Contents

1 Basic

1.1 Default_code

```cpp
#include <bits/stdc++.h>
using namespace std;
#define MP make_pair
#define pb push_back
#define pf push_front
#define ppb pop_back
#define ppf pop_front

#define F first
#define S second
using ll = long long;
using pii = pair<int, int>;
using pll = pair<long long, long long>;
using pdd = pair<double, double>;
#define noTLE ios::sync_with_stdio(0), cin.tie(0),cout.
    tie(0);
#define debug(x) cerr << #x << " = " << x << "\n"

int read(){
    int res = 0 , f = 1;
    char ch = getchar();
    while (ch < '0' || ch > '9'){
        if (ch == '-') f = -f;
        ch = getchar();
    }
    while (ch >= '0' && ch <= '9'){
        res = res * 10 + ch - 48;
        ch = getchar();
    }
    return res * f;
}
void print(int x){
  if(x == 0){
        putchar(48);
        return;
    }
    int len = 0, dg[20];
    while (x > 0){
        dg[++len] = x % 10;
        x /= 10;
    }
    for (int i = len; i >= 1; --i)
        putchar(dg[i] + 48);
}
```

1.2 vimrc

```
*g++ file.cpp -o file.exe(compile)*
*./file.exe(run)*
set nocp nu rnu cin ai hls is si ru sc cul ic
set wrap autowrite scs lbr sm sta
set ts=4 sw=4 mouse=a bg=dark
hi cursorline cterm=bold ctermbg=NONE
syntax enable
:inoremap {<CR> {<CR>}<Esc>ko
```

2 Python

2.1 Syntax

```python
# input many integers :
a, b = map(int, input().split())
# input array :
a = list(map(int, input().split()))
x = [int(i) for i in a]
# input 2D array
n = int(input())
a = [[int(i) for i in input().split()] for j in range(n
    )]
# set
st = set()
st.add()
st.remove()
x in st
```

```
max(st)
# list
lst = []
lst.pop(1) # pop index 1
lst.append()
# others
mp = {'A' : 1, 'B' : 2}
mp['A']
```

3 Data_structure

3.1 SegmentTree

```
struct SegmentTree{
    int seg[N * 4],lazy[N * 4];

    #define ls rt << 1
    #define rs rt << 1 | 1

    void pull(int rt){
        //seg[rt] = seg[ls] + seg[rs];
    }
    void push(int l,int r,int rt){
        if (l == r || lazy[rt] == 0)
            return;
        //
    }
    void build(int l, int r, int rt, int *data){
        if (l == r){
            seg[rt] = data[l];
            return;
        }
        int mid = l + r >> 1;
        build(l, mid, ls, data);
        build(mid + 1, r, rs, data);
        pull(rt);
    }
    void upd(int l, int r, int rt, int ql, int qr, int
        k){
        if (l >= ql && r <= qr){
            //seg[rt]
            return;
        }
        push(l, r, rt);
        int mid = l + r >> 1;
        if (ql <= mid)
            upd(l, mid, ls, ql, qr, k);
        if (qr > mid)
            upd(mid + 1, r, rs, ql, qr, k);
        pull(rt);
    }
    int qy(int l,int r, int rt, int ql, int qr){
        if (l >= ql && r <= qr)
            return seg[rt];
        push(l, r, rt);
        int mid = l + r >> 1;
        if (qr <= mid)
            return qy(l, mid, ls, ql, qr);
        if (ql > mid)
            return qy(mid + 1, r, rs, ql, qr);
        return qy(l, mid, ls, ql, qr) + qy(mid + 1, r,
            rs, ql, qr);
    }
};
```

3.2 Sparse_table

```
struct Sparse_table{
    int st[__lg(N) + 1][N]; // st[i][j] => [j, j+(1<<i)
        )

    void bd_st(int n, int *data){
        for (int i = 1; i <= n; i++)
            st[0][i] = data[i];
        for (int i = 1; (1 << i) <= n; i++)
            for (int j = 1; j + (1 << i) <= n + 1; j++)
                st[i][j] = max(st[i - 1][j], st[i - 1][
                    j + (1 << i - 1)]);
        }
    int qy(int l, int r){
        int len = __lg(r - l + 1);
        return max(st[len][l], st[len][r - (1 << len) +
            1]);
    }
};
```

3.3 Treap

```
struct Treap{
    int sz[N], ch[N][2], pri[N], val[N];
    int root, cnt, z, x, y;

    void pull(int rt) {
        sz[rt] = 1 + sz[ch[rt][0]] + sz[ch[rt][1]];
    }

    int new_node(int x){
        sz[++cnt] = 1;
        val[cnt] = x;
        pri[cnt] = rand();
        return cnt;
    }

    void split(int now, int k, int &x, int &y){
        if (!now) x = y = 0;
        else{
            if (val[now] <= k)
                x = now, split(ch[now][1], k, ch[now
                    ][1], y);
            else
                y = now, split(ch[now][0], k, x, ch[now
                    ][0]);
            pull(now);
        }
    }

    int Merge(int A,int B){
        if (!A || !B)
            return A + B;
        if (pri[A] > pri[B]){
            ch[A][1] = Merge(ch[A][1], B);
            pull(A);
            return A;
        }
        else {
            ch[B][0] = Merge(A, ch[B][0]);
            pull(B);
            return B;
        }
    }

    int kth(int now, int k){
        while(1){
            if (k <= sz[ch[now][0]])
                now = ch[now][0];
            else if (k == sz[ch[now][0]] + 1)
                return now;
            else
                k -= sz[ch[now][0]] + 1, now = ch[now
                    ][1];
        }
    }
};
```

