*Digits OF N!*

A simple method is to first calculate factorial of n, then count trailing 0s in the result (We can count trailing 0s by repeatedly dividing the factorial by 10 till the remainder is 0).

The above method can cause overflow for a slightly bigger numbers as factorial of a number is a big number (See factorial of 20 given in above examples). The idea is to consider prime factors of a factorial n. A trailing zero is always produced by prime factors 2 and 5. If we can count the number of 5s and 2s, our task is done. Consider the following examples.

**n = 5:** There is one 5 and 3 2s in prime factors of 5! (2 \* 2 \* 2 \* 3 \* 5). So count of trailing 0s is 1.

**n = 11:** There are two 5s and three 2s in prime factors of 11! (2 8 \* 34 \* 52 \* 7). So count of trailing 0s is 2.

We can easily observe that the number of 2s in prime factors is always more than or equal to the number of 5s. So if we count 5s in prime factors, we are done. How to count total number of 5s in prime factors of n!? A simple way is to calculate floor(n/5). For example, 7! has one 5, 10! has two 5s. It is done yet, there is one more thing to consider. Numbers like 25, 125, etc have more than one 5. For example if we consider 28!, we get one extra 5 and number of 0s become 6. Handling this is simple, first divide n by 5 and remove all single 5s, then divide by 25 to remove extra 5s and so on. Following is the summarized formula for counting trailing 0s.

Trailing 0s in n! = Count of 5s in prime factors of n!

= floor(n/5) + floor(n/25) + floor(n/125) + ....

int findTrailingZeros(int n)

{

    // Initialize result

    int count = 0;

    // Keep dividing n by powers of

    // 5 and update count

    for (int i = 5; n / i >= 1; i \*= 5)

        count += n / i;

    return count;

}