Guion de Video: Notación O Grande

Parte 1: Introducción

Saludos

// Pendiente

Parte 2: O(1) - Constante

O(1), o complejidad constante, indica que el tiempo de ejecución de un algoritmo es fijo, sin importar el tamaño de la entrada. Por ejemplo, acceder a un elemento en un array por su índice toma el mismo tiempo sin importar cuán grande sea el array.

Parte 3: O(n) - Lineal

O(n), o complejidad lineal, significa que el tiempo de ejecución crece proporcionalmente con el tamaño de la entrada. Un ejemplo es recorrer un array para encontrar un elemento específico; el tiempo aumenta linealmente con el número de elementos en el array.

Parte 4: O(log n) - Logarítmica

O(log n), o complejidad logarítmica, significa que el tiempo de ejecución crece de manera logarítmica con el tamaño de la entrada. Un ejemplo es la búsqueda binaria en un array ordenado, donde el tiempo de búsqueda aumenta lentamente a medida que aumenta el tamaño del array.

Parte 5: O(n^2) - Cuadrática

O(n^2), o complejidad cuadrática, indica que el tiempo de ejecución crece proporcionalmente al cuadrado del tamaño de la entrada. Un ejemplo es el algoritmo de ordenamiento por burbuja, que requiere un número de operaciones que aumenta cuadráticamente con el número de elementos.

Parte 6: O(n!) - Factorial

O(n!) indica que el tiempo de ejecución de un algoritmo crece factorialmente con respecto al tamaño de la entrada. Esto significa que si el tamaño de la entrada (n) aumenta, el tiempo de ejecución aumenta extremadamente rápido, siguiendo la fórmula factorial: n! = n × (n-1) × (n-2) × ...

Parte 7: Cierre

// Pendiente