Contents

1 Basic

1.1 Default code

```
#define wiwihorz
#include <bits/stdc++.h>
#pragma GCC optimize("Ofast")
#pragma loop-opt(on)

#define rep(i, a, b) for(int i = a; i <= b; i++)
#define rrep(i, a, b) for(int i = b; i >= a; i--)
#define all(x) x.begin(), x.end()
#define ceil(a, b) ((a + b - 1) / (b))

#define INF 1000000000000000000
#define MOD 1000000007
#define eps (1e-9)

using namespace std;

#define int long long int
#define lld long double
#define pii pair<int, int>
#define random mt19937 rnd(chrono::steady_clock::now().
    time_since_epoch().count())

#ifdef wiwihorz
#define print(a...) cerr << "Line" << __LINE__ << ": ",
     kout("[" + string(#a) + "] = ", a)
void vprint(auto L, auto R) {while(L < R) cerr << *L <<
     " \n"[next(L) == R], ++L; }
void kout() { cerr << endl; }
template<class T1, class ... T2> void kout(T1 a, T2 ...
     e) { cerr << a << " ", kout(e...); }
#else
#define print(...) 0
#define vprint(...) 0
#endif

signed main() {
  ios::sync_with_stdio(false), cin.tie(0);

  return 0;
}
```

1.2 FasterIO

```
namespace io {
    const int SIZE = 1e7 + 10;
    char inbuff[SIZE];
    char *l, *r;
    inline void init() {
        l = inbuff;
        r = inbuff + fread(inbuff, 1, SIZE, stdin);
    }
    inline char gc() {
        if (l == r) init();
        return (l != r) ? *(l++) : EOF;
    }
    void read(int &x) {
        x = 0; char ch = gc();
        while(!isdigit(ch)) ch = gc();
        while(isdigit(ch)) x = x * 10 + ch - '0', ch =
            gc();
    }
} using io::read;
```

1.3 Check

```
g++ -std=c++14 $1 -o ac
g++ -std=c++14 $2 -o wa
g++ -std=c++14 $3 -o gen
for i in $(seq 1 10000);
do
  echo $i
  ./gen > input
```

```
  ./ac < input > out_ac
  ./wa < input > out_wa
  diff out_ac out_wa || break
done
```

2 Data Structure

2.1 Disjoint set - Path Compression

```
struct DSU {
  int n;
  vector<int> par, rk;
  void init_(int n) {
    par.assign(n + 1, 0);
    rk.assign(n + 1, 0);
    rep(i, 1, n) par[i] = i;
  }
  int find_par(int x) {
    if(par[par[x]] == par[x]) return par[x];
    else return par[x] = find_par(par[x]);
  }
  void unite(int a, int b) {
    int aa = find_par(a), bb = find_par(b);
    if(aa == bb) return;
    if(rk[aa] > rk[bb]) par[bb] = aa;
    else if(rk[bb] > rk[aa]) par[aa] = bb;
    else par[bb] = aa, rk[aa] ++;
    return;
  }
  bool same(int a, int b) {
    return find_par(a) == find_par(b);
  }
}dsu;
```

2.2 Disjoint set - Undo

```
struct DSU {
    int n, cnt, op;
    vector<int> par, sz;
    stack<pair<int*, int>> st;
    void init_(int _n) {
        n = _n, cnt = n, op = 0;
        par.assign(n + 1, 0);
        sz.assign(n + 1, 1);
        rep(i, 1, n) par[i] = i;
        return;
    }
    int find_par(int a) {
        while(par[a] != a) a = par[a];
        return par[a];
    }
    bool same(int a, int b) {
        return find_par(a) == find_par(b);
    }
    void unite(int a, int b) {
        int aa = find_par(a);
        int bb = find_par(b);
        if(aa == bb) return;
        cnt --, op ++;
        if(sz[aa] < sz[bb]) swap(aa, bb);
        st.push({&par[bb], par[bb]});
        st.push({&sz[aa], sz[aa]});
        par[bb] = aa, sz[aa] += sz[bb];
        return;
    }
    void undo() {
        assert(op);
        rep(i, 0, 1) {
          pair<int*, int> aa = st.top();
          st.pop(), *aa.first = aa.second;
        }
        op --, cnt ++;
    }
} dsu;
```

2.3 Sparse Table

```
int n, lg;
vector<int> a;
vector<vector<int>> st;
void build() {
    lg = 32 - __builtin_clz(n);
    st.assign(lg + 1, vector<int>(n + 1, -INF));
    rep(i, 1, n) st[0][i] = a[i];
    rep(i, 1, lg) rep(j, 1, n) {
        st[i][j] =
        max(
            st[i - 1][j],
            st[i - 1][j + (1 << (i - 1))]
        );
    }
    return;
}
int query(int L, int R) {
    int x = 31 - __builtin_clz(R - L + 1);
    return max(st[x][L], st[x][R - (1 << x) + 1]);
}
```