3.4 Splay_tree

```
struct SplayTree {
    int root, tot, fa[N], ch[2][N], val[N], cnt[N], sz[
        N];
    void maintain(int x) {
        sz[x] = sz[ch[0][x]] + sz[ch[1][x]] + cnt[x];
    }
    bool get(int x) { // x is left(0) or right(1)
        return x == ch[1][fa[x]];
    }
    void clear(int x) {
        ch[0][x] = ch[1][x] = fa[x] = val[x] = sz[x] =
            cnt[x] = 0;
    }
    void rotate(int x) {
        int y = fa[x], z = fa[y], chk = get(x);
        ch[chk][y] = ch[chk ^ 1][x];

        if (ch[chk ^ 1][x])
            fa[ch[chk ^ 1][x]] = y;

        ch[chk ^ 1][x] = y;
        fa[y] = x;
        fa[x] = z;

        if (z)
            ch[y == ch[1][z]][z] = x;

        maintain(y);
        maintain(x);
    }
    void splay(int x) { // O(log n)
        for (int f = fa[x]; f = fa[x], f; rotate(x))
            if (fa[f])
                rotate(get(x) == get(f) ? f : x);

        root = x;
    }
    void ins(int x) {
        if (!root) {
            val[++tot] = x;
            cnt[tot]++;
            root = tot;
            maintain(root);
            return;
        }

        int now = root, f = 0;

        while (1) {
            if (val[now] == x) {
                cnt[now]++;
                maintain(now);
                maintain(f);
                splay(now);
                return;
            }

            f = now;
            now = ch[val[now] < x][now];

            if (!now) {
                val[++tot] = x;
                cnt[tot]++;
                fa[tot] = f;
                ch[val[f] < x][f] = tot;
                maintain(tot);
                maintain(f);
                splay(tot);
                return;
            }
        }
    }
    int rk(int x) { // the rank of value x
        int res = 0, now = root;

        while (1) {
            if (x < val[now])
                now = ch[0][now];
            else {
                res += sz[ch[0][now]];

                if (x == val[now]) {
                    splay(now);
                    return res + 1;
                }

                res += cnt[now];
                now = ch[1][now];
            }
        }
    }
    int kth(int k) { // the kth value in splay tree
        int now = root;

        while (1) {
            if (ch[0][now] && k <= sz[ch[0][now]])
                now = ch[0][now];
            else {
                k -= (sz[ch[0][now]] + cnt[now]);

                if (k <= 0) {
                    splay(now);
                    return val[now];
                }

                now = ch[1][now];
            }
        }
    }
    int pre() { // biggest integer smaller than val[
        root], ins x first and del x later
        int now = ch[0][root];

        if (!now)
            return now;

        while (ch[1][now])
            now = ch[1][now];

        splay(now);
        return now;
    }
    int nxt() { // smallest integer bigger than val[
        root], same as pre()
        int now = ch[1][root];

        if (!now)
            return now;

        while (ch[0][now])
            now = ch[0][now];

        splay(now);
        return now;
    }
    void del(int x) {
        rk(x); // splay value x to root

        if (cnt[root] > 1) {
            cnt[root]--;
            maintain(root);
            return ;
        }

        if (!ch[0][root] && !ch[1][root]) {
            clear(root);
            root = 0;
            return ;
        }

        if (!ch[0][root]) {
            int now = root;
            root = ch[1][root];
            fa[root] = 0;
            clear(now);
            return ;
        }

        if (!ch[1][root]) {
            int now = root;
            root = ch[0][root];
            fa[root] = 0;
            clear(now);
            return ;
        }

        int now = root, y = pre();
        fa[ch[1][now]] = y;
        ch[1][y] = ch[1][now];
        clear(now);
        maintain(root);
    }
    /*
    merge two splay tree:
        Let roots of two splay tree be x and y
        if x or y is null tree
            return another one
        else
            splay the biggest value x' of x tree to
                root
            set ch[1][x'] = y
            return x'
    */
};
```

3.5 Trie

```
int idx, cnt[N];
struct Trie{
    int ch[26];
} tr[N];
void add(string s){
    int u = 0;
    for (int i = 0; i < s.size(); i++){
        int w = s[i] - 'a';
        if (tr[u].ch[w] == 0)
            tr[u].ch[w] = ++idx;
        u = tr[u].ch[w];
        cnt[u]++;
    }
}
void del(string s){
    int u = 0;
    for (int i = 0; i < s.size(); i++){
        int w = s[i] - 'a';
        int nxt = tr[u].ch[w];
        cnt[nxt]--;
        if (cnt[nxt] == 0)
            tr[u].ch[w] = 0;
        u = nxt;
    }
}
bool match(string s){
    int u = 0;
    for (int i = 0; i < s.size(); i++){
        int w = s[i] - 'a';
        if (cnt[tr[u].ch[w]] > 0)
            u = tr[u].ch[w];
        else
            return false;
    }
    return true;
}
```

3.6 Persistent_SegmentTree

```
struct Persistent_ST{
    int rt[N * 20], lc[N * 20], rc[N * 20], seg[N *
        20], idx;

    void build(int l,int r,int &p, int *data){
        p = ++idx;
        if(l == r){
            seg[p] = data[l];
            return;
        }
        int mid = l + r >> 1;
        build(l, mid, lc[p], data);
        build(mid + 1, r, rc[p], data);
    }

    void upd(int l,int r,int &p,int pre,int q,int k){
        p = ++idx;
        lc[p] = lc[pre], rc[p] = rc[pre], seg[p] = seg[
            pre];
        if(l == r){
            seg[p] = k;
            return;
        }
        int mid = l + r >> 1;
        if (q <= mid)
            upd(l, mid, lc[p], lc[pre], q, k);
        else
            upd(mid + 1, r, rc[p], rc[pre], q, k);
    }

    int qy(int l,int r,int p,int q){
        if (l == r)
            return seg[p];
        int mid = l + r >> 1;
        if (q <= mid)
            return qy(l, mid, lc[p], q);
        else
            return qy(mid + 1, r, rc[p], q);
    }
};
```

3.7 Lichao_tree

```
struct Lichaotree{
    struct line{
        int m, k;
        int operator()(const int &x){
            return x * m + k;
        }
    } seg[C << 2];

    #define ls rt << 1
    #define rs rt << 1 | 1
    void build(int l, int r, int rt){
        seg[rt] = {0, (int)1e18};
        if (l == r) return;
        int mid = l + r >> 1;
        build(l, mid, ls);
        build(mid + 1, r, rs);
    }
    void ins(int l, int r, int rt, line L){
        if (l == r){
            if (L(l) < seg[rt](l))
                seg[rt] = L;
            return;
        }
        int mid = l + r >> 1;
        if (seg[rt].m < L.m)
            swap(seg[rt], L);
        if (seg[rt](mid) > L(mid)){
            swap(seg[rt], L);
            ins(l, mid, ls, L);
        }
        else
            ins(mid + 1, r, rs, L);
    }
    int qy(int l, int r, int rt, int x){
        if (l == r)
            return seg[rt](x);
        int mid = l + r >> 1;
        if (x < mid)
            return min(seg[rt](x), qy(l, mid, ls, x));
        return min(seg[rt](x), qy(mid + 1, r, rs, x));
    }
};
```

