D. Zero Tree

A *tree* is a graph with n vertices and exactly $n - 1$ edges; this graph should meet the following condition: there exists exactly one shortest (by number of edges) path between any pair of its vertices.

A *subtree* of a tree T is a tree with both vertices and edges as subsets of vertices and edges of T.

You're given a tree with n vertices. Consider its vertices numbered with integers from 1 to n. Additionally an integer is written on every vertex of this tree. Initially the integer written on the i-th vertex is equal to v_i. In one move you can apply the following operation:

1. Select the subtree of the given tree that includes the vertex with number 1.
2. Increase (or decrease) by one all the integers which are written on the vertices of that subtree.

Calculate the minimum number of moves that is required to make all the integers written on the vertices of the given tree equal to zero.

Input

The first line of the input contains n ($1 \leq n \leq 10^5$). Each of the next $n - 1$ lines contains two integers a_i and b_i ($1 \leq a_i, b_i \leq n$; $a_i \neq b_i$) indicating there's an edge between vertices a_i and b_i. It's guaranteed that the input graph is a tree.

The last line of the input contains a list of n space-separated integers $v_1, v_2, ..., v_n$ ($|v_i| \leq 10^9$).

Output

Print the minimum number of operations needed to solve the task.

Please, do not write the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use the `cin`, `cout` streams or the `%I64d` specifier.

Examples

input
3 1 2 1 3 1 -1 1

output
3