Factorial Trailing Zeroes

Given an integer n, return the number of trailing zeroes in $n!$.

Note: Your solution should be in logarithmic time complexity.

Credits:
Special thanks to @ts for adding this problem and creating all test cases.

Solution 1

This question is pretty straightforward.

Because all trailing 0 is from factors 5 * 2.

But sometimes one number may have several 5 factors, for example, 25 have two 5 factors, 125 have three 5 factors. In the n! operation, factors 2 is always ample. So we just count how many 5 factors in all number from 1 to n.

One line code:

Java:

```
return n == 0 ? 0 : n / 5 + trailingZeroes(n / 5);
```

C++:

```
return n == 0 ? 0 : n / 5 + trailingZeroes(n / 5);
```

Python:

```
return 0 if n == 0 else n / 5 + self.trailingZeroes(n / 5)
```

written by xcv58 original link here

Solution 2

The idea is:

1. The ZERO comes from 10.
2. The 10 comes from 2 x 5
3. And we need to account for all the products of 5 and 2. likes 4×5 = 20 ...
4. So, if we take all the numbers with 5 as a factor, we'll have way more than enough even numbers to pair with them to get factors of 10

Example One

How many multiples of 5 are between 1 and 23? There is 5, 10, 15, and 20, for four multiples of 5. Paired with 2's from the even factors, this makes for four factors of 10, so: **23! has 4 zeros**.

Example Two

How many multiples of 5 are there in the numbers from 1 to 100?

because 100 ÷ 5 = 20, so, there are twenty multiples of 5 between 1 and 100.

but wait, actually 25 is 5×5, so each multiple of 25 has an extra factor of 5, e.g. 25 × 4 = 100， which introduces extra of zero.

So, we need know how many multiples of 25 are between 1 and 100? Since 100 ÷ 25 = 4, there are four multiples of 25 between 1 and 100.

Finally, we get 20 + 4 = 24 trailing zeroes in 100!

The above example tell us, we need care about 5, 5×5, 5×5×5, 5×5×5×5 ....

Example Three

By given number 4617.

5^1 : 4617 ÷ 5 = 923.4, so we get 923 factors of 5

5^2 : 4617 ÷ 25 = 184.68, so we get 184 additional factors of 5

5^3 : 4617 ÷ 125 = 36.936, so we get 36 additional factors of 5

5^4 : 4617 ÷ 625 = 7.3872, so we get 7 additional factors of 5

5^5 : 4617 ÷ 3125 = 1.47744, so we get 1 more factor of 5

5^6 : 4617 ÷ 15625 = 0.295488, which is less than 1, so stop here.

Then 4617! has 923 + 184 + 36 + 7 + 1 = 1151 trailing zeroes.

C/C++ code

```
int trailingZeroes(int n) {
    int result = 0;
    for(long long i=5; n/i>0; i*=5){
        result += (n/i);
    }
    return result;
}
```

---------update-----------

To avoid the integer overflow as **@localvar** mentioned below(in case of 'n >=1808548329'), the expression " i <= INT*MAX/5" is not a good way to prevent overflow, because 5^13 is > INT*MAX/5 and it's valid.

So, if you want to use "multiply", consider define the 'i' as 'long long' type.

Or, take the solution **@codingryan** mentioned in below answer!

written by haoel original link here

Solution 3

10 is the product of 2 and 5. In n!, we need to know how many 2 and 5, and the number of zeros is the minimum of the number of 2 and the number of 5.

Since multiple of 2 is more than multiple of 5, the number of zeros is dominant by the number of 5.

Here we expand

```
  2147483647!
=2 * 3 * ...* 5 ... *10 ... 15* ... * 25 ... * 50 ... * 125 ... * 250...
=2 * 3 * ...* 5 ... * (5^1*2)...(5^1*3)...*(5^2*1)...*(5^2*2)...*(5^3*1)...*(5^3*
2)... (Equation 1)
```

We just count the number of 5 in Equation 1.

Multiple of 5 provides one 5, multiple of 25 provides two 5 and so on.

Note the duplication: multiple of 25 is also multiple of 5, so multiple of 25 only provides one extra 5.

Here is the basic solution:

```
return n/5 + n/25 + n/125 + n/625 + n/3125+...;
```

You can easily rewrite it to a loop.

written by gqq original link here