int)vec.size() != m + 1) return false;
    return true;
}
int main() {
    cin >> n >> m;
    rep(i, 1, m) {
        int u, v;
        cin >> u >> v;
        idx++;
        g[u].pb({v, idx});
        g[v].pb({u, idx});
        d[u]++, d[v]++;
    }
    cout << (euler() ? "Yes" : "No") << '\n';
}
```

Hungary algorithm

```
vector<int> g[maxn];
int idx;
int a[N][N], use[N][N], p[maxn], vis[maxn];
```

```
bool find(int x) {
    vis[x] = 1;
    for (auto y : g[x]) {
        if (!p[y] || (!vis[p[y]] && find(p[y]))) {
            p[y] = x;
            return true;
        }
    }
    return false;
}

int match() {
    int res = 0;
    fill(p + 1, p + idx + 1, 0);
    for (int i = 1; i <= idx; i++) {
        fill(vis + 1, vis + idx + 1, 0);
        if (find(i)) res++;
    }
    return res;
}
```

KM

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

// L <= R, 左边完全匹配
// 最小权完备匹配

template<class T>
pair<T, vector<int>> hungarian(const vector<vector<T>> &a) {
  if (a.empty()) return {0, {}};
  int n = a.size() + 1, m = a[0].size() + 1;
  vector<T> u(n), v(m); // 顶标
  vector<int> p(m), ans(n - 1);
  for (int i = 1; i < n; i++) {
    p[0] = i;
    int j0 = 0;
    vector<T> dist(m, numeric_limits<T>::max());
    vector<int> pre(m, -1);
    vector<bool> done(m + 1);
    do { // dijkstra
      done[j0] = true;
      int i0 = p[j0], j1;
      T delta = numeric_limits<T>::max();
      for (int j = 1; j < m; j++) if (!done[j]) {
        auto cur = a[i0 - 1][j - 1] - u[i0] - v[j];
        if (cur < dist[j]) dist[j] = cur, pre[j] = j0;
        if (dist[j] < delta) delta = dist[j], j1 = j;
      }
```

```
      for (int j = 0; j < m; j++) {
        if (done[j]) u[p[j]] += delta, v[j] -= delta;
        else dist[j] -= delta;
      }
      j0 = j1;
    } while (p[j0]);
    while (j0) { // update alternating path
      int j1 = pre[j0];
      p[j0] = p[j1], j0 = j1;
    }
  }
  for (int j = 1; j < m; j++) {
    if (p[j]) ans[p[j] - 1] = j - 1;
  }
  return {-v[0], ans}; // min cost
}

int L, R, m;
int main() {
  scanf("%d%d%d", &L, &R, &m);
  R = max(L, R);
  auto a = vector<vector<ll>>(L, vector<ll>(R, 0));
  for (int i = 0; i < m; i++) {
    int u, v, w;
    scanf("%d%d%d", &u, &v, &w);
    --u; --v;
    a[u][v] = -w;
  }
  auto [val, ans] = hungarian(a);
  printf("%lld\n", -val);
  for (int i = 0; i < L; i++) {
    if (a[i][ans[i]] >= 0) ans[i] = -1;
    printf("%d%c", ans[i] + 1, " \n"[i == L - 1]);
  }
}
```

kosaraju

```
vector<int> e[maxn], erev[maxn];
vector<int> c, out;
vector<vector<int>> scc;
int vis[maxn];
void dfs(int u) {
    vis[u] = 1;
    for (auto v : e[u]) if (!vis[v]) dfs(v);
    out.pb(u);
}
void dfs_rev(int u) {
    vis[u] = 1;
    for (auto v : erev[u]) if (!vis[v]) dfs_rev(v);
    c.pb(u);
```

```
}
void solve() {
    cin >> n >> m;
    rep(i, 1, m) {
        int u, v;
        cin >> u >> v;
        e[u].pb(v);
        erev[v].pb(u);
    }
    rep(i, 1, n) if (!vis[i]) dfs(i);
    fill(vis + 1, vis + n + 1, 0);
    reverse(ALL(out));
    for (auto v : out) if (!vis[v]) {
            c.clear();
            dfs_rev(v);
            scc.pb(c);
        }
}
```

MCMF

```
template<typename T>
struct MinCostGraph {
    static const int V = 20100;
    static const int E = 201000;
    int s, t, vtot;
    int head[V], etot;
    T dis[V], flow, cost;
    int pre[V];
    bool vis[V];

    struct edge {
        int v, nxt;
        T f, c;
    } e[E * 2];
    void addedge(int u,int v, T f, T c, T f2 = 0){
        e[etot]= {v, head[u], f, c}; head[u] = etot++;
        e[etot]= {u, head[v], f2, -c}; head[v] = etot++;
    }

    bool spfa() {
        T inf = numeric_limits<T>::max() / 2;
        for (int i = 1; i <= vtot; i++) {
            dis[i] = inf;
            vis[i] = false;
            pre[i] = -1;
        }
        dis[s] = 0;
        vis[s] = true;
        queue<int> q;
        q.push(s);
```

```
        while (!q.empty()) {
            int u = q.front();
            for (int i = head[u]; ~i; i = e[i].nxt) {
                int v = e[i].v;
                if (e[i].f && dis[v] > dis[u] + e[i].c) {
                    dis[v] = dis[u] + e[i].c;
                    pre[v] = i;
                    if (!vis[v]) {
                        vis[v] = 1;
                        q.push(v);
                    }
                }
            }
            q.pop();
            vis[u] = false;
        }
        return dis[t] != inf;
    }

    void augment() {
        int u = t;
        T f = numeric_limits<T>::max();
        while (~pre[u]) {
            f = min(f, e[pre[u]].f);
            u = e[pre[u] ^ 1].v;
        }
        flow += f;
        cost += f * dis[t];
        u = t;
        while (~pre[u]) {
            e[pre[u]].f -= f;
            e[pre[u] ^ 1].f += f;
            u = e[pre[u] ^ 1].v;
        }
    }

    pair<T, T> solve() {
        flow = 0;
        cost = 0;
        while (spfa()) augment();
        return {flow, cost};
    }
    void init(int s_, int t_, int vtot_) {
        s = s_;
        t = t_;
        vtot = vtot_;
        etot = 0;
        for (int i = 1; i <= vtot; i++) head[i] = -1;
    }
};
```

MCMFfast

```
template <typename flow_t = int, typename cost_t = long long>
struct MCMF_SSPA {
    int N;
    vector<vector<int>> adj;
    struct edge_t {
        int dest;
        flow_t cap;
        cost_t cost;
    };
    vector<edge_t> edges;

    vector<char> seen;
    vector<cost_t> pi;
    vector<int> prv;

    explicit MCMF_SSPA(int N_) : N(N_), adj(N), pi(N, 0), prv(N) {}

    void addEdge(int from, int to, flow_t cap, cost_t cost) {
        assert(cap >= 0);
        int e = int(edges.size());
        edges.emplace_back(edge_t{to, cap, cost});
        edges.emplace_back(edge_t{from, 0, -cost});
        adj[from].push_back(e);
        adj[to].push_back(e+1);
    }

    const cost_t INF_COST = numeric_limits<cost_t>::max() / 4;
    const flow_t INF_FLOW = numeric_limits<flow_t>::max() / 4;
    vector<cost_t> dist;
    __gnu_pbds::priority_queue<pair<cost_t, int>> q;
    vector<typename decltype(q)::point_iterator> its;
    void path(int s) {
        dist.assign(N, INF_COST);
        dist[s] = 0;

        its.assign(N, q.end());
        its[s] = q.push({0, s});

        while (!q.empty()) {
            int i = q.top().second; q.pop();
            cost_t d = dist[i];
            for (int e : adj[i]) {
                if (edges[e].cap) {
                    int j = edges[e].dest;
                    cost_t nd = d + edges[e].cost;
                    if (nd < dist[j]) {
                        dist[j] = nd;
                        prv[j] = e;
                        if (its[j] == q.end()) {
                            its[j] = q.push({-(dist[j] - pi[j]), j});
```

```
                        } else {
                            q.modify(its[j], {-(dist[j] - pi[j]), j});
                        }
                    }
                }
            }
        }

        swap(pi, dist);
    }

    vector<pair<flow_t, cost_t>> maxflow(int s, int t) {
        assert(s != t);
        flow_t totFlow = 0; cost_t totCost = 0;
        vector<pair<flow_t, cost_t>> res;
        while (path(s), pi[t] < INF_COST) {
            flow_t curFlow = numeric_limits<flow_t>::max();
            for (int cur = t; cur != s; ) {
                int e = prv[cur];
                int nxt = edges[e^1].dest;
                curFlow = min(curFlow, edges[e].cap);
                cur = nxt;
            }
            totFlow += curFlow;
            totCost += pi[t] * curFlow;
            for (int cur = t; cur != s; ) {
                int e = prv[cur];
                int nxt = edges[e^1].dest;
                edges[e].cap -= curFlow;
                edges[e^1].cap += curFlow;
                cur = nxt;
            }

            res.emplace_back(totFlow, totCost);
        }
        return res;
    }
};
```