3.8 Link-Cut_tree

```
#include <bits/stdc++.h>
using namespace std;
#define MP make_pair
#define pb push_back
#define pf push_front
#define ppb pop_back
#define ppf pop_front

#define F first
#define S second

using ll = long long;
using pii = pair<int, int>;
using pll = pair<ll, ll>;
using pdd = pair<double, double>;
```

```cpp
#define noTLE ios::sync_with_stdio(0), cin.tie(0), cout
    .tie(0);
#define debug(x) cerr << #x << " = " << x << '\n'

const int N = 1e5 + 25;

struct LCT{
  int top, c[N][2], fa[N], xr[N], q[N], rev[N], val[N];
  void pull(int x) { xr[x] = xr[c[x][0]] ^ xr[c[x][1]]
      ^ val[x]; }
  void push(int x){
    if(rev[x]){
      rev[c[x][0]] ^= 1;
      rev[c[x][1]] ^= 1;
      rev[x] = 0;
      swap(c[x][0], c[x][1]);
    }
  }
  bool isroot(int x) { return c[fa[x]][0] != x && c[fa[
      x]][1] != x; }
  void rotate(int x){
    int y = fa[x], z = fa[y], l, r;
    if (c[y][0] == x)
      l = 0;
    else
      l = 1;
    r = l ^ 1;
    if (!isroot(y)){
      if (c[z][0] == y)
        c[z][0] = x;
      else
        c[z][1] = x;
    }
    fa[x] = z, fa[y] = x, fa[c[x][r]] = y;
    c[y][l] = c[x][r];
    c[x][r] = y;
    pull(y), pull(x);
  }
    void splay(int x){
    top = 1;
    q[top] = x;
    for (int i = x; !isroot(i); i = fa[i])
      q[++top] = fa[i];
    for (int i = top; i; i--)
      push(q[i]);
    while (!isroot(x)){
      int y = fa[x], z = fa[y];
      if (!isroot(y)){
        if ((c[y][0] == x) ^ (c[z][0] == y))
          rotate(x);
        else
          rotate(y);
      }
      rotate(x);
    }
  }
    void access(int x){
    for (int t = 0; x; t = x, x = fa[x])
      splay(x), c[x][1] = t, pull(x);
  }
  void makeroot(int x) { access(x), splay(x), rev[x] ^=
       1; }
  int find(int x){
    access(x), splay(x);
    while (c[x][0])
      x = c[x][0];
    return x;
  }
  void split(int x, int y) { makeroot(x), access(y),
      splay(y); }
  void cut(int x, int y){
    split(x, y);
    if (c[y][0] == x && c[x][1] == 0)
      c[y][0] = 0, fa[x] = 0;
  }
  void link(int x, int y){
    makeroot(x);
    fa[x] = y;
  }
  bool connected(int x, int y) { return find(x) == find
      (y); }
} lct;

signed main(){
  noTLE;
  int n, m;
  cin >> n >> m;
  for (int i = 1; i <= n; ++i)
    cin >> lct.val[i];
  while (m--){
    int opt, x, y;
    cin >> opt >> x >> y;
    if (opt == 0){ // xor between x and y
      lct.split(x, y);
      cout << lct.xr[y] << '\n';
    }
    else if (opt == 1){
      if (!lct.connected(x, y))
        lct.link(x, y);
    }
    else if (opt == 2){
      if (lct.connected(x, y))
        lct.cut(x, y);
    }
    else // change value
      lct.access(x), lct.splay(x), lct.val[x] = y, lct.
          pull(x);
  }
}
```

4 Flow

4.1 Dinic

```cpp
//O(MN^2)
int idx = 2, h[N], lev[N];
struct edge{
    int to, nxt, val;
}e[M];
int n, m, ed, st;
void add(int u,int v,int vl){
    e[idx].to = v;
    e[idx].nxt = h[u];
    e[idx].val = vl;
    h[u] = idx++;
}
bool bfs(){
    memset(lev, -1, sizeof(lev));
    queue<int> q;
    q.push(st);
    lev[st] = 1;
    while (!q.empty()){
        int now = q.front();
        q.pop();
        for (int i = h[now]; i != 0; i = e[i].nxt){
            int x = e[i].to, vl = e[i].val;
            if (vl && lev[x] == -1){
                lev[x] = lev[now] + 1;
                q.push(x);
            }
        }
    }
    return lev[ed] != -1;
}
int dfs(int now, int in){
    if (now == ed)
        return in;
    int out = 0;
    for (int i = h[now]; i != 0; i = e[i].nxt){
        int x = e[i].to, vl = e[i].val;
        if (vl && lev[x] == lev[now] + 1){
            int tmp = dfs(x, min(vl, in));
            e[i].val -= tmp;
            e[i ^ 1].val += tmp;
            in -= tmp;
            out += tmp;
        }
    }
    if (out == 0) lev[now] = -1;
    return out;
}
int dinic(){
    int res = 0;
    while(bfs())
        res += dfs(st, inf);
    return res;
}
```

4.2 Minimum cost maximum flow

```

int s, t;
int h[N], idx = 2, dis[N], last[N], pre[N], fl[N];
bool vis[N];
struct edge{
    int to, nxt, vl, cost;
} e[M];
void add(int u, int v, int val, int c){
    e[idx].to = v;
    e[idx].nxt = h[u];
    e[idx].cost = c;
    e[idx].vl = val;
    h[u] = idx++;
}

bool spfa(){
    fill(dis, dis + t + 1, 1e18);
    fill(fl, fl + t + 1, 1e18);
    queue<int> q;
    q.push(s);
    dis[s] = 0;
    vis[s] = 1;
    pre[s] = -1;
    while (!q.empty()){
        int now = q.front();
        q.pop();
        vis[now] = 0;
        for (int i = h[now]; i; i = e[i].nxt){
            int v = e[i].to, c = e[i].cost;
            if (e[i].vl && dis[v] > dis[now] + c){
                dis[v] = dis[now] + c;
                fl[v] = min(fl[now], e[i].vl);
                pre[v] = now;
                last[v] = i;
                if (!vis[v])
                    q.push(v), vis[v] = 1;
            }
        }
    }
    return dis[t] != 1e18;
}
int ans;
void dinic(){
    while (spfa()){
        int i = t;
        ans += fl[i] * dis[i];
        while (i != s){
            e[last[i]].vl -= fl[t];
            e[last[i] ^ 1].vl += fl[t];
            i = pre[i];
        }
    }
}
```

5 Graph

5.1 Dijkstra

```
vector<pii> adj[N];
int dis[N];
int dijkstra(int s, int t){
    priority_queue<pii, vector<pii>, greater<pii>> pq;
    for(int i = 0;i <= n; i++)
        dis[i] = 2e9;
    dis[s] = 0;
    pq.push(MP(0, s));
    while(!pq.empty()){
        int now_dis = pq.top().F, now_pos = pq.top().S;
        pq.pop();
        if(now_dis != dis[now_pos]) continue;
        if(now_pos == t) break;
        for(auto j : adj[now_pos]){
            if(dis[j.F] > dis[now_pos] + j.S){
                dis[j.F] = dis[now_pos] + j.S;
                pq.push(MP(dis[j.F], j.F));
            }
        }
    }
    return dis[t];
}
```

5.2 Kth_shrtest_path

```
priority_queue<pll, vector<pll>, greater<pll>> pq;
priority_queue<ll> best[N];
int n, k; // kth shortest path
void kth_shortest_path(){
    best[st].push(0);
    pq.push(MP(0, st));
    while (!pq.empty()){
        ll now = pq.top().S, now_dis = pq.top().F;
        pq.pop();
        if (now_dis > best[now].top()) continue;
        for (auto x : v[now]){
            if (best[x.F].size() < k){
                best[x.F].push(now_dis + x.S);
                pq.push(MP(now_dis + x.S, x.F));
            }
            else if (!best[x.F].empty() && now_dis + x.
                S < best[x.F].top()){
                best[x.F].pop();
                best[x.F].push(now_dis + x.S);
                pq.push(MP(now_dis + x.S, x.F));
            }
        }
    }
}
```