2.4 Persistent Segment Tree

```
#include <iostream>
#include <vector>
#include <algorithm>
#define rep(i, a, b) for(int i = a; i <= b; i++)
#define all(x) x.begin(), x.end()
#define MAXP 5000000
using namespace std;
int n, m, p = 0, mx;
struct node {
  int lch, rch, val;
  node() {
    lch = rch = val = 0;
  }
};
vector<node> seg(MAXP, node());
vector<int> a, v, T;
int get_new() {return ++p;}
void pull(int nd) {
  seg[nd].val = seg[seg[nd].lch].val + seg[seg[nd].rch
      ].val;
  return;
}
int get_copy(int nd) {
  ++p;
  seg[p].lch = seg[nd].lch;
  seg[p].rch = seg[nd].rch;
  seg[p].val = seg[nd].val;
  return p;
}
void modify(int pre, int cur, int L, int R, int id, int
     val) {
  if(L == R) {
    seg[cur].val += val;
  }
  else {
    int mid = (L + R) / 2;
    if(id <= mid) {
      if(!seg[pre].lch) seg[cur].lch = get_new();
      else seg[cur].lch = get_copy(seg[pre].lch);
      modify(seg[pre].lch, seg[cur].lch, L, mid, id,
          val);
    }
    else {
      if(!seg[pre].rch) seg[cur].rch = get_new();
      else seg[cur].rch = get_copy(seg[pre].rch);
      modify(seg[pre].rch, seg[cur].rch, mid + 1, R, id
          , val);
    }
    pull(cur);
  }
}
int build(int L, int R) {
  if(L == R) return get_new();
  else {
```

```
    int mid = (L + R) / 2, nd = get_new();
    seg[nd].lch = build(L, mid);
    seg[nd].rch = build(mid + 1, R);
    return nd;
  }
}
void init() {
  rep(i, 1, n) v.push_back(a[i]);
  sort(all(v)), v.resize(unique(all(v)) - v.begin());
  mx = v.size();
  rep(i, 1, n) {
    a[i] = lower_bound(v.begin(), v.end(), a[i]) - v.
        begin() + 1;
  }
  T.assign(n + 1, 0);
  T[0] = build(1, mx);
  rep(i, 1, n) {
    T[i] = get_copy(T[i - 1]);
    modify(T[i - 1], T[i], 1, mx, a[i], 1ll);
  }
  return;
}
int query(int lt, int rt, int L, int R, int k) {
  if(L == R) return v[L - 1];
  else {
    int mid = (L + R) / 2;
    int ll = seg[lt].lch, rr = seg[rt].lch;
    int vv = seg[rr].val - seg[ll].val;
    if(vv >= k) return query(seg[lt].lch, seg[rt].lch,
        L, mid, k);
    else return query(seg[lt].rch, seg[rt].rch, mid +
        1, R, k - vv);
  }
}
```

2.5 Li Chao Tree

```
struct leechao {
  const int MAXN = 100000;
  struct ln {
    lld m, k;
    int id, flag;
    ln() : m(0), k(0), id(-INF), flag(0) {}
    ln(lld _m, lld _k, int _id) {
      m = _m, k = _k, id = _id, flag = 1;
    }
    lld operator()(lld x) {
      if(!flag) return -INF;
      else return m * x + k;
    }
  };
  int n, p, root;
  vector<ln> seg;
  vector<int> lch, rch;
  void init_(int _n) {
    lch.assign(MAXN, 0);
    rch.assign(MAXN, 0);
    seg.assign(MAXN, ln());
    n = _n, p = 0, root = get_new();
  }
  int get_new() { return ++ p;}
  void insert(int nd, int L, int R, ln a) {
    if(L == R) {
      if(a(L) > seg[nd](L)) seg[nd] = a;
      return;
    }
    int mid = (L + R) / 2;

    if(a.m < seg[nd].m) swap(seg[nd], a);
    if(a(mid) > seg[nd](mid)) {
      swap(a, seg[nd]);
      if(!lch[nd]) lch[nd] = get_new();
      insert(lch[nd], L, mid, a);
    }
    else {
      if(!rch[nd]) rch[nd] = get_new();
      insert(rch[nd], mid + 1, R, a);
    }
  }
  void modify(int nd, int L, int R, int l, int r, ln a)
      {
    if(l > R || r < L) return;
    else if(l <= L && r >= R) insert(nd, L, R, a);
    else {
      int mid = (L + R) / 2;
      if(!lch[nd]) lch[nd] = get_new();
      if(!rch[nd]) rch[nd] = get_new();
      modify(lch[nd], L, mid, l, r, a);
      modify(rch[nd], mid + 1, R, l, r, a);
    }
  }
  pair<double, int> query(int nd, int L, int R, int x)
      {
    if(!nd) return {-INF, INF};
    else if(L == R) return {seg[nd](x), -seg[nd].id};
    else {
      int mid = (L + R) / 2;
      pair<double, int> p = {seg[nd](x), -seg[nd].id};
      if(x <= mid) return max(query(lch[nd], L, mid, x)
          , p);
      else return max(query(rch[nd], mid + 1, R, x), p)
          ;
    }
  }
  void add(int x0, int y0, int x1, int y1, int id) {
    if(x0 > x1) swap(x0, x1), swap(y0, y1);
    if(x0 == x1) modify(root, 1, n, x0, x0, ln(0, max(
        y0, y1), id));
    else {
      lld m = (lld)(y1 - y0) / (lld)(x1 - x0);
      lld k = y0 - m * x0;
      modify(root, 1, n, x0, x1, ln(m, k, id));
    }
  }
  int ask(int x) {
    pair<double, int> p = query(root, 1, n, x);
    return max(0ll, -p.second);
  }
} st;
```

