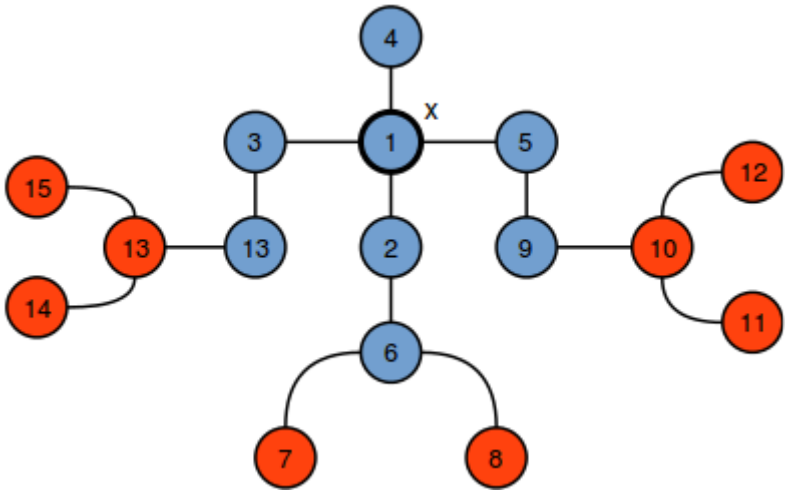
Problem Statement 1

Jenny loves experimenting with trees. Her favorite tree has nodes connected by edges, and each edge is unit in length. She wants to cut a subtree (i.e., a connected part of the original tree) of radius from this tree by performing the following two steps:

1. Choose a node, , from the tree.
2. Cut a subtree consisting of all nodes which are not further than units from node .

For example, the blue nodes in the diagram below depict a subtree centered at that has radius :



Given , , and the definition of Jenny's tree, find and print the number of different subtrees she can cut out. Two subtrees are considered to be different if they are not isomorphic.

Input Format

The first line contains two space-separated integers denoting the respective values of and .
Each of the next subsequent lines contains two space-separated integers, and , describing a bidirectional edge in Jenny's tree having length .

Constraints

- 1<= n<=3000
- 0<=r <=3000
- 1 <= x,y <=n

Subtasks

For 50% of the max score:

- 1<= n <= 500
- 0<= r <= 500

Output Format

Print the total number of different possible subtrees.