5.3 euler_tour

```
set<int> adj[N];
vector<int> ans;
void dfs(int now){
    while (!adj[now].empty()){
        int x = *adj[now].begin();
        adj[now].erase(adj[now].find(x));
        adj[x].erase(adj[x].find(now));
        dfs(x);
    }
    ans.pb(now);
}

bool euler_tour(){
    int st = 1, cnt = 0;
    for(int i = n; i; i--){
        if (adj[i].size() % 2 == 1)
            st = i, cnt++;
    }
    if (cnt > 2) return false;
    else{
        return true;
        dfs(st);
        reverse(ans.begin(), ans.end());
    }
}
```

5.4 Hungarian

```
int n;
struct bipartite_graph_matching{
    int adj[N][N], a[N], vis[N];

    void init(){
        memset(adj, 0, sizeof(adj));
        memset(a, -1, sizeof(a));
    }
    bool DFS(int x){
        if (vis[x])
            return false;
        vis[x] = 1;
        for (int i = 1; i <= n; i++){
            if (adj[x][i] && (a[i] == -1 || DFS(a[i])))
                return a[i] = x, 1;
        }
        return false;
    }
    int match(){
        int res = 0;
        for(int i = 1; i <= n; i++){
            memset(vis, 0, sizeof(vis));
            res += DFS(i);
        }
        return res;
    }
};
```

5.5 2-SAT

```
const int N; // range * 2
vector<int> v[N];
bool instk[N];
stack<int> stk;
int dfn[N], low[N], idx, blg[N], scc;
// a -> b = if a then b
void tarjan(int now){
    dfn[now] = low[now] = ++idx;
    instk[now] = 1;
    stk.push(now);
    for (auto x : v[now]){
        if (!dfn[x]){
            tarjan(x);
            low[now] = min(low[now], low[x]);
        }
        else if (instk[x])
            low[now] = min(low[now], dfn[x]);
    }
    if (dfn[now] == low[now]){
        scc++;
        for (int top = -1; top != now; stk.pop()){
            top = stk.top();
            blg[top] = scc;
            instk[top] = 0;
        }
    }
}
bool twosat(){
    for (int i = 1; i <= 2 * n; i++)
        if (!dfn[i])
            tarjan(i);
    bool ok = true;
    for (int i = 1; i <= n; i++)
        if (blg[i] == blg[i + n])
            ok = false;
    if (!ok)
        return false;
    else{
        return true;
        // construct ans
        vector<int> ans;
        for(int i = 1; i <= n; i++)
            if(blg[i] < blg[i + n])
                ans.pb(0); // choose i
            else
                ans.pb(1); // choose i + n
    }
}
```

5.6 SCC

```
void tarjan(int now){
    dfn[now] = low[now] = ++idx;
    stk.push(now);
    instk[now] = true;
    for (auto x : v[now]){
        if (!dfn[x]){
            tarjan(x);
            low[now] = min(low[now], low[x]);
        }
        else if (instk[x])
            low[now] = min(low[now], dfn[x]);
    }
    if (low[now] == dfn[now]){
        scc_idx++;
        for (int top = -1; top != now; stk.pop()){
            top = stk.top();
            blg[top] = scc_idx;
            instk[top] = false;
        }
    }
}
```

5.7 BCC

```
void tarjan(int now, int pre){
    dfn[now] = low[now] = ++idx;
    for (int i = h[now]; i; i = e[i].nxt){
        int v = e[i].to;
        if (v == pre || vis[i]) continue;
        vis[i] = vis[i ^ 1] = 1;
        stk.push(i);
        if (!dfn[v]){
            tarjan(v, now);
            low[now] = min(low[now], low[v]);
            if (low[v] >= dfn[now] && now != pre){
                bcc_cnt++;
                for (int top = -1; top != i; stk.pop())
                    {
                    top = stk.top();
                    if (blg[e[top].to] != bcc_cnt)
                        blg[e[top].to] = bcc_cnt,
                            bcc_node[bcc_cnt].pb(e[top
                            ].to);
                    if (blg[e[top ^ 1].to] != bcc_cnt)
                        blg[e[top ^ 1].to] = bcc_cnt,
                            bcc_node[bcc_cnt].pb(e[top
                            ^ 1].to);
                    bcc_edge[bcc_cnt].pb(top);
                    bcc_edge[bcc_cnt].pb(top ^ 1);
                }
            }
        }
        else
            low[now] = min(low[now], dfn[v]);
    }
    //if (now == pre && ch > 0){}
}
```

5.8 Tree_Isomorphism

```
const int MOD = 1e9 + 7;
const int bas = 107;
vector<int> v[N];
int sz[N], dep[N], h[N], p[N];
pii has[N];
int n, rtmx, rt, rtt;
void findrt(int now, int pre){
    sz[now] = 1;
    int mx = 0;
    for (auto x : v[now]){
        if (x == pre)
            continue;
        findrt(x, now);
        sz[now] += sz[x];
        mx = max(mx, sz[x]);
    }
    mx = max(mx, n - sz[now]);
    if (mx < rtmx)
        rt = now, rtmx = mx, rtt = 0;
    else if (mx == rtmx)
        rtt = now;
}
void dfs(int now, int pre){
    h[now] = dep[now] * p[1] % MOD;
    sz[now] = 1;
    for (auto x : v[now]){
        if (x == pre)
            continue;
        dep[x] = dep[now] + 1;
        dfs(x, now);
    }
    vector<pii> tmp;
    for (auto x : v[now]){
        if (x == pre)
            continue;
        tmp.pb({h[x], sz[x]});
    }
    for (auto x : tmp){
        (h[now] += x.F * p[x.S] % MOD) %= MOD;
        sz[now] += x.S;
    }
}
signed main(){
    int t;
    cin >> t;
    p[0] = 1;
    for (int i = 1; i <= 50; i++)
        (p[i] = p[i - 1] * bas) %= MOD;
    for (int k = 1; k <= t; k++){
        cin >> n;
        rtmx = MOD;
        for (int i = 0; i <= n; i++)
            v[i].clear(), sz[i] = dep[i] = 0;
        for (int i = 1; i <= n; i++){
            int x;
            cin >> x;
            if (x)
                v[i].pb(x), v[x].pb(i);
        }
        findrt(1, -1);
        dep[rt] = 1;
        dfs(rt, -1);
        has[k].F = h[rt];
        if (rtt){
            dep[rtt] = 1;
            dfs(rtt, -1);
            has[k].S = h[rtt];
        }
        if (has[k].S > has[k].F) swap(has[k].F, has[k].
            S);
    }
    // if has[i] == has[j] => tree isomorphism
}
```

