Problem Name: Rotating Tree

Topic: Trees

Tags: Trees, Array

Language used: C++

Difficulty: Hard

Problem Statement:

You are given a binary tree with N nodes and N-1 edges rooted at node 1, but this tree has a unique property of cyclically rotating its nodes at every level after each second. For example if nodes at level 3 are [4, 5, 6, 7] then after 1 second they will be [7, 4, 5,6]. You need to find the sum of middle elements of all the levels of the tree after K seconds. If there are X nodes at some level then the middle node would be $((X/2)+1)^{th}$ node.
Initially the nodes at each level are in ascending order of node value.
Note: Here floor division is used for finding the middle element.

Input Format:

The first line of input contains N denoting the number of nodes.
Next N-1 lines contain two integers u, v denoting an edge between nodes u and v.
Next line contains an integer K, denoting the seconds.

Output Format:

Output the sum of middle elements of all the levels of the tree after K seconds.

Constraints:

$1 <= N <= 1000$
$1 <= u, v <= N$
$1 <= K <= 10^7$

Sample Input 1:

7
1 2

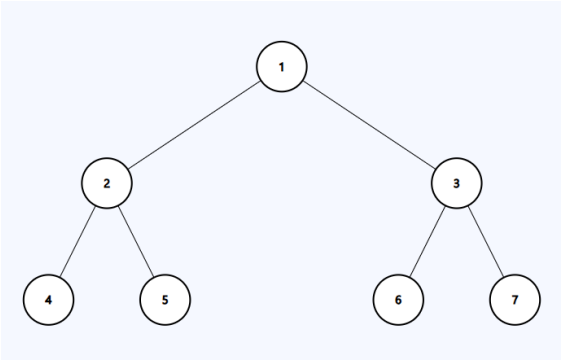
1 3
2 4
2 5
3 6
3 7
2

Sample Output 1:

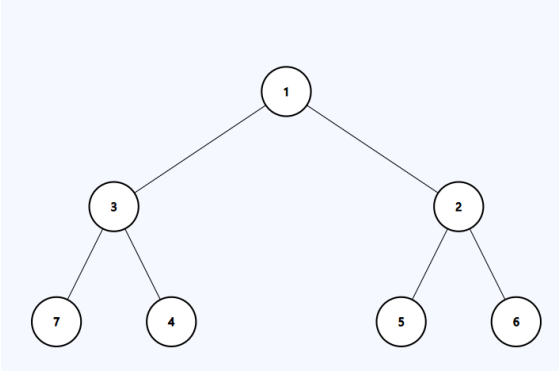
8

Explanation of Sample Input 1:

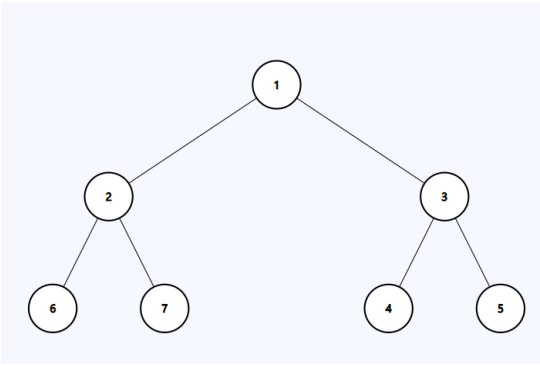
Initially the tree would look like this,



After 1 second, tree would be like,



After 2 seconds, tree would be like,



Now the middle element at level 0 is 1.

Middle element at level 1 is 3, as there are 2 nodes at level 1 and the middle node would be $((2/2)+1)^{th}$ node, which would be the second node of level 1.

Middle element at level 2 is 4, as there are 4 nodes at level 2 and the middle node would be $((4/2)+1)^{th}$ node, which would be the third node of level 2.

Thus, sum = 1 + 3 + 4 = 8.

Sample Input 2:

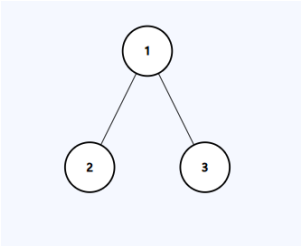
```
3
1 3
1 2
1
```

Sample Output 2:

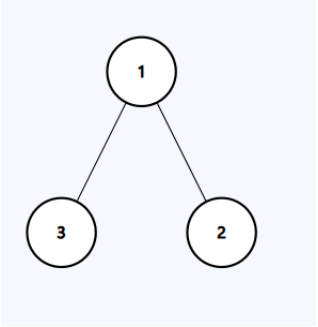
```
3
```

Explanation of Sample Input 1:

Initially the tree would look like this,



After 1 second, it would be like,



Now the sum of middle elements is 1 + 2 =3.

Code:

```
#include <bits/stdc++.h>

using namespace std;
long int calculate(vector<long int>& temp, long int K){
   sort(temp.begin(),temp.end());
   int middle= temp.size()/2 + 1;
   int size = temp.size();
   K%=size;
   int index;
   if (K >= middle)
     index = (size - K) + (middle - 1);
   else
     index = (middle - K - 1);
   return temp[index];
}
int main()
{
   long int n,u,v,ans=0,i,K;
   cin>>n;
   vector<long int>adj[n+1];
   for(i=0;i<n-1;i++){
      cin>>u>>v;
      adj[u].push_back(v);
   }
   cin>>K;
   queue<vector<long int>>q;
   q.push({1, 0});
   int cur=0;
   vector<long int>temp;
   while(!q.empty()){
      long int node = q.front()[0];
      long int level = q.front()[1];
      q.pop();
      if(level==cur)temp.push_back(node);
      else{
         ans+= calculate(temp, K);
         temp.clear();
         cur=level;
         temp.push_back(node);
      }
      for(auto it:adj[node]){
         q.push({it, level+1});
      }

   }
   ans+= calculate(temp, K);
   cout<<ans<<endl;

   return 0;
}
```

C++
Python
Java