Tree Dash

You have come across a *rooted tree*. This tree consists of N vertices, numbered from 1 to N. One of these vertices, labelled R, serves as the *root*.

Each non-root vertex i has a parent vertex P_i, where $P_i \neq i$. If you start at any vertex and repeatedly move to its parent, then its parent's parent, and so on, you will eventually reach the root vertex. The vertices encountered on this path are called the *ancestors* of i.

Vertex i is a *child* of vertex P_i. The children of a vertex, along with their children, and so on, are called the *descendants* of i.

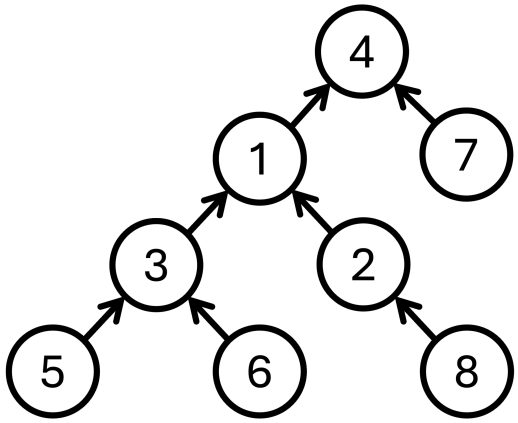

Figure 1: An example tree with $N = 8$. The root is $R = 4$ and each non-root has an arrow to its parent. Vertex 1 has one ancestor (4) and five descendants (2, 3, 5, 6, 8), while vertex 8 has three ancestors (1, 2, 4) and no descendants. The root has no ancestors and $N - 1$ descendants.

If you are at vertex i in this tree, you can travel to any of the following vertices:

- The ancestor of i with the minimum index (if i has at least one ancestor).
- The ancestor of i with the maximum index (if i has at least one ancestor).
- The descendant of i with the minimum index (if i has at least one descendant).
- The descendant of i with the maximum index (if i has at least one descendant).

A vertex u can reach another vertex v if it is possible to travel from u to v, potentially using some intermediate vertices.

Each vertex has a *weight* W_i. Compute the sum of $W_u \times W_v$ across all ordered pairs of vertices (u, v) such that $u \neq v$ and vertex u can reach v.

Subtasks and Constraints

For all subtasks:

- $2 \leq N \leq 500\,000$.
- $1 \leq R \leq N$.
- $1 \leq P_i \leq N$ for all $i \neq R$.
- $1 \leq W_i \leq 1000$ for all i.
- If you start at any vertex and repeatedly move to its parent, then its parent's parent, and so on, you will eventually reach the root vertex.

Additional constraints for each subtask are given below.

Subtask	Points	Additional constraints
1	20	$N \leq 100$ and the tree is a *line* (see $*$ below).
2	15	$N \leq 3000$.
3	25	The tree is a line (see $*$ below).
4	20	The root is 1 and it has only one child, which is N. In other words, $R = 1$, $P_N = 1$, and $P_i \neq 1$ for all $2 \leq i \leq N - 1$.
5	20	No additional constraints.

$*$: a *line* is a tree where every vertex has at most 1 child. Sample input 2 is a line.

Input

- The first line of input contains the two integers N and R.
- The second line contains N integers $P_1, P_2, \ldots, P_N$. Since R does not have a parent, P_R is replaced with a 0.
- The third line contains N integers $W_1, W_2, \ldots, W_N$.

Output

Output a single integer, the sum of $W_u \times W_v$ across all pairs $u \neq v$ where vertex u can reach v.

Note: Your solution may involve integers which are large. Consider using 64-bit integers ('long long' in C++) in your solution.

