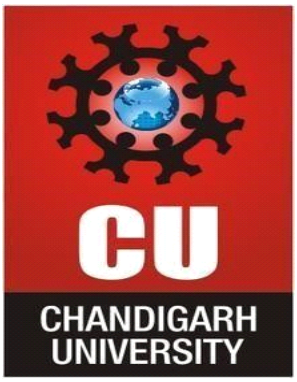
CHANDIGARH UNIVERSITY

UNIVERSITY INSTITUTE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



Submitted By: Rajiv Paul	**Submitted To:** Urvashi Malhotra
Subject Name	Competitive Coding-I
Subject Code	20CSP-314
Branch	BE-CSE
Semester	5th

LAB INDEX

Sr. No	Program	Date	Evaluatio				Sign
			L W (12	V V (8)	F W (10	Tot al (30	
1.							
2.							
3.							
4.							
5.							
6.	To solve the following hacker rank problems based on Trees.	20/10/22					
7.							
8.							
9.							
10.							

EXPERIMENT – 2.2

Student Name:Rajiv Paul **UID:20BCS1812**
Branch: CSE **Section/Group: 20BCS_WM_702(A)**
Semester: 5th **Date of Performance:20/10/2022**
Subject Name: Competitive Coding **Subject Code: 20CSP-314**

AIM OF THE EXPERIMENT:

To solve the following hacker rank problems based on Trees.

Problem 1: https://www.hackerrank.com/challenges/tree-huffman-decoding/problem?isFullScreen=true

1. PROGRAM CODE:

```
import java.util.*;


abstract class Node implements Comparable<Node> {
   public  int frequency; // the frequency of this tree
   public  char data;
   public  Node left, right;
   public Node(int freq) {
     frequency = freq;
   }

   // compares on the frequency
   public int compareTo(Node tree) {
      return frequency - tree.frequency;
   }
```

```
}

class HuffmanLeaf extends Node {


    public HuffmanLeaf(int freq, char val) {
        super(freq);
        data = val;
    }
}

class HuffmanNode extends Node {

    public HuffmanNode(Node l, Node r) {
        super(l.frequency + r.frequency);
        left = l;
        right = r;
    }

}

class Decoding {

  void decode(String S, Node root)
{
    StringBuilder sb = new StringBuilder();
    Node c = root;
    for (int i = 0; i < S.length(); i++) {
```

