November 4, 2020: Runtime Analysis

- Considering worst-case scenarios and big inputs (10^9)
- For analyzing, we'll use the RAM Model, with the following principles
  - High-level operations ($+, -, \div, \times$, logical statements, setting values) take constant time (is independent of input size)
  - Memory access takes constant time
- To denote the runtime of a program/function $f(n)$, we'll use $T(f(n))$
- Let's say we have a function f. What is the most mathematically correct way to denote $T(f(n))$

```
f(n){
  ans = 0 ←
  for i from 1 to n:
    ans += i ←
  return ans ←
}
```

$n \times c$ time

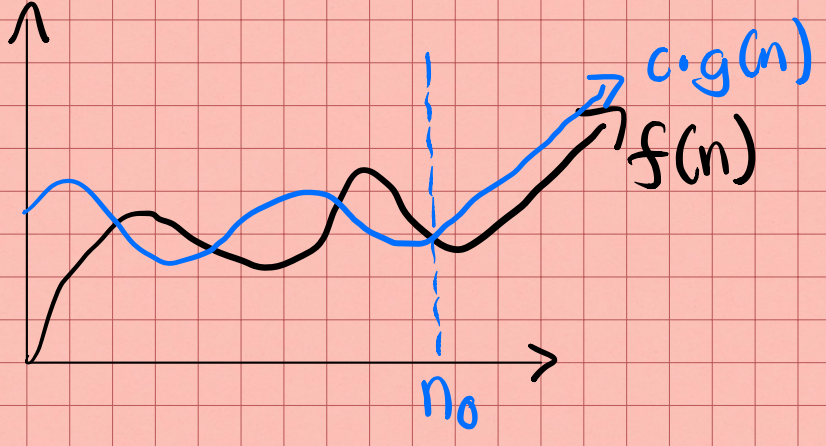
A. $T(f(n)) = O(n^2)$

B. $T(f(n)) \in O(\sqrt{n})$

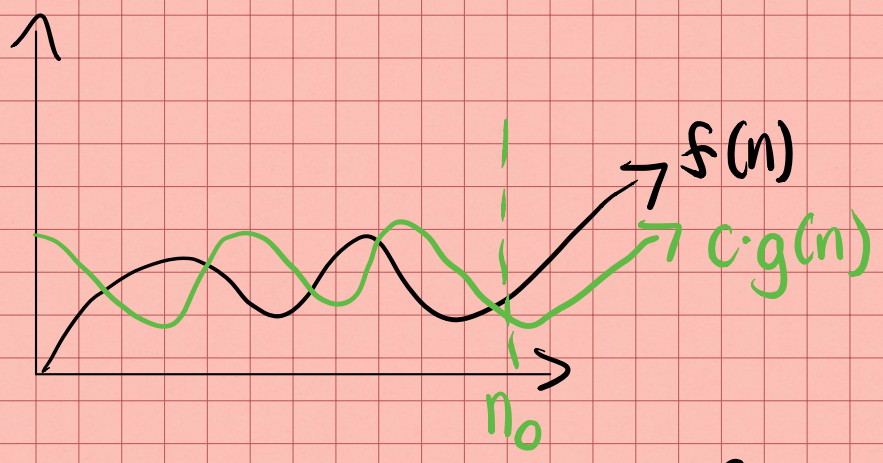
C. $T(f(n)) \in O(n^n)$

D. $T(f(n)) \notin O(n)$ ✓✓✓

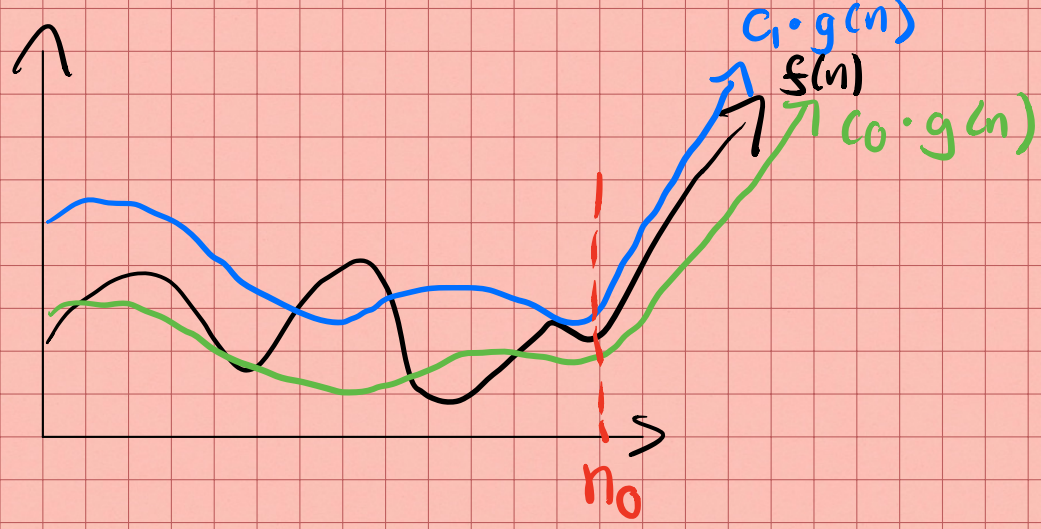
- The most common ways for analyzing runtimes (all borrowed from math
  - Big-O Notation: $O(g(n)) = \{ f(n) \mid \exists$ positive c and n_0 s.t. $\forall n \geq n_0, 0 \leq f(n) \leq c \cdot g(n) \}$
    - "Upper bound"



- Big-Ω Notation: $\Omega(g(n)) = \{f(n) \mid \exists$ positive c and n_0 s.t. $\forall n \geq n_0, 0 \leq c \cdot g(n) \leq f(n)\}$
  - "Lower bound"



- Big-Θ Notation: $\Theta(g(n)) = \{f(n) \mid \exists$ positive constants c_0, c_1, n_0 s.t. $\forall n \geq n_0, 0 \leq c_0 \cdot g(n) \leq f(n) \leq c_1 \cdot g(n)\}$
  - "Tight bound"; $\Theta(g(n)) = O(g(n)) \cap \Omega(g(n))$



* Note that people abuse notation, $f(n) = O(g(n))$ *

- Examples (or Θ or Ω)
  - $\underbrace{3^{60}}_{f(n)} = O(1)$; Constant time
    $n_0 = 1, c = 3^{60}$
  - $\frac{n^2}{80} - 50n = O(n^2)$
  - Just take the biggest power

- Calculate 2^n.

```
A(power) {
  if power = 1, return 2
  else return A(power - 1) x 2
}
```

$$2^3 \rightarrow 2^2 \times 2 \rightarrow (2 \times 2) \times 2 \rightarrow 4 \times 2 = 8$$

O(n) time

```
B(power) {
  if power = 1, return 2
  else:
    temp = B(⌊power/2⌋)
    if power odd, return temp x temp x 2
    if even return temp x temp
}
```

$$2^{17} \rightarrow (2^8 \times 2^8) \times 2 \rightarrow (2^4 \times 2^4) \times 2^8 \times 2$$

$$\vdots$$

O(lg n) time

$\lg = \log_2$