2.6 Treap

```
#define MAXP 1000005
int n, m, p, root, a, b, c, d, e;
struct node {
  int lch, rch, pri, sz;
  int mn, val, tag, rev;
  node() {
    lch = rch = 0, sz = 1;
    pri = rand(), mn = INF;
    val = tag = rev = 0;
  }
}trp[MAXP];
int get_new() {
  ++p, trp[p].pri = rand();
  return p;
}
void pull(int nd) {
  int lch = trp[nd].lch, rch = trp[nd].rch;
  trp[nd].sz = 1 + trp[lch].sz + trp[rch].sz;
  trp[nd].mn = min(
    min(trp[nd].val, trp[lch].mn + trp[lch].tag),
    trp[rch].mn + trp[rch].tag
  );
}
void push(int nd) {
  int lch = trp[nd].lch, rch = trp[nd].rch;
  if(trp[nd].tag) {
    trp[nd].val += trp[nd].tag;
    trp[nd].mn += trp[nd].tag;
    trp[lch].tag += trp[nd].tag;
    trp[rch].tag += trp[nd].tag;
    trp[nd].tag = 0;
  }
  if(trp[nd].rev) {
    trp[lch].rev ^= 1;
    trp[rch].rev ^= 1;
    swap(trp[nd].lch, trp[nd].rch);
    trp[nd].rev = 0;
```

```
  }
  return;
}
int merge(int pre, int suf) {
  if(pre == 0) return suf;
  if(suf == 0) return pre;
  push(pre), push(suf);
  if(trp[pre].pri > trp[suf].pri) {
    trp[pre].rch = merge(trp[pre].rch, suf);
    pull(pre);
    return pre;
  }
  else {
    trp[suf].lch = merge(pre, trp[suf].lch);
    pull(suf);
    return suf;
  }
}
pii split(int nd, int k) {
  if(nd == 0) return pii(0, 0);
  push(nd);
  int lch = trp[nd].lch, rch = trp[nd].rch;
  if(trp[lch].sz >= k) {
    pii p = split(lch, k);
    trp[nd].lch = p.second;
    pull(nd);
    return pii(p.first, nd);
  }
  else {
    pii p = split(rch, k - trp[lch].sz - 1);
    trp[nd].rch = p.first;
    pull(nd);
    return pii(nd, p.second);
  }
}
void insert(int id, int val) {
  pii p = split(root, id );
  int nd = get_new();
  trp[nd].val = trp[nd].mn = val;
  root = merge(merge(p.first, nd), p.second);
  return;
}
void reve(int l, int r) {
  pii p = split(root, l - 1);
  a = p.first, p = split(p.second, r - l + 1);
  b = p.first, c = p.second;
  trp[b].rev ^= 1;
  root = merge(merge(a, b), c);
  return;
}
void revo(int l, int r, int t) {
  pii p = split(root, l - 1);
  a = p.first, p = split(p.second, r - l + 1);
  b = p.first, c = p.second;
  int k = (t % (r - l + 1) + (r - l + 1)) % (r - l + 1)
      ;
  p = split(b, r - k - l + 1);
  root = merge(merge(a, p.second), merge(p.first, c));
  return;
}
void add(int l, int r, int val) {
  pii p = split(root, l - 1);
  a = p.first, p = split(p.second, r - l + 1);
  b = p.first, c = p.second;
  trp[b].tag += val;
  root = merge(merge(a, b), c);
  return;
}
void erase(int id) {
  pii p = split(root, id);
  c = p.second;
  p = split(p.first, id - 1);
  root = merge(p.first, c);
  return;
}
int query(int l, int r) {
  pii p = split(root, l - 1);
  a = p.first, p = split(p.second, r - l + 1);
  b = p.first, c = p.second;
  int ans = trp[b].mn + trp[b].tag;
  root = merge(merge(a, b), c);
  return ans;
}
void trav(int nd) {
  if(!nd) return;
  push(nd);
  trav(trp[nd].lch);
  cerr << trp[nd].val << " ";
  trav(trp[nd].rch);
  return;
}
```

3 Graph

3.1 D and L

```
void dfs(int x, int par) {
  L[x] = D[x] = ++timestamp;
  for(auto i : mp[x]) if(i != par) {
    int to = es[i] ^ x;
    if(!D[to]) { //  
      dfs(to, i);
      L[x] = min(L[x], L[to]);
    }
    else L[x] = min(L[x], D[to]);
  }
  return;
}
```

3.2 Articulation Point

```
void dfs(int x, int par) {
  D[x] = L[x] = ++timestamp;
  int ch = 0, isap = 0;
  for(auto i : mp[x]) if(i != par) {
    if(!D[i]) {
      dfs(i, x), ch ++;
      L[x] = min(L[x], L[i]);
    }
    else L[x] = min(L[x], D[i]);
    if(L[i] >= D[x]) isap = 1;
  }
  if(x == par && ch < 2) isap = 0;
  if(isap) ap.push_back(x);
}
```

3.3 Bridge

```
void dfs(int x, int par) {
  L[x] = D[x] = ++timestamp;
  for(auto i : mp[x]) if(i != par) {
    int to = es[i] ^ x;
    if(!D[to]) {
      dfs(to, i);
      L[x] = min(L[x], L[to]);
    }
    else L[x] = min(L[x], D[to]);
    if(L[to] > D[x]) bridge.push_back(i);
  }
  return;
}
```

3.4 Biconnected Component

```
void dfs(int x, int par) {
    D[x] = L[x] = ++timestamp;
    for(auto i : mp[x]) if(i != par) {
        int to = es[i] ^ x;
        if(!D[to]) {
            pre[ii ++] = i;
            dfs(to, i);
            L[x] = min(L[x], L[to]);
            if(D[x] <= L[to]) {
                bccid ++;
```

```
                bcc.push_back(vector<int>());
                while(pre[ii] != i) {
                    ii --, bcc[bccid].push_back(pre[ii
                        ]);
                }
            }
        }
        else if(D[to] <= D[x]) {
            pre[ii ++] = i;
            L[x] = min(L[x], D[to]);
        }
    }
    return;
}
```

3.5 Strongly Connected Component

```
void dfs(int x) {
    D[x] = L[x] = ++ timestamp;
    pre[ii++] = x, instack[x] = 1;
    for(auto i : mp[x]) {
        if(!D[i]) dfs(i), L[x] = min(L[x], L[i]);
        else if(instack[i]) L[x] = min(L[x], D[i]);
    }
    if(L[x] == D[x]) {
        sccid ++;
        while(pre[ii] != x) {
            ii --, instack[pre[ii]] = 0;
            id[pre[ii]] = sccid;
        }
    }
    return;
}
```