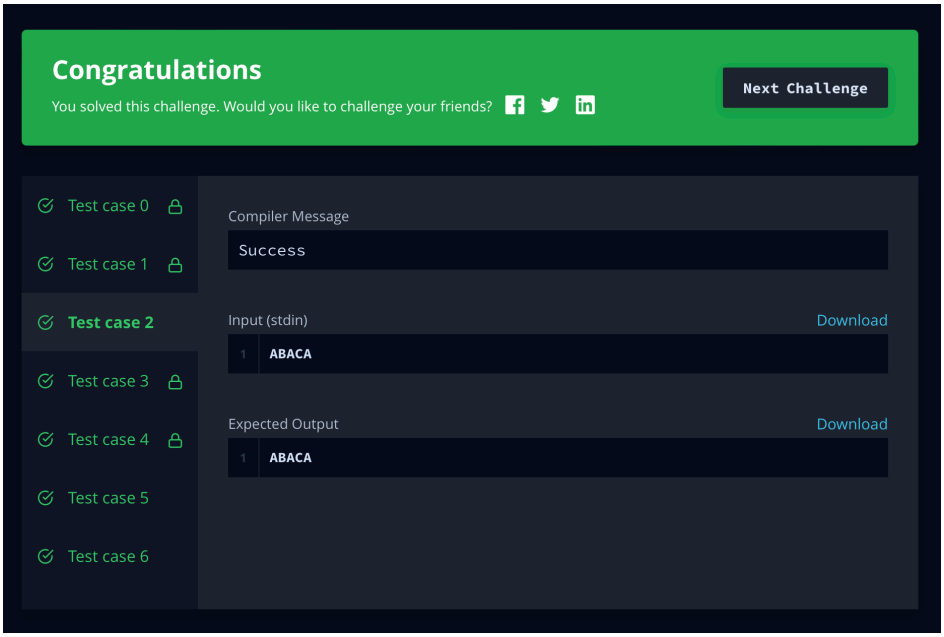
```
            c = S.charAt(i) == '1' ? c.right : c.left;
            if (c.left == null && c.right == null) {
                sb.append(c.data);
                c = root;
            }
        }
        System.out.print(sb);
    }

    }

    public class Solution {

        // input is an array of frequencies, indexed by character code
        public static Node buildTree(int[] charFreqs) {

            PriorityQueue<Node> trees = new PriorityQueue<Node>();
            // initially, we have a forest of leaves
            // one for each non-empty character
            for (int i = 0; i < charFreqs.length; i++)
                if (charFreqs[i] > 0)
                    trees.offer(new HuffmanLeaf(charFreqs[i], (char)i));

            assert trees.size() > 0;

            // loop until there is only one tree left
            while (trees.size() > 1) {
                // two trees with least frequency
```

```
            Node a = trees.poll();
            Node b = trees.poll();

            // put into new node and re-insert into queue
            trees.offer(new HuffmanNode(a, b));
        }

        return trees.poll();
    }

    public static Map<Character,String> mapA=new
HashMap<Character ,String>();

    public static void printCodes(Node tree, StringBuffer prefix) {

        assert tree != null;

        if (tree instanceof HuffmanLeaf) {
            HuffmanLeaf leaf = (HuffmanLeaf)tree;

            // print out character, frequency, and code for this leaf (which is just the
prefix)
            //System.out.println(leaf.data + "\t" + leaf.frequency + "\t" + prefix);
            mapA.put(leaf.data,prefix.toString());

        } else if (tree instanceof HuffmanNode) {
            HuffmanNode node = (HuffmanNode)tree;
```

```
        // traverse left
        prefix.append('0');
        printCodes(node.left, prefix);
        prefix.deleteCharAt(prefix.length()-1);

        // traverse right
        prefix.append('1');
        printCodes(node.right, prefix);
        prefix.deleteCharAt(prefix.length()-1);
    }
}

public static void main(String[] args) {
    Scanner input = new Scanner(System.in);

    String test= input.next();

    // we will assume that all our characters will have
    // code less than 256, for simplicity
    int[] charFreqs = new int[256];

    // read each character and record the frequencies
    for (char c : test.toCharArray())
        charFreqs[c]++;

    // build tree
    Node tree = buildTree(charFreqs);
```

```
        // print out results
        printCodes(tree, new StringBuffer());
        StringBuffer s = new StringBuffer();

        for(int i = 0; i < test.length(); i++) {
            char c = test.charAt(i);
            s.append(mapA.get(c));
        }

        //System.out.println(s);
        Decoding d = new Decoding();
        d.decode(s.toString(), tree);

    }
}
```

2. OUTPUT:



Problem 2: https://www.hackerrank.com/challenges/balanced-forest/problem?isFullScreen=true

1. **PROGRAM CODE:**

```
import java.io.File;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.HashSet;
import java.util.List;
import java.util.Map;
import java.util.Scanner;
import java.util.Set;

public class Solution {
private static Scanner scn;
private static int n;
private static long ret;
private static int[] c, p;
private static long[] s;
private static List<Integer>[] adj;
private static void visit(int k, int i) {
s[i] = c[i];
for(int j : adj[i]) {
   if(j == k) {
      continue;
```

