TECNOLOGÍAS DE AUTOMATIZACIÓN

# ¿Cómo definirías a la complejidad computacional?

La complejidad computacional es una rama de la informática teórica que estudia los recursos necesarios (como tiempo y espacio) que requiere un algoritmo para resolver un problema, en función del tamaño de la entrada.

# ¿Cuál es el principal propósito de la complejidad computacional?

El propósito principal es clasificar los problemas según su grado de dificultad computacional y determinar cuán eficiente puede ser un algoritmo para resolverlos. Esto ayuda a saber si un problema es tratable (resoluble en un tiempo razonable) o no.

# ¿Qué se analiza?

***Se analizan principalmente dos recursos:***

* Tiempo de ejecución: Cuánto tarda el algoritmo en completarse.
* Espacio de memoria: Cuánta memoria utiliza durante su ejecución. Además, se estudia cómo crecen estos recursos a medida que aumenta el tamaño de los datos de entrada.

# Ejemplos:

***Dato interesante:*** Se sospecha que P ≠ NP, pero aún no está demostrado.

***¿Por qué se sospecha que P ≠ NP?*** Se sospecha que P es distinto de NP porque, aunque es fácil verificar la solución de muchos problemas (NP), no se ha encontrado un algoritmo eficiente (de clase P) para resolverlos.

# Ejemplos:

***Imagina que estás jugando Sudoku:***

* Si alguien te da una solución completa del Sudoku, vos podés **verificar fácilmente** si está bien (cumple las reglas) → eso está en **NP**.
* Pero **resolverlo desde cero**, cuando el tablero está medio vacío, puede tomarte mucho tiempo y prueba/error → aún **no hay un algoritmo eficiente** que lo resuelva siempre rápido → por eso **no está en P (al menos, no se sabe cómo)**.

***Entonces:***

* Si algún día alguien encuentra un algoritmo eficiente para un problema **NP-completo** (como el Sudoku, TSP, etc.), significaría que **P = NP**.
* Pero **nadie lo ha logrado** en décadas de investigación.
* Y muchos problemas NP-completos son tan complicados que **parece muy poco probable** que tengan una solución rápida.

***En resumen:***

Se sospecha que P ≠ NP porque hay muchos problemas que podemos verificar fácilmente, pero resolverlos nos lleva muchísimo tiempo, y nadie ha encontrado un truco para hacerlos rápidos.

# Ejemplo Visual:

