A

bfs

```cpp
/**
 * @see https://www.luogu.com.cn/problem/P1451
 */
#include <bits/stdc++.h>

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

int main() {
    int n, m;
    std::cin >> n >> m;

    std::vector<std::vector<int>> mp(n, std::vector<int>(m));
    for (int i = 0; i < n; ++i) {
        std::string row;
        std::cin >> row;
        for (int j = 0; j < m; ++j) {
            mp[i][j] = row[j] - '0';
        }
    }

    std::vector<std::vector<bool>> vis(n, std::vector<bool>(m, false));
    int ans = 0;
    auto bfs = [&](int x, int y) {
        int n = mp.size(), m = mp[0].size();
        std::queue<std::pair<int, int>> q;
        q.push({x, y});
        vis[x][y] = true;

        while (!q.empty()) {
            auto [rx, ry] = q.front();
            q.pop();

            for (int i = 0; i < 4; ++i) {
                int nx = rx + dx[i];
                int ny = ry + dy[i];
                if (nx >= 0 && nx < n && ny >= 0 && ny < m && !vis[nx][ny]
&& mp[nx][ny] != 0) {
                    vis[nx][ny] = true;
                    q.push({nx, ny});
                }
            }
        }
    };

    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < m; ++j) {
            if (mp[i][j] != 0 && !vis[i][j]) {
                bfs(i, j);
                ++ans;
```

```
            }
        }
    }

    std::cout << ans << std::endl;
    return 0;
}
```

B

二分

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

using i64 = long long;

int main() {
    int n, c;
    std::cin >> n >> c;
    std::vector<int> a(n);
    for(int i = 0; i < n; ++i) {
        std::cin >> a[i];
    }
    std::sort(a.begin(), a.end());
    i64 ans = 0;
    for(int i = 0; i < n; ++i) {
        auto end = std::upper_bound(a.begin(), a.end(), a[i] + c) -
a.begin();
        auto start = std::lower_bound(a.begin(), a.end(), a[i] + c) -
a.begin();
        ans += end - start;
    }
    std::cout << ans << std::endl;
}
```

C

并查集

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

const int N = 5e4 + 3;
int fa[N];

void init(int n) {
    for(int i = 1; i <= n; ++i) {
        fa[i] = i;
    }
}

int query(int x) {
    if(x == fa[x]) {
        return x;
    }
```

```
    return fa[x] = query(fa[x]);
}

void merge(int x, int y) {
    fa[query(y)] = query(x);
}

bool same(int x, int y) {
    return query(x) == query(y);
}

int main() {
    int n, m, p;
    std::cin >> n >> m >> p;
    init(n);
    while(m--) {
        int mi, mj;
        std::cin >> mi >> mj;
        merge(mi, mj);
    }
    while(p--) {
        int pi, pj;
        std::cin >> pi >> pj;
        std::cout << (same(pi, pj) ? "Yes" : "No") << std::endl;
    }
}
```

D

模拟

```
#include <iostream>
#include <list>

const int N = 1e5 + 9;

std::list<int> nums;
std::list<int>::iterator pos[N];
bool vis[N];

int main() {
    int n;
    std::cin >> n;

    nums.push_front(1);
    pos[1] = nums.begin();

    for (int i = 2; i <= n; ++i) {
        int k, p;
        std::cin >> k >> p;

        if (p == 0) {
            pos[i] = nums.insert(pos[k], i);
        } else {
            auto next = std::next(pos[k]);
```

```
            pos[i] = nums.insert(next, i);
        }
    }

    int m;
    std::cin >> m;
    while (m--) {
        int x;
        std::cin >> x;
        if (!vis[x]) {
            nums.erase(pos[x]);
        }
        vis[x] = true;
    }

    for (auto num : nums) {
        std::cout << num << ' ';
    }
    std::cout << std::endl;
}
```

E

模拟

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

struct Diary {
    int id, cnt;
};

struct Cell {
    std::stack<Diary> diaries;
};

int main() {
    int n, m, k, t;
    std::cin >> n >> m >> k >> t;

    std::vector<std::vector<std::stack<Diary>>> box(n,
std::vector<std::stack<Diary>>(m));
    std::vector<std::vector<int>> indexes(n, std::vector<int>(m, 0));

    for (int i = 0; i < t; ++i) {
        int a, x, y;
        std::cin >> a >> x >> y;

        int x1 = x - 1;
        int y1 = y - 1;

        if (box[x1][y1].size() < k) {
            std::cout << -1 << std::endl;
        } else {
            auto st = box[x1][y1];
            int min_id = INT_MAX;
```

```
            while(!st.empty()) {
                min_id = std::min(min_id, st.top().id);
                st.pop();
            }

            std::stack<Diary> tmp;
            while(!box[x1][y1].empty() && box[x1][y1].top().id != min_id) {
                tmp.push(box[x1][y1].top());
                box[x1][y1].pop();
            }
            box[x1][y1].pop();

            std::cout << min_id << ' ' << tmp.size() << std::endl;

            while(!tmp.empty()) {
                box[x1][y1].push(tmp.top());
                tmp.pop();
            }
        }
        box[x1][y1].push({a, int(box[x1][y1].size())});
    }
}
```

F

只需要分别求出从根节点到两个子节点的路径异或和，再将两者异或即可消掉公共的路径部分，留下两子节点的路径异或和

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

const int N = 1e5 + 3;

struct Edge {
    int to, next, weight;
} edge[N << 1];

int head[N], cnt;

int xor_path[N]; // xor_path[i] = j：表示节点i到根节点的路径异或和为j

void init() {
    for(auto& h : head) {
        h = -1;
    }
    for(auto& e : edge) {
        e.next = -1;
    }
    cnt = 0;
}

void add_edge(int from, int to, int weight) {
    edge[cnt].to = to;
    edge[cnt].weight = weight;
    edge[cnt].next = head[from];
    head[from] = cnt++;
```

```cpp
}

void for_each(int cur, const std::function<void(int)>& func) {
    for(int i = head[cur]; ~i; i = edge[i].next) {
        func(i);
    }
}

void dfs(int cur, int parent) {
    for_each(cur, [&](int i) {
        int neighbor = edge[i].to;
        int weight = edge[i].weight;
        if(neighbor == parent) {
            return; // continue
        }
        xor_path[neighbor] = xor_path[cur] ^ weight;
        dfs(neighbor, cur);
    });
}

int main() {
    int n;
    std::cin >> n;
    init();
    for(int i = 1; i <= n - 1; ++i) {
        int u, v, w;
        std::cin >> u >> v >> w;
        add_edge(u, v, w);
        add_edge(v, u, w);
    }

    dfs(1, -1);

    int m;
    std::cin >> m;
    while(m--) {
        int u, v;
        std::cin >> u >> v;
        std::cout << (xor_path[u] ^ xor_path[v]) << std::endl;
    }
}
```