C. Kefa and Park

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

Kefa decided to celebrate his first big salary by going to the restaurant.

He lives by an unusual park. The park is a rooted tree consisting of *n* vertices with the root at vertex 1. Vertex 1 also contains Kefa's house. Unfortunaely for our hero, the park also contains cats. Kefa has already found out what are the vertices with cats in them.

The leaf vertices of the park contain restaurants. Kefa wants to choose a restaurant where he will go, but unfortunately he is very afraid of cats, so there is no way he will go to the restaurant if the path from the restaurant to his house contains more than *m* **consecutive** vertices with cats.

Your task is to help Kefa count the number of restaurants where he can go.

**Input**

The first line contains two integers, *n* and *m* (2 ≤ *n* ≤ 105, 1 ≤ *m* ≤ *n*) — the number of vertices of the tree and the maximum number of consecutive vertices with cats that is still ok for Kefa.

The second line contains *n* integers *a*1, *a*2, ..., *an*, where each *ai* either equals to 0 (then vertex*i* has no cat), or equals to 1 (then vertex *i* has a cat).

Next *n* - 1 lines contains the edges of the tree in the format "*xi* *yi*" (without the quotes) (1 ≤ *xi*, *yi* ≤ *n*, *xi* ≠ *yi*), where *xi* and *yi* are the vertices of the tree, connected by an edge.

It is guaranteed that the given set of edges specifies a tree.

**Output**

A single integer — the number of distinct leaves of a tree the path to which from Kefa's home contains at most *m* consecutive vertices with cats.

**Examples**

**input**

4 1  
1 1 0 0  
1 2  
1 3  
1 4

**output**

2

**input**

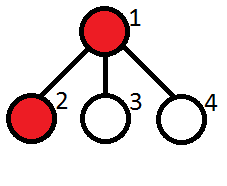
7 1  
1 0 1 1 0 0 0  
1 2  
1 3  
2 4  
2 5  
3 6  
3 7

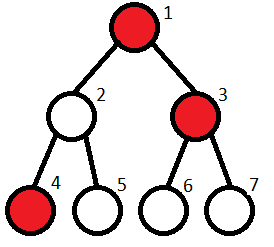
**output**

2

**Note**

Let us remind you that a *tree* is a connected graph on *n* vertices and *n* - 1 edge. A *rooted* tree is a tree with a special vertex called *root*. In a rooted tree among any two vertices connected by an edge, one vertex is a parent (the one closer to the root), and the other one is a child. A vertex is called a *leaf*, if it has no children.

Note to the first sample test:The vertices containing cats are marked red. The restaurants are at vertices 2, 3, 4. Kefa can't go only to the restaurant located at vertex 2.

Note to the second sample test:The restaurants are located at vertices 4, 5, 6, 7. Kefa can't go to restaurants 6, 7.

这道题进入了一个逻辑陷阱……我把里面的edge当成了有向边，结果这个是无向的……然后就思考在读入一个无向图以后怎么把这个无向图变成一个有向的树……但是最后发现的确只需要改一下判断的条件，将叶节点的判断条件从child数组为empty变成没有可以去的节点就可以了……

把大象放进冰箱要几个步骤？

三步，一，打开冰箱，二，放进大象，三，关上门

#include<iostream>

#include<vector>

#define MAX 1000

using namespace std;

struct tree

{

int id;

vector<tree\*>child;

};

vector<int>cat;

vector<tree\*>store;

vector<int>vst;

int vertic\_num, cat\_toler;

int cnt = 0;

void dfs(int index, int current\_cat)

{

vst[index] = 1;

if (cat[index])

current\_cat += cat[index];

else

current\_cat = 0;

if (current\_cat > cat\_toler)

return;

int flag = 0;

for (int i = 0; i < store[index]->child.size(); i++)

{

if (!vst[store[index]->child[i]->id])

{

flag = 1;

dfs(store[index]->child[i]->id, current\_cat);

}

}

if (!flag)

{

cnt++;

}

}

int main()

{

cin >> vertic\_num >> cat\_toler;

cat.resize(vertic\_num);

store.resize(vertic\_num);

vst.resize(vertic\_num,0);

for (int i = 0; i < vertic\_num; i++)

{

store[i] = new tree;

store[i]->id = i;

}

for (int i = 0; i < vertic\_num; i++)

{

cin >> cat[i];

}

for (int i = 1; i < vertic\_num; i++)

{

int father, son;

cin >> father >> son;

store[--father]->child.push\_back(store[--son]);

store[son]->child.push\_back(store[father]);

}

dfs(0, 0);

cout << cnt;

}