Crack-A-Hack

Black White Tree

Course: Algorithmic Problem Solving

Course Code: 17ECSE309

Aditya Kumar Verma

USN: 01FE15BCS008

Introduction

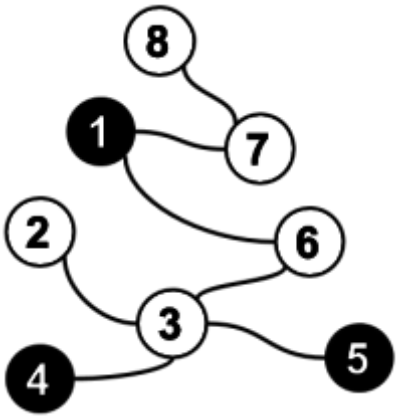
*A **tree** is a connected graph with no cycles and there's only one way to get from one node to another, but this isn't true in general graphs.*

Signed strangeness of a tree is simply the difference between the number of black nodes and the number of white nodes in the tree, without the absolute value.

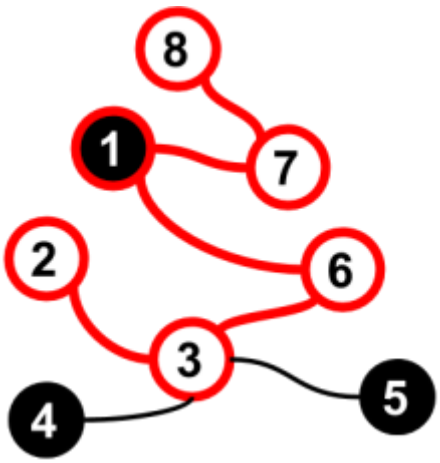
Problem

Given a tree with black and white nodes, find a subtree from the tree which have maximum strangeness.

SAMPLE INPUT



EXPECTED OUTPUT



Solution

The first step is to find the maximum and minimum possible signed strangeness of any subtree. We can compute it using dynamic programming*: Let Smax(i) be the maximum possible signed strangeness for any tree rooted at node i.*

Now, any tree rooted at i will contain i itself, so the color of i affects Smax(i) by either 1 or -1. However, we can be greedy when choosing the subtrees. For example, suppose we want to compute Smax(i). Consider some child j:

$$Smax(i) = val(i) + \sum max(0, Smax(j)) \qquad [\ j \text{ is a child of } i\ ]$$

Here, val(i) is defined as 1 if i is black and -1 otherwise.

Using this recurrence, we can compute Smax(i) for all i with a single pass through the tree in O(N) time!

The Implementation:

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

#define N 111111

vector<int> adj[N];
int col[N];
int parent[N];
int mx[N];
int mn[N];
vector<int> res;
int ans, ansi;

void compute(int i, int p) {
    parent[i] = p;
    mx[i] = +col[i];
    mn[i] = -col[i];
    for (int j : adj[i]) {
        if (j == p) continue;
        compute(j, i);
        mx[i] += max(0, mx[j]);
        mn[i] += max(0, mn[j]);
    }
    int curr = max(mx[i], mn[i]);
    if (ans < curr) {
        ans = curr;
        ansi = i;
    }
}

void get(int i, bool ismx) {
    res.push_back(i);
    for (int j : adj[i]) {
        if (j == parent[i]) continue;
        if ((ismx ? mx : mn)[j] > 0) get(j, ismx);
    }
}
```

```
int main() {
    int n;
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &col[i]);
        if (!col[i]) col[i] = -1;
        parent[i] = -1;
    }
    for (int i = 1; i < n; i++) {
        int a, b;
        scanf("%d%d", &a, &b);
        a--, b--;
        adj[a].push_back(b);
        adj[b].push_back(a);
    }

    compute(0, -1);
    get(ansi, ans == mx[ansi]);

    printf("%d\n%d\n", ans, int(res.size()));
    for (int i = 0; i < res.size(); i++) {
        printf("%d%c", res[i] + 1, " \n"[i == res.size() - 1]);
    }
}
```

References

- Problem
  - https://www.hackerrank.com/contests/university-codesprint-3/challenges/black-white-tree
- Trees Explanation
  - https://en.wikipedia.org/wiki/Tree_(graph_theory)