Problem

Submissions

Discussion Coming Soon

Havva loves trees and she has lots of trees in the inzva's garden. However, one of her trees, Hibiscus, is very special to her. One can compare Hibiscus to the 'tree' concept in computer science: Each branch of Havva's tree represents an edge (link). Likewise, each frond in Havva's tree can be considered as a node (vertex). More formally, Hibiscus is a tree that has N nodes and $N-1$ edges. Also, since it is a tree, a path exists between any two nodes. One day after breakfast Havva realized that there are lizards on the leaf nodes of the Hibiscus. She is very scared of lizards and wants you to help!

Initially, you are on the root. You can visit any adjacent node in 1 second, and you can carry at most one lizard at a time. After picking up a lizard, you have to go back to the root and leave it back to the ground from the root of the Hibiscus. What is the minimum number of seconds you need to clean up Hibiscus from lizards?

Note: Nodes of Hibiscus are numerated from 1 to N and root node is the node numbered 1.

Note 2: In a **tree** data structure, the node which does not have a child is called a **leaf node**.

Input Format

The first line contains an integer, N.

Each of the following $N-1$ lines contains two integers, u_i and v_i, representing an edge between u_i and v_i.

Output Format

Print an integer, the minimum number of seconds you need to clean up trees.

Constraints

- $1 \leq N \leq 10^5$
- $1 \leq u_i \; v_i \leq N$

Sample Input 1

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

Sample Output 1

```
6
```

Explanation 1

$1 \rightarrow 2$
$2 \rightarrow 3$
$3 \rightarrow 4$ (pick up lizard from the node 4)
$4 \rightarrow 3$
$3 \rightarrow 2$
$2 \rightarrow 1$

Sample Input 2

Sample Output 2

```
12
```

```
6
1 2
2 3
2 4
1 5
5 6
```

C++ (GCC 9.2.0) Bright **Memory Limit (kB) :** 256000 **Time Limit (s) :** 1

```cpp
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;
#define MAX 100005
vector<int> tree[MAX];
vector <bool> visited(MAX, false);
int sum = 0;

void dfs(int node, int depth){
    visited[node] = true;
    bool isLeaf = true;
    for(auto child: tree[node]){
        if(!visited[child]){
            dfs(child, depth + 1);
            isLeaf = false;
        }
    }
    if(isLeaf == true)
        sum += depth;
}

int main() {
    int N, u, v;
    cin >> N;
    for(int i = 0; i < N - 1; i++){
        cin >> u >> v;
        tree[u].push_back(v);
        tree[v].push_back(u);
    }
```

Upload File Test against custom test case Run Code Submit

Sample Test Case 0

Sample Test Case 1

Accepted

Input(stdin)

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

Output(stdin)

```
6
```

Expected Output

```
6
```