Jump Graph

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

Given a permutation of numbers from 1 to n, we can create a directed graph with n vertices based on it. In such a graph, one can jump from any element of the permutation to any other, as long as the elements skipped along the way (excluding the starting one) are smaller than the target. Such a graph is called the *jump graph* of the permutation.

Formally, from vertex i there's an edge to vertex j ($i \neq j$), if p_j is larger than every p_k for k between i and j (not counting positions i or j). In other words, for $i \neq j$ the edge $i \to j$ exists, if $p_j > p_k$ for every k such that $\min(i, j) < k < \max(i, j)$.

Your task is, given a permutation, for each vertex in its jump graph, determine the sum of distances to all other vertices. The distance from one vertex to another is the number of edges on the shortest path from the first vertex to the second. It can be proven that the jump graph is always strongly connected, so there exists a path between any two vertices.

Input

The first line of the standard input contains a single integer n ($2 \leq n \leq 300\,000$), indicating the length of the permutation.

The second line contains a sequence $p_1, p_2, \ldots, p_{n-1}, p_n$ ($1 \leq p_i \leq n$; $p_i \neq p_j$ for $i \neq j$) which is a permutation of integers from 1 to n.

Output

The only line of the output should contain n integers: the i-th of them should indicate the sum of distances from the i-th vertex of the jump graph to all other vertices.

Example

standard input	standard output
6 1 6 3 2 5 4	11 7 7 7 6 8

Note

The adjacency matrix of the jump graph for the sample permutation is given below. If there's an edge from the i-th vertex to the j-th, then in the i-th row and j-th column there's a value of 1, otherwise, there's a value of 0.

	1	2	3	4	5	6
1	**0**	**1**	**2**	**3**	**2**	**3**
2	**1**	**0**	**1**	**2**	**1**	**2**
3	**2**	**1**	**0**	**1**	**1**	**2**
4	**2**	**1**	**1**	**0**	**1**	**2**
5	**2**	**1**	**1**	**1**	**0**	**1**
6	**2**	**1**	**2**	**2**	**1**	**0**