Teamfulness

Time limit: 4 seconds
Memory limit: 1024 MiB

Anton was recently walking around Kaunas during EJOI when he discovered N spots where participants hang out. The spots are numbered from 0 to $N-1$. Each spot is occupied by members of exactly one team, and teams are numbered from 0 to $K-1$. Note that members of the same team may occupy **arbitrarily many** spots. Some teams may occupy no spots at all.

The spots are connected by $N-1$ two-way roads such that there is exactly one *simple path* between any two spots, so they form a tree. Recall that a simple path between two spots is a sequence of distinct spots in which every two consecutive spots are connected by a road. The *length* of a path is the number of roads it uses, that is one less than the number of spots it visits.

Anton wants to take a walk along a simple path, visiting as many spots as possible. A simple path is called *interesting* if its length is the maximum possible among all simple paths in the tree. The *teamfulness* of a path is the number of distinct teams Anton encounters along it.

Your task is to find the sum of teamfulness over all different interesting paths. Two interesting paths are considered the same if and only if they visit exactly the same set of spots (in particular, traversing a path in the opposite direction does not produce a different path).

Implementation details

You should implement the following function:

```
long long teamfulness(int N, int K, std::vector<int> a,
                      std::vector<int> u, std::vector<int> v)
```

- N: the number of spots;
- K: the number of teams;
- a: an array of N integers, where a_i is the team occupying spot i, for each $0 \le i < N$;
- u, v: arrays of $N-1$ integers, where u_i and v_i are the two spots connected by the i-th road, for each $0 \le i < N-1$.

This function is called exactly once per test and must return the sum of teamfulness over all interesting paths.

Constraints

- $3 \leq N \leq 10^6$
- $1 \leq K \leq N$
- $0 \leq a_i < K$ for each $0 \leq i < N$
- $0 \leq u_i, v_i < N$ for each $0 \leq i < N - 1$

Examples

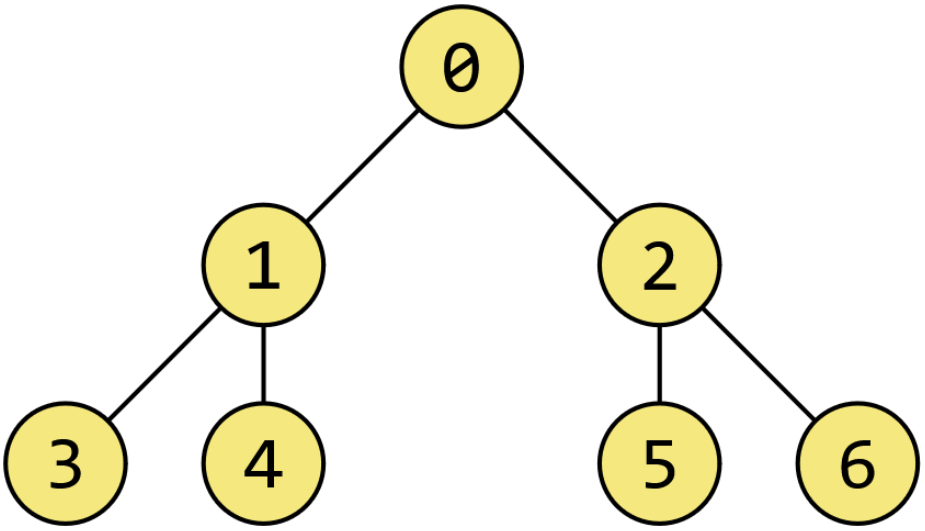
Sample grader input	Sample grader output
6 3 1 0 0 1 2 1 0 1 0 2 0 3 0 4 0 5	21
7 1 0 0 0 0 0 0 0 0 1 0 2 1 3 1 4 2 5 2 6	4
6 3 0 1 2 0 1 2 0 1 1 2 2 3 1 4 2 5	11

Explanation of example 1

The maximum length of a simple path is 2 (2 roads, 3 spots), hence interesting paths will have length 2. There are two interesting paths with teamfulness 3, seven interesting paths with teamfulness 2, and one interesting path with teamfulness 1, for a total sum of 21.

Explanation of example 2

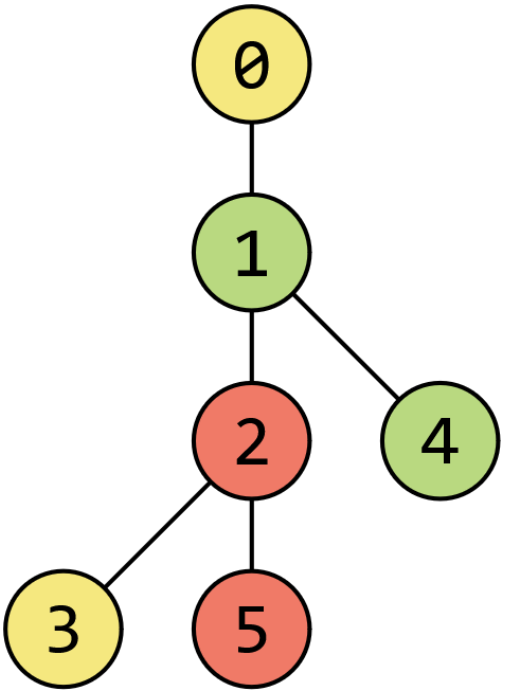
The spots and the roads connecting them are illustrated as follows (team 0 is colored in yellow):



The teamfulness of every path is 1 because there is only one team (team 0) that hangs out at every spot. There are 4 interesting paths (of length 4), hence the sum of teamfulness over all interesting paths is 4 as well.

Explanation of example 3

The spots and the roads connecting them are illustrated as follows (team 0 is colored in yellow, team 1 is colored in green, team 2 is colored in red):



Interesting paths have length 3. There are four interesting paths in total: three have teamfulness 3, and one has teamfulness 2. Therefore, the total sum of teamfulness over all interesting paths is 11.

Subtasks

Subtask	Points	N	K	Additional constraints
0	0	–	–	The examples.
1	4	$\leq 10^6$	$\leq N$	Every spot is directly connected to at most 2 other spots.
2	7			There is a spot that is directly connected to every other spot.
3	9	≤ 200	$\leq N$	–
4	10	$\leq 2 \cdot 10^3$		
5	10	$\leq 10^6$	$= 1$	
6	9		≤ 2	
7	11	$\leq 2 \cdot 10^5$	≤ 50	
8	12		$\leq N$	
9	13	$\leq 10^6$	$\leq N$	The length of an interesting path is odd.
10	15			–

Sample grader

The input format is the following:

- line 1: two integers – the values of N and K;
- line 2: N integers $a_0, a_1, \ldots, a_{N-1}$, where a_i is the team occupying spot i;
- line $3 + i$: two integers u and v – the two spots connected by the i-th road.

The output format is the following:

- line 1: one integer – the return value of the call.