Worksheet - 6

Student Name: ANIKET KUMAR **UID:**20BCS5306
Branch: BE-CSE **Semester:** 5th
Subject Name: Competitive Coding **Section/Group:** 20BCS_WM-703/B

Task-1: Tree-huffman-decoding

https://www.hackerrank.com/challenges/tree-huffman-decoding/problem?isFullScreen=true

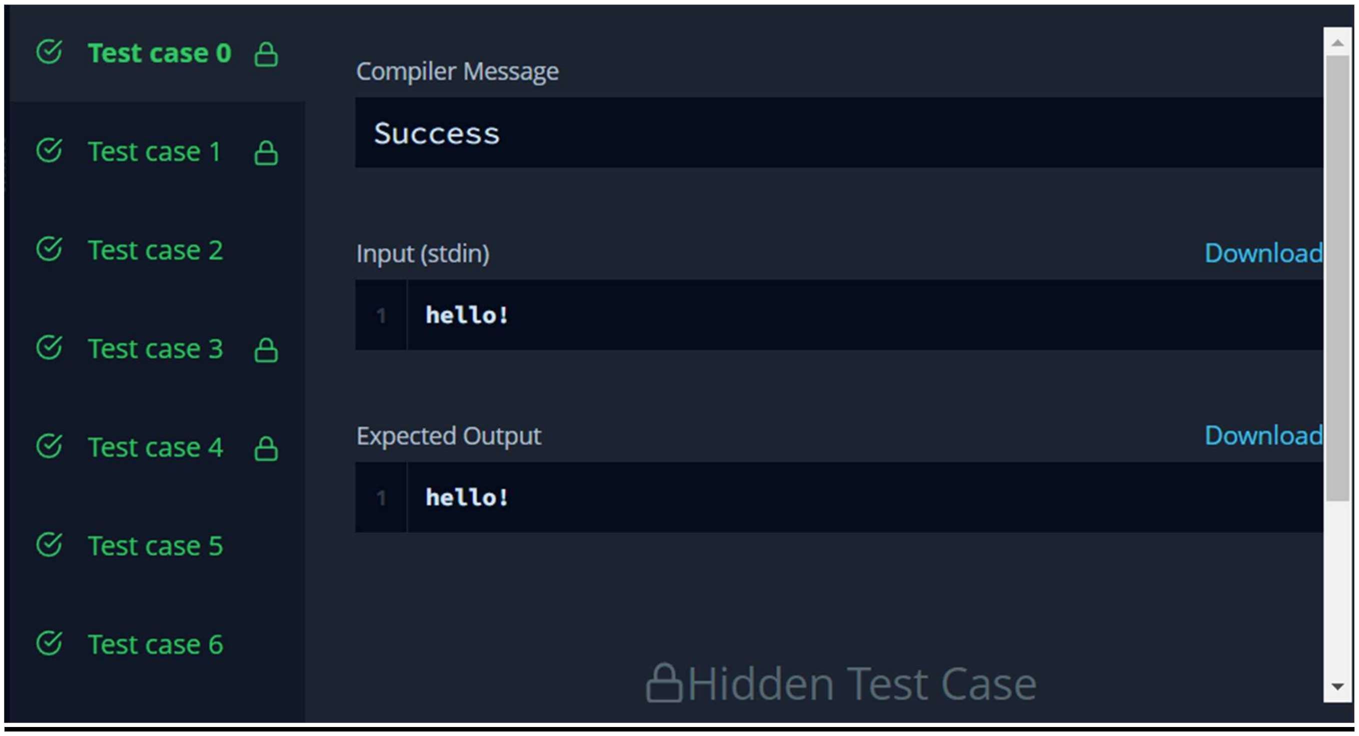
Code:

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

typedef struct node {
   int freq;
   char data;
   node * left;
   node * right;
} node;

struct deref:public binary_function<node*, node*, bool> {
   bool operator()(const node * a, const node * b)const {
      return a->freq > b->freq;
   }
};

typedef priority_queue<node *, vector<node*>, deref> spq;

node * huffman_hidden(string s) {

   spq pq;
   vector<int>count(256,0);
```

```
    for(int i = 0; i < s.length(); i++ ) {
        count[s[i]]++;
    }

    for(int i=0; i < 256; i++) {

        node * n_node = new node;
        n_node->left = NULL;
        n_node->right = NULL;
        n_node->data = (char)i;
        n_node->freq = count[i];

        if( count[i] != 0 )
            pq.push(n_node);

