

Dos Retos, Un Mismo Código. ¿Qué Priorizas?



El Desafío: Ordenar 1,000 programas de TV para un cliente.



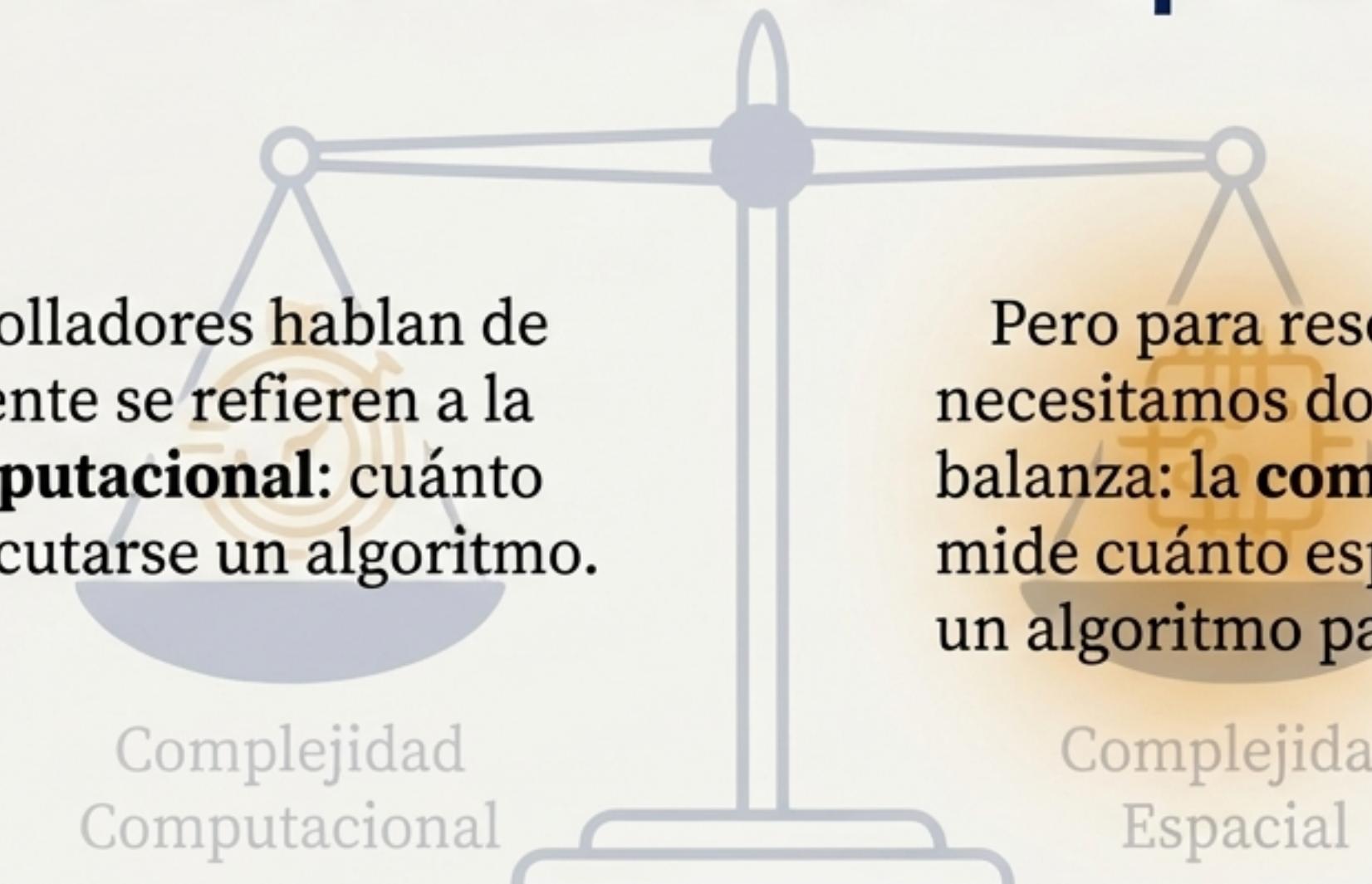
El Desafío: Ordenar 1,000,000 de videos para millones de usuarios.

