Running time: The running time depends on the size of its input, expressed as $T(n)$ ($T(n) \geq 0$) where n is the input.

↳ growth rate: $T(n) = cn$.

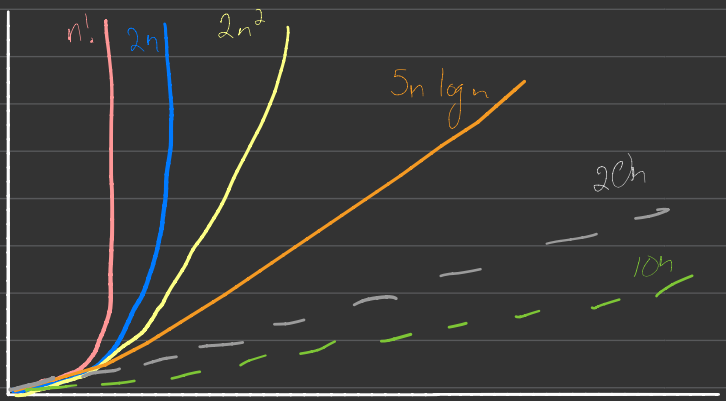
↳ C is the constant.

What is the growth rate for the following code:

```
sum = 0;
for (i = 1; i <= n; i++)
    for (j = 1; j <= n; j++)
        sum++;
```

Assuming incrementing is constantly increasing, use C_2 for time. And the total number of operations is n^2.
Using the expression $T(n) = cn$, we know $T(n) = C_2 n^2$.

Growth rates:



Best, Worst, and Average Cases:

- Best case: when 1 integer is examined
- Worst case: when all n integers need to be examined
- Average case: $n/2$

Big-O, Big-Ω, Big Θ:

- Big-O: upper bound for growth (often running time)
  $f(n)$ for the upper bound,
  For $T(n)$ a non-negatively valued function $T(n) \to O(f(n))$ if there exists 2 positive constants c and n_0 s.t. $T(n) \leq cf(n)$ for all $n > n_0$.

- Big-Ω: lower bound
  $T(n) \to \Omega(g(n))$

- Big-Θ: when upper/lower bounds are the same within a constant factor.
  if $f(n)$ is $\Theta(g(n))$ then $g(n)$ is $\Theta(f(n))$