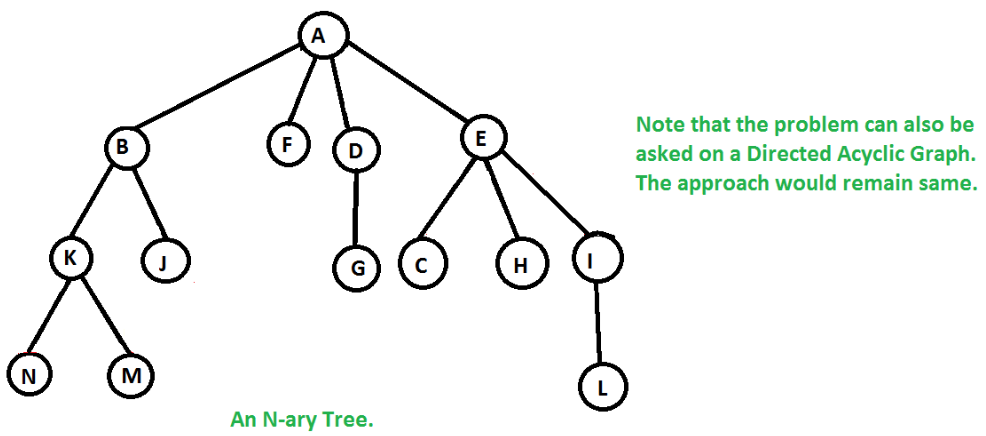
Program Name/Purpose- Number of ways to traverse an N-ary(or a Directed Acyclic Graph) tree starting from the root vertex.

Programming Paradigm Used- Factorial, Permutation, Combinatorics, Observation

Algorithm Used- Level Order Traversal(in case of tree)/ Breadth First Search(in case of Directed Acyclic Graph)

Data Structures Used- A queue for level order traversal, a structure having a "key" field that holds the data of the node and a "pointer" to another structure to point towards its children.

Explanation- Suppose we have a given N-ary tree as shown below-



An N-ary Tree.

Now we have to find the number of ways of traversing the **whole tree** starting from the **root vertex.**

There can be many such ways. Some of them are listed below-

1) **N->M->K->J->B->F->D->E->C->H->I->L->A (kind-of depth first traversal).**
2) **A->B->F->D->E->K->J->G->C->H->I->N->M->L (level order traversal)**
3) ....
4) ....
.
.
.

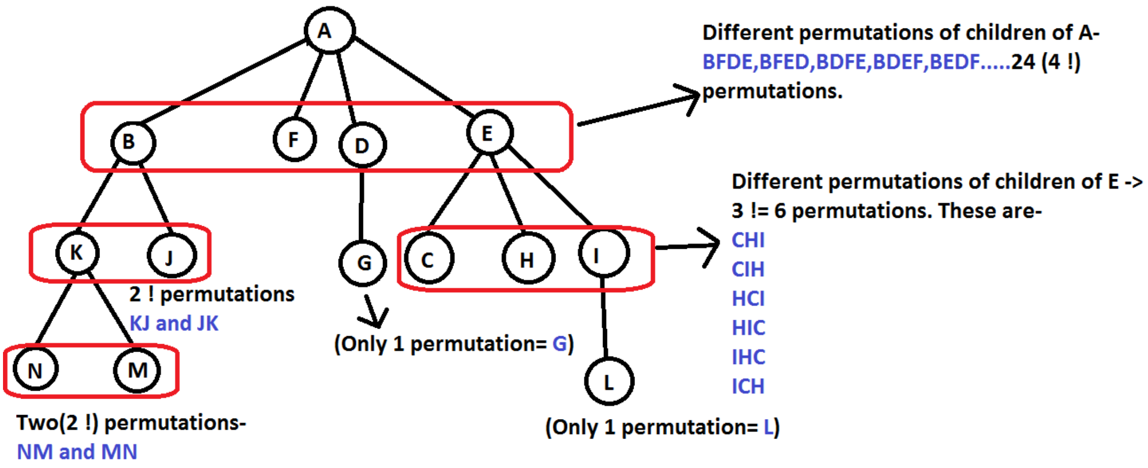
and so on....

As it might be clear, there can be many such ways.
How to get the count of all such ways?

Simple.
<u>The count of all such ways is the product of factorials of the number of children of each node.</u>

How?

Refer to the below figure for clear understanding-



Here,

'A' has four children, so – 4 ! permutations possible
'B' has two children, so – 2 ! permutations possible
'F' has no children, so – 0 ! permutations possible
'D' has one children, so – 1 ! permutations possible
'E' has three children, so – 3 ! permutations possible
'K' has two children, so – 2 ! permutations possible
'J' has no children, so – 0 ! permutations possible
'G' has no children, so – 0 ! permutations possible
'C' has no children, so – 0 ! permutations possible
'H' has no children, so – 0 ! permutations possible
'I' has one children, so – 1 ! permutations possible
'N' has no children, so – 0 ! permutations possible
'M' has no children, so – 0 ! permutations possible
'L' has no children, so – 0 ! permutations possible

Hence all such ways are- 4 ! * 2 ! * 0 ! * 1 ! * 3 ! * 2 ! * 0 ! * 0 ! * 0 ! * 0 ! * 1 ! * 0 ! * 0 ! = **576 ways**

That's a huge number of ways and sadly among them only few proves to be useful, like-
inorder,level-order,preorder,postorder (arranged according to the popularity of these traversals)

Time Complexity- Since we are visiting each node once during the level order traversal and getting the count of their children is a **O(1)** operation, hence overall time complexity is – **O(N)**,
where N= number of nodes in the N-ary tree

Space Complexity- Since we are only using a **queue** and a **structure** for every node, so overall space complexity is also **O(N).**

Alternatives to solve this problem-

Any other traversal technique (depth-first search etc.) will also work fine.

Common Pitfalls-

Since, products of factorials can tend to grow very huge, so it may overflow. It is preferable to use data types like- *unsigned long long int* in C/C++, as the number of ways can never be a negative number. In Java and Python there are *Big Integer* to take care of overflows.