El Eterno Balance: Velocidad vs. Memoria



Algunos algoritmos consumen mucha memoria pero son rápidos. Otros no usan memoria pero son lentos. La elección depende de tus necesidades.

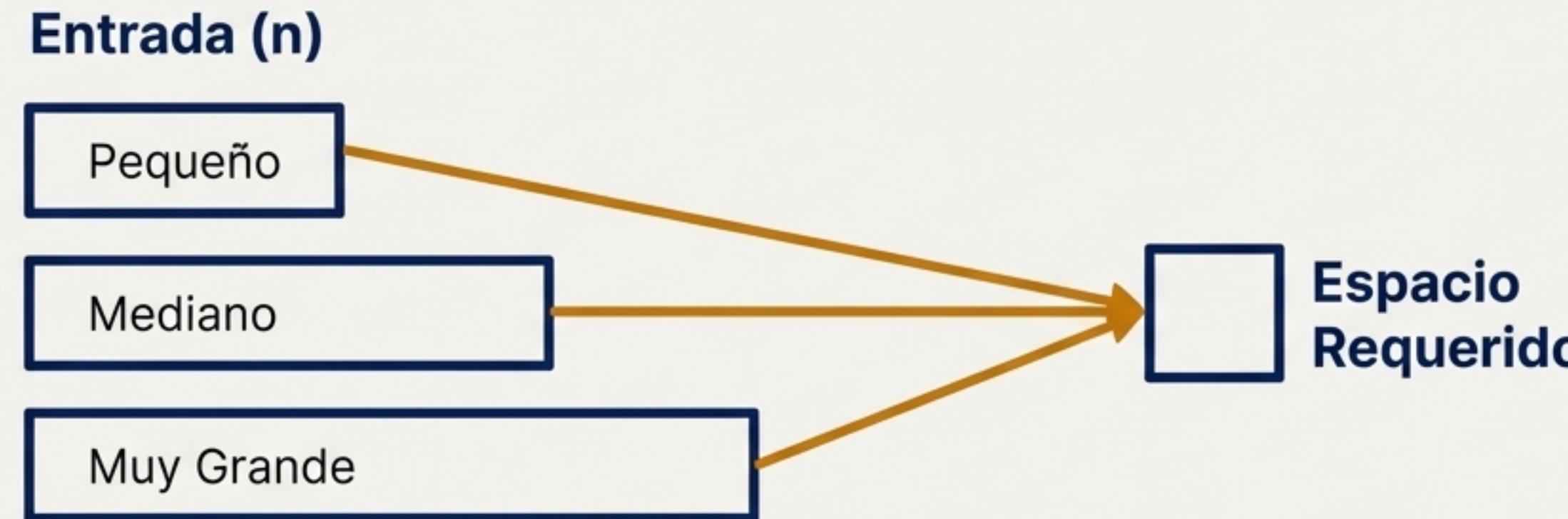
La Mayoría Mide el Tiempo. La Clave Está en el Espacio. La Clave Está en el Espacio.



Cuando los desarrolladores hablan de 'Big O', generalmente se refieren a la **complejidad computacional**: cuánto tiempo tarda en ejecutarse un algoritmo.

Pero para resolver nuestro dilema, necesitamos dominar el otro lado de la balanza: la **complejidad espacial**, que mide cuánto espacio (RAM, disco) necesita un algoritmo para completarse.

