Capítulo 1: Diseñar algoritmos eficientes aplicando estructuras de control y recursividad

1. Fundamentos y métricas de eficiencia:

Definición de eficiencia algoritmica:
La eficiencia mide recursos usados por un algoritmo, principalmente tiempo y memoria. Se expresa mediante notación asintótica (O, Ω, Θ). Sirve para comparar soluciones independientemente del hardware, como un índice rápido que predice comportamiento ante entradas crecientes.

Notación O: propósito y límites:
La notación O describe el peor caso temporal. No da constantes ni coste exacto, pero permite ordenar alternativas. Es útil para evitar soluciones que escalan mal cuando la entrada crece de 1.000 a 100.000 elementos.

Complejidad espacial:
Indica memoria adicional requerida además de la entrada. Diferencia entre algoritmos in situ (O(1)) y los que crean estructuras auxiliares. Es clave para sistemas embebidos y cargas simultáneas en servidores web.

Medir y comparar algoritmos (práctica):
Tú debes ejecutar pruebas con tamaños crecientes y medir tiempos medios y máximos. Asegúrate de repetir 5–10 veces para mitigar ruido. Usa entradas aleatorias y casos límite para evaluar comportamiento real.

Errores comunes al estimar costes:
Confundir tiempo de ejecución con número de operaciones simples, ignorar constantes significativas en problemas pequeños y no considerar entrada típica. Estos fallos conducen a elecciones subóptimas en proyectos reales.

Ejemplo 1:
Imagina que comparas dos funciones de búsqueda: una lineal en una lista de 10.000 y otra binaria en una lista ordenada; la binaria reduce operaciones de ~10.000 a ~14, notable en tiempo y consumo de CPU.

Estructura	Costo temporal típico	Uso típico
if / else	O(1)	Decisiones simples por elemento
for (iterativo)	O(n)	Recorrer colecciones
while / do-while	O(n) (depende de la condición)	Condiciones de parada dinámicas
Recursión simple	O(n) a O(2^n)	Problemas divididos, árboles