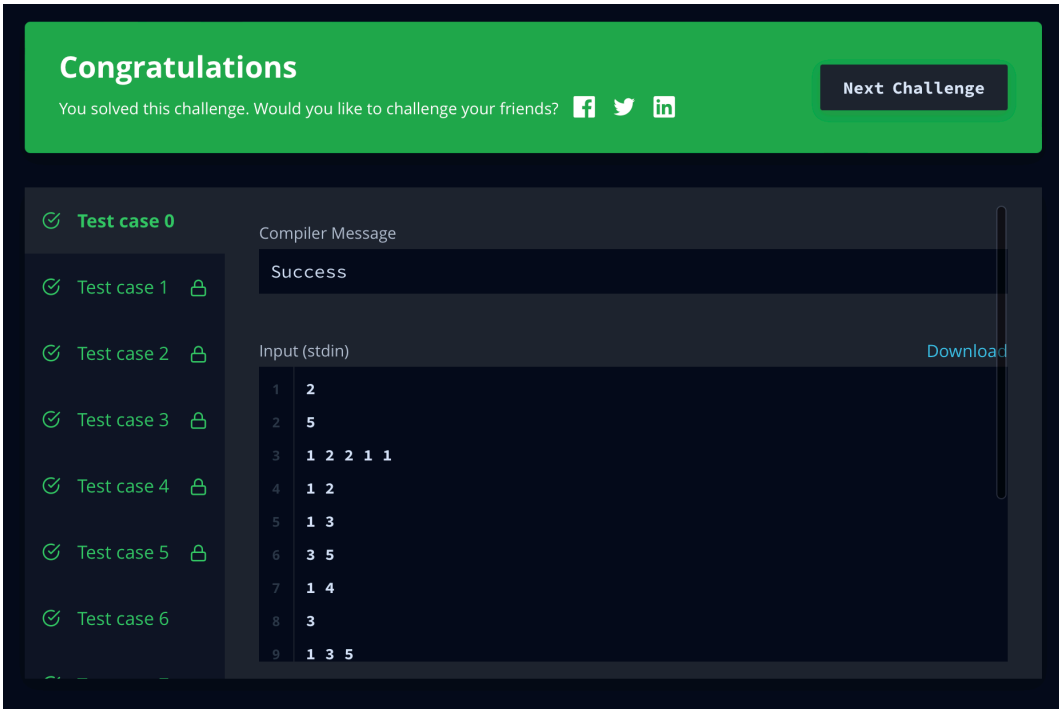
```
        }
p[j] = i;
visit(i,j);
s[i] += s[j];
}
}
private static void check(long x, long y, long z) {
long[] t = new long[] {x, y, z};
for (int i = 0; i< 3; i++) {
    for (int j = i + 1; j < 3; j++) {
if (t[i] != t[j]) {
    continue;
}
long h = -t[i] + -t[j] + t[0] + t[1] + t[2];
if (h <= t[i]) {
if (ret < 0) {
ret = t[i] - h; }
else {
ret = Math.min(ret, t[i] - h);
}
}
}
}
}

private static void solve() {
ret = -1; n =scn.nextInt();
c = new
```

```
int[n]; s = new long[n];
adj = new List[n];
p =new int[n];
Arrays.fill(p,-1);
for (int i = 0; i < n;++i) {
    c[i] =scn.nextInt();
adj[i] = new ArrayList<Integer>();
}
for (int i = 0; i < n - 1; i++) {
int x = scn.nextInt();
int y= scn.nextInt();
x--;
y--;

adj[x].add(y);
adj[y].add(x);
}
visit(-1, 0);
Map<Long, Set<Integer>> sSet = new HashMap<Long, Set<Integer>>();
for (int i = 0; i < n; ++i) {
    if (sSet.containsKey(s[i])) {
        if (s[i]* 3 >= s[0]) {
            long h = s[i] * 3 - s[0];
if (ret < 0) {
ret = h; }
else {
ret = Math.min(ret, h);
}
```

```
}
}
Set<Integer> si = sSet.get(s[i]);
if (si == null) {
si = new HashSet<Integer>();
}
si.add(i);
sSet.put(s[i], si);
}
for (int i = 0; i < n; ++i) {

   if (s[i] * 3 < s[0] || s[i] * 2 > s[0]) {
continue;
}
long t = s[0] - s[i] * 2;
Set<Integer> si = sSet.get(t);
if (si == null) {
   continue;
}
for (int j : si) {
int k = j;
boolean ok =true;
while (k >= 0) {
if (k == i) {
   ok =false;
   break;
}
k = p[k];
```

```
}
if (ok) {
    long h = s[i] *3 - s[0];
if (ret < 0) ret= h;
else ret = Math.min(ret, h);
}
}
}
for (int i= 0; i < n; ++i) {
int j = i;
while (j >= 0) {
j = p[j];
if (j>= 0) {
check(s[i], s[j] - s[i], s[0] - s[j]);
}
}
}
System.out.println(ret);
}
public static void main(String[] args) {
scn = new Scanner(System.in); int
nTest = scn.nextInt();
for (int i = 0; i < nTest; ++i) {
solve();
}
}
}
```

2. OUTPUT:


Congratulations
You solved this challenge. Would you like to challenge your friends?
Next Challenge
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6
Compiler Message
Success
Input (stdin)
Download
2
5
1 2 2 1 1
1 2
1 3
3 5
1 4
3
1 3 5