run.sh

```
g++ input.cpp -o input
g++ hacked.cpp -o hacked
g++ std.cpp -o std

for ((cnt = 1;;cnt++))
do
    echo -ne "\r${cnt}th hack"

    ./input > data.in
    ./hacked < data.in > data.result
    ./std < data.in > data.out

    if ! cmp -s data.out data.result
    then
        echo -e "\nhack success"
        break
    fi
done
```

data.cpp

```
#include <iostream>
#include <algorithm>
#include <chrono>
#include <random>
#include <set>
#define endl '\n'
#define ll long long
using namespace std;
mt19937 rng(chrono::steady_clock::now().time_since_epoch().count());
ll r(ll L, ll R)
{
    return uniform_int_distribution<ll>(L, R)(rng);
}
vector<pair<int, int>> edge;
void graph(int n, int root = -1, int m = -1)
{
    vector<pair<int, int>> t;
    for (int i = 1; i < n; i++)
    { // 先建立一棵以0为根节点的树
        t.emplace_back(i, r(0, i - 1));
    }
    set<pair<int, int>> uni;
    if (root == -1)
        root = r(0, n - 1); // 确定根节点
    for (auto [x, y] : t)
    { // 偏移建树
        x = (x + root) % n + 1;
        y = (y + root) % n + 1;
        edge.emplace_back(x, y);
        uni.emplace(x, y);
    }
    if (m != -1)
    { // 如果是图，则在树的基础上继续加边
        for (int i = n; i <= m; i++)
        {
            while (true)
            {
                int x = r(1, n), y = r(1, n);
                if (x == y)
                    continue; // 拒绝自环
                if (uni.count({x, y}))
                    continue; // 拒绝重边
                edge.emplace_back(x, y);
                uni.emplace(x, y);
            }
        }
    }
    random_shuffle(edge.begin(), edge.end()); // 打乱节点
    // for (auto [x, y] : edge)
    // {
    //     cout << x << " " << y << endl;
    // }
}
vector<int> ans;
vector<int> edgek[50];
void dfs(int u, int fa){
    ans.push_back(u);
    for (auto v : edgek[u]){
        if (v == fa) continue;
        dfs(v, u);
    }
}
void solve()
{
    int n = r(3, 20), qk = r(1, 3);
    graph(n);
    for (auto [x, y] : edge){
        edgek[x].push_back(y);
        edgek[y].push_back(x);
    }
    cout << n << " " << qk << endl;
    dfs(1, 0);
    for (auto [u, v] : edge){
        cout << u << " " << v << endl;
    }
    vector<int> ansk(n + 5);
    for (int i = 1; i <= n; i++) ansk[i] = i;
    random_shuffle(ansk.begin() + 1, ansk.begin() + 1 + n);
```

```cpp
    for (int i = 1; i <= qk; i++){
        int L = r(0, ans.size() - 2), R = r(L + 1, ans.size() - 1);
        cout << R - L + 1 << " ";
        int f = r(0, 1);
        for (int j = L; j <= R; j++){
            if (f)cout << ans[j] << " ";
            else cout << ansk[j] << " ";
        }
        cout << endl;
    }
}
signed main()
{
    ios::sync_with_stdio(false);
    cin.tie(nullptr), cout.tie(nullptr);
    int t = 1;
    // cout << t << endl;
    while (t--)
    {
        solve();
    }
}
```