3.6 Cactus

```
// DP on cactus
/*
Divide into three cases
1. bridge => merge
2. back edge => label it
3. forward edge => operate cycle
*/
#define wiwihorz
#include <bits/stdc++.h>
#pragma GCC optimize("Ofast")
#pragma loop-opt(on)

#define rep(i, a, b) for(int i = a; i <= b; i++)
#define rrep(i, a, b) for(int i = b; i >= a; i--)
#define all(x) x.begin(), x.end()
#define ceil(a, b) ((a + b - 1) / (b))
#define INF 1000000000
using namespace std;
namespace solver {
  int n, timestamp;
  vector<int> es, L, D, bad, pa;
  vector<vector<int>> mp, dp;
  void init_(int _n) {
    n = _n, timestamp = 0;
    es.clear();
    L.assign(n + 1, 0);
    D.assign(n + 1, 0);
    pa.assign(n + 1, 0);
    mp.assign(n + 1, vector<int>());
    dp.assign(2, vector<int>());
    dp[1].assign(n + 1, 1);
    dp[0].assign(n + 1, 0);
  }
  void add_edge(int a, int b) {
    mp[a].push_back(es.size());
    mp[b].push_back(es.size());
    es.push_back(a ^ b);
    bad.push_back(0);
    if(a == b) dp[0][a] = INF;
  }
  void dfs(int x, int par) {
    L[x] = D[x] = ++timestamp;
    for(auto i : mp[x]) if(i != par) {
      int to = es[i] ^ x;
      if(!D[to]) pa[to] = x, dfs(to, i);
      if(bad[i]) {
        vector<int> v1, v2;
        v1 = {0, INF}, v2 = {0, 0};
        for(int j = to; j != x; j = pa[j]) {
          v2[0] = v1[1] + dp[0][j];
          v2[1] = min(v1[0], v1[1]) + dp[1][j];
          v1.swap(v2);
        }
        dp[0][x] += v1[1];
        v1 = {INF, 0}, v2 = {0, 0};
        for(int j = to; j != x; j = pa[j]) {
          v2[0] = v1[1] + dp[0][j];
          v2[1] = min(v1[0], v1[1]) + dp[1][j];
          v1.swap(v2);
        }
        dp[1][x] += min(v1[0], v1[1]);
      }
      else if(D[to] < D[x]) {
        bad[i] = 1;
        L[x] = min(L[x], D[to]);
      }
      else {
        L[x] = min(L[x], L[to]);
        if(L[to] > D[x]) {
          dp[0][x] += dp[1][to];
          dp[1][x] += min(dp[1][to], dp[0][to]);
        }
      }
    }
  }
};
```

3.7 Block-Cut Tree

```
void dfs(int x, int par) {
    cnt ++, pre[ii++] = x;
    L[x] = D[x] = ++timestamp;
    for(auto i : mp1[x]) if(i != par) {
        int to = es[i] ^ x;
        if(D[to]) L[x] = min(D[to], L[x]);
        else {
            dfs(to, i);
            L[x] = min(L[x], L[to]);
            if(L[to] >= D[x]) {
                bccid ++;
                mp[n + bccid].push_back(x);
                mp[x].push_back(n + bccid);
                while(pre[ii] != to) {
                    ii --;
                    mp[n + bccid].push_back(to);
                    mp[to].push_back(n + bccid);
                }
            }
        }
    }
    return;
}
```

3.8 Centroid Decomposition

```
vector<int> es, cost, yes;
vector<vector<int>> mp;
struct centroid_deco {
  int n;
  vector<int> sz, mx, pa, dep, vis;
  vector<vector<int>> dis;
  vector<int> cnt, sum, minu;
  void init_(int _n) {
    n = _n;
    sz.assign(n + 1, 0);
    mx.assign(n + 1, 0);
    pa.assign(n + 1, 0);
    dep.assign(n + 1, 0);
    cnt.assign(n + 1, 0);
    sum.assign(n + 1, 0);
```

```
    minu.assign(n + 1, 0);
    vis.assign(n + 1, 0);
    dis.assign(32, vector<int>(n + 1, 0));
    deco(1);
    return;
  }
  void get_sz(int x, int par) {
    sz[x] = 1, mx[x] = 0;
    for(auto i : mp[x]) {
      int to = es[i] ^ x;
      if(to == par || vis[to]) continue;
      get_sz(to, x);
      sz[x] += sz[to];
      mx[x] = max(mx[x], sz[to]);
    }
  }
  int get_cen(int x, int par, int tot) {
    int best = x;
    for(auto i : mp[x]) {
      int to = es[i] ^ x;
      if(to == par || vis[to]) continue;
      int cur = get_cen(to, x, tot);
      if(max(tot - sz[cur], mx[cur]) <
        max(tot - sz[best], mx[best])) best = cur;
    }
    return best;
  }
  void get_dis(int x, int par, int d, int tot = 0) {
    dis[d][x] = tot;
    for(auto i : mp[x]) {
      int to = es[i] ^ x;
      if(to == par || vis[to]) continue;
      get_dis(to, x, d, tot + cost[i]);
    }
    return;
  }
  void deco(int x, int par = -1, int d = 0) {
    get_sz(x, x);
    int c = get_cen(x, x, sz[x]);
    vis[c] = 1, dep[c] = d, pa[c] = par;
    get_dis(c, c, d);
    for(auto i : mp[c]) {
      int to = es[i] ^ c;
      if(vis[to]) continue;
      deco(to, c, d + 1);
    }
  }
  void modify(int x) {
    int cur = x;
    while(cur != -1) {
      cnt[cur] ++ ;
      sum[cur] += dis[dep[cur]][x];
      if(dep[cur]) minu[cur] += dis[dep[cur] - 1][x];
      cur = pa[cur];
    }
  }
  int query(int x) {
    int cur = x, ans = sum[cur];
    while(pa[cur] != -1) {
      int f = pa[cur], c = cur;
      ans += sum[f] - minu[c]
        + (cnt[f] - cnt[c]) * dis[dep[f]][x];
      cur = pa[cur];
    }
    return ans;
  }
} deco;
```