Sample Input 0

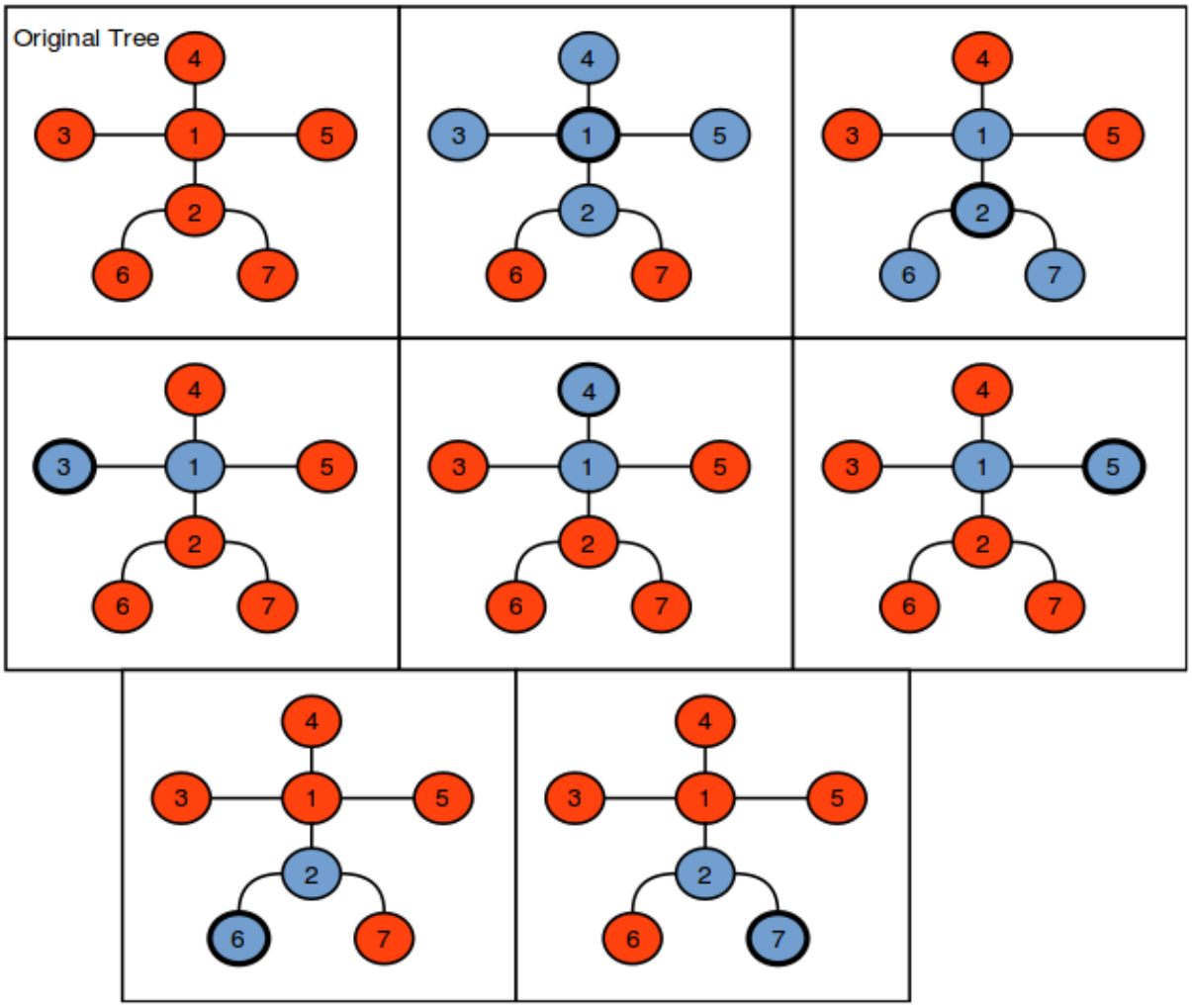
7 1
1 2
1 3
1 4
1 5
2 6
2 7

Sample Output 0

3

Explanation 0

In the diagram below, blue nodes denote the possible subtrees:



The last subtrees are considered to be the same (i.e., they all consist of two nodes connected by one edge), so we print as our answer.

Sample Input 1

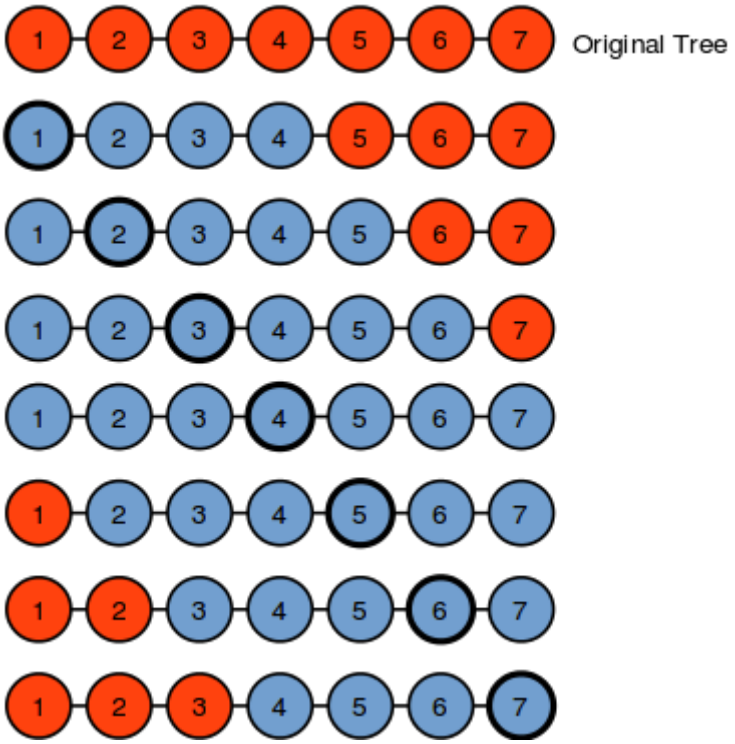
7 3
1 2
2 3
3 4
4 5
5 6
6 7

Sample Output 1

4

Explanation 1

In the diagram below, blue nodes denote the possible subtrees:



Here, we have four possible different subtrees.

Problem Statement 2

Problem Statement 2
Encryption

Lexicographical order is often known as alphabetical order when dealing with strings. A string is greater than another string if it comes later in a lexicographically sorted list.
Given a word, create a new word by swapping some or all of its characters. This new word must meet two criteria:

- It must be greater than the original word
- It must be the smallest word that meets the first condition

For example, given the word w=abcd , the next largest word is abdc.
Complete the function biggerIsGreater below to create and return the new string meeting the criteria. If it is not possible, return no answer.
Function Description
Complete the biggerIsGreater function in the editor below. It should return the smallest lexicographically higher string possible from the given string or no answer.
biggerIsGreater has the following parameter(s):

- w: a string

Input Format
The first line of input contains T, the number of test cases.
Each of the next T lines contains w .
Constraints

- 1<= T <= 10^5
- 1<= |w| <= 100

- will contain only letters in the range ascii[a..z].

Output Format
For each test case, output the string meeting the criteria. If no answer exists, print no answer.
Sample Input 0
5
ab
bb
hefg
dhck
dkhc

Sample Output 0

ba
no answer
hegf
dhkc
hcdk

Explanation 0

- Test case 1:
- ba is the only string which can be made by rearranging ab. It is greater.
- Test case 2:
- It is not possible to rearrange bb and get a greater string.
- Test case 3:
- hegf is the next string greater than hefg.
- Test case 4:
- dhkc is the next string greater than dhck.
- Test case 5:
- hcdk is the next string greater than dkhc.
-

Sample Input 1

6
lmno
dcba
dcbb
abdc
abcd
fedcbabcd

Sample Output 1
lmon
no answer
no answer
acbd
abdc
fedcbabdc