# **Leetcode 172 – Factorial Trailing Zeroes** using **recursion**

## Problem Understanding

Given an integer n, return the number of **trailing zeroes** in n! (n factorial).

### Key Idea:

* Trailing zeroes come from factors of **10**, i.e., **2 × 5**.
* Since there are always more 2s than 5s in a factorial, the count of **5s** determines the number of trailing zeroes.

## Optimized Java Solution (Recursive)

class Solution {

public int trailingZeroes(int n) {

if (n == 0) return 0;

return n / 5 + trailingZeroes(n / 5);

}

}

### How It Works:

* At each step, count how many numbers ≤ n are divisible by 5, 25, 125, ...
* Use recursion to add all those contributions.
* Base case: when n == 0, stop recursion.

## Dry Run Using Table

Let’s dry run for n = 100:

|  |  |  |
| --- | --- | --- |
| Call | Value Returned | Explanation |
| trailingZeroes(100) | 20 + T(20) | 100 / 5 = 20 |
| trailingZeroes(20) | 4 + T(4) | 20 / 5 = 4 |
| trailingZeroes(4) | 0 + T(0) | 4 / 5 = 0 |
| trailingZeroes(0) | 0 | Base case |
| **Backtrack:** |  |  |
| T(4) returns | 0 |  |
| T(20) returns | 4 + 0 = 4 |  |
| T(100) returns | 20 + 4 = 24 | ✅ Final Answer |

## Time / Space Complexity

|  |  |
| --- | --- |
| Metric | Value |
| Time | O(log₅ n) |
| Space | O(log₅ n) |

## Alternate Approaches

### 1. ****Iterative (Recommended for Performance)****

public int trailingZeroes(int n) {

int count = 0;

while (n > 0) {

n /= 5;

count += n;

}

return count;

}

* Same logic, but avoids recursion.
* ✅ Time: O(log₅ n), Space: O(1)

### 2. ****Brute Force (Not Recommended)****

* Compute n! using BigInteger
* Convert to string and count trailing '0'

BigInteger fact = BigInteger.ONE;

for (int i = 2; i <= n; i++) {

fact = fact.multiply(BigInteger.valueOf(i));

}

// count trailing zeros from the string

❌ Very inefficient and fails for large n due to factorial overflow.

### 3. ****Recursive (Educational Version)****

* Great for understanding recurrence and base case.
* Trade-off: uses call stack (not tail recursive in Java).