3.9 Dinic

```
struct Dinic {
  struct edge {
    int to, cap, rev;
  };
  vector<int> level, iter;
  vector<vector<edge>> mp;
  int n, s, t;
  void init(int _n, int _s, int _t) {
    n = _n, s = _s, t = _t;
    mp.assign(n + 1, vector<edge>());
    level.assign(n + 1, INF);
    iter.assign(n + 1, 0);
  }
  void add_edge(int u, int v, int cap) {
    mp[u].push_back({v, cap, signed(mp[v].size())});
    mp[v].push_back({u, 0, signed(mp[u].size()) - 1});
  }
  bool bfs() {
    level.assign(n + 1, INF);
    iter.assign(n + 1 , 0);
    level[s] = 0;
    queue<int> q; q.push(s);
    while(q.size()) {
      int cur = q.front(); q.pop();
      for(auto i : mp[cur]) {
        if(level[i.to] == INF && i.cap > 0) {
          q.push(i.to);
          level[i.to] = level[cur] + 1;
        }
      }
    }
    if(level[t] == INF) return false;
    return true;
  }
  int dfs(int cur, int flow) {
    if(cur == t) return flow;
    for(int &k = iter[cur]; k < mp[cur].size(); k ++) {
      edge &i = mp[cur][k];
      if(level[i.to] != level[cur] + 1 || i.cap <= 0)
          continue;
      int res = dfs(i.to, min(flow, i.cap));
      if(res > 0) {
        i.cap -= res;
        mp[i.to][i.rev].cap += res;
        return res;
      }
    }
    return 0;
  }
  int flow() {
    int ans = 0;
    while(bfs()) {
      int res = 0;
      while(res = dfs(s, INF), res) {
        ans += res;
      }
    }
    return ans;
  }
};
```

3.10 Eular

```
int n = 50, m, s;
vector<int> es, deg, ans, vis;
vector<vector<int>> mp;
void init_() {
    es.clear(), ans.clear();
    deg.assign(n + 1, 0);
    mp.assign(n + 1, vector<int>());
    return;
}
void dfs(int x, int par) {
    for(auto i : mp[x]) {
        if(i == par || vis[i]) continue;
        vis[i] = 1, dfs(es[i] ^ x, i);
        ans.push_back(i);
    }
    return;
}
bool solve() {
    rrep(i, 1, n) {
        if(deg[i] & 1) return false;
        else if(deg[i]) s = i;
    }
    vis.assign(m, 0);
    dfs(s, -1);
    if(ans.size() == m) return true;
    else return false;
```

```
}
```

3.11 Heavy-Light Decomposition

```
vector<vector<int>> mp;
vector<int> es, cost, x, y;
struct segment_tree {
  int n;
  vector<int> seg;
  void init_(int _n) {
    n = _n;
    seg.assign(2 * n + 1, 0);
    return;
  }
  int get(int L, int R) {
    return (L + R) | (L != R);
  }
  void pull(int L, int R) {
    int mid = (L + R) / 2, nd = get(L, R);
    int lch = get(L, mid), rch = get(mid + 1, R);
    seg[nd] = max(seg[lch], seg[rch]);
    return;
  }
  void modify(int L, int R, int id, int val) {
    int mid = (L + R) / 2, nd = get(L, R);
    if(L == R) seg[nd] = val;
    else {
      if(id <= mid) modify(L, mid, id, val);
      else modify(mid + 1, R, id, val);
      pull(L, R);
    }
    return;
  }
  int query(int L, int R, int l, int r) {
    int mid = (L + R) / 2, nd = get(L, R);
    if(l > R || r < L) return -INF;
    else if(l <= L && r >= R) return seg[nd];
    else {
      return max(
        query(L, mid, l, r),
        query(mid + 1, R, l, r)
      );
    }
  }
};
struct HLD {
  int n, timestamp;
  vector<int> dep, pa, rt;
  vector<int> hson, id, sz;
  segment_tree st;
  void init_(int _n) {
    n = _n, timestamp = 0;
    dep.assign(n + 1, 0);
    pa.assign(n + 1, 0);
    rt.assign(n + 1, 0);
    hson.assign(n + 1, 0);
    id.assign(n + 1, 0);
    sz.assign(n + 1, 0);
    get_info(1, 1);
    deco(1, 1, 1);
    st.init_(n);
  }
  void get_info(int x, int par, int d = 0) {
    sz[x] = 1, pa[x] = par, dep[x] = d;
    int mxid = 0;
    for(auto i : mp[x]) {
      int to = es[i] ^ x;
      if(to == par) continue;
      get_info(to, x, d + 1);
      sz[x] += sz[to];
      if(sz[to] > sz[mxid]) mxid = to;
    }
    hson[x] = mxid;
    return;
  }
  void deco(int x, int top, int par) {
    rt[x] = top, id[x] = ++timestamp;
    if(hson[x]) deco(hson[x], top, x);
    for(auto i : mp[x]) {
      int to = es[i] ^ x;
      if(to == par || to == hson[x]) continue;
      deco(to, to, x);
    }
    return;
  }
  void change(int u, int v, int w) {
    if(dep[u] < dep[v]) swap(u, v);
    st.modify(1, n, id[u], w);
  }
  int ask(int u, int v) {
    int ans = -INF;
    while(rt[u] != rt[v]) {
      if(dep[rt[u]] < dep[rt[v]]) swap(u, v);
      ans = max(ans, st.query(1, n, id[rt[u]], id[u]));
      u = pa[rt[u]];
    }
    if(u != v) {
      if(dep[u] < dep[v]) swap(u, v);
      ans = max(ans, st.query(1, n, id[v] + 1, id[u]));
    }
    return ans;
  }
} hld;
```