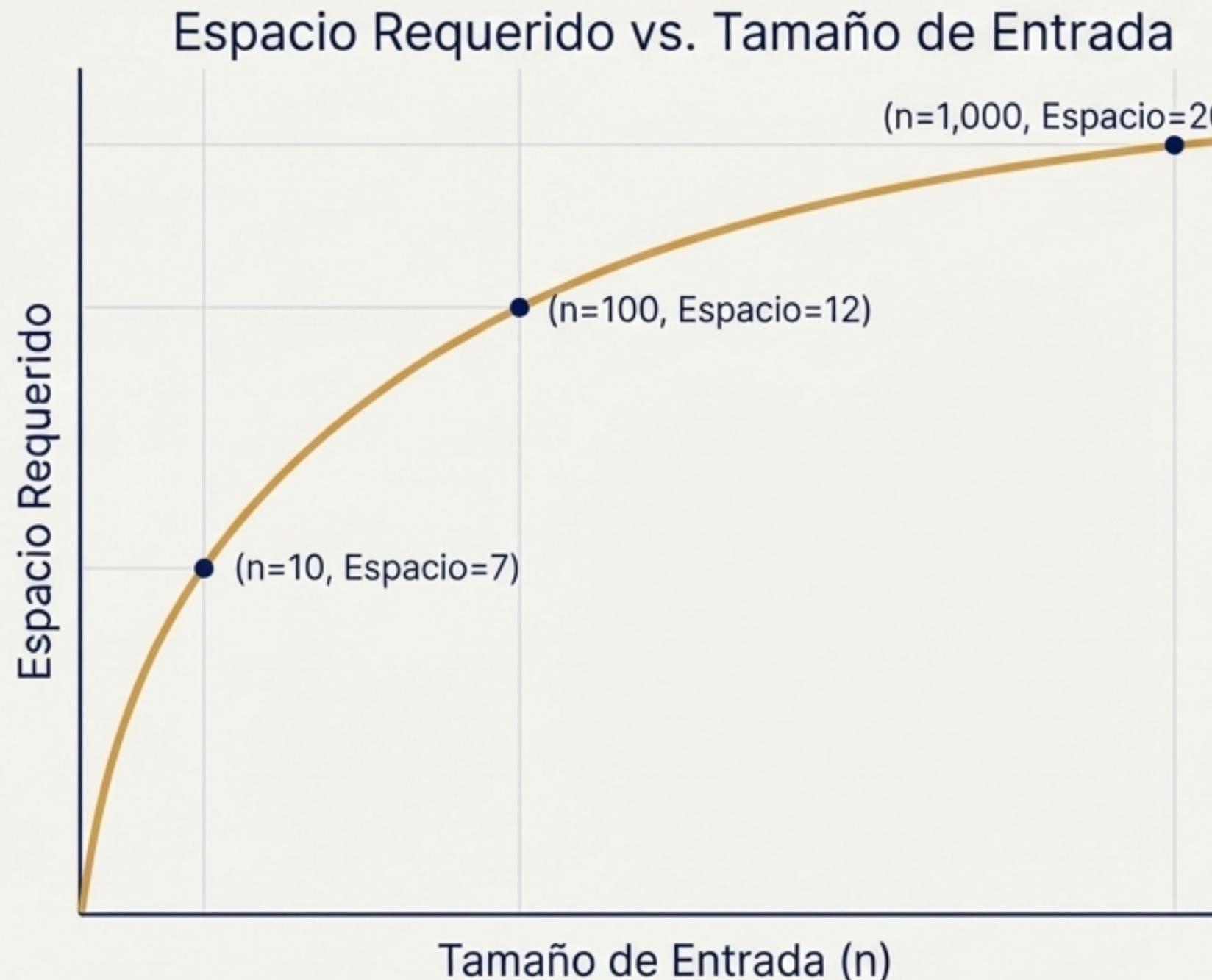
Complejidad Constante: O(1)



El algoritmo no crea estructuras de datos adicionales, o crea un número fijo de ellas sin import el tamaño de la entrada. Solo utiliza el espacio que se le asignó al principio.

Sin importar el tamaño de la entrada, nuestro espacio es fijo.

Complejidad Logarítmica: $O(\log n)$



El espacio adicional que necesita el algoritmo crece a un ritmo cada vez menor a medida que la entrada aumenta.

