# Semana 2 Control de Lectura

# Capítulo 3: Notación Asintótica

## Objetivo del Capítulo

Comprender y aplicar la notación asintótica para analizar la eficiencia de algoritmos en términos de tiempo y espacio, ignorando constantes y términos de menor orden.

## Secciones Clave

### 1. Introducción a la Notación Asintótica

Propósito: Comparar el crecimiento de funciones que representan el costo de algoritmos.

Enfoque: Comportamiento de algoritmos cuando el tamaño de entrada (n) tiende a infinito.

### 2. Definiciones Fundamentales

O-grande (O) – Límite Superior

Definición: f(n) ∈ O(g(n)) si existen constantes c > 0 y n₀ ≥ 0 tales que:

f(n) ≤ c · g(n) para todo n ≥ n₀

Interpretación: El algoritmo no es peor que g(n).

Ejemplo: 3n² + 5n ∈ O(n²) con c = 4, n₀ = 2

Omega (Ω) – Límite Inferior

Definición: f(n) ∈ Ω(g(n)) si existen constantes c > 0 y n₀ ≥ 0 tales que:

f(n) ≥ c · g(n) para todo n ≥ n₀

Interpretación: El algoritmo no es mejor que g(n).

Ejemplo: n log n ∈ Ω(n)

Theta (Θ) – Límite Ajustado

Definición: f(n) ∈ Θ(g(n)) si f(n) ∈ O(g(n)) y f(n) ∈ Ω(g(n))

Interpretación: El algoritmo tiene un crecimiento exacto como g(n).

Ejemplo: 2n² + n ∈ Θ(n²)

### 3. Otras Notaciones

o-pequeña (o): Límite superior estricto (ejemplo: n ∈ o(n log n))

ω-pequeña: Límite inferior estricto (ejemplo: n² ∈ ω(n))

### 4. Propiedades y Operaciones

Transitividad: Si f(n) ∈ O(g(n)) y g(n) ∈ O(h(n)), entonces f(n) ∈ O(h(n))

Suma y Multiplicación:

O(f(n)) + O(g(n)) = O(max(f(n), g(n)))

O(f(n)) · O(g(n)) = O(f(n) · g(n))

### 5. Aplicaciones en Algoritmos

Búsqueda binaria: O(log n)

Ordenación por inserción: O(n²) en caso peor, Θ(n) en caso mejor

Mergesort: Θ(n log n) en todos los casos

### 6. Análisis de Casos

Caso peor: Máximo tiempo posible (ejemplo: Quicksort → O(n²))

Caso medio: Esperado en entradas aleatorias (ejemplo: Quicksort → Θ(n log n))

Caso mejor: Mínimo tiempo posible (ejemplo: Ordenación por inserción → Θ(n))