MCMFfull

```
template <typename T, typename C>
class MCMF {
 public:
  static constexpr T eps = (T) 1e-9;

  struct edge {
    int from;
    int to;
    T c;
    T f;
```

```
  C cost;
};

int n;
vector<vector<int>> g;
vector<edge> edges;
vector<C> d;
vector<C> pot;
__gnu_pbds::priority_queue<pair<C, int>> q;
vector<typename decltype(q)::point_iterator> its;
vector<int> pe;
const C INF_C = numeric_limits<C>::max() / 2;

explicit MCMF(int n_) : n(n_), g(n), d(n), pot(n, 0), its(n), pe(n) {}

int add(int from, int to, T forward_cap, T backward_cap, C edge_cost) {
  assert(0 <= from && from < n && 0 <= to && to < n);
  assert(forward_cap >= 0 && backward_cap >= 0);
  int id = static_cast<int>(edges.size());
  g[from].push_back(id);
  edges.push_back({from, to, forward_cap, 0, edge_cost});
  g[to].push_back(id + 1);
  edges.push_back({to, from, backward_cap, 0, -edge_cost});
  return id;
}

void expath(int st) {
  fill(d.begin(), d.end(), INF_C);
  q.clear();
  fill(its.begin(), its.end(), q.end());
  its[st] = q.push({pot[st], st});
  d[st] = 0;
  while (!q.empty()) {
    int i = q.top().second;
    q.pop();
    its[i] = q.end();
    for (int id : g[i]) {
      const edge &e = edges[id];
      int j = e.to;
      if (e.c - e.f > eps && d[i] + e.cost < d[j]) {
        d[j] = d[i] + e.cost;
        pe[j] = id;
        if (its[j] == q.end()) {
          its[j] = q.push({pot[j] - d[j], j});
        } else {
          q.modify(its[j], {pot[j] - d[j], j});
        }
      }
    }
  }
  swap(d, pot);
}
```

```
pair<T, C> calc(int st, int fin) {  // max_flow_min_cost
  T flow = 0;
  C cost = 0;
  bool ok = true;
  for (auto& e : edges) {
    if (e.c - e.f > eps && e.cost + pot[e.from] - pot[e.to] < 0) {
      ok = false;
      break;
    }
  }
  if (ok) {
    expath(st);
  } else {
    vector<int> deg(n, 0);
    for (int i = 0; i < n; i++) {
      for (int eid : g[i]) {
        auto& e = edges[eid];
        if (e.c - e.f > eps) {
          deg[e.to] += 1;
        }
      }
    }
    vector<int> que;
    for (int i = 0; i < n; i++) {
      if (deg[i] == 0) {
        que.push_back(i);
      }
    }
    for (int b = 0; b < (int) que.size(); b++) {
      for (int eid : g[que[b]]) {
        auto& e = edges[eid];
        if (e.c - e.f > eps) {
          deg[e.to] -= 1;
          if (deg[e.to] == 0) {
            que.push_back(e.to);
          }
        }
      }
    }
    fill(pot.begin(), pot.end(), INF_C);
    pot[st] = 0;
    if (static_cast<int>(que.size()) == n) {
      for (int v : que) {
        if (pot[v] < INF_C) {
          for (int eid : g[v]) {
            auto& e = edges[eid];
            if (e.c - e.f > eps) {
              if (pot[v] + e.cost < pot[e.to]) {
                pot[e.to] = pot[v] + e.cost;
                pe[e.to] = eid;
              }
```

```
            }
          }
        }
      }
    } else {
      que.assign(1, st);
      vector<bool> in_queue(n, false);
      in_queue[st] = true;
      for (int b = 0; b < (int) que.size(); b++) {
        int i = que[b];
        in_queue[i] = false;
        for (int id : g[i]) {
          const edge &e = edges[id];
          if (e.c - e.f > eps && pot[i] + e.cost < pot[e.to]) {
            pot[e.to] = pot[i] + e.cost;
            pe[e.to] = id;
            if (!in_queue[e.to]) {
              que.push_back(e.to);
              in_queue[e.to] = true;
            }
          }
        }
      }
    }
  }
  // debug(pot[fin]);
  while (pot[fin] < INF_C) {  // < 0
    T push = numeric_limits<T>::max();
    int v = fin;
    while (v != st) {
      const edge &e = edges[pe[v]];
      push = min(push, e.c - e.f);
      v = e.from;
    }
    v = fin;
    while (v != st) {
      edge &e = edges[pe[v]];
      e.f += push;
      edge &back = edges[pe[v] ^ 1];
      back.f -= push;
      v = e.from;
    }
    flow += push;
    cost += push * pot[fin];
    expath(st);
  }
  return {flow, cost};
 }
};
```

prim&heap

```
vector<PII> g[maxn];
int dist[maxn], use[maxn];
int prim(int st) {
    fill(dist + 1, dist + n + 1, bit(29));
    fill(use + 1, use + n + 1, 0);
    dist[st] = 0;
    priority_queue<PII, vector<PII>, greater<PII>> heap;
    heap.push({0, st});
    int res = 0;
    while (heap.size()) {
        auto [w, id] = heap.top();
        heap.pop();
        if (use[id]) continue;
        use[id] = 1;
        res += dist[id];
        for (auto [v, w1] : g[id]) {
            if (dist[v] > w1) {
                dist[v] = w1;
                heap.push({dist[v], v});
            }
        }
    }
    return res;
}
```

PushRelabel

```
/**
 * Author: Simon Lindholm
 * Date: 2015-02-24
 * License: CC0
 * Source: Wikipedia, tinyKACTL
 * Description: Push-relabel using the highest label selection rule and the gap heuristic.
Quite fast in practice.
 *  To obtain the actual flow, look at positive values only.
 * Time: $O(V^2\sqrt E)$
 * Status: Tested on Kattis and SPOJ, and stress-tested
 */
#pragma once

struct PushRelabel {
  typedef vector<int> vi;
  struct Edge {
    int dest, back;
    ll f, c;
  };
  vector<vector<Edge>> g;
  vector<ll> ec;
```

```
  vector<Edge*> cur;
  vector<vi> hs; vi H;
  PushRelabel(int n) : g(n), ec(n), cur(n), hs(2*n), H(n) {}

  void addEdge(int s, int t, ll cap, ll rcap=0) {
    if (s == t) return;
    g[s].push_back({t, SZ(g[t]), 0, cap});
    g[t].push_back({s, SZ(g[s])-1, 0, rcap});
  }

  void addFlow(Edge& e, ll f) {
    Edge &back = g[e.dest][e.back];
    if (!ec[e.dest] && f) hs[H[e.dest]].push_back(e.dest);
    e.f += f; e.c -= f; ec[e.dest] += f;
    back.f -= f; back.c += f; ec[back.dest] -= f;
  }
  ll calc(int s, int t) {
    int v = SZ(g); H[s] = v; ec[t] = 1;
    vi co(2*v); co[0] = v-1;
    rep(i,0,v-1) cur[i] = g[i].data();
    for (Edge& e : g[s]) addFlow(e, e.c);

    for (int hi = 0;;) {
      while (hs[hi].empty()) if (!hi--) return -ec[s];
      int u = hs[hi].back(); hs[hi].pop_back();
      while (ec[u] > 0)  // discharge u
        if (cur[u] == g[u].data() + SZ(g[u])) {
          H[u] = 1e9;
          for (Edge& e : g[u]) if (e.c && H[u] > H[e.dest]+1)
            H[u] = H[e.dest]+1, cur[u] = &e;
          if (++co[H[u]], !--co[hi] && hi < v)
            rep(i,0,v-1) if (hi < H[i] && H[i] < v)
              --co[H[i]], H[i] = v + 1;
          hi = H[u];
        } else if (cur[u]->c && H[u] == H[cur[u]->dest]+1)
          addFlow(*cur[u], min(ec[u], cur[u]->c));
        else ++cur[u];
    }
  }
  bool leftOfMinCut(int a) { return H[a] >= SZ(g); }
};
```

spfa判负环

```
bool spfa() {
    queue<int> q; // dist初值不影响负环的判断，存在负环即会一直更新
    rep(i, 1, n) {
        q.push(i);
        ins[i] = 1;
        num[i] = 0;
    }
```

```
    while (q.size()) {
        int p = q.front();
        q.pop();
        ins[p] = 0;
        for (auto [v, w] : g[p]) {
            if (dist[v] > dist[p] + w) {
                dist[v] = dist[p] + w;
                num[v] = num[p] + 1;
                if (num[v] >= n) return true;
                if (!ins[v]) {
                    ins[v] = 1;
                    q.push(v);
                }
            }
        }
    }
    return false;
}
```

tarjan边双

```
vector<PII> g[maxn];
stack<int> stk;
int dfn[maxn], low[maxn], idx, tot, belong[maxn];
vector<int> bcc[maxn];

void dfs(int x, int f) {
    low[x] = dfn[x] = ++idx;
    stk.push(x);
    for (auto [y, id] : g[x]) {
        if (!dfn[y]) {
            dfs(y, id);
            low[x] = min(low[x], low[y]);
        } else {
            if (id != f) low[x] = min(low[x], dfn[y]);
        }
    }
    if (low[x] >= dfn[x]) {
        ++tot;
        while (true) {
            int cnt = stk.top();
            stk.pop();
            belong[cnt] = tot;
            bcc[tot].pb(cnt);
            if (cnt == x) break;
        }
    }
}
```