    }

    while( pq.size() != 1 ) {

        node * left = pq.top();
        pq.pop();
        node * right = pq.top();
        pq.pop();
        node * comb = new node;
        comb->freq = left->freq + right->freq;
        comb->data = '\0';
        comb->left = left;
        comb->right = right;
        pq.push(comb);

    }

    return pq.top();

}

void print_codes_hidden(node * root, string code, map<char, string>&mp) {

    if(root == NULL)
```

```
        return;

    if(root->data != '\0') {
        mp[root->data] = code;
    }

    print_codes_hidden(root->left, code+'0', mp);
    print_codes_hidden(root->right, code+'1', mp);

}

/*
The structure of the node is

typedef struct node {

    int freq;
    char data;
    node * left;
    node * right;

} node;

*/


void decode_huff(node * root,string s)
{
    string ans = "";
    node* n = root;
    for(auto itr = s.begin(); itr != s.end();itr++){
        node* next;
        if(*itr == '0'){
            next = n -> left;
        }
        else{
            next = n -> right;
        }
        if(next -> data == '\0'){
            n = next;
        }
        else{
            ans += next -> data;
            n = root;
        }
    }
    cout << ans << endl;
}

int main() {

    string s;
    std::cin >> s;

    node * tree = huffman_hidden(s);
    string code = "";
    map<char, string>mp;

    print_codes_hidden(tree, code, mp);

    string coded;

    for( int i = 0; i < s.length(); i++ ) {
        coded += mp[s[i]];
    }

    decode_huff(tree,coded);

    return 0;
}
```

Hacker Rank Test Case / Output:



Compiler Message

```
Success
```

Input (stdin)

```
hello!
```

Expected Output

```
hello!
```

Task-2: Balanced-forest problem

https://www.hackerrank.com/challenges/balanced-forest/problem?isFullScreen=true

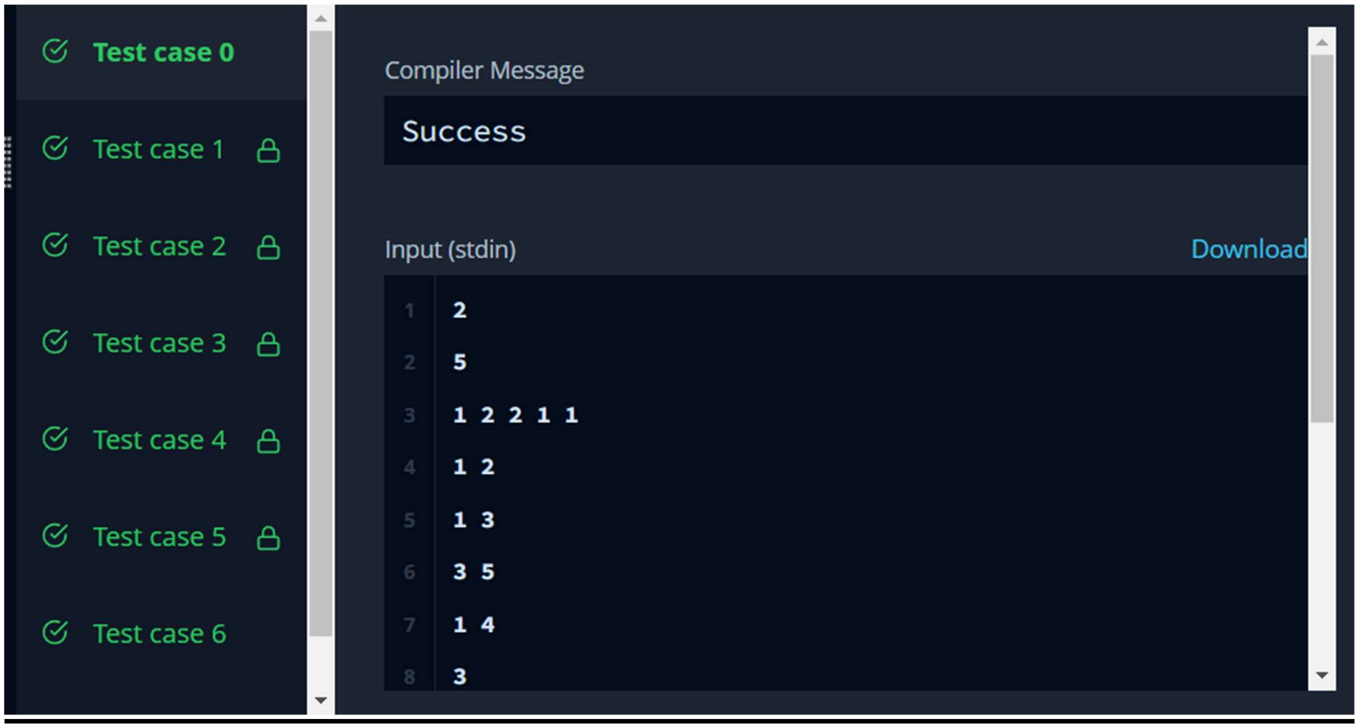
Code:

