You want to spend your next vacation in a foreign country. In the summer you are free for N consecutive days. By a strange coincidence, in the foreign country there are N villages numbered from 0 to N−1. On each day there will be a festival in one of the villages; on day K there will be a festival in village number K.

You plan to begin your vacation on some day A and end it on some day B (0 ≤ A ≤ B < N). During day K you will attend the festival organized in village K, and in the evening (if K < B) you will take a bus to village K+1.

The villages are connected by N−1 bidirectional roads in such a way that there is exactly one path (not necessarily direct) between any two villages. If the path between villages K and K+1 is not direct, the bus will travel through some other village(s). Whenever you travel through a village, you are extremely sad if you did not and will not attend the festival in this village, which would totally ruin your vacations. In other words, if during travel between villages K and K+1 the bus passes through some village J, following inequalities must be satisfied: A ≤ J ≤ B.

You are given a map of the road network in the form of array T of length N. This means that if T[K] = J and K ≠ J, then there is a direct road between villages K and J.

Write a program that calculates how many possible vacations you can consider, i.e., how many pairs of days (A, B) satisfy the constraints that you will visit all the villages you see during your bus trips (visited villages numbers create continuous range of integers).

Write a function:

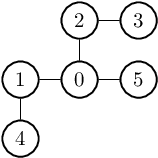
int solution(vector<int> &T);

that, given a non-empty array T consisting of N integers describing a road network between N villages, returns your number of possible vacations.

For example, the following array

T[0] = 2 T[1] = 0 T[2] = 2 T[3] = 2 T[4] = 1 T[5] = 0

encodes a country of N = 6 villages depicted in the figure below:



For this array the function should return 12, since there are twelve possible vacations: (0, 0), (0, 1), (0, 2), (0, 3), (0, 4), (0, 5), (1, 1), (2, 2), (2, 3), (3, 3), (4, 4), (5, 5).

Every pair in which A = B is a correct vacation. Pair (2, 3) is also correct, since, during the bus trip from village 2 to village 3, you do not pass through any other village.

Pair (0, 3) is correct. Paths the bus goes through are following: (0 −> 1), (1 −> 0 −> 2), (2 −> 3). Every bus path covers villages only from the continuous range (0, 3).

Pair (2, 4) is not correct, since, during the bus trip from village 3 to village 4, you pass through villages 0 and 1, but you do not attend festivals in these villages.

Pair (1, 4) is not correct. You would start on day A = 1 in village 1 and finish on day B = 4 in village 4. In the first evening you would take the bus from village 1 to 2. The path of the bus would lead through village 0 (the path is 1 −> 0 −> 2), in which the festival happened on day J = 0, i.e. before day A, hence condition A ≤ J ≤ B is violated. As you will not ever visit festival in a village you pass through, you would be extremely sad. You would pass through village 0 again after the day K = 3 during travel to village B = 4.

Write an efficient algorithm for the following assumptions:

* N is an integer within the range [2..1,000];
* each element of array T is an integer within the range [0..N−1];
* there is exactly one (possibly indirect) connection between any two distinct villages.