Ejemplo Concreto

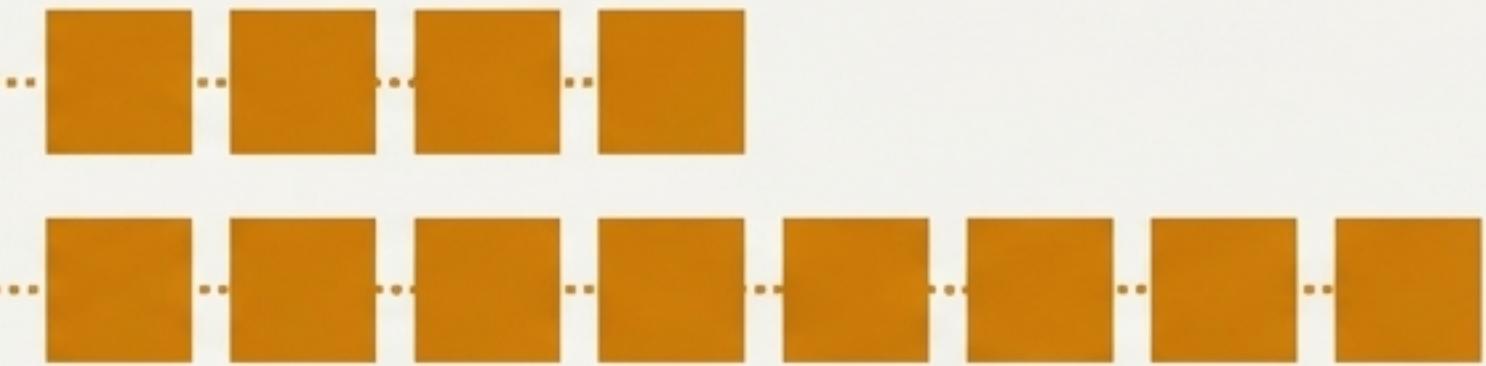
- Para un array de **10** elementos, crea **7** arrays adicionales.
- Para un array de **100** elementos, crea **12** arrays adicionales.
- Para un array de **1,000** elementos, crea **20** arrays adicionales.

Complejidad Lineal: $O(n)$

Entrada (n)



