$D[x]$ = the number of different topological orderings in the subtree of x.

By selecting the ordering for each subtree of y (where y is a child of x), the only remaining part is to merge the orderings. This can be done very easy using a lot of combinations or applying the number of anagrams. The formula for the number of anagrams is $n! / (\prod f[x]!)$, where n is the number of letters and $f[x]$ the number of occurrences of letter x.

$D[x] = (\prod d[y]) * [(v[x] - 1)! / (\prod v[y]!)]$, for each y which is a child of x. $V[x]$ is the number of nodes in the subtree of x.

By opening this recursion and divide all the factorials, we can obtain a formula for the number of topological orderings: $(n - 1)! / (\prod v[x])$, for each x which is a node in the tree different from the root.

$(n - 1)!$ can be obtained very easy from the helper which is $n!$. We just have to divide it by n (multiply it with the inverse modular).

The current complexity is O(M * N * log VALMAX). For each query we loop throw all N values $v[x]$ and divide the answer using inverse modular.

In order to make the program fast it was necessary to analyze the test cases and observe the fact that the tree is generated random. Upon testing we can deduce that there are not that many different values for $v[x]$, so for each query we can apply the formula and loop only throw all the different values of $v[x]$.

Complexity: O(M * C * log VALMAX), where C is a constant for the number of different values of $v[x]$. For N = 3.000.000, this value is close to C = 2000.