**Worksheet-2.2**

**Student Name:-** Pushpraj Roy **UID:-** 20BCS9866

**Branch:-** BE- CSE **Section/Group:-** WM\_617 “A”

**Subjetct Code:-** 20CSP-314 **Semester:-** 5th

**Subject Name:-** Competitive Coding Lab

**Problem 1:- Tree-huffman-decoding**

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

**Code:-**

/\*

The structure of the node is

typedef struct node

{

int freq;

char data;

node \* left;

node \* right;

}node;

\*/

void decode\_huff\_helper(node\* root, string s, int& index){

if (root->left == NULL && root->right == NULL){

cout<<root->data;

return;

}

if (s[index] == '0'){

index++;

decode\_huff\_helper(root->left, s, index);

}

else {

index++;

decode\_huff\_helper(root->right, s, index);

}

}

void decode\_huff(node \* root,string s)

{

int index = 0;

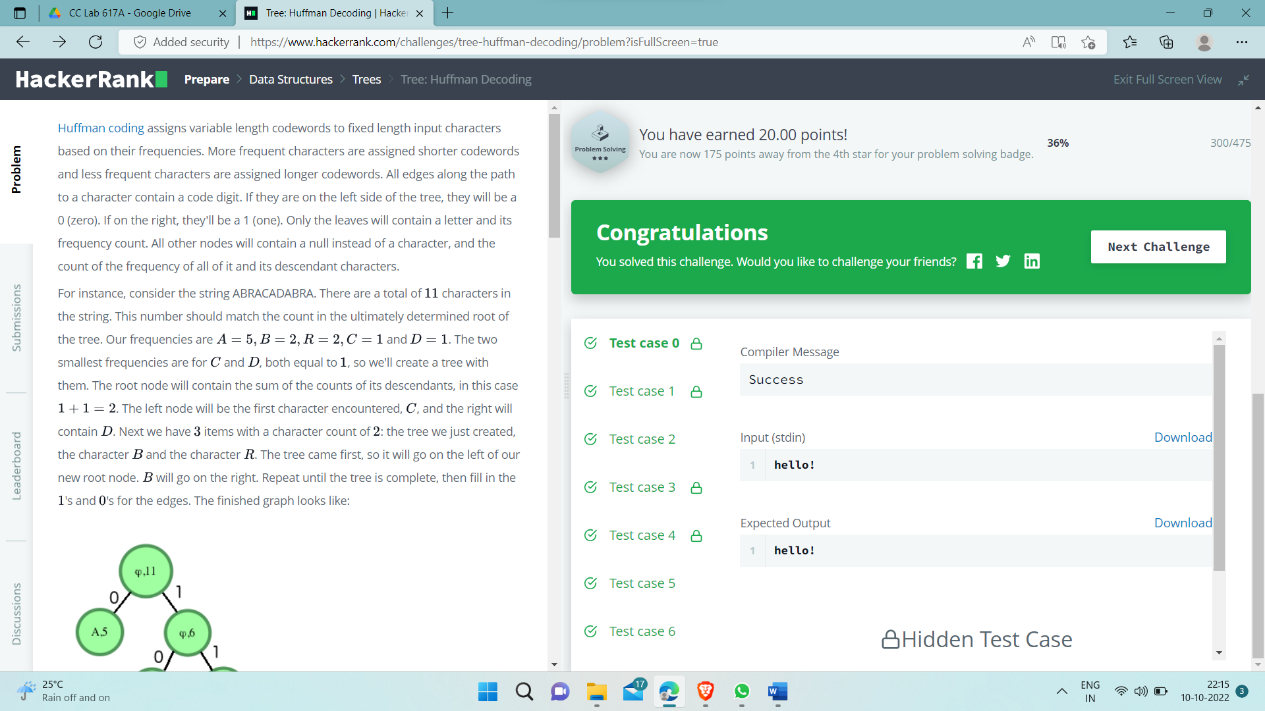
while(index < s.size()){

decode\_huff\_helper(root, s, index);

}

}

**Output:-**



**Problem 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;

}

**Output:**