```
#include <iostream>
#include <cstdio>
#include <vector>
#include <algorithm>
#include <string>
#include <set>
#include <map>
#include <queue>
#include <stack>
#include <deque>
#include <cassert>
#include <stdlib.h>

using namespace std;


typedef long long ll;

const ll INF = (ll) 1e18;
const int N = (int) 5e4 + 10;

vector<int> g[N];
ll c[N];
ll f[N];
ll res = INF;
ll tot = 0;
bool was[N];

void upd(ll a, ll b, ll c) {
   if (a == b && c <= a)
```

```
        res = min(res, a - c);
    if (a == c && b <= a)
        res = min(res, a - b);
    if (b == c && a <= b)
        res = min(res, b - a);
}

set<ll>* unite(set<ll>* a, set<ll>* b) {
    if (a->size() > b->size())
        swap(a, b);
    for (ll x : *a) {
        if (b->count(tot - 2 * x))
            upd(tot - 2 * x, x, x);
        if (b->count(x))
            upd(x, x, tot - 2 * x);
        if ((tot - x) % 2 == 0 && b->count((tot - x) / 2))
            upd((tot - x) / 2, x, (tot - x) / 2);
    }
    for (ll x : *a) {
        b->insert(x);
    }
    delete a;
    return b;
}

set<ll>* dfs(int v) {
    was[v] = true;
    f[v] = c[v];
    set<ll>* sv = new set<ll>();
    for (int to : g[v])
        if (!was[to]) {
            set<ll>* sto = dfs(to);
            f[v] += f[to];
            sv = unite(sv, sto);
        }
    if (f[v] % 2 == 0 && sv->count(f[v] / 2))
        upd(f[v] / 2, f[v] / 2, tot - f[v]);
    if (sv->count(tot - f[v]))
        upd(tot - f[v], 2 * f[v] - tot, tot - f[v]);
    if (sv->count(2 * f[v] - tot))
        upd(2 * f[v] - tot, tot - f[v], tot - f[v]);
    sv->insert(f[v]);
    return sv;
}

void solve() {
    int n;
    cin >> n;
    for (int i = 0; i < N; i++) {
        was[i] = false;
        g[i].clear();
        c[i] = 0;
    }
    tot = 0;
    res = INF;
    for (int i = 0; i < n; i++) {
        cin >> c[i];
        tot += c[i];
    }
    for (int i = 0; i < n - 1; i++) {
        int x, y;
        cin >> x >> y;
        --x;
        --y;
        g[x].push_back(y);
        g[y].push_back(x);
    }
    set<ll>* s = dfs(0);
    //for (int i = 0; i < n; i++)
    //    cerr << f[i] << " ";
    //cerr << endl;
    delete s;
    if (res == INF)
        res = -1;
    cout << res << endl;
    // cerr << "----------" << endl;
}
```

```
int main() {
    ios_base::sync_with_stdio(0);
    int p;
    cin >> p;
    while (p--) {
        solve();
    }
    return 0;
}
```

Hacker Rank Test Case / Output:



Compiler Message

```
Success
```

Input (stdin)

```
2
5
1 2 2 1 1
1 2
1 3
3 5
1 4
3
```