Purpose of the Project-

Apart from the solution mentioned above there are three other possible solutions to the above problem-

1) **Brute Force** - O(N!) time and O(N) space

2) **Smart Brute Force-** O(**sqrt**(N!)) time and O(N) space

3) **Dynamic Programming** – O(N^2 **sqrt**(N) **log**(N)) time and O(N^2) space. [Most elegant solution]

The detailed explanation of each of the above methods are-

Method 1 (Brute Force)-

We calculate the factorial of the given number - num and then we run a loop from i=1 to i= num! and check whether i is a divisor of num! or not. If it is a divisor then we count it otherwise we neglect it.

Time Complexity-

O(N!), because we are running a loop from i=1 to i=N!.

Space Complexity-

O(N), because of the space taken by recursion stack in calculating the factorial of N

Limitations-

1) Inefficient
2) In languages like C/C++, the factorial value overflows for N > 20.

Method 2 (Intelligent Brute Force)-

We calculate the factorial of the given number - num and then we run a loop from i=1 to i= **sqrt**(num!) and check whether i is a divisor of num! or not. If it is a divisor and if it is a square of num! then we count only once otherwise we count two numbers (num! / i and i both).

Time Complexity-

O(**sqrt**(N!)), because we are running a loop from i=1 to i=**sqrt**(N!).

Space Complexity-

O(N), because of the space taken by recursion stack in calculating the factorial of N

Limitations-

1) Although more efficient than the method 2 but still very inefficient.
2) In languages like C/C++, the factorial value overflows for N > 20.

Method 3 (Dynamic Programming) [Elegant Solution]-

There is also an elegant method to solve this using Dynamic Programming.

It might be visible that there is a recurrence relation-

count_of_prime_divisors (N!) = count_of_prime_divisors ((N-1)!) + count_of_prime_divisors (N)

Then using the above recurrence relation we create and fill a DP table of N*N size.

The rows contains the numbers from 1 to N and we only fill those rows with index as prime, i.e- i=2, i=3, i=5...etc will be filled.

Then once we have calculate the count of all the respective primes, then we use the concept-

"Any positive integer can be expressed as product of power of its prime factors. Suppose a number n = p_1^{a1} x p_2^{a2} x p_3^{a3}, …., p_k^{ak} where p_1, p_2, p_3, …., p_k are distinct primes and a1, a2, a3,………….., ak are their respective exponents.
Then the no of divisors of n = (a1+1) x (a2+1) x (a3+1)…x (ak+1)"

Time Complexity-

$O(N^2$ **sqrt**(N) **log**$(N))$

Space Complexity-

$O(N^2)$, because of the space taken by the 2D DP table.