Sample Input 1

```
8 4
4 1 1 0 3 3 4 2
1 1 1 1 1 1 1 1
```

Sample Output 1

```
30
```

Sample Input 2

```
6 5
6 5 4 2 0 3
1 2 3 4 5 6
```

Sample Output 2

```
228
```

Sample Input 3

```
12 1
0 3 12 3 6 12 8 2 8 12 5 1
70 92 86 72 22 79 20 98 42 56 76 34
```

Sample Output 3

```
228776
```

Explanation

Sample input 1 shown in Figure 1 on the first page:

1. Vertex 1 can reach vertices $2, 4, 8$.
2. Vertex 2 can reach vertices $1, 4, 8$.
3. Vertex 3 can reach vertices $1, 2, 4, 5, 6, 8$.

4. Vertex 4 can reach vertices $1, 2, 8$.
5. Vertex 5 can reach vertices $1, 2, 4, 8$.
6. Vertex 6 can reach vertices $1, 2, 4, 8$.
7. Vertex 7 can reach vertices $1, 2, 4, 8$.
8. Vertex 8 can reach vertices $1, 2, 4$.

Sample input 2 is a line shown in Figure 2:

1. Vertex 1 can reach vertices $2, 5, 6$.
2. Vertex 2 can reach vertices $1, 5, 6$.
3. Vertex 3 can reach vertices $1, 2, 5, 6$.
4. Vertex 4 can reach vertices $1, 2, 5, 6$.
5. Vertex 5 can reach vertices $1, 2, 6$.
6. Vertex 6 can reach vertices $1, 2, 5$.

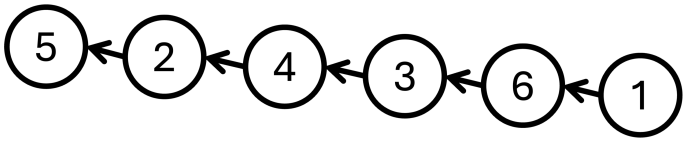

Figure 2: Sample input 2. Each vertex has $W_i = i$.

Sample input 3 is a tree satisfying the constraints of subtask 4, shown in Figure 3:

1. Vertex 1 can reach vertices $2, 7, 9, 11, 12$.
2. Vertex 2 can reach vertices $1, 7, 9, 11, 12$.
3. Vertex 3 can reach vertices $1, 2, 7, 9, 11, 12$.
4. Vertex 4 can reach vertices $1, 2, 7, 9, 11, 12$.
5. Vertex 5 can reach vertices $1, 2, 7, 9, 11, 12$.
6. Vertex 6 can reach vertices $1, 2, 5, 7, 9, 11, 12$.
7. Vertex 7 can reach vertices $1, 2, 9, 11, 12$.
8. Vertex 8 can reach vertices $1, 2, 7, 9, 11, 12$.
9. Vertex 9 can reach vertices $1, 2, 7, 11, 12$.
10. Vertex 10 can reach vertices $1, 2, 7, 9, 11, 12$.
11. Vertex 11 can reach vertices $1, 2, 7, 9, 12$.
12. Vertex 12 can reach vertices $1, 2, 7, 9, 11$.

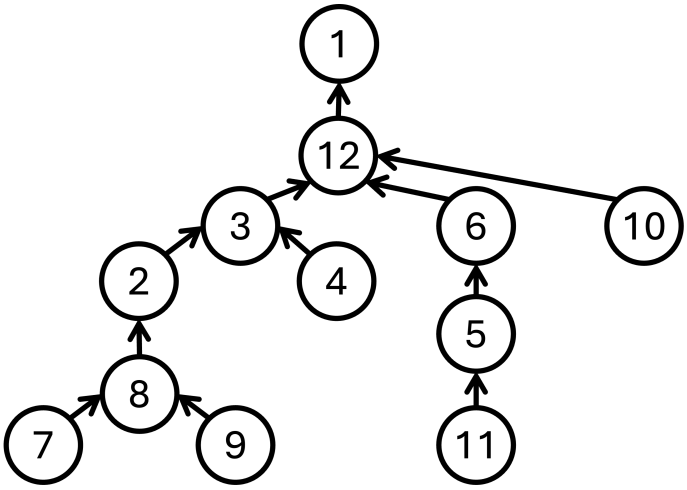

Figure 3: Sample input 3.