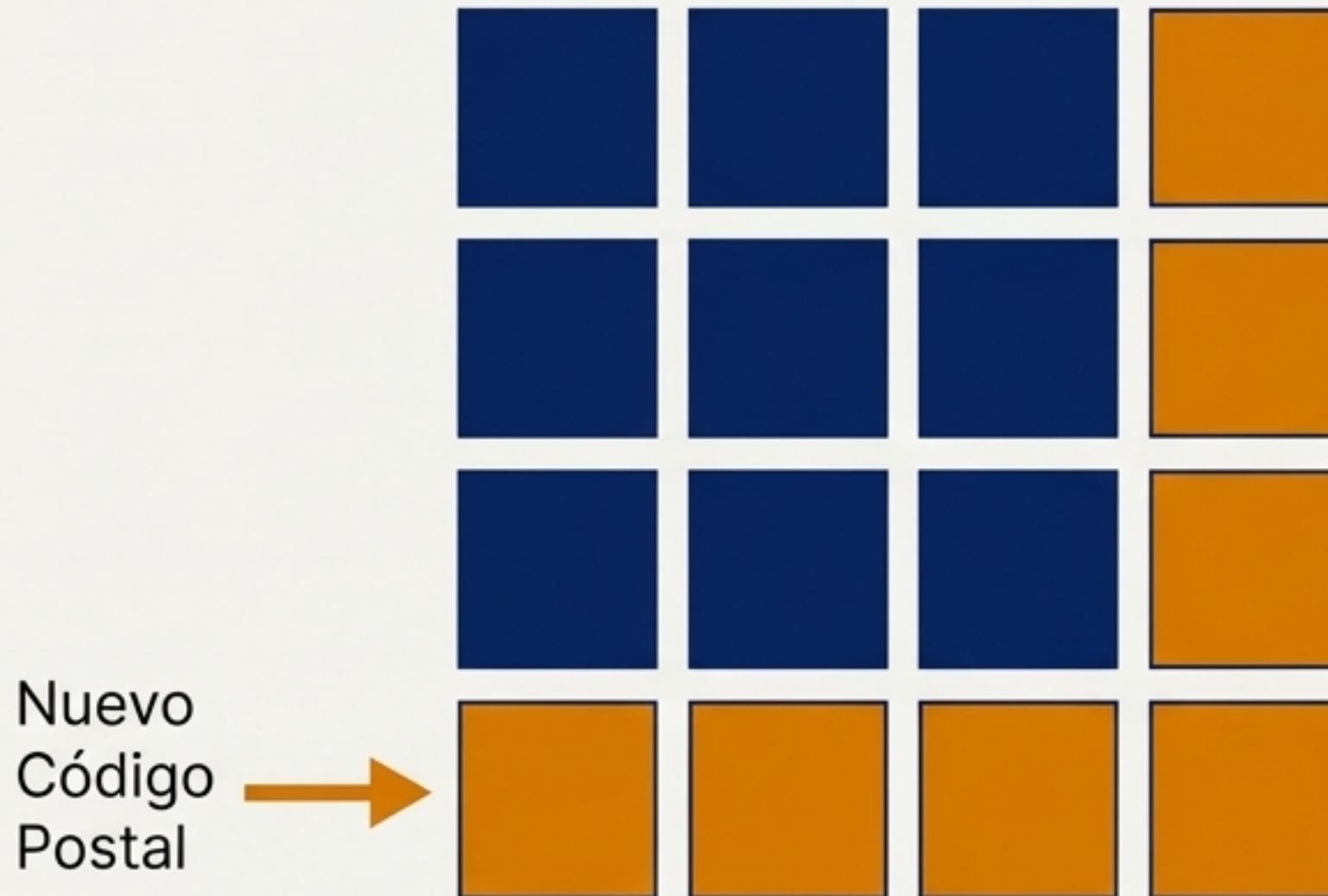
Espacio Adicional



Por cada elemento en los datos de entrada, el algoritmo necesita crear un espacio de memoria adicional. La necesidad de espacio crece en proporción directa al tamaño de la entrada.

A más datos de entrada, más espacio necesitamos.