6 Math

6.1 Bignumber

```
string s;
char c;
int la, lb, a[100], b[100], res[100];

void add(){
    int cy = 0;
    for(int i = 0; i < max(la, lb); i++){
        res[i] = a[i] + b[i] + cy;
        cy = res[i] / 10;
        res[i] %= 10;
    }
}

void sub(){
    int bw = 0;
    for(int i = 0; i < 100; i++){
        res[i] = a[i] - b[i] - bw;
        if (res[i] < 0)
            bw = 1, res[i] += 10;
        else
            bw = 0;
    }
}

void mul(){
    memset(res, 0, sizeof(res));
    for (int i = 0; i < la; i++){
        for (int j = 0; j < lb; j++){
            res[i+j] += a[i]*b[j];
        }
    }
    for (int i = 0; i < 100; i++){
        res[i+1] += res[i]/10;
        res[i] %= 10;
    }
}
bool cmp(int x){
    for (int i = lb-1; i >= 0; i--){
        if (a[i+x] < b[i]) return 0;
        if (a[i+x] > b[i]) return 1;
    }
    return 1;
}
void mns(int x){
    for (int i = 0; i < lb; i++)
        a[i+x] -= b[i];
}
void div(){
    memset(res, 0, sizeof(res));
    for (int i = la-lb; i >= 0; i--){
        int cnt = 0;
        while (cmp(i)){
            mns(i);
            cnt++;
        }
        res[i] = cnt;
    }
}
void print(){
    bool flag = false;
    for (int i = 99; i >= 0; i--){
        if (res[i] != 0) flag = true;
        if (flag) cout << res[i];
    }
    if (!flag) cout << 0;
    cout << "\n";
}

signed main() {
    string s;
    cin >> s;
    la = s.length();
    memset(a, 0, sizeof(a));
    for (int i = 0; i < la; i++)
        a[la - i - 1] = s[i]-'0';
    cin >> s;
    lb = s.length();
    memset(b, 0, sizeof(b));
    for (int i = 0; i < lb; i++)
        b[lb - i - 1] = s[i] - '0';
}
```

6.2 Exgcd

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

6.3 Linear_sieve

```
bool prime[N];
vector<int> p;
void linear_sieve(){
    for (int i = 0; i < N; i++)
        prime[i] = 1;
    prime[0] = prime[1] = 0;
    for (int i = 2; i < N; i++){
        if (prime[i])
            p.pb(i);
        for (auto x : p){
            if (x * i >= N)
                break;
            prime[x * i] = 0;
            if (i % x == 0)
                break;
        }

    }
}
```

6.4 Linear_inv

```
int inv[N];
void linear_inv(int p){
    inv[1] = 1;
    for (int i = 2; i < N; i++)
        inv[i] = (inv[p % i] * (p - p / i)) % p;
}
```

6.5 Gaussian_Elimination(mod)

```
int a[N][N];
int n, MOD;
void gaussian_elimination_mod(){
    for (int i = 1; i <= n; i++){
        if (a[i][i] == 0){
            int tmp = i;
            for (int j = i + 1; j <= n; j++)
                if (a[j][i]){
                    tmp = j;
                    break;
                }
            for (int j = 1; j <= n + 1; j++)
                swap(a[i][j], a[tmp][j]);
        }
        int tmp = a[i][i];
        for (int j = i; j <= n + 1; j++)
            (a[i][j] *= fpow(tmp, MOD - 2)) %= MOD;
        for (int j = 1; j <= n; j++){
            if (i == j) continue;
            tmp = a[j][i];
            for (int k = 1; k <= n + 1; k++)
                a[j][k] = ((a[j][k] - a[i][k] * tmp %
                    MOD) + MOD) % MOD;
        }
    }
    // x_i = a[i][n + 1]
}
```

6.6 Euler_phi

```
int phi[N];
bool isp[N];
vector<int> prime;
void euler_phi_function(){
    fill(isp, isp + N, 1);
    isp[0] = isp[1] = 0;
    phi[1] = 1;
    for (int i = 2; i < N; i++){
        if (isp[i])
            prime.pb(i), phi[i] = i - 1;
        for (int j = 0; i * prime[j] < N; j++){
            isp[i * prime[j]] = 0;
            if (i % prime[j] == 0){
                phi[i * prime[j]] = phi[i] * prime[j];
                break;
            }
            else
                phi[i * prime[j]] = phi[i] * phi[prime[
                    j]];
        }
    }
}
```

6.7 Chinese remainder theorem

```
// res % m[i] == r[i]
int CRT(int n, int *m, int *r){
    int M = 1, res = 0;
    for (int i = 1; i <= n; i++)
        M *= m[i];
    for (int i = 1; i <= n; i++){
        int tm = M / m[i], x, y;
        exgcd(tm, m[i], x, y);
        res = ((res + r[i] * tm * x % M)) % M;
    }
    return (res + M) % M;
}
```

6.8 Miller Rabin

```
// n < 2^32 {2, 7, 61}
// n < 2^64 {2, 325, 9375, 28178, 450775, 9780504,
    1795265022}
using ll = long long;
ll mult(ll a, ll b, ll m){
    return ((a % m) * (b % m)) % m;
}
ll fpow(ll a, ll b, ll m){
    ll r = 1;
    a %= m;
    for(;; b >>= 1){
        if(b & 1)
            (r *= a) %= m;
        (a *= a) %= m;
    }
    return r;
}
bool miller_rabin(ll a, ll n){
    if(n < 2)
        return true;
    if(n % 2 == 0)
        return n != 2;
    a %= n;
    if(!a)
        return false;
    ll u = n - 1;
    int t = 0;
    while(!(u % 2)){
        u /= 2;
        t++;
    }
    ll x = fpow(a, u, n);
    for(int i = 0; i < t; i++){
        ll nx = mult(x, x, n);
        if(nx == 1 && x != 1 && x != n - 1)
            return true;
        x = nx;
    }
    return x != 1;
}
```

6.9 Hamel Basis

```
//maximmum xor
int bas[50];
void ins(int x){
    for (int i = 20; i >= 0; i--){
        if ((x >> i) & 1){
            if (!bas[i])
                bas[i] = x;
            x ^= bas[i];
        }
    }
}
int q_mx(){
    int r = 0;
    for (int i = 20; i >= 0; i--){
        if ((r ^ bas[i]) > r)
            r ^= bas[i];
    }
    return r;
}
```

