	n	$n \log_2 n$	n^2	n^3	1.5^n	2^n	$n!$
$n = 10$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	4 sec
$n = 30$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	18 min	10^{25} years
$n = 50$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	11 min	36 years	very long
$n = 100$	< 1 sec	< 1 sec	< 1 sec	1 sec	12,892 years	10^{17} years	very long
$n = 1{,}000$	< 1 sec	< 1 sec	1 sec	18 min	very long	very long	very long
$n = 10{,}000$	< 1 sec	< 1 sec	2 min	12 days	very long	very long	very long
$n = 100{,}000$	< 1 sec	2 sec	3 hours	32 years	very long	very long	very long
$n = 1{,}000{,}000$	1 sec	20 sec	12 days	31,710 years	very long	very long	very long

WALL-CLOCK RUNTIMES WITH A PROCESSOR PERFORMING 1 MILLION OPERATIONS PER SECOND

DEF: AN ALGORITHM HAS A (WORST-CASE) RUNTIME OF $f(n)$ IF THE ALGORITHM NEVER MAKES MORE THAN $f(n)$ OPERATIONS WHEN RUN ON INPUTS OF SIZE n.

HOW CAN WE DETERMINE THE RUNTIME OF AN ALGORITHM?

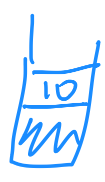
```
DEF ×INC(A):
    RETURN A+1

×INC(10)
```

↑ THIS PIECE OF CODE REQUIRES AT MOST SOME CONSTANT NUMBER, SAY c, OF OPERATIONS.

COMPUTING c IS VERY HARD, BUT IT IS EASY TO OBSERVE THAT c IS SOME CONSTANT ($c \stackrel{?}{=} 27$, $c \stackrel{?}{=} 115$)

① "10" WILL BE PUSHED ONTO THE STACK



② IT ADDS THE VARIABLE "A" TO THE NAMESPACE

③ IT ASSIGNS THE VALUE 10 (WHICH IS POPPED FROM THE STACK) TO THE VARIABLE A.

④ PYTHON RECOVERS THE VALUE OF A FROM MEMORY, IT SUMS 1 TO IT, AND RETURNS THE RESULT

SUPPOSE THAT MAGICALLY WE DETERMINE THAT $c = 27$. OPERATIONS ARE NEEDED TO RUN INC(A).

```
DEF F(n):
    x = 0
    FOR i IN RANGE(n):
        x = INC(i)
    RETURN x
```

```
INC(A):
    RETURN A+1
```

("INC(A) TAKES 27 OPERATIONS")

$$\boxed{c'' + c' \cdot n + 27 \cdot n} \approx (c' + 27) \cdot n$$

```
y = 0
FOR i IN RANGE(n):
    y += F(i)
```

$$\boxed{c'''(c'' + c'n + 27n)\, n} = (c' + 27)c''' \cdot n^2 + c'' \cdot c''' n$$

$$c''' \cdot c'' \cdot n + c''' \cdot c' \cdot n^2 + c''' 27 \cdot n^2$$

O, Ω, Θ NOTATIONS

TO AVOID GETTING STUCK INTO TOO MANY CONSTANTS, WE JUST DISREGARD THEM ALTOGETHER.

SUPPOSE THAT $T(n)$ IS THE RUNTIME OF AN ALGORITHM.

DEF: IF $T(n)$, AND $f(n)$, ARE NON-NEGATIVE, AND INCREASING, FUNCTIONS, WE SAY THAT "$T(n)$ IS $O(f(n))$" IF $\exists\ c, n_0 > 0$ S.T. $\forall\ n \geq n_0$, $T(n) \leq c \cdot f(n)$.

$T(n) = O(f(n))$

$T(n) \leq O(f(n))$

$$T(n) = p\, n^2 + q\, n + r \qquad \text{WITH } p, q, r \geq 0 \text{ AND } p \neq 0.$$

THEN $\forall\ n \geq n_0 = 1$, IT HOLDS THAT $q\,n \leq q\,n^2$ AND $r \leq r\,n^2$.

THUS, $T(n) \leq p\,n^2 + q\,n^2 + r\,n^2$
$= (p + q + r) \cdot n^2$

IF I CHOOSE $c = p + q + r$,
I GET THAT $\forall\ n \geq n_0 = 1$, $T(n) \leq c \cdot n^2$.

$(f(n) = n^2)$

$T(n) \leq O(n^2)$

$T(n) = O(n^2)$

THIS IS TIGHTER/PREFERABLE

THIS IS WEAKER.

$T(n) \leq c \cdot n^3$

$T(n) = O(n^3)$

TRUE OR FALSE?

① "$100 n^2 = O(n^{1.9})$" ?

② "$n = O(1000 n^{1.1})$" ?

③ "$n \leq O(2^n)$" ?

④ "$n \leq O(1.1^n)$" ?

⑤ "$n \leq O(0.9^n)$" ?