Green University of Bangladesh
Department of Computer Science and Engineering(CSE)

Faculty of Sciences and Engineering

Semester: (Fall, Year:2021), B.Sc. in CSE (Day)

Lab Project Report

Course Title: Algorithms Lab

Course Code: CSE 206 Sections: DC

LAB PROJECT NAME: CUT THE TREE

Student Details

Name		ID
1.	JIHAD	202002082

Submission Date : 03/01/2022

Course Teacher's Name : Zeseya Sharmin

[For Teachers use only: Don't Write Anything inside this box]

Lab Report Status

Marks: …… **Signature:.....**

Comments:..... **Date:.....**

TITLE OF THE PROJECT

CUT THE TREE

In this problem (Cut the Tree) we have given a tree and we need to determine which edge to cut so that the resulting tress have a minimal difference between them and then return that difference. remember the difference between their sums is equal to the difference between two trees.

OBJECTIVES/AIM

To find the minimum achievable absolute difference of tree sums.
The first line contains an integer n, the number of vertices in the tree.
The second line contains n space-separated integers, where each integer u denotes the node [u] data value data [u], .Each of the n-1 subsequent lines contains two space-separated integers u and v that describe edge u - v in tree t .

PROCEDURE / ANALYSIS / DESIGN

There is an undirected tree where each vertex is numbered from 1 to ,n and each contains a data value. The sum of a tree is the sum of all its nodes' data values. If an edge is cut, two smaller trees are formed. The difference between two trees is the absolute value of the difference in their sums. Given a tree, determine which edge to cut so that the resulting trees have a minimal difference between them, then return that difference.

Sample Input

```
STDIN              Function
-----              --------
6                  data[] size n = 6
100 200 100 500 100 600    data = [100, 200, 100, 500, 100, 600]
1 2                edges = [[1, 2], [2, 3], [2, 5], [4, 5], [5, 6]]
2 3
2 5
4 5
5 6
```

Sample Output

400

ALGORITHM

- ✓ Make an adjacency list out of the edges. This is effectively the tree.
- ✓ Recursively find the sum of each subtree, and record that in an array.
- ✓ Scan the array for the element with a sum as close as possible to half the total for the tree.
- ✓ Return the difference times 2.

FUNCTION DESCRIPTION : **Cut The Tree** has the following parameter(s):

- *int data[n]:* an array of integers that represent node values
- *int edges[n-1][2]:* an 2 dimensional array of integer pairs where each pair represents nodes connected by the edge.

IMPLEMENTATION

```
import java.io.*;
import java.math.*;
import java.security.*;
import java.text.*;
import java.util.*;
import java.util.concurrent.*;
import java.util.function.*;
import java.util.regex.*;
import java.util.stream.*;
import static java.util.stream.Collectors.joining;
import static java.util.stream.Collectors.toList;

public class Solution {
   static ArrayList<Integer>[ ] graph;
   static int[ ] w;
   static int[ ] total;

public static void main(String[ ] args) {
      Scanner sc = new Scanner(System.in);
      int n = sc.nextInt();
      w = new int[n+1];
      graph = new ArrayList[n+1];
      total = new int[n+1];
      for(int i=1; i<=n; ++i) {
         w[i] = sc.nextInt();
         graph[i] = new ArrayList<Integer>();
      }
      for(int i=1; i<n; ++i) {
         int x = sc.nextInt();
         int y = sc.nextInt();
         graph[x].add(y);
         graph[y].add(x);
      }

      int totalweight = genWeights(1, 0);
      int minDiff = Integer.MAX_VALUE;
```

