**Semana 6 Control de lectura**

**Recursión y retroceso**

**¿Qué es la recursión?**

La recursión es una técnica de programación en la que una función se llama a sí misma para resolver un problema. Es una forma natural y poderosa de expresar soluciones a problemas que pueden dividirse en subproblemas más pequeños del mismo tipo.

**Características clave:**

* **Caso base (o condición de parada):** toda función recursiva debe tener al menos un caso que no implique una llamada recursiva.
* **Llamada recursiva:** la función debe reducir el problema de forma que se acerque al caso base.

El capítulo enfatiza que una mala definición recursiva puede llevar a bucles infinitos o a un uso ineficiente de recursos.

**Diseño de algoritmos recursivos**

El capítulo propone un enfoque estructurado para construir funciones recursivas:

1. **Identificar el problema general.**
2. **Dividir el problema en subproblemas más pequeños.**
3. **Resolver los subproblemas de forma recursiva.**
4. **Combinar las soluciones para formar la solución completa.**

**Ejemplos:**

* **Factorial de