tarjan点双

```
vector<int> g[maxn];
stack<int> stk;
int dfn[maxn], low[maxn], idx, tot, cut[maxn];
vector<int> bcc[maxn];

void dfs(int x, int f) {
    low[x] = dfn[x] = ++idx;
    stk.push(x);
    int ch = 0;
    for (auto y : g[x]) {
        if (!dfn[y]) {
            ch++;
            dfs(y, x);
            low[x] = min(low[x], low[y]);
            if (low[y] >= dfn[x]) {
                cut[x] = 1;
                ++tot;
                bcc[tot].pb(x);
                while (true) {
                    int cnt = stk.top();
                    stk.pop();
                    bcc[tot].pb(cnt);
                    if (cnt == y) break;
                }
            }
        } else {
            if (y != f) low[x] = min(low[x], dfn[y]);
        }
    }
    if (x == 1 && ch <= 1) cut[x] = 0;
}
```

tarjan割边

```
vector<PII> g[maxn];
stack<int> stk;
int dfn[maxn], ins[maxn], low[maxn];
int idx, tot;
VI ans;
void dfs(int x, int f) {
    low[x] = dfn[x] = ++idx;
    stk.push(x);
    ins[x] = 1;
    for (auto [y, id] : g[x]) {
        if (!dfn[y]) {
            dfs(y, id);
            low[x] = min(low[x], low[y]);
        } else {
```

```
            if (ins[y] && id != f) low[x] = min(low[x], dfn[y]);
        }
    }
    if (low[x] >= dfn[x]) {
        ++tot;
        while (true) {
            int cnt = stk.top();
            stk.pop();
            ins[cnt] = 0;
            if (cnt == x) break;
        }
        if (f != 0) ans.pb(f);
    }
}
```

tarjan割点

```
vector<int> g[maxn], ans;
stack<int> stk;
int dfn[maxn], cut[maxn], low[maxn], idx;

void dfs(int x, int f) {
    low[x] = dfn[x] = ++idx;
    stk.push(x);
    int ch = 0;
    for (auto y : g[x]) {
        if (!dfn[y]) {
            ch++;
            dfs(y, x);
            low[x] = min(low[x], low[y]);
            if (low[y] >= dfn[x]) cut[x] = 1;
        } else {
            if (y != f) low[x] = min(low[x], dfn[y]);
        }
    }
    if (x == 1 && ch <= 1) cut[x] = 0;
    if (cut[x]) ans.pb(x);
}
```

tarjan强连通分量

```
vector<int> g[maxn];
stack<int> stk;
int dfn[maxn], ins[maxn], low[maxn], belong[maxn];
int idx, tot;

void dfs(int x) {
    low[x] = dfn[x] = ++idx;
    ins[x] = 1;
    stk.push(x);
```

```
    for (auto y : g[x]) {
        if (!dfn[y]) {
            dfs(y);
            low[x] = min(low[x], low[y]);
        } else {
            if (ins[y]) low[x] = min(low[x], dfn[y]);
        }
    }
    if (low[x] >= dfn[x]) {
        ++tot;
        while (true) {
            int cnt = stk.top(); stk.pop();
            ins[cnt] = 0;
            belong[cnt] = tot;
            if (cnt == x) break;
        }
    }
}
```

Basic

bitset

```
template <int len = 1>
void solve(int n) {
    if (n > len) {
        solve<std::min(len*2, MAXLEN)>(n);
        return;
    }
    // solution using bitset<len>
}

struct Bitset {
    vector<ull> b;
    int n;
    Bitset(int x = 0) {
        n = x;
        b.resize((n + 63) / 64, 0);
    }

    int get(int x) {
        return (b[x >> 6] >> (x & 63)) & 1;
    }

    void set(int x, int y) {
        b[x >> 6] |= 1ULL << (x & 63);
        if (!y) b[x >> 6] ^= 1ULL << (x & 63);
    }
```

```
    Bitset &operator&=(const Bitset &another) {
        rep(i, 0, min(SZ(b), SZ(another.b)) - 1) {
            b[i] &= another.b[i];
        }
        return (*this);
    }

    Bitset operator&(const Bitset &another)const {
        return (Bitset(*this) &= another);
    }

    Bitset &operator|=(const Bitset &another) {
        rep(i, 0, min(SZ(b), SZ(another.b)) - 1) {
            b[i] |= another.b[i];
        }
        return (*this);
    }

    Bitset operator|(const Bitset &another)const {
        return (Bitset(*this) |= another);
    }

    Bitset &operator^=(const Bitset &another) {
        rep(i, 0, min(SZ(b), SZ(another.b)) - 1) {
            b[i] ^= another.b[i];
        }
        return (*this);
    }

    Bitset operator^(const Bitset &another)const {
        return (Bitset(*this) ^= another);
    }

    Bitset &operator>>=(int x) {
        if (x & 63) {
            rep(i, 0, SZ(b) - 2) {
                b[i] >>= (x & 63);
                b[i] ^= (b[i + 1] << (64 - (x & 63)));
            }
            b.back() >>= (x & 63);
        }

        x >>= 6;
        rep(i, 0, SZ(b) - 1) {
            if (i + x < SZ(b)) b[i] = b[i + x];
            else b[i] = 0;
        }
        return (*this);
    }

    Bitset operator>>(int x)const {
```

```
        return (Bitset(*this) >>= x);
    }

    Bitset &operator<<=(int x) {
        if (x & 63) {
            for (int i = SZ(b) - 1; i >= 1; i--) {
                b[i] <<= (x & 63);
                b[i] ^= b[i - 1] >> (64 - (x & 63));
            }
            b[0] <<= x & 63;
        }

        x >>= 6;
        for (int i = SZ(b) - 1; i >= 0; i--) {
            if (i - x >= 0) b[i] = b[i - x];
            else b[i] = 0;
        }
        return (*this);
    }

    Bitset operator<<(int x)const {
        return (Bitset(*this) <<= x);
    }
};
```

fastIO

```
static struct FastInput {
  static constexpr int BUF_SIZE = 1 << 20;
  char buf[BUF_SIZE];
  size_t chars_read = 0;
  size_t buf_pos = 0;
  FILE *in = stdin;
  char cur = 0;

  inline char get_char() {
    if (buf_pos >= chars_read) {
      chars_read = fread(buf, 1, BUF_SIZE, in);
      buf_pos = 0;
      buf[0] = (chars_read == 0 ? -1 : buf[0]);
    }
    return cur = buf[buf_pos++];
  }

  template <typename T>
  inline void tie(T) {}

  inline explicit operator bool() {
    return cur != -1;
  }
```

```
inline static bool is_blank(char c) {
  return c <= ' ';
}

inline bool skip_blanks() {
  while (is_blank(cur) && cur != -1) {
    get_char();
  }
  return cur != -1;
}

inline FastInput& operator>>(char& c) {
  skip_blanks();
  c = cur;
  get_char();
  return *this;
}

inline FastInput& operator>>(string& s) {
  if (skip_blanks()) {
    s.clear();
    do {
      s += cur;
    } while (!is_blank(get_char()));
  }
  return *this;
}

template <typename T>
inline FastInput& read_integer(T& n) {
  // unsafe, doesn't check that characters are actually digits
  n = 0;
  if (skip_blanks()) {
    int sign = +1;
    if (cur == '-') {
      sign = -1;
      get_char();
    }
    do {
      n += n + (n << 3) + cur - '0';
    } while (!is_blank(get_char()));
    n *= sign;
  }
  return *this;
}

template <typename T>
inline typename enable_if<is_integral<T>::value, FastInput&>::type operator>>(T& n) {
  return read_integer(n);
}

#if !defined(_WIN32) || defined(_WIN64)
```

```
  inline FastInput& operator>>(__int128& n) {
    return read_integer(n);
  }
  #endif

  template <typename T>
  inline typename enable_if<is_floating_point<T>::value, FastInput&>::type operator>>(T& n) {
    // not sure if really fast, for compatibility only
    n = 0;
    if (skip_blanks()) {
      string s;
      (*this) >> s;
      sscanf(s.c_str(), "%lf", &n);
    }
    return *this;
  }
} fast_input;

#define cin fast_input

static struct FastOutput {
  static constexpr int BUF_SIZE = 1 << 20;
  char buf[BUF_SIZE];
  size_t buf_pos = 0;
  static constexpr int TMP_SIZE = 1 << 20;
  char tmp[TMP_SIZE];
  FILE *out = stdout;

  inline void put_char(char c) {
    buf[buf_pos++] = c;
    if (buf_pos == BUF_SIZE) {
      fwrite(buf, 1, buf_pos, out);
      buf_pos = 0;
    }
  }

  ~FastOutput() {
    fwrite(buf, 1, buf_pos, out);
  }

  inline FastOutput& operator<<(char c) {
    put_char(c);
    return *this;
  }

  inline FastOutput& operator<<(const char* s) {
    while (*s) {
      put_char(*s++);
    }
    return *this;
  }
```

```
inline FastOutput& operator<<(const string& s) {
  for (int i = 0; i < (int) s.size(); i++) {
    put_char(s[i]);
  }
  return *this;
}

template <typename T>
inline char* integer_to_string(T n) {
  // beware of TMP_SIZE
  char* p = tmp + TMP_SIZE - 1;
  if (n == 0) {
    *--p = '0';
  } else {
    bool is_negative = false;
    if (n < 0) {
      is_negative = true;
      n = -n;
    }
    while (n > 0) {
      *--p = (char) ('0' + n % 10);
      n /= 10;
    }
    if (is_negative) {
      *--p = '-';
    }
  }
  return p;
}

template <typename T>
inline typename enable_if<is_integral<T>::value, char*>::type stringify(T n) {
  return integer_to_string(n);
}

#if !defined(_WIN32) || defined(_WIN64)
inline char* stringify(__int128 n) {
  return integer_to_string(n);
}
#endif

template <typename T>
inline typename enable_if<is_floating_point<T>::value, char*>::type stringify(T n) {
  sprintf(tmp, "%.17f", n);
  return tmp;
}

template <typename T>
inline FastOutput& operator<<(const T& n) {
  auto p = stringify(n);
  for (; *p != 0; p++) {
    put_char(*p);
```

```
    }
    return *this;
  }
} fast_output;

#define cout fast_output
```