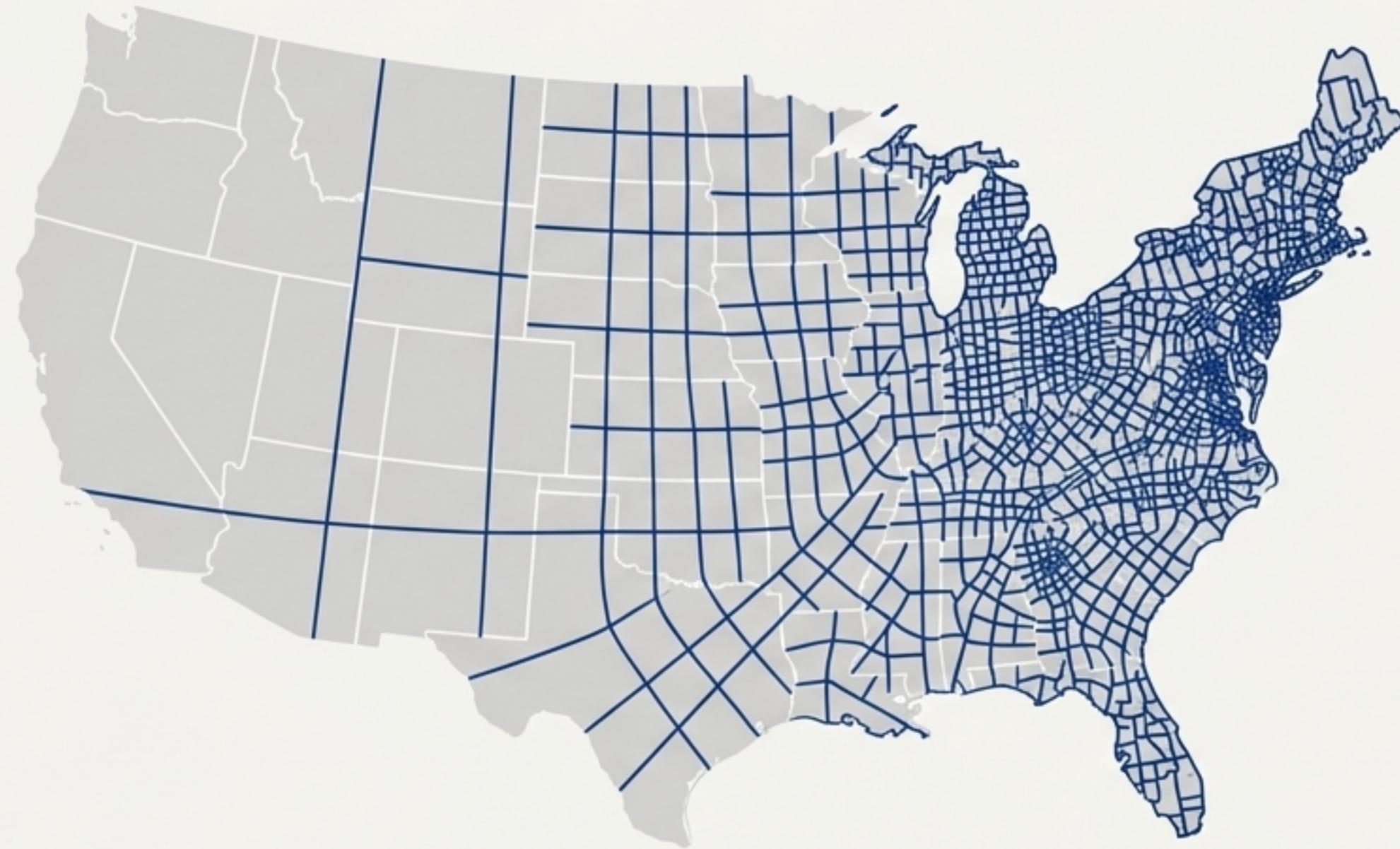
Complejidad Cuadrática: $O(n^2)$



Imagina una app que calcula la distancia entre códigos postales. Por cada nuevo código postal que añadimos a nuestro sistema, tenemos que calcular y almacenar la distancia entre este y todos los demás códigos postales existentes.

Hay casi 42,000 códigos postales en EE. UU.
Esto significa que por cada nuevo código, debemos añadir casi 42,000 nuevos registros de distancia.

El Caso de Estudio: La Matriz de Códigos Postales



crecimiento exponencial

Con una complejidad espacial de $O(n^2)$, el tamaño de nuestra base de datos explotaría. Por cada código postal nuevo, añadimos una cantidad de datos igual al total de códigos ya existentes.

¿Es esta una buena idea?

La Respuesta: Depende del Costo



Costo Alto (Única Vez)

Pagar por almacenar una base de datos masiva.



Costo Recurrente Muy Alto

Pagar por cada llamada a una API externa para obtener la distancia.

Una empresa hizo exactamente esto. La base de datos era enorme, pero terminó siendo mucho más barata que las llamadas a la API.

Aun así, $O(n^2)$ en complejidad espacial es raro y una gran señal de alerta ('big red flag').

Nuestro Dilema, Resuelto

Armados con nuestro nuevo vocabulario de complejidad espacial, ahora podemos tomar una decisión informada para cada escenario.

crecimiento
exponencial



La Solución para la PlayStation 3: Priorizar el Espacio



Lógica de Decisión

La PS3 tiene poca memoria disponible. Debemos sacrificar velocidad para conservar espacio. El procesador puede hacer más trabajo para compensar.

Elección del Algoritmo

Elegimos algoritmos con complejidad espacial **Constante ($O(1)$)** o **Logarítmica ($O(\log n)$)**. Aceptamos que la ordenación pueda tardar un poco más, porque quedarnos sin memoria no es una opción.

