```java
/*TOURISTS Uses BFS to find the layers of the graph multiple times on nodes according to
 * the rule and sums them up recursively to find the total sum of lengths of paths for
 * the tourists --  sumPath(int i);
 * */

import java.util.*;

class tourists_BFS{
    static Node[] G;
    public static int N;
    static int pathCost;
    static int[] layer ;
    static boolean[] visited;

    static class Node {
        List<Edge> adj;
        int n;
        public Node(int N){
            adj = new ArrayList<Edge>();
            n=N;
        }
    }
    static class Edge{
        int to, weight;
        public Edge(int t, int w){
            to=t;
            weight = w;
        }
    }
    public static void makeGraph(int n){
    G = new Node[n];
        for(int i =0; i<n; i++){
            G[i]=new Node(i);
        }
        layer = new int[G.length];
        visited = new boolean[G.length];
    }
    public static void addEdge(int u,int v, int w){
        G[u].adj.add(new Edge(v,w));
        G[v].adj.add(new Edge(u,w));
    }
    public static int charN(char c){
        return c;
    }
    public static void bfs(int s){
        Arrays.fill(visited, false);
        Arrays.fill(layer, -1);
        layer[s] = 0;
        Queue<Integer> q = new LinkedList<Integer>();
        q.add(s);
        int cur = -1;
        while (!q.isEmpty()){
            cur = q.peek();
            visited[cur] = true;
            for (Edge e : G[cur].adj){
                if(!(visited[e.to])){
                    q.add(G[e.to].n);
                    visited[e.to] = true;
                    layer[e.to] = layer[cur] + 1;
                }
            }
            q.poll();
        }
    }
    public static void sumPath(int k){
        for(int i = k; i<N/2+1; i++){
            bfs(i-1);
            for(int j =2; i*j<N+1; j++){
                pathCost+=layer[i*j-1]+1;
            }
        }
    }

    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        N = scan.nextInt();
        makeGraph(N);
        int u = -1;
        int v = -1;
        for(int k =0; k<N-1; k++){

                u = scan.nextInt();
                v = scan.nextInt();

                addEdge(u-1,v-1,1);
        }
        sumPath(1);
        System.out.println(pathCost);
        scan.close();
    }
}

```