7 Geometry

7.1 Geomerty_Default

```
using pdd = pair<double, double>;
const double eps = 1e-6;
#define X first
#define Y second
pdd operator+(pdd a, pdd b){
    return pdd(a.X + b.X, a.Y + b.Y);
}
pdd operator-(pdd a, pdd b){
    return pdd(a.X - b.X, a.Y - b.Y);
}
pdd operator*(pdd a, double b){
    return pdd(a.X * b, a.Y * b);
}
pdd operator/(pdd a, double b){
    return pdd(a.X / b, a.Y / b);
}
double dot(pdd a, pdd b){
    return a.X * b.X + a.Y * b.Y;
}
double cross(pdd a, pdd b){
    return a.X * b.Y - a.Y * b.X;
}
double abs2(pdd a){
    return dot(a, a);
}
double abs(pdd a){
    return sqrt(dot(a, a));
}
int sign(double a){
    return fabs(a) < eps ? 0 : a > 0 ? 1 : -1;
}
int ori(pdd a, pdd b, pdd c){
    return sign(cross(b - a, c - a));
}
bool btw(pdd a, pdd b, pdd c){
    if (sign(ori(a, b, c)) != 0)
        return 0;
    return sign(dot(a - c, b - c)) <= 0;
}
bool seg_intersection(pdd a, pdd b, pdd c, pdd d){
    int a123 = ori(a, b, c);
    int a124 = ori(a, b, d);
    int a341 = ori(c, d, a);
    int a342 = ori(c, d, b);
    if (!a123 && !a124)
        return btw(a, b, c) || btw(a, b, d) || btw(c, d
            , a) || btw(c, d, b);
    return a123 * a124 <= 0 && a341 * a342 <= 0;
}
```

7.2 Convexhull

```
vector<pii> convexhull(){
    sort(node.begin(), node.end());
    vector<pii> hull, ans;
    for (int i = 0; i < node.size(); i++){
        while (hull.size() >= 2){
            int sz = hull.size() - 1;
            if (cross({hull[sz].X - hull[sz - 1].X,
                hull[sz].Y - hull[sz - 1].Y}, {node[i].
                X - hull[sz - 1].X, node[i].Y - hull[sz
                 - 1].Y}) >= 0)
                hull.pop_back();
            else
                break;
        }
        hull.pb(node[i]);
    }
    ans = hull;
    hull.clear();
    for (int i = 0; i < node.size(); i++){
        while (hull.size() >= 2){
            int sz = hull.size() - 1;
            if (cross({hull[sz].X - hull[sz - 1].X,
                hull[sz].Y - hull[sz - 1].Y}, {node[i].
                X - hull[sz - 1].X, node[i].Y - hull[sz
                 - 1].Y}) <= 0)
                hull.pop_back();
            else
                break;
        }
        hull.pb(node[i]);
    }
    for (int i = hull.size() - 2; i >= 0; i--)
        ans.pb(hull[i]);
    int area = 0;
    for (int i = 1; i < ans.size() ; i++)
        area += (cross(ans[i], ans[i - 1]));
    area /= 2;
    return ans;
}
```

7.3 Closest_pair

```
vector<pii> v;
double dis(int a, int b) { return sqrt((double)(v[a].F
    - v[b].F) * (v[a].F - v[b].F) + (double)(v[a].S - v
    [b].S) * (v[a].S - v[b].S)); }
bool cmpbyX(pii a, pii b) { return a.F < b.F || (a.F ==
     b.F && a.S < b.S); }
bool cmpbyY(int a, int b) { return v[a].S < v[b].S || (
    v[a].S == v[b].S && v[a].F < v[b].F); }
double solve(int l,int r){
    double d = 1 << 30;
    if (l == r)
        return d;
    if (l + 1 == r)
        return dis(l, r);
    int mid = l + r >> 1;
    double d1 = solve(l, mid);
    double d2 = solve(mid + 1, r);
    d = min(d1, d2);
    vector<int> tmp;
    for (int i = l; i <= r; i++)
        if (abs(v[i].F - v[mid].F) <= d)
            tmp.pb(i);
    sort(tmp.begin(), tmp.end(), cmpbyY);
    for (int i = 0; i < tmp.size(); i++)
        for (int j = i + 1; j < tmp.size() && v[tmp[j
            ]].S - v[tmp[i]].S < d; j++)
            d = min(d, dis(tmp[i], tmp[j]));
    return d;
}
```

7.4 Farthest_pair

```
int cross(pii a, pii b) { return a.X * b.Y - a.Y * b.X;
     }
int dis(pii a, pii b){return (a.X - b.X) * (a.X - b.X)
    + (a.Y - b.Y) * (a.Y - b.Y);}

signed main(){
    vector<pii> convex = convexhull();
    if (convex.size() == 2)
        cout << dis(convex[0], convex[1]) << '\n';
    else{
        int j = 2, ans = 0, m = convex.size();
        for (int i = 0; i < convex.size(); i++){
            while (cross({convex[i].X - convex[j].X,
                convex[i].Y - convex[j].Y}, {convex[(i
                + 1) % m].X - convex[j].X, convex[(i +
                1) % m].Y - convex[j].Y}) <= cross({
                convex[i].X - convex[(j + 1) % m].X,
                convex[i].Y - convex[(j + 1) % m].Y}, {
                convex[(i + 1) % m].X - convex[(j + 1)
                % m].X, convex[(i + 1) % m].Y - convex
                [(j + 1) % m].Y}))
                ans = max(ans, max(dis(convex[i],
                    convex[j]), dis(convex[(i + 1) % m
                    ], convex[j]))), (j += 1) %= m;
            ans = max(ans, max(dis(convex[i], convex[j
                ]), dis(convex[(i + 1) % m], convex[j])
                ));
        }
        cout << ans << '\n';
    }
}
```

7.5 Smallest_enclosing_circle

```
const double eps = 1e-8;
const int N = 5;
pdd p[N], O;
double r;
double dis(pdd a, pdd b) { return sqrt((a.X - b.X) * (a
    .X - b.X) + (a.Y - b.Y) * (a.Y - b.Y)); }
void solve(int i, int j, int k){
    double a = p[j].X - p[i].X;
    double b = p[j].Y - p[i].Y;
    double c = (p[j].X * p[j].X - p[i].X * p[i].X) / 2
        + (p[j].Y * p[j].Y - p[i].Y * p[i].Y) / 2;
    double d = p[k].X - p[i].X;
    double e = p[k].Y - p[i].Y;
    double f = (p[k].X * p[k].X - p[i].X * p[i].X) / 2
        + (p[k].Y * p[k].Y - p[i].Y * p[i].Y) / 2;
    O.X = (c * e - b * f) / (a * e - b * d), O.Y = (b *
         d - a * b) / (c * d - a * e);
    r = dis(O, p[i]);
}
pair<pii, int> smallest_enclosing_circle(){
    random_shuffle(p + 1, p + n + 1);
    O = p[1], r = 0;
    for (int i = 2; i <= n; i++){
        if (dis(p[i], O) > r + eps){
            O = p[i], r = 0;
            for (int j = 1; j < i; j++){
                if (dis(O, p[j]) > r + eps){
                    O.X = (p[i].X + p[j].X) / 2;
                    O.Y = (p[i].Y + p[j].Y) / 2;
                    r = dis(O, p[j]);
                    for (int k = 1; k < j; k++){
                        if (dis(O, p[k]) > r + eps)
                            solve(i, j, k);
                    }
                }
            }
        }
    }
    return MP(O, r);
}
```

7.6 Rectangles_area

```
const int N;
struct Node{
    int x, y1, y2, ok; //left bound 1, right bound 1
    bool operator <(const Node &tmp)const{
        return x < tmp.x;
    }
} node[N * 2];
struct Seg{
    int len, sum;
} seg[N * 8];
void pull(int l,int r,int rt){
    if (seg[rt].sum > 0) seg[rt].len = r - l + 1;
    else if (r != l) seg[rt].len = seg[ls].len + seg[rs
        ].len;
    else seg[rt].len = 0;
}

void upd(int l,int r,int rt,int ql,int qr,int k){
    if (r + 1 <= ql || l >= qr) return;
    if (l >= ql && r + 1 <= qr){
        seg[rt].sum += k;
        pull(l, r, rt);
        return;
    }
    int mid = l + r >> 1;
    upd(l, mid, ls, ql, qr, k);
    upd(mid + 1, r, rs, ql, qr, k);
    pull(l, r, rt);
}

int rectangles_area(){
    sort(node, node + 2 * n);
    int last = node[0].x;
    long long ans = 0;
    for (int i = 0; i < n; i++){
        ans += 1LL * (node[i].x - last) * seg[1].len;
        upd(1, N + 1, 1, node[i].y1, node[i].y2, node[i
            ].ok);
        last = node[i].x;
    }
    cout << ans << '\n';
}
```

8 String

8.1 KMP

```
int f[N]; // failure function, longest common prefix
    and suffix(s[0~f[i]-1] == s[i-f[i]+1~i])
// f[i + 1] => s[i]
vector<int> match(string a,string b){
    vector<int> ans;
    f[0] = -1, f[1] = 0;
    for (int i = 1, j = 0; i < b.size(); f[++i] = ++j){
        if (b[i] == b[j])
            f[i] = f[j];
        while (j != -1 && b[i] != b[j])
            j = f[j];
    }
    for (int i = 0, j = 0; i - j + b.size() <= a.size()
    ; ++i, ++j){
        while (j != -1 && a[i] != b[j])
            j = f[j];
        if (j == b.size() - 1)
            ans.pb(i - j);
    }
    return ans;
}
```

8.2 Z_algorithm

```
//z[i] = longest common prefix(s[0~z[i]] == s[i~i+z[i
    ]])
//match: let s = a + b, calculate Z value of s
int z[N];
vector<int> Z_val(string s){
    int l = 0, r = 0;
    vector<int> ans;
    for (int i = 0; i < s.size(); i++){
        z[i] = max(0, min(z[i - l], r - i));
        while (i + z[i] < s.size() && s[z[i]] == s[i +
            z[i]])
            l = i, z[i]++, r = i + z[i];
        if (i + z[i] == s.size())
            ans.pb(s.size() - i);
    }
    return ans;
}
```

8.3 Smallest_rotation

```
string smallest_rotation(string s) {
    int sz = s.size(), i = 0, j = 1;
    s += s;
    while (i < sz && j < sz) {
        int k = 0;
        while (k < sz && s[i + k] == s[j + k]) k++;
        if (s[i + k] <= s[j + k])
            j += k + 1;
        else
            i += (k + 1);
        if (i == j)
            j++;
    }
    int ans = i < sz ? i : j;
    return s.substr(ans, sz);
}
```

8.4 Manacher

```
int f[N];
int manacher(string tmp){
    string tmp, s;
    cin >> tmp;
    for (int i = 0; i < tmp.size(); i++){
        s += '*';
        s += tmp[i];
    }
    s += '*';
    int l = 0, r = -1, ans = 0;
    for (int i = 0; i < s.size(); i++){
        f[i] = min(r - i + 1, f[r + l - i]);
        while (i - f[i] >= 0 && i + f[i] < s.size() &&
            s[i - f[i]] == s[i + f[i]])
            f[i]++;
        f[i]--;
        if (i + f[i] > r){
            r = i + f[i];
            l = i - f[i];
        }
        ans = max(ans, f[i]);
    }
    cout << ans << '\n';
}
```

8.5 AC_automaton

```
// fail[i] point to the longest prefix == longest
    suffix of i
const int N = 1e6 + 25;
struct AC_automaton{
    int tr[26][N], fail[N], e[N], idx = 0;
    void clear(){
        for (int i = 0; i <= idx; i++){
            e[i] = fail[i] = 0;
            for (int j = 0; j < 26; j++)
                tr[j][i] = 0;
        }
        idx = 0;
    }
    void ins(string s){
        int now = 0;
        for (int i = 0; i < s.size(); i++){
            if (!tr[s[i] - 'a'][now])
                tr[s[i] - 'a'][now] = ++idx;
            now = tr[s[i] - 'a'][now];
        }
        e[now]++;
    }
    void build(){
        queue<int> q;
        for (int i = 0; i < 26; i++)
            if (tr[i][0])
                q.push(tr[i][0]);
        while (!q.empty()){
            int now = q.front();
            q.pop();
            for (int i = 0; i < 26; i++){
                if (tr[i][now]){
                    fail[tr[i][now]] = tr[i][fail[now
                        ]];
                    q.push(tr[i][now]);
                }
                else
                    tr[i][now] = tr[i][fail[now]];
            }
        }
    }
    int query(string s){// calculate how many s_i in S
        int now = 0, ans = 0;
        for (int i = 0; i < s.size(); i++){
            now = tr[s[i] - 'a'][now];
            for (int j = now; j && e[j] != -1; j = fail
                [j])
                ans += e[j], e[j] = -1;
        }
        return ans;
    }
} ac;
```

8.6 Suffix_Array

```
const int N = 1e6 + 25;
int sa[N], x[N], y[N], cnt[N];
// sa[i] = i-th smallest suffix's index (1-base)
// O(nlogn)
void build_SA(string s){
    int n = s.size(), m = 256;
    for (int i = 1; i <= n; i++)
        x[i] = s[i - 1], cnt[x[i]]++;
    for (int i = 2; i <= m; i++)
        cnt[i] += cnt[i - 1];
    for (int i = n; i; i--)
        sa[cnt[x[i]]--] = i;
    for (int k = 1; k <= n; k <<= 1){
        int id = 0;
        for (int i = n - k + 1; i <= n; i++)
            y[++id] = i;
        for (int i = 1; i <= n; i++)
            if (sa[i] > k)
                y[++id] = sa[i] - k;
        for (int i = 0; i <= m; i++)
            cnt[i] = 0;
        for (int i = 1; i <= n; i++)
            cnt[x[i]]++;
        for (int i = 2; i <= m; i++)
            cnt[i] += cnt[i - 1];
        for (int i = n; i; i--){
            sa[cnt[x[y[i]]]--] = y[i];
            y[i] = 0;
        }
        swap(x, y);
        id = 1, x[sa[1]] = 1;
        for (int i = 2; i <= n; i++){
            if (y[sa[i]] == y[sa[i - 1]] && y[sa[i] + k
                ] == y[sa[i - 1] + k])
                x[sa[i]] = id;
            else
                x[sa[i]] = ++id;
        }
        if (id == n)
            break;
        m = id;
    }
}
```

9 Others

9.1 CDQ

```
struct node{
    int y, z, id;
};
vector<node> a[N]; // (y, z, time)
bool cmp(node A, node B) { return A.y < B.y; }
int bit[N], ans[N];
int n;
void upd(int p, int k){
    for (int i = p; i < N; i += i & -i)
        bit[i] += k;
}
int qy(int p){
    int res = 0;
    for (int i = p; i; i -= i & -i)
        res += bit[i];
    return res;
}
void solve(int l, int r){
    if (l == r)
        return;
    int mid = l + r >> 1;
    solve(l, mid);
    solve(mid + 1, r);
    vector<node> left, right;
    for (int i = l; i <= mid; i++)
        for (auto x : a[i])
            left.pb(x);
    for (int i = mid + 1; i <= r; i++)
        for (auto x : a[i])
            right.pb(x);
    sort(left.begin(), left.end(), cmp);
    sort(right.begin(), right.end(), cmp);
    for (auto x : right)
        upd(x.z, 1);
    int j = 0;
    for (int i = 0; i < left.size(); i++){
        while (j < right.size() && right[j].y <= left[i
            ].y)
            upd(right[j++].z, -1);
        ans[left[i].id] += (qy(n) - qy(left[i].z));
    }
    for (j; j < right.size(); j++)
        upd(right[j].z,-1);
}
```

9.2 Digital_dp

```
int dp[N], a[N]; //dp[]... record everything you want,
    a[] record the number
//init dp => -1
int dfs(int pos, bool lim, bool zero){ //dfs(pos,
    mx_number?, leading_zero?, ...)
    if (pos <= 0)
        return ; //something
    if (!lim && !zero && dp[pos] != -1)
        return dp[pos];
    int up = lim ? a[pos] : 1;
    int res = 0;
    for (int i = 0; i <= up; i++)
        res += dfs(pos - 1, lim && (i == up), zero && (
            i == 0));
    if (!lim && !zero) dp[pos] = res;
    return res;
}

int solve(int now){
    int len = 0;
    for (; now; now /= 10)
        a[++len] = now % 10;
    return dfs(len, 1, 1);
}
```

9.3 Matrix_fpow

```
#define matrix vector<vector<int>>
matrix operator*(const matrix &a, const matrix &b){
    matrix c = vector<vector<int>>(a.size(), vector<int
        >(b[0].size(), 0));
    for (int i = 0; i < a.size(); i++)
        for (int j = 0; j < b[0].size(); j++)
            for (int k = 0; k < b.size(); k++)
                (c[i][j] += a[i][k] * b[k][j]);
    return c;
}
matrix fpow(matrix &a, int p){
    matrix I;
    for(int i = 0;i < a.size(); i++){
        vector<int> tmp;
        for(int j = 0; j < a.size(); j++)
            if(j == i)
                tmp.pb(1);
            else
                tmp.pb(0);
        I.pb(tmp);
    }
    for (; p; p >>= 1){
        if (p & 1)
            I = I * a;
        a = a * a;
    }
    return c;
}
```

9.4 Mo's_algorithm

```
struct query{
    int l, r, id, bid;
    bool operator<(const query& tmp) const{ return bid
        < tmp.bid || (bid == tmp.bid && r < tmp.r) ;}
};
void add(int x){
    //do something
}

void sub(int x){
    //do something
}
signed main(){
    cin >> n;
    for(int i = 0; i < n; i++) cin >> a[i];
    vector<query> Q;
    int k = sqrt(n);
    for(int i = 0; i < q; i++){
        int l, r;
        cin >> l >> r;
        Q.pb({l, r, i, l / k});
    }
    int l = 0, r = -1;
    for(int i = 0; i < q; i++){
        while(l < Q[i].l) sub(a[l++]);
        while(l > Q[i].l) add(a[--l]);
        while(r < Q[i].r) add(a[++r]);
        while(r > Q[i].r) sub(a[r--]);
        ans[Q[i].id] = // answer
    }
}
```

9.5 time_segment_tree

```
const int N = 1e5 + 25;
int ans;
int f[N], sz[N], res[N];
map<pii, int> mp;
vector<pii> seg[N << 2];
stack<pii> ud_sz, ud_f;
#define ls rt << 1
#define rs rt << 1 | 1

int Find(int x){
  return f[x] == x ? x : Find(f[x]);
}
bool uni(int a, int b){
  int p = Find(a), q = Find(b);
  if (p == q)
    return 0;
  if (sz[p] < sz[q])
    swap(p, q);
  ud_sz.push(MP(p, sz[p]));
  ud_f.push(MP(q, f[q]));
  ans--;

  sz[p] += sz[q];
  f[q] = p;
  return 1;
}
void upd(int rt, int l, int r, int ql, int qr, pii edg)
    {
  if (l >= ql && r <= qr){
    seg[rt].pb(edg);
    return;
  }
  int mid = l + r >> 1;
  if (ql <= mid)
    upd(ls, l, mid, ql, qr, edg);
  if (qr > mid)
    upd(rs, mid + 1, r, ql, qr, edg);
}
void traversal(int rt, int l, int r){
  int cnt = 0;
  for (auto i : seg[rt]){
    if (uni(i.F, i.S))
      cnt++;
  }

  if (l == r)
    res[l] = ans;
  else{
    int mid = l + r >> 1;
    traversal(ls, l, mid);
    traversal(rs, mid + 1, r);

  }
  while (cnt--){
    pii x = ud_sz.top(), y = ud_f.top();
    ud_sz.pop();
    ud_f.pop();
    sz[x.F] = x.S;
    f[y.F] = y.S;
    ans++;
  }
}
signed main(){
  noTLE;
  int n, m, k;
  cin >> n >> m >> k;
  ans = n;
  for (int i = 1; i <= n; ++i)
    f[i] = i, sz[i] = 1;
  for (int i = 0; i < m; ++i){
    int a, b;
    cin >> a >> b;
    if (a > b)
      swap(a, b);
    mp[MP(a, b)] = 0;
  }
  for (int i = 1; i <= k; ++i){
    int t, a, b;
    cin >> t >> a >> b;
    if (a > b)
      swap(a, b);
    if (t == 1)
      mp[MP(a, b)] = i;
    else{
      upd(1, 0, k, mp[MP(a, b)], i - 1, MP(a, b));
      mp.erase(mp.find(MP(a, b)));
    }
  }
  for (auto i : mp)
    upd(1, 0, k, i.S, k, i.F);
  traversal(1, 0, k);
  for (int i = 0; i <= k; ++i)
    cout << res[i] << ' ';
  cout << '\n';
}
```