3.12 Hungary

```
int n, m, ii;
vector<int> vis, match;
vector<vector<int>> mp;
void init_(int _n, int _m) {
  n = _n, m = _m ii = 0;
  mp.assign(n + 1, vector<int>());
  vis.assign(m + 1, 0);
  match.assign(m + 1, -1);
  return;
}
bool dfs(int x) {
  for(auto i : mp[x]) {
    if(vis[i] == ii) continue;
    vis[i] = ii;
    if(match[i] == -1 || dfs(match[i])) {
      match[i] = x;
      return true;
    }
  }
  return false;
}
int solve() {
  int ans = 0;
  rep(i, 1, n) ans += dfs(i);
  return ans;
}
```

3.13 Jellyfish

```
namespace solver {
  int n, ii;
  vector<int> cyc, id, len, es, to, pre, d;
  vector<vector<int>> mp;
  void init_(int _n) {
    n = _n, ii = 0, mx = 0;
    cyc.assign(n + 1, 0);
    id.assign(n + 1, 0);
    len.assign(n + 1, 0);
    es.assign(n + 1, 0);
    to.assign(n + 1, -1);
    pre.assign(n + 1, 0);
    d.assign(n + 1, 0);
    mp.assign(n + 1, vector<int>());
  }
  void dfs1(int x, int par) {
    for(auto i : mp[x]) if(i != par) {
      int t = es[i] ^ x;
      if(!pre[t]) pre[t] = i, dfs1(t, i);
      else if(!cyc[t]) {
        int cur = x;
        while(cur != t) {
```

```
        id[cur] = ++ii;
        to[pre[cur]] = ii + 1;
        cyc[cur] = 1;
        cur = es[pre[cur]] ^ cur;
      }
      id[t] = ++ii;
      to[i] = 1;
      cyc[t] = 1;
    }
  }
}
void dfs2(int x, int par, int dd, int id) {
  len[id] = max(len[id], dd);
  int ch1 = 0, ch2 = 0;
  for(auto i : mp[x]) if(i != par) {
    int t = es[i] ^ x;
    if(!cyc[t]) {
      dfs2(t, i, dd + 1, id);
      int cur = d[t];
      d[x] = max(d[x], cur);
      if(cur > ch1) swap(cur, ch1);
      if(cur > ch2) swap(cur, ch2);
    }
  }
  d[x] = max(d[x], dd);
}
void solve() {
  pre[1] = -1, dfs1(1, -1);
  rep(i, 1, n) if(cyc[i]) {
    dfs2(i, -1, 1, id[i]);
  }
  len.resize(2 * ii + 1);
  rep(i, 1, ii) len[ii + i] = len[i];
}
};
```

3.14 KM algo

```
const int P = 500;
struct KM {
  int n, ii, slack[P];
  int Lx[P], Ly[P], match[P];
  int visx[P], visy[P];
  int a[P][P];
  void init_(int _n) {
    n = _n, ii = 0;
    rep(i, 1, n) {
      slack[i] = INF;
      visx[i] = visy[i] = 0;
      Lx[i] = Ly[i] = 0;
      match[i] = -1;
      rep(j, 1, n) a[i][j] = 0;
    }
  }
  bool dfs(int x, bool aug) {
    if(visx[x] == ii) return false;
    visx[x] = ii;
    rep(i, 1, n) if(visy[i] != ii) {
      int cur = Lx[x] + Ly[i] - a[x][i];
      if(cur == 0) {
        visy[i] = ii;
        if(match[i] == -1 || dfs(match[i], aug)) {
          if(aug) match[i] = x;
          return true;
        }
      }
      else slack[i] = min(slack[i], cur);
    }
    return false;
  }
  bool augment() {
    rep(i, 1, n) if(visy[i] != ii && slack[i] == 0) {
      visy[i] = ii;
      if(match[i] == -1 || dfs(match[i], false)) {
        return true;
      }
    }
    return false;
  }
  void relabel() {
    int ans = INF;
    rep(i, 1, n) if(visy[i] != ii)
      ans = min(ans, slack[i]);
    rep(i, 1, n) {
      if(visy[i] == ii) Ly[i] += ans;
      else slack[i] -= ans;
      if(visx[i] == ii) Lx[i] -= ans;
    }
  }
  int solve() {
    rep(i, 1, n) rep(j, 1, n)
      Lx[i] = max(Lx[i], a[i][j]);
    int ans = 0;
    rep(i, 1, n) {
      rep(j, 1, n) slack[j] = INF;
      ii ++;
      if(dfs(i, true)) continue;
      while(!augment()) relabel();
      ii ++;
      assert(dfs(i, true));
    }
    rep(i, 1, n) ans += a[match[i]][i];
    return ans;
  }
}km;
signed main() {
  ios::sync_with_stdio(false), cin.tie(0);
  int n;
  while(cin >> n && n) {
    km.init_(n);
    rep(i, 1, n) rep(j, 1, n) {
      int x; cin >> x;
      km.a[i][j] = max(0ll, x);
    }
    cout << km.solve() << "\n";
  }
  return 0;
}
```

3.15 Prim

```
#include <iostream>
#include <queue>
#define int long long int
#define MAXN 200
#define rep(i, a, b) for(int i = a; i <= b; i++)
#define INF 1000000000000000000
#define pii pair<int, int>
using namespace std;
int n, mp[MAXN][MAXN];
int dis[MAXN], vis[MAXN];
int solve() {
  rep(i, 1, n) dis[i] = INF;
  rep(i, 1, n) vis[i] = 0;
  dis[1] = 0;
  priority_queue<pii, vector<pii>, greater<pii>> pq;
  pq.push(pii(0, 1));
  int ans = 0;
  while(pq.size()) {
    int cur = pq.top().second; pq.pop();
    if(vis[cur]) continue;
    ans += dis[cur], vis[cur] = 1;
    rep(i, 1, n) {
      if(!vis[i] && dis[i] > mp[cur][i]) {
        dis[i] = mp[cur][i];
        pq.push(pii(dis[i], i));
      }
    }
  }
  return ans;
}
signed main() {
  ios::sync_with_stdio(false), cin.tie(0);
  while(cin >> n) {
    rep(i, 1, n) rep(j, 1, n) {
      cin >> mp[i][j];
    }
    cout << solve() << "\n";
  }
  return 0;
```

```
}
```

3.16 Smallest Mean Cycle

```
struct min_mean_cycle {
  int n, A, B;
  vector<vector<int>> mp, dp;
  void init_(int _n) {
    n = _n , A = 1, B = 0;
    mp.assign(n + 1, vector<int>(n + 1, INF));
    dp.assign(n + 2, vector<int>(n + 1, INF));
    dp[0][0] = 0;
    rep(i, 1, n) mp[0][i] = 0;
  }
  void solve() {
    rep(i, 1, n + 1) rep(k, 0, n) rep(j, 0, n) {
      dp[i][j] = min(dp[i][j], dp[i - 1][k] + mp[k][j])
          ;
    }
    rep(j, 1, n) {
      int curA = 0, curB = 1;
      rep(i, 1, n) {
        if(dp[n + 1][j] == INF || dp[i][j] == INF)
            continue;
        int aa = dp[n + 1][j] - dp[i][j];
        int bb = n + 1 - i;
        if(curA * bb < curB * aa) curA = aa, curB = bb;
      }
      if(curA && A * curB > B * curA) A = curA, B =
          curB;
    }
    if(A == 1 && B == 0) A = -1, B = -1;
    else {
      int gcd = __gcd(A, B);
      A /= gcd, B /= gcd;
    }
  }

} a;
```

4 Math

4.1 FFT and NTT

```
/*
NTT params :
p       g r k
998244353     3   119 23
2013265921    31  15  27
206158430281  7   15  37

How to NTT :
1. w(n, m) <- a^m, w(n / 2, m) = a^2m
2. p = (2^k) * r + 1
3. we have : g ^ (p - 1) = 1
4. we need : a^(2^k) = 1
5. thus a = g^r(cycle size = 2^k)
*/

struct FFT {
  int n, k;
  const double PI = acos(-1);
  const cp I = cp(0, 1);
  vector<int> rev;
  vector<cp> omega, iomega;
  void init_(int _n) {
    assert(__builtin_popcount(_n));
    n = _n, k = 31 - __builtin_clz(n);
    rev.assign(n, 0);
    rep(i, 0, n - 1) {
      rep(j, 0, k) if((i >> j) & 1)
        rev[i] |= (1 << (k - j));
    }
    cp w = exp(2. * PI / (double) n * I);
    cp iw = exp(-2. * PI / (double) n * I)
    omega.assign(n, 1);
    iomega.assign(n, 1);
    rep(i, 1, n - 1) {
      omega[i] = omega[i - 1] * w;
      iomega[i] = iomega[i - 1] * iw;
    }
  }
  void transform(vector<cp> &a, cp* b) {
    rep(i, 0, n - 1) if(i > rev[i])
      swap(a[i], a[rev[i]]);
    rep(l, 1, k) {
      int len = (1 << l), mid = (1 << (l - 1));
      for(int i = 0; i < n; i += len) {
        rep(j, 0, mid - 1) {
          cp t = a[i + j + mid] * b[i + j];
          a[i + j + mid] = a[i + j] - t;
          a[i + j] = a[i + j] + t;
        }
      }
    }
  }
  void fft(bool inv, vector<cp> &a) {
    transform(a, (inv ? iomega : omega));
    rep(i, 0, n - 1) a[i] /= n;
  }
};
```

4.2 Chinese Remainder Theorum

```
vector<int> a, r;
pii exgcd(int a, int b) {
  if(b == 0) return pii(a > 0 ? 1 : -1, 0);
  else {
    pii p = exgcd(b, a % b);
    return pii(p.second, p.first - a / b * p.second);
  }
}
int __gcd(int a, int b) {
  if(b == 0) return a;
  else return __gcd(b, a%b);
}
int crt(int n) {
  int aa = 1, rr = 0;
  rep(i, 1, n) {
    pii p = exgcd(aa, -a[i]);
    int c = r[i] - rr, gcd = __gcd(aa, a[i]);
    if(c % gcd) return -1;
    p.first = p.first * (c / gcd) % a[i];
    rr = aa * p.first + rr;
    aa = aa / gcd * a[i];
    rr = rr % aa;
  }
  return (rr + aa) % aa;
}
signed main() {
  ios::sync_with_stdio(false), cin.tie(0);
  int n;
  while(cin >> n) {
    a.assign(n + 1, 0);
    r.assign(n + 1, 0);
    rep(i, 1, n)
      cin >> a[i] >> r[i];
    cout << crt(n) << "\n";
  }
  return 0;
}
```

4.3 Discrete Log

```
int pow_(int a, int times, int mod) {
  int ans = 1;
  for(;times > 0; times >>= 1, a = (a * a) % mod) {
    if(times & 1) ans = (ans * a) % mod;
  }
  return ans;
}
int solve(int a, int b, int n) {
  map<int, int> mp;
  int ans = INF, k = sqrt(n) + 1;
```

```
  rrep(i, 0, k) mp[pow_(a, i * k, n)] = i;
  rep(i, 0, k) {
    int cur = b * pow_(a, i, n) % n;
    if(mp.find(cur) != mp.end()) {
      int aa = mp[cur];
      if(aa * k - i >= 0)
      ans = min(ans, aa * k - i);
    }
  }
  if(ans == INF) return -1;
  else return ans;
}
signed main() {
  ios::sync_with_stdio(false), cin.tie(0);
  int a, b, n;
  while(cin >> a >> b >> n) {
    int ans = solve(a, b, n);
    if(ans == -1) cout << "NOT FOUND\n";
    else cout << ans << "\n";
  }
  return 0;
}
```

4.4 EXgcd

```
pii exgcd(int a, int b) {
  if(b == 0) return {a, 0};
  else {
    pii p = exgcd(b, a % b);
    return {p.second, p.first - a / b * p.second};
  }
}
signed main() {
  ios::sync_with_stdio(false), cin.tie(0);
  int A, B;
  while(cin >> A >> B) {
    int gcd = __gcd(A, B);
    A /= gcd, B /= gcd;
    pii ans = exgcd(A, B);
    cout << ans.first << " "
      << ans.second << " " << gcd << "\n";
  }
  return 0;
}
```

4.5 Gauss Elimination

```
int MOD, a[200][200], n;
int pow_(int a, int times) {
    int ans = 1;
    for(;times > 0; times >>= 1, a = a * a % MOD) {
        if(times & 1) ans = ans * a % MOD;
    }
    return ans;
}
void sswap(int aa, int bb) {
    rep(i, 1, n + 1) {
        swap(a[aa][i], a[bb][i]);
    }
    return;
}
int ddiv(int a, int b) {
    return a * pow_(b, MOD - 2) % MOD;
}
void solve() {
    rep(i, 1, n) {
        int piv = i;
        while(!a[piv][i]) piv ++;
        sswap(i, piv);
        rep(j, 1, n + 1) if(j != i) {
            a[i][j] = ddiv(a[i][j], a[i][i]);
        }
        a[i][i] = 1;
        rep(j, 1, n) if(j != i) {
            int mul = a[j][i];
            rep(k, 1, n + 1) {
                a[j][k] -= mul * a[i][k];
                a[j][k] %= MOD;
            }
        }
    }
    rep(i, 1, n) rep(j, 1, n + 1) {
        a[i][j] = (a[i][j] % MOD + MOD) % MOD;
    }
    return;
}
signed main() {
    ios::sync_with_stdio(false), cin.tie(0);
    cin >> n >> MOD;
    rep(i, 1, n) cin >> a[i][n + 1];
    rep(i, 1, n) rep(j, 1, n) {
        cin >> a[j][i];
    }
    solve();
    rep(i, 1, n) {
        cout << a[i][n + 1] << " \n"[i == n];
    }
    return 0;
}
```

4.6 Linear Basis

```
int n;
vector<int> a, base;
int solve() {
  if(a[n] == 0) return -1;
  base.assign(31, 0);
  rep(i, 1, n) {
    rrep(j, 0, 30) {
      if(!(a[i] >> j)) continue;
      if(!base[j]) {
        base[j] = a[i];
        break;
      }
      a[i] ^= base[j];
    }
  }
  int cnt = 0;
  rep(i, 0, 30) cnt += bool(base[i]);
  return cnt;
}
signed main() {
  ios::sync_with_stdio(false), cin.tie(0);
  cin >> n;
  a.assign(n + 1, 0);
  rep(i, 1, n) {
    cin >> a[i];
    a[i] ^= a[i - 1];
  }
  cout << solve() << "\n";
  return 0;
}
```

4.7 Miller Rabin

```
random;
int pow_(int a, int times, int mod) {
    int ans = 1;
    for(;times > 0;times>>=1, a = a * a % mod) {
        if(times & 1) ans = ans * a % mod;
    }
    return ans;
}
bool miller_rabin(int n, int d) {
    int a = rnd() % (n - 1) + 1;
    int x = pow_(a, d, n);
    if(x == 1 || x == n - 1) return true;
    else {
        while(d != (n - 1)) {
            d <<= 1;
            x = x * x % n;
            if(x == n - 1) return true;
        }
    }
    return false;
}
```

```
bool prime(int n) {
    if(n == 2) return true;
    if(!(n & 1) || n == 1) return false;
    int x = n - 1;
    while(!(x & 1)) x >>= 1;
    rep(i, 1, 10) {
        if(!miller_rabin(n, x)) return false;
    }
    return true;
}
signed main() {
    ios::sync_with_stdio(false), cin.tie(0);
    int n;
    while(cin >> n) {
        if(prime(n)) cout << "Yes\n";
        else cout << "No\n";
    }
    return 0;
}
```

4.8 Mobius Transform

```
int n;
vector<int> a, fac, p, mu;
vector<int> cnt;
void build() {
  fac.assign(MAXN, 1);
  mu.assign(MAXN, 1);
  rep(i, 2, MAXN - 1) {
    if(fac[i] == 1) {
      p.push_back(i);
      mu[i] = -1;
    }
    for(auto j : p) {
      if(i * j >= MAXN) break;
      fac[i * j] = j;
      mu[i * j] = -mu[i];
      if(i % j == 0) {
        mu[i * j] = 0;
        break;
      }
    }
  }
  return;
}
int pow_(int a, int times) {
  int ans = 1;
  for(;times > 0; times >>= 1, a = (a * a) % MOD) {
    if(times & 1) ans = (ans * a) % MOD;
  }
  return ans;
}
int solve() {
  cnt.assign(MAXN, 0);
  rep(i, 1, n) {
    int nn = sqrt(a[i]);
    rep(j, 1, nn) {
      if(a[i] % j == 0) {
        cnt[a[i] / j] ++;
        cnt[j] ++;
      }
    }
    if(nn * nn == a[i]) cnt[nn] --;
  }
  int ans = 0;
  rep(i, 1, MAXN - 1) {
    ans = (ans + (pow_(2, cnt[i]) - 1) * mu[i]) % MOD;
  }
  return (ans % MOD + MOD) % MOD;
}
signed main() {
  ios::sync_with_stdio(false), cin.tie(0);
  cin >> n, build();
  a.assign(n + 1, 0);
  rep(i, 1, n) cin >> a[i];
  cout << solve() << "\n";
  return 0;
}
```