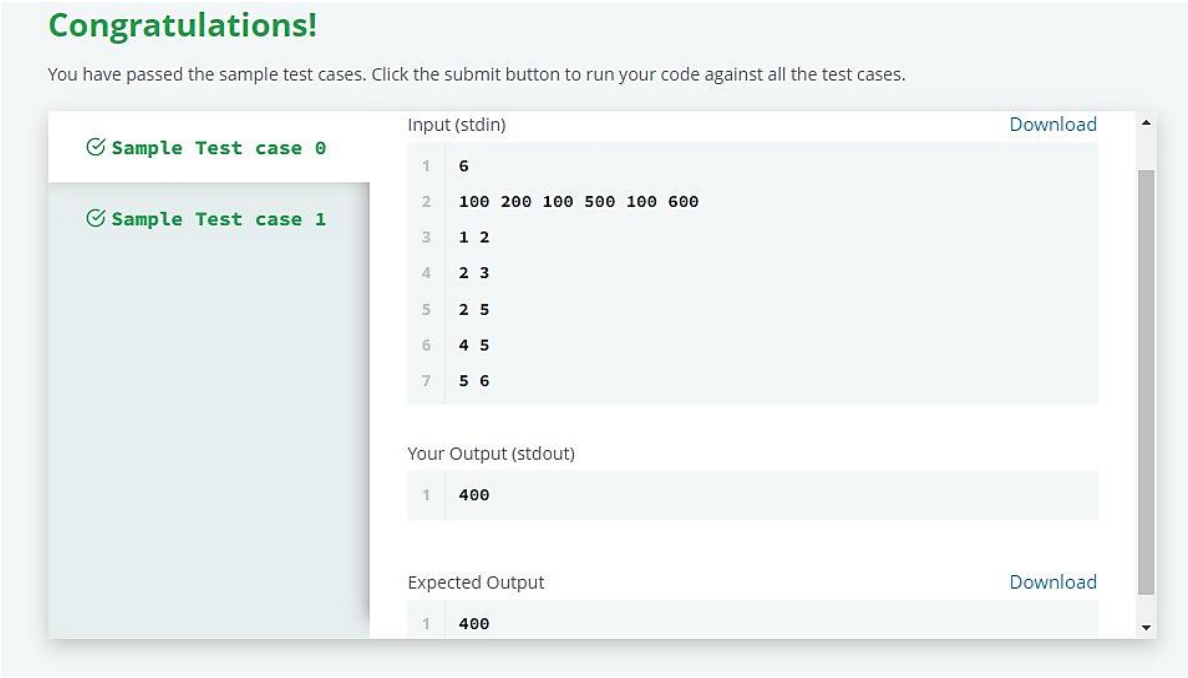
```
        for(int i=2; i<=n; ++i)
            if(minDiff > Math.abs(totalweight - (total[i]<<1)))
                minDiff = Math.abs(totalweight - (total[i]<<1));

        System.out.println(minDiff);
    }

    public static int genWeights(int node, int parent) {
        ArrayList<Integer> adj = graph[node];
        Iterator<Integer> it = adj.iterator();
        int totalw = 0;
        while(it.hasNext()) {
            int child = it.next();
            if(child != parent)
                totalw += genWeights(child, node);
        }
        totalw += w[node];
        total[node] = totalw;
        return totalw;
    }
}
```

TEST RESULT / OUTPUT

We can see that my program has run correctly. And displayed the output.

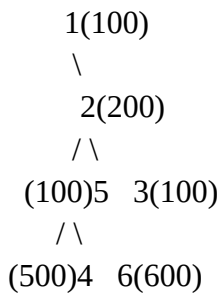


ANALYSIS AND DISCUSSION

This problem is a simple application of Recursion and Depth First Traversal of a tree. You need to solve this problem using post order traversal technique, considering the tree as a rooted tree. (Hint: During dfs, each edge will be traversed only once, thats when you find T1-T2 when this edge is cut).

SUMMARY:

we can represent tree as



Cutting the edge at 1 2 would result in Tree_diff = 1500-100 = 1400
Cutting the edge at 2 3 would result in Tree_diff = 1500-100 = 1400
Cutting the edge at 2 5 would result in Tree_diff = 1200-400 = 800
Cutting the edge at 4 5 would result in Tree_diff = 1100-500 = 600
Cutting the edge at 5 6 would result in Tree_diff = 1000-600 = 400

The minimum difference is 400.

REFERENCE:

https://www.hackerrank.com/challenges/cut-the-tree/problem?isFullScreen=true