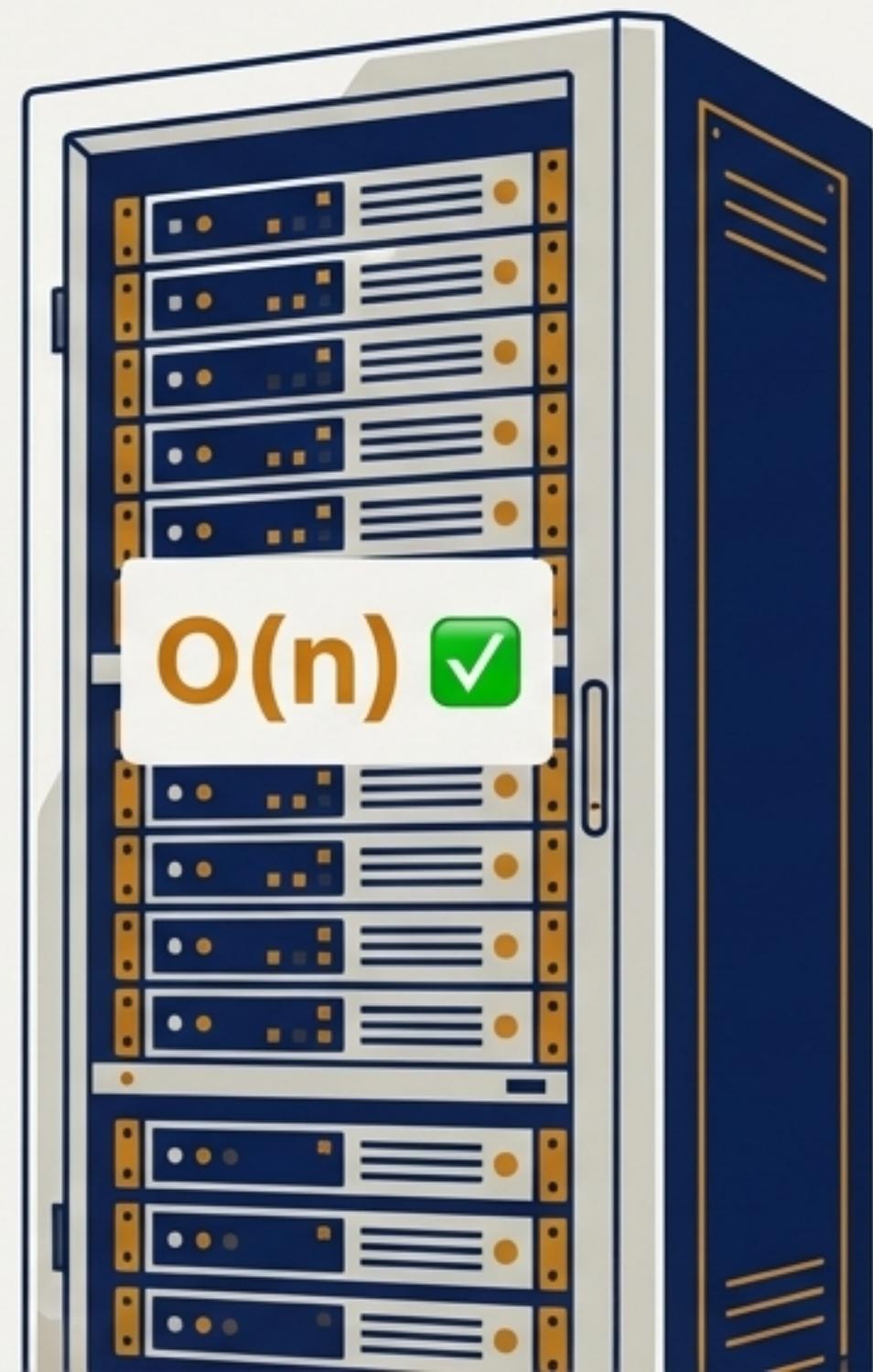
La Solución para el Servidor en la Nube: Priorizar la Velocidad

Lógica de Decisión

En un servidor potente no estamos limitados ni por memoria ni por cómputo. La prioridad es responder al usuario lo más rápido posible.

Elección del Algoritmo

Podemos permitirnos algoritmos con complejidad espacial **Lineal ($O(n)$)** o superior si nos ofrecen una mejora significativa en la velocidad de respuesta. Usar más memoria es un intercambio aceptable.



**La Complejidad Algorítmica
no es un juicio de valor.**

Es una herramienta de diseño.