Semana 4 control de lectura

**Notación Asintótica**

**3.1 Introducción**

* El objetivo principal del capítulo es estudiar cómo **evaluar la eficiencia de los algoritmos**.
* Se enfoca en **estimar matemáticamente la cantidad de recursos** (como tiempo o memoria) que consume un algoritmo en función del tamaño del problema.
* La notación asintótica permite **simplificaciones útiles** al estudiar el comportamiento del algoritmo a gran escala (para entradas grandes).
* Se evita depender de factores como la implementación específica, el lenguaje de programación o la velocidad del hardware.

**3.2 Una notación para “el orden de”**

* Se introduce una función f:N→R+f: \mathbb{N} \rightarrow \mathbb{R}^+f:N→R+ para representar el consumo de recursos.
* Se usa la notación O(g(n))O(g(n))O(g(n)) para expresar que una función f(n)f(n)f(n) está **acotada superiormente** por una constante multiplicada por otra función g(n)g(n)g(n), cuando nnn es suficientemente grande.
* Esta notación **no refleja casos particulares**, sino el comportamiento general cuando el tamaño del problema crece.
* También se introduce la idea de **funciones dominantes**: cuando una función crece más rápido que otra, domina su comportamiento asintótico.

**Ejemplo y propiedades:**

* Se analiza un ejemplo con f(n)=27n3+111n2+112f(n) = 27n^3 + 111n^2 + 112f(n)=27n3+111n2+112 y se muestra cómo simplificarlo a O(n3)O(n^3)O(n3), descartando términos menores y constantes.
* Se discute la **relación entre funciones**: si f(n)∈O(g(n))f(n) \in O(g(n))f(n)∈O(g(n)) y g(n)∈O(h(n))g(n) \in O(h(n))g(n)∈O(h(n)), entonces f(n)∈O(h(n))f(n) \in O(h(n))f(n)∈O(h(n)).
* También se menciona cómo combinar funciones en notaciones OOO y cómo manejar sumas y multiplicaciones.

**Importancia práctica:**

* Permite **comparar algoritmos** independientemente de la plataforma o lenguaje.
* Ofrece una forma **estándar y objetiva** de describir eficiencia.
* Aunque no proporciona valores exactos de tiempo de ejecución, **ayuda a predecir el comportamiento a gran escala**.

### Notaciones estándar y funciones comunes

Se describen funciones que aparecen frecuentemente en el análisis de algoritmos:

* Constantes: 1, 10, etc.
* Logarítmicas: log n, log₂ n
* Polinómicas: n, n², n³
* Exponenciales: 2ⁿ, eⁿ
* Factoriales: n!

También se presentan dos notaciones **menos comunes pero útiles**:

* **o(g(n))** – crecimiento estrictamente menor que g(n).
* **ω(g(n))** – crecimiento estrictamente mayor que g(n).