FastMod

```
Description: Compute a % b about 5 times faster than usual, where b is
constant but not known at compile time. Returns a value congruent to a
(mod b) in the range [0, 2b).

typedef unsigned long long ull;
struct FastMod {
    ull b, m;
    FastMod(ull b) : b(b), m(-1ULL / b) {}
    ull reduce(ull a) { // a % b + (0 or b)
        return a - (ull)((__uint128_t(m) * a) >> 64) * b;
    }
};
```

intervalContainer

```
Description: Add and remove intervals from a set of disjoint intervals.
Will merge the added interval with any overlapping intervals in the set when
adding. Intervals are [inclusive, exclusive).
Time: O (log N)

set<pii>::iterator addInterval(set<pii>& is, int L, int R) {
    if (L == R) return is.end();
    auto it = is.lower_bound({L, R}), before = it;
    while (it != is.end() && it->first <= R) {
        R = max(R, it->second);
        before = it = is.erase(it);
    }
    if (it != is.begin() && (--it)->second >= L) {
        L = min(L, it->first);
        R = max(R, it->second);
        is.erase(it);
    }
    return is.insert(before, {L, R});
}
void removeInterval(set<pii>& is, int L, int R) {
    if (L == R) return;
    auto it = addInterval(is, L, R);
    auto r2 = it->second;
    if (it->first == L) is.erase(it);
    else (int&)it->second = L;
    if (R != r2) is.emplace(R, r2);
```

```
}
```

lineContainer

```
/**
 * Author: Simon Lindholm
 * Date: 2017-04-20
 * License: CC0
 * Source: own work
 * Description: Container where you can add lines of the form kx+m, and query maximum values
at points x.
 *  Useful for dynamic programming (``convex hull trick'').
 * Time: O(\log N)
 * Status: stress-tested
 */
#pragma once

struct Line {
    mutable ll k, m, p;
    bool operator<(const Line& o) const { return k < o.k; }
    bool operator<(ll x) const { return p < x; }
};

struct LineContainer : multiset<Line, less<>> {
    // (for doubles, use inf = 1/.0, div(a,b) = a/b)
    static const ll inf = LLONG_MAX;
    ll div(ll a, ll b) { // floored division
        return a / b - ((a ^ b) < 0 && a % b); }
    bool isect(iterator x, iterator y) {
        if (y == end()) return x->p = inf, 0;
        if (x->k == y->k) x->p = x->m > y->m ? inf : -inf;
        else x->p = div(y->m - x->m, x->k - y->k);
        return x->p >= y->p;
    }
    void add(ll k, ll m) {
        auto z = insert({k, m, 0}), y = z++, x = y;
        while (isect(y, z)) z = erase(z);
        if (x != begin() && isect(--x, y)) isect(x, y = erase(y));
        while ((y = x) != begin() && (--x)->p >= y->p)
            isect(x, erase(y));
    }
    ll query(ll x) {
        assert(!empty());
        auto l = *lower_bound(x);
        return l.k * x + l.m;
    }
};
```

mint

```
template<int MOD, int RT> struct mint {
    static const int mod = MOD;
    static constexpr mint rt() { return RT; } // primitive root for FFT
    int v; explicit operator int() const { return v; } // explicit -> don't silently convert
to int
    mint():v(0) {}
    mint(ll _v) { v = int((-MOD < _v && _v < MOD) ? _v : _v % MOD);
        if (v < 0) v += MOD; }
    bool operator==(const mint& o) const {
        return v == o.v; }
    friend bool operator!=(const mint& a, const mint& b) {
        return !(a == b); }
    friend bool operator<(const mint& a, const mint& b) {
        return a.v < b.v; }

    mint& operator+=(const mint& o) {
        if ((v += o.v) >= MOD) v -= MOD;
        return *this; }
    mint& operator-=(const mint& o) {
        if ((v -= o.v) < 0) v += MOD;
        return *this; }
    mint& operator*=(const mint& o) {
        v = int((ll)v*o.v%MOD); return *this; }
    mint& operator/=(const mint& o) { return (*this) *= inv(o); }
    friend mint pow(mint a, ll p) {
        mint ans = 1; assert(p >= 0);
        for (; p; p /= 2, a *= a) if (p&1) ans *= a;
        return ans; }
    friend mint inv(const mint& a) { assert(a.v != 0);
        return pow(a,MOD-2); }

    mint operator-() const { return mint(-v); }
    mint& operator++() { return *this += 1; }
    mint& operator--() { return *this -= 1; }
    friend mint operator+(mint a, const mint& b) { return a += b; }
    friend mint operator-(mint a, const mint& b) { return a -= b; }
    friend mint operator*(mint a, const mint& b) { return a *= b; }
    friend mint operator/(mint a, const mint& b) { return a /= b; }
};

const int MOD=998244353;
using mi = mint<MOD,5>; // 5 is primitive root for both common mods

namespace simp {
    vector<mi> fac,ifac,invn;
    void check(int x) {
        if (fac.empty()) {
            fac={mi(1),mi(1)};
            ifac={mi(1),mi(1)};
            invn={mi(0),mi(1)};
```

```
        }
        while (SZ(fac)<=x) {
            int n=SZ(fac),m=SZ(fac)*2;
            fac.resize(m);
            ifac.resize(m);
            invn.resize(m);
            for (int i=n;i<m;i++) {
                fac[i]=fac[i-1]*mi(i);
                invn[i]=mi(MOD-MOD/i)*invn[MOD%i];
                ifac[i]=ifac[i-1]*invn[i];
            }
        }
    }
    mi gfac(int x) {
        check(x); return fac[x];
    }
    mi ginv(int x) {
        check(x); return invn[x];
    }
    mi gifac(int x) {
        check(x); return ifac[x];
    }
    mi binom(int n,int m) {
        if (m < 0 || m > n) return mi(0);
        return gfac(n)*gifac(m)*gifac(n - m);
    }
}
```

pbds

```
#include <bits/extc++.h>
using namespace __gnu_cxx;
using namespace __gnu_pbds;

#include<ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/tree_policy.hpp>
#include<ext/pb_ds/hash_policy.hpp>
#include<ext/pb_ds/trie_policy.hpp>
#include<ext/pb_ds/priority_queue.hpp>

pairing_heap_tag：配对堆
thin_heap_tag：斐波那契堆
binomial_heap_tag：二项堆
binary_heap_tag：二叉堆

__gnu_pbds::priority_queue<PII, greater<PII>, pairing_heap_tag> q;
__gnu_pbds::priority_queue<PII, greater<PII>, pairing_heap_tag>::point_iterator its[N];

its[v] = q.push({dis[v], v});
q.modify(its[v], {dis[v], v});
```

```
可以将两个优先队列中的元素合并（无任何约束）
使用方法为a.join(b)
此时优先队列b内所有元素就被合并进优先队列a中，且优先队列b被清空

cc_hash_table<string, int> mp1拉链法
gp_hash_table<string, int> mp2查探法
```

simulate anneal

```
pair<db,db> p[N];
db ans = 1e10;
db rd(db l, db r) {
    uniform_real_distribution<db> u(l,r);
    // uniform_int_distribution<ll> u(l,r);
    default_random_engine e(rng());
    return u(e);  // e(rng)
}

db dist(pair<db,db> a, pair<db,db> b) {
    db dx = a.fi - b.fi;
    db dy = a.se - b.se;
    // sqrtl() for long double
    return sqrt(dx * dx + dy * dy);
}

db eval(pair<db,db> x) {
    db res = 0;
    rep(i, 1, n) res += dist(p[i], x);
    ans = min(ans, res);
    return res;
}

void simulate_anneal() {
    pair<db,db> cnt(rd(0, 10000), rd(0, 10000));
    for (double k = 10000; k > 1e-5; k *= 0.99) {
        // [start, end, step]
        pair<db,db> np(cnt.fi + rd(-k, k), cnt.se + rd(-k, k));
        db delta = eval(np) - eval(cnt);
        if (exp(-delta / k) > rd(0, 1)) cnt = np;
    }
}
```

sort

```
void merge_sort(int q[], int l, int r) {
    if (l >= r) return;
    int mid = l + r >> 1;
    merge_sort(q, l, mid);
    merge_sort(q, mid + 1, r);
```

```
    int k = 0, i = l, j = mid + 1;
    while (i <= mid && j <= r)
        if (q[i] <= q[j])
            tmp[k++] = q[i++];
        else
            tmp[k++] = q[j++];

    while (i <= mid)
        tmp[k++] = q[i++];
    while (j <= r)
        tmp[k++] = q[j++];

    for (i = l, j = 0; i <= r; i++, j++) q[i] = tmp[j];
}

void quick_sort(int q[], int l, int r) {
    if (l >= r) return;
    int i = l - 1, j = r + 1, x = q[l + r >> 1];
    while (i < j) {
        do i ++ ; while (q[i] < x);
        do j -- ; while (q[j] > x);
        if (i < j) swap(q[i], q[j]);
    }
    quick_sort(q, l, j), quick_sort(q, j + 1, r);
}

template<class T>
void radixsort(T *a, ll n) {
    int base = 0;
    rep(i, 1, n) sa[i] = i;
    rep(k, 1, 5) {
        rep(i, 0, 255) c[i] = 0;
        rep(i, 1, n) c[(a[i] >> base) & 255]++;
        rep(i, 1, 255) c[i] += c[i - 1];
        per(i, n, 1) {
            rk[sa[i]] = c[(a[sa[i]] >> base) & 255]--;
        }
        rep(i, 1, n) sa[rk[i]] = i;
        base += 7;
    }
}
```

逆波兰表达式

```
class Solution:
  def calculate(self, s: str) -> int:
    sign = ['+', '-', '*', '/', '(', ')']
    v = []; num = ''
    for c in s:
      if c in sign:
        if num:
```

```
        v.append(num); num = ''
      if c == '-' and (not v or v[-1] == '('):
        v.append('0')
      v.append(c)
    elif c.isnumeric():
      num += c
  if num: v.append(num)

  stk0 = []; stk1 = []
  for e in v:
    if e.isnumeric():
      stk0.append(e)
    elif e in ['+', '-']:
      while stk1 and stk1[-1] in ['*', '/', '+', '-']:
        stk0.append(stk1.pop())
      stk1.append(e)
    elif e in ['*', '/', '(']:
      stk1.append(e)
    else:
      while stk1 and stk1[-1] != '(':
        stk0.append(stk1.pop())
      stk1.pop()
  while stk1:
    stk0.append(stk1.pop())

  res = []
  for e in stk0:
    if e.isnumeric():
      res.append(int(e))
    else:
      v = res.pop(); u = res.pop()
      if e == '+':
        res.append(u + v)
      if e == '-':
        res.append(u - v)
      if e == '*':
        res.append(u * v)
      if e == '/':
        res.append(u // v)
  return res[0]
```

对拍 loop

```cpp
int main() {
    std::ios::sync_with_stdio(false);
    cin.tie(0); cout.tie(0);
    while (T) { //一直循环，直到找到不一样的数据
        system("data.exe > in.txt");
        system("my.exe < in.txt > my.txt");
        system("std.exe < in.txt > std.txt");
        cout << T++ << ":" << endl;
        if (system("fc std.txt my.txt")) //当 fc 返回1时，说明这时数据不一样
            break;
    }
}
```

Computation geometry

point&line

```cpp
typedef double db;
const db EPS = 1e-9;

inline int sign(db a) { return a < -EPS ? -1 : a > EPS; }

inline int cmp(db a, db b) { return sign(a - b); }

struct P {
  db x, y;
  P() {}
  P(db _x, db _y) : x(_x), y(_y) {}
  P operator+(P p) { return {x + p.x, y + p.y}; }
  P operator-(P p) { return {x - p.x, y - p.y}; }
  P operator*(db d) { return {x * d, y * d}; }
  P operator/(db d) { return {x / d, y / d}; }

  bool operator<(P p) const {
    int c = cmp(x, p.x);
    if (c) return c == -1;
    return cmp(y, p.y) == -1;
  }

  bool operator==(P o) const {
    return cmp(x, o.x) == 0 && cmp(y, o.y) == 0;
  }

  db dot(P p) { return x * p.x + y * p.y; }
  db det(P p) { return x * p.y - y * p.x; }

  db distTo(P p) { return (*this - p).abs(); }
  db alpha() { return atan2(y, x); }
```

```
  void read() { cin >> x >> y; }
  void write() {cout << "(" << x << "," << y << ")" << endl;}
  db abs() { return sqrt(abs2());}
  db abs2() { return x * x + y * y; }
  P rot90() { return P(-y, x);}
  P unit() { return *this / abs(); }
  int quad() const { return sign(y) == 1 || (sign(y) == 0 && sign(x) >= 0); }
  P rot(db an) { return {x * cos(an) - y * sin(an), x * sin(an) + y * cos(an)}; }
};

#define cross(p1,p2,p3) ((p2.x-p1.x)*(p3.y-p1.y)-(p3.x-p1.x)*(p2.y-p1.y))
#define crossOp(p1,p2,p3) sign(cross(p1,p2,p3))

// 直线 p1p2, q1q2 是否恰有一个交点
bool chkLL(P p1, P p2, P q1, P q2) {
  db a1 = cross(q1, q2, p1), a2 = -cross(q1, q2, p2);
  return sign(a1 + a2) != 0;
}

// 求直线 p1p2, q1q2 的交点
P isLL(P p1, P p2, P q1, P q2) {
  db a1 = cross(q1, q2, p1), a2 = -cross(q1, q2, p2);
  return (p1 * a2 + p2 * a1) / (a1 + a2);
}

// 判断区间 [l1, r1], [l2, r2] 是否相交
bool intersect(db l1, db r1, db l2, db r2) {
  if (l1 > r1) swap(l1, r1); if (l2 > r2) swap(l2, r2);
  return !( cmp(r1, l2) == -1 || cmp(r2, l1) == -1 );
}

// 线段 p1p2, q1q2 相交
bool isSS(P p1, P p2, P q1, P q2) {
  return intersect(p1.x, p2.x, q1.x, q2.x) && intersect(p1.y, p2.y, q1.y, q2.y) &&
         crossOp(p1, p2, q1) * crossOp(p1, p2, q2) <= 0 && crossOp(q1, q2, p1)
         * crossOp(q1, q2, p2) <= 0;
}

// 线段 p1p2, q1q2 严格相交
bool isSS_strict(P p1, P p2, P q1, P q2) {
  return crossOp(p1, p2, q1) * crossOp(p1, p2, q2) < 0 && crossOp(q1, q2, p1)
         * crossOp(q1, q2, p2) < 0;
}

// m 在 a 和 b 之间
bool isMiddle(db a, db m, db b) {
  /*if (a > b) swap(a, b);
  return cmp(a, m) <= 0 && cmp(m, b) <= 0;*/
  return sign(a - m) == 0 || sign(b - m) == 0 || (a < m != b < m);
}

bool isMiddle(P a, P m, P b) {
```

```
  return isMiddle(a.x, m.x, b.x) && isMiddle(a.y, m.y, b.y);
}

// 点 p 在线段 p1p2 上
bool onSeg(P p1, P p2, P q) {
  return crossOp(p1, p2, q) == 0 && isMiddle(p1, q, p2);
}
// q1q2 和 p1p2 的交点 在 p1p2 上?

// 点 p 严格在 p1p2 上
bool onSeg_strict(P p1, P p2, P q) {
  return crossOp(p1, p2, q) == 0 && sign((q - p1).dot(p1 - p2)) * sign((q - p2).dot(p1 - p2))
< 0;
}

// 求 q 到 直线p1p2 的投影（垂足） ⚠️ : p1 != p2
P proj(P p1, P p2, P q) {
  P dir = p2 - p1;
  return p1 + dir * (dir.dot(q - p1) / dir.abs2());
}

// 求 q 以 直线p1p2 为轴的反射
P reflect(P p1, P p2, P q) {
  return proj(p1, p2, q) * 2 - q;
}

// 求 q 到 线段p1p2 的最小距离
db nearest(P p1, P p2, P q) {
  if (p1 == p2) return p1.distTo(q);
  P h = proj(p1, p2, q);
  if (isMiddle(p1, h, p2))
    return q.distTo(h);
  return min(p1.distTo(q), p2.distTo(q));
}

// 求 线段p1p2 与 线段q1q2 的距离
db disSS(P p1, P p2, P q1, P q2) {
  if (isSS(p1, p2, q1, q2)) return 0;
  return min(min(nearest(p1, p2, q1), nearest(p1, p2, q2)), min(nearest(q1, q2, p1),
nearest(q1, q2, p2)));
}

// 极角排序
sort(p, p + n, [&](P a, P b) {
  int qa = a.quad(), qb = b.quad();
  if (qa != qb) return qa < qb;
  else return sign(a.det(b)) > 0;
});
```

polygon

```
db area(vector<P> ps){
  db ret = 0; rep(i,0,ps.size()) ret += ps[i].det(ps[(i+1)%ps.size()]);
  return ret/2;
}

int contain(vector<P> ps, P p){ //2:inside,1:on_seg,0:outside
  int n = ps.size(), ret = 0;
  rep(i,0,n){
    P u=ps[i],v=ps[(i+1)%n];
    if(onSeg(u,v,p)) return 1;
    if(cmp(u.y,v.y)<=0) swap(u,v);
    if(cmp(p.y,u.y) > 0 || cmp(p.y,v.y) <= 0) continue;
    ret ^= crossOp(p,u,v) > 0;
  }
  return ret*2;
}

vector<P> convexHull(vector<P> ps) {
  int n = ps.size(); if(n <= 1) return ps;
  sort(ps.begin(), ps.end());
  vector<P> qs(n * 2); int k = 0;
  for (int i = 0; i < n; qs[k++] = ps[i++])
    while (k > 1 && crossOp(qs[k - 2], qs[k - 1], ps[i]) <= 0) --k;
  for (int i = n - 2, t = k; i >= 0; qs[k++] = ps[i--])
    while (k > t && crossOp(qs[k - 2], qs[k - 1], ps[i]) <= 0) --k;
  qs.resize(k - 1);
  return qs;
}

vector<P> convexHullNonStrict(vector<P> ps) {
  //caution: need to unique the Ps first
  int n = ps.size(); if(n <= 1) return ps;
  sort(ps.begin(), ps.end());
  vector<P> qs(n * 2); int k = 0;
  for (int i = 0; i < n; qs[k++] = ps[i++])
    while (k > 1 && crossOp(qs[k - 2], qs[k - 1], ps[i]) < 0) --k;
  for (int i = n - 2, t = k; i >= 0; qs[k++] = ps[i--])
    while (k > t && crossOp(qs[k - 2], qs[k - 1], ps[i]) < 0) --k;
  qs.resize(k - 1);
  return qs;
}

db convexDiameter(vector<P> ps){
  int n = ps.size(); if(n <= 1) return 0;
  int is = 0, js = 0; rep(k,1,n) is = ps[k]<ps[is]?k:is, js = ps[js] < ps[k]?k:js;
  int i = is, j = js;
  db ret = ps[i].distTo(ps[j]);
  do{
    if((ps[(i+1)%n]-ps[i]).det(ps[(j+1)%n]-ps[j]) >= 0)
      (++j)%=n;
```

```
    else
      (++i)%=n;
    ret = max(ret,ps[i].distTo(ps[j]));
  }while(i!=is || j!=js);
  return ret;
}

vector<P> convexCut(const vector<P>&ps, P q1, P q2) {
  vector<P> qs;
  int n = ps.size();
  rep(i,0,n){
    P p1 = ps[i], p2 = ps[(i+1)%n];
    int d1 = crossOp(q1,q2,p1), d2 = crossOp(q1,q2,p2);
    if(d1 >= 0) qs.push_back(p1);
    if(d1 * d2 < 0) qs.push_back(isLL(p1,p2,q1,q2));
  }
  return qs;
}

void reorderPolygon(vector<P> &ps) {
    size_t pos = 0;
    for(size_t i = 1; i < ps.size(); i++){
        if(ps[i].y < ps[pos].y || (ps[i].y == ps[pos].y && ps[i].x < ps[pos].x))
            pos = i;
    }
    rotate(ps.begin(), ps.begin() + pos, ps.end());
}

vector<P> minkowski(vector<P> p, vector<P> q){
    if(p.empty()) return q;
    // the first vertex must be the lowest
    reorderPolygon(p);
    reorderPolygon(q);
    // must ensure cyclic indexing
    p.push_back(p[0]);
    p.push_back(p[1]);
    q.push_back(q[0]);
    q.push_back(q[1]);
    // main part
    vector<P> result;
    size_t i = 0, j = 0;
    while(i < p.size() - 2 || j < q.size() - 2){
        result.push_back(p[i] + q[j]);
        auto cross = (p[i + 1] - p[i]).det(q[j + 1] - q[j]);
        if(cross >= 0)
            ++i;
        if(cross <= 0)
            ++j;
    }
    return result;
}
```

```
bool convexContain(const vector<P> &l, P p, bool strict = true) {
    int a = 1, b = l.size() - 1, r = !strict;
    if (l.size() < 3) return r && onSeg(l[0], l.back(), p);
    if (crossOp(l[0], l[a], l[b]) > 0) swap(a, b);
    if (crossOp(l[0], l[a], p) >= r || crossOp(l[0], l[b], p)<= -r)
        return false;
    while (abs(a - b) > 1) {
        int c = (a + b) / 2;
        (crossOp(l[0], l[c], p) > 0 ? b : a) = c;
    }
    return sign(cross(l[a], l[b], p)) < r;
}
```

circle

```
int type(P o1,db r1,P o2,db r2){
  db d = o1.distTo(o2);
  if(cmp(d,r1+r2) == 1) return 4;
  if(cmp(d,r1+r2) == 0) return 3;
  if(cmp(d,abs(r1-r2)) == 1) return 2;
  if(cmp(d,abs(r1-r2)) == 0) return 1;
  return 0;
}

vector<P> isCL(P o,db r,P p1,P p2){
  if (cmp(abs((o-p1).det(p2-p1)/p1.distTo(p2)),r)>0) return {};
  db x = (p1-o).dot(p2-p1), y = (p2-p1).abs2(), d = x * x - y * ((p1-o).abs2() - r*r);
  d = max(d,(db)0.0); P m = p1 - (p2-p1)*(x/y), dr = (p2-p1)*(sqrt(d)/y);
  return {m-dr,m+dr}; //along dir: p1->p2
}

vector<P> isCC(P o1, db r1, P o2, db r2) { //need to check whether two circles are the same
  db d = o1.distTo(o2);
  if (cmp(d, r1 + r2) == 1) return {};
  if (cmp(d,abs(r1-r2))==-1) return {};
  d = min(d, r1 + r2);
  db y = (r1 * r1 + d * d - r2 * r2) / (2 * d), x = sqrt(r1 * r1 - y * y);
  P dr = (o2 - o1).unit();
  P q1 = o1 + dr * y, q2 = dr.rot90() * x;
  return {q1-q2,q1+q2};//along circle 1
}

// extanCC, intanCC : -r2, tanCP : r2 = 0
vector<pair<P, P>> tanCC(P o1, db r1, P o2, db r2) {
  P d = o2 - o1;
  db dr = r1 - r2, d2 = d.abs2(), h2 = d2 - dr * dr;
  if (sign(d2) == 0|| sign(h2) < 0) return {};
  h2 = max((db)0.0, h2);
  vector<pair<P, P>> ret;
  for (db sign : {-1, 1}) {
    P v = (d * dr + d.rot90() * sqrt(h2) * sign) / d2;
```

```
    ret.push_back({o1 + v * r1, o2 + v * r2});
  }
  if (sign(h2) == 0) ret.pop_back();
  return ret;
}

db rad(P p1,P p2){
  return atan2l(p1.det(p2),p1.dot(p2));
}

db areaCT(db r, P p1, P p2){
  vector<P> is = isCL(P(0,0),r,p1,p2);
  if(is.empty()) return r*r*rad(p1,p2)/2;
  bool b1 = cmp(p1.abs2(),r*r) == 1, b2 = cmp(p2.abs2(), r*r) == 1;
  if(b1 && b2){
    P md=(is[0]+is[1])/2;
    if(sign((p1-md).dot(p2-md)) <= 0)
      return r*r*(rad(p1,is[0]) + rad(is[1],p2))/2 + is[0].det(is[1])/2;
    else return r*r*rad(p1,p2)/2;
  }
  if(b1) return (r*r*rad(p1,is[0]) + is[0].det(p2))/2;
  if(b2) return (p1.det(is[1]) + r*r*rad(is[1],p2))/2;
  return p1.det(p2)/2;
}

P inCenter(P A, P B, P C) {
  double a = (B - C).abs(), b = (C - A).abs(), c = (A - B).abs();
  return (A * a + B * b + C * c) / (a + b + c);
}

P circumCenter(P a, P b, P c) {
  P bb = b - a, cc = c - a;
  double db = bb.abs2(), dc = cc.abs2(), d = 2 * bb.det(cc);
  return a - P(bb.y * dc - cc.y * db, cc.x * db - bb.x * dc) / d;
}

P othroCenter(P a, P b, P c) {
  P ba = b - a, ca = c - a, bc = b - c;
  double Y = ba.y * ca.y * bc.y,
  A = ca.x * ba.y - ba.x * ca.y,
  x0 = (Y + ca.x * ba.y * b.x - ba.x * ca.y * c.x) / A,
  y0 = -ba.x * (x0 - c.x) / ba.y + ca.y;
  return {x0, y0};
}

pair<P,db> min_circle(vector<P> ps){
    random_shuffle(ps.begin(), ps.end());
    int n = ps.size();
    P o = ps[0]; db r = 0;
    rep(i,1,n) if(o.distTo(ps[i]) > r + EPS){
        o = ps[i], r = 0;
        rep(j,0,i) if(o.distTo(ps[j]) > r + EPS){
```

```
            o = (ps[i] + ps[j]) / 2; r = o.distTo(ps[i]);
            rep(k,0,j) if(o.distTo(ps[k]) > r + EPS){
                o = circumCenter(ps[i],ps[j],ps[k]);
                r = o.distTo(ps[i]);
            }
        }
    }
    return {o,r};
}
```

圆面积并

```
db intergal(db x,db y,db r,db L,db R){
    return r*r*(R-L) + x*r*(sinl(R) - sinl(L)) + y*r*(-cosl(R) + cosl(L));
}

db calc_area_circle(P c,db r,db L,db R){
  return intergal(c.x,c.y,r,L,R) / 2;
}

db norm(db x){
  while(x < 0) x += 2 * PI;
  while(x > 2 * PI) x -= 2 * PI;
  return x;
}

P cs[N]; db rs[N];

void work(){
  vector<int> cand = {};
  rep(i,0,m){
    bool ok = 1;
    rep(j,0,m) if(i!=j){
      if(rs[j] > rs[i] + EPS && rs[i] + cs[i].distTo(cs[j]) <= rs[j] + EPS){
        ok = 0; break;
      }
      if(cs[i] == cs[j] && cmp(rs[i],rs[j]) == 0 && j < i){
        ok = 0; break;
      }
    }
    if(ok) cand.pb(i);
  }

  rep(i,0,cand.size()) cs[i] = cs[cand[i]], rs[i] = rs[cand[i]];
  m = cand.size();

  db area = 0;

  //work
  rep(i,0,m){
    vector<pair<db,int>> ev = {{0,0},{2*PI,0}};
```

```
    int cur = 0;

    rep(j,0,m) if(j!=i){
      auto ret = isCC(cs[i],rs[i],cs[j],rs[j]);
      if(!ret.empty()){
        db l = (ret[0] - cs[i]).alpha();
        db r = (ret[1] - cs[i]).alpha();
        l = norm(l); r = norm(r);
        ev.pb({l,1});ev.pb({r,-1});
        if(l > r) ++cur;
      }
    }

    sort(ev.begin(), ev.end());
    rep(j,0,ev.size() - 1){
      cur += ev[j].se;
      if(cur == 0){
        area += calc_area_circle(cs[i],rs[i],ev[j].fi,ev[j+1].fi);
      }
    }
  }
}
```

all

```
typedef double db;
const db EPS = 1e-9;

inline int sign(db a) { return a < -EPS ? -1 : a > EPS; }

inline int cmp(db a, db b){ return sign(a-b); }

struct P {
    db x, y;
    P() {}
    P(db _x, db _y) : x(_x), y(_y) {}
    P operator+(P p) { return {x + p.x, y + p.y}; }
    P operator-(P p) { return {x - p.x, y - p.y}; }
    P operator*(db d) { return {x * d, y * d}; }
    P operator/(db d) { return {x / d, y / d}; }

    bool operator<(P p) const {
        int c = cmp(x, p.x);
        if (c) return c == -1;
        return cmp(y, p.y) == -1;
    }

    bool operator==(P o) const{
        return cmp(x,o.x) == 0 && cmp(y,o.y) == 0;
    }
```

```
    db dot(P p) { return x * p.x + y * p.y; }
    db det(P p) { return x * p.y - y * p.x; }

    db distTo(P p) { return (*this-p).abs(); }
    db alpha() { return atan2(y, x); }
    void read() { cin>>x>>y; }
    void write() {cout<<"("<<x<<","<<y<<")"<<endl;}
    db abs() { return sqrt(abs2());}
    db abs2() { return x * x + y * y; }
    P rot90() { return P(-y,x);}
    P unit() { return *this/abs(); }
    int quad() const { return sign(y) == 1 || (sign(y) == 0 && sign(x) >= 0); }
    P rot(db an){ return {x*cos(an)-y*sin(an),x*sin(an) + y*cos(an)}; }
};

struct L{ //ps[0] -> ps[1]
    P ps[2];
    P& operator[](int i) { return ps[i]; }
    P dir() { return ps[1] - ps[0]; }
    bool include(P p) { return sign((ps[1] - ps[0]).det(p - ps[0])) > 0; }
    L push(){ // push eps outward
        const double eps = 1e-6;
        P delta = (ps[1] - ps[0]).rot90().unit() * eps;
        return {{ps[0] - delta, ps[1] - delta}};
    }
};

#define cross(p1,p2,p3) ((p2.x-p1.x)*(p3.y-p1.y)-(p3.x-p1.x)*(p2.y-p1.y))
#define crossOp(p1,p2,p3) sign(cross(p1,p2,p3))

bool chkLL(P p1, P p2, P q1, P q2) {
    db a1 = cross(q1, q2, p1), a2 = -cross(q1, q2, p2);
    return sign(a1+a2) != 0;
}

P isLL(P p1, P p2, P q1, P q2) {
    db a1 = cross(q1, q2, p1), a2 = -cross(q1, q2, p2);
    return (p1 * a2 + p2 * a1) / (a1 + a2);
}

P isLL(L l1,L l2){ return isLL(l1[0],l1[1],l2[0],l2[1]); }

bool intersect(db l1,db r1,db l2,db r2){
    if(l1>r1) swap(l1,r1); if(l2>r2) swap(l2,r2);
    return !( cmp(r1,l2) == -1 || cmp(r2,l1) == -1 );
}

bool isSS(P p1, P p2, P q1, P q2){
    return intersect(p1.x,p2.x,q1.x,q2.x) && intersect(p1.y,p2.y,q1.y,q2.y) &&
    crossOp(p1,p2,q1) * crossOp(p1,p2,q2) <= 0 && crossOp(q1,q2,p1)
            * crossOp(q1,q2,p2) <= 0;
```

```
}

bool isSS_strict(P p1, P p2, P q1, P q2){
    return crossOp(p1,p2,q1) * crossOp(p1,p2,q2) < 0 && crossOp(q1,q2,p1)
            * crossOp(q1,q2,p2) < 0;
}

bool isMiddle(db a, db m, db b) {
    return sign(a - m) == 0 || sign(b - m) == 0 || (a < m != b < m);
}

bool isMiddle(P a, P m, P b) {
    return isMiddle(a.x, m.x, b.x) && isMiddle(a.y, m.y, b.y);
}

bool onSeg(P p1, P p2, P q){
    return crossOp(p1,p2,q) == 0 && isMiddle(p1, q, p2);
}

bool onSeg_strict(P p1, P p2, P q){
    return crossOp(p1,p2,q) == 0 && sign((q-p1).dot(p1-p2)) * sign((q-p2).dot(p1-p2)) < 0;
}

P proj(P p1, P p2, P q) {
    P dir = p2 - p1;
    return p1 + dir * (dir.dot(q - p1) / dir.abs2());
}

P reflect(P p1, P p2, P q){
    return proj(p1,p2,q) * 2 - q;
}

db nearest(P p1,P p2,P q){
    P h = proj(p1,p2,q);
    if(isMiddle(p1,h,p2))
        return q.distTo(h);
    return min(p1.distTo(q),p2.distTo(q));
}

db disSS(P p1, P p2, P q1, P q2){
    if(isSS(p1,p2,q1,q2)) return 0;
    return min(min(nearest(p1,p2,q1),nearest(p1,p2,q2)),
min(nearest(q1,q2,p1),nearest(q1,q2,p2)));
}

db rad(P p1,P p2){
    return atan2l(p1.det(p2),p1.dot(p2));
}

db incircle(P p1, P p2, P p3){
    db A = p1.distTo(p2);
    db B = p2.distTo(p3);
```

```
    db C = p3.distTo(p1);
    return sqrtl(A*B*C/(A+B+C));
}

//polygon

db area(vector<P> ps){
    db ret = 0; rep(i,0,ps.size()) ret += ps[i].det(ps[(i+1)%ps.size()]);
    return ret/2;
}

int contain(vector<P> ps, P p){ //2:inside,1:on_seg,0:outside
    int n = ps.size(), ret = 0;
    rep(i,0,n){
        P u=ps[i],v=ps[(i+1)%n];
        if(onSeg(u,v,p)) return 1;
        if(cmp(u.y,v.y)<=0) swap(u,v);
        if(cmp(p.y,u.y) >0 || cmp(p.y,v.y) <= 0) continue;
        ret ^= crossOp(p,u,v) > 0;
    }
    return ret*2;
}

vector<P> convexHull(vector<P> ps) {
    int n = ps.size(); if(n <= 1) return ps;
    sort(ps.begin(), ps.end());
    vector<P> qs(n * 2); int k = 0;
    for (int i = 0; i < n; qs[k++] = ps[i++])
        while (k > 1 && crossOp(qs[k - 2], qs[k - 1], ps[i]) <= 0) --k;
    for (int i = n - 2, t = k; i >= 0; qs[k++] = ps[i--])
        while (k > t && crossOp(qs[k - 2], qs[k - 1], ps[i]) <= 0) --k;
    qs.resize(k - 1);
    return qs;
}

vector<P> convexHullNonStrict(vector<P> ps) {
    //caution: need to unique the Ps first
    int n = ps.size(); if(n <= 1) return ps;
    sort(ps.begin(), ps.end());
    vector<P> qs(n * 2); int k = 0;
    for (int i = 0; i < n; qs[k++] = ps[i++])
        while (k > 1 && crossOp(qs[k - 2], qs[k - 1], ps[i]) < 0) --k;
    for (int i = n - 2, t = k; i >= 0; qs[k++] = ps[i--])
        while (k > t && crossOp(qs[k - 2], qs[k - 1], ps[i]) < 0) --k;
    qs.resize(k - 1);
    return qs;
}

db convexDiameter(vector<P> ps){
    int n = ps.size(); if(n <= 1) return 0;
    int is = 0, js = 0; rep(k,1,n) is = ps[k]<ps[is]?k:is, js = ps[js] < ps[k]?k:js;
    int i = is, j = js;
```

```
    db ret = ps[i].distTo(ps[j]);
    do{
        if((ps[(i+1)%n]-ps[i]).det(ps[(j+1)%n]-ps[j]) >= 0)
            (++j)%=n;
        else
            (++i)%=n;
        ret = max(ret,ps[i].distTo(ps[j]));
    }while(i!=is || j!=js);
    return ret;
}

vector<P> convexCut(const vector<P>&ps, P q1, P q2) {
    vector<P> qs;
    int n = ps.size();
    rep(i,0,n){
        P p1 = ps[i], p2 = ps[(i+1)%n];
        int d1 = crossOp(q1,q2,p1), d2 = crossOp(q1,q2,p2);
        if(d1 >= 0) qs.pb(p1);
        if(d1 * d2 < 0) qs.pb(isLL(p1,p2,q1,q2));
    }
    return qs;
}

//min_dist

db min_dist(vector<P>&ps,int l,int r){
    if(r-l<=5){
        db ret = 1e100;
        rep(i,l,r) rep(j,l,i) ret = min(ret,ps[i].distTo(ps[j]));
        return ret;
    }
    int m = (l+r)>>1;
    db ret = min(min_dist(ps,l,m),min_dist(ps,m,r));
    vector<P> qs; rep(i,l,r) if(abs(ps[i].x-ps[m].x)<= ret) qs.pb(ps[i]);
    sort(qs.begin(), qs.end(),[](P a,P b) -> bool {return a.y<b.y; });
    rep(i,1,qs.size()) for(int j=i-1;j>=0&&qs[j].y>=qs[i].y-ret;--j)
        ret = min(ret,qs[i].distTo(qs[j]));
    return ret;
}

int type(P o1,db r1,P o2,db r2){
    db d = o1.distTo(o2);
    if(cmp(d,r1+r2) == 1) return 4;
    if(cmp(d,r1+r2) == 0) return 3;
    if(cmp(d,abs(r1-r2)) == 1) return 2;
    if(cmp(d,abs(r1-r2)) == 0) return 1;
    return 0;
}

vector<P> isCL(P o,db r,P p1,P p2){
    db x = (p1-o).dot(p2-p1), y = (p2-p1).abs2(), d = x * x - y * ((p1-o).abs2() - r*r);
    if(sign(d) < 0) return {};
```

```
    d = max(d,0.0); P m = p1 - (p2-p1)*(x/y), dr = (p2-p1)*(sqrt(d)/y);
    return {m-dr,m+dr}; //along dir: p1->p2
}

vector<P> isCC(P o1, db r1, P o2, db r2) { //need to check whether two circles are the same
    db d = o1.distTo(o2);
    if (cmp(d, r1 + r2) == 1) return {};
    d = min(d, r1 + r2);
    db y = (r1 * r1 + d * d - r2 * r2) / (2 * d), x = sqrt(r1 * r1 - y * y);
    P dr = (o2 - o1).unit();
    P q1 = o1 + dr * y, q2 = dr.rot90() * x;
    return {q1-q2,q1+q2};//along circle 1
}

vector<P> tanCP(P o, db r, P p) {
    db x = (p - o).abs2(), d = x - r * r;
    if (sign(d) <= 0) return {}; // on circle => no tangent
    P q1 = o + (p - o) * (r * r / x);
    P q2 = (p - o).rot90() * (r * sqrt(d) / x);
    return {q1-q2,q1+q2}; //counter clock-wise
}


vector<L> extanCC(P o1, db r1, P o2, db r2) {
    vector<L> ret;
    if (cmp(r1, r2) == 0) {
        P dr = (o2 - o1).unit().rot90() * r1;
        ret.pb({{o1 + dr, o2 + dr}}), ret.pb({{o1 - dr, o2 - dr}});
    } else {
        P p = (o2 * r1 - o1 * r2) / (r1 - r2);
        vector<P> ps = tanCP(o1, r1, p), qs = tanCP(o2, r2, p);
        rep(i,0,min(ps.size(),qs.size())) ret.pb({{ps[i], qs[i]}}); //c1 counter-clock wise
    }
    return ret;
}

vector<L> intanCC(P o1, db r1, P o2, db r2) {
    vector<L> ret;
    P p = (o1 * r2 + o2 * r1) / (r1 + r2);
    vector<P> ps = tanCP(o1,r1,p), qs = tanCP(o2,r2,p);
    rep(i,0,min(ps.size(),qs.size())) ret.pb({{ps[i], qs[i]}}); //c1 counter-clock wise
    return ret;
}

db areaCT(db r, P p1, P p2){
    vector<P> is = isCL(P(0,0),r,p1,p2);
    if(is.empty()) return r*r*rad(p1,p2)/2;
    bool b1 = cmp(p1.abs2(),r*r) == 1, b2 = cmp(p2.abs2(), r*r) == 1;
    if(b1 && b2){
        if(sign((p1-is[0]).dot(p2-is[0])) <= 0 &&
            sign((p1-is[0]).dot(p2-is[0])) <= 0)
        return r*r*(rad(p1,is[0]) + rad(is[1],p2))/2 + is[0].det(is[1])/2;
```

```
        else return r*r*rad(p1,p2)/2;
    }
    if(b1) return (r*r*rad(p1,is[0]) + is[0].det(p2))/2;
    if(b2) return (p1.det(is[1]) + r*r*rad(is[1],p2))/2;
    return p1.det(p2)/2;
}

bool parallel(L l0, L l1) { return sign( l0.dir().det( l1.dir() ) ) == 0; }

bool sameDir(L l0, L l1) { return parallel(l0, l1) && sign(l0.dir().dot(l1.dir()) ) == 1; }

bool cmp (P a,  P b) {
    if (a.quad() != b.quad()) {
        return a.quad() < b.quad();
    } else {
        return sign( a.det(b) ) > 0;
    }
}

bool operator < (L l0, L l1) {
    if (sameDir(l0, l1)) {
        return l1.include(l0[0]);
    } else {
        return cmp( l0.dir(), l1.dir() );
    }
}

bool check(L u, L v, L w) {
    return w.include(isLL(u,v));
}

vector<P> halfPlaneIS(vector<L> &l) {
    sort(l.begin(), l.end());
    deque<L> q;
    for (int i = 0; i < (int)l.size(); ++i) {
        if (i && sameDir(l[i], l[i - 1])) continue;
        while (q.size() > 1 && !check(q[q.size() - 2], q[q.size() - 1], l[i])) q.pop_back();
        while (q.size() > 1 && !check(q[1], q[0], l[i])) q.pop_front();
        q.push_back(l[i]);
    }
    while (q.size() > 2 && !check(q[q.size() - 2], q[q.size() - 1], q[0])) q.pop_back();
    while (q.size() > 2 && !check(q[1], q[0], q[q.size() - 1])) q.pop_front();
    vector<P> ret;
    for (int i = 0; i < (int)q.size(); ++i) ret.push_back(isLL(q[i], q[(i + 1) % q.size()]));
    return ret;
}

P inCenter(P A, P B, P C) {
    double a = (B - C).abs(), b = (C - A).abs(), c = (A - B).abs();
    return (A * a + B * b + C * c) / (a + b + c);
}
```

```
P circumCenter(P a, P b, P c) {
    P bb = b - a, cc = c - a;
    double db = bb.abs2(), dc = cc.abs2(), d = 2 * bb.det(cc);
    return a - P(bb.y * dc - cc.y * db, cc.x * db - bb.x * dc) / d;
}

P othroCenter(P a, P b, P c) {
    P ba = b - a, ca = c - a, bc = b - c;
    double Y = ba.y * ca.y * bc.y,
    A = ca.x * ba.y - ba.x * ca.y,
    x0 = (Y + ca.x * ba.y * b.x - ba.x * ca.y * c.x) / A,
    y0 = -ba.x * (x0 - c.x) / ba.y + ca.y;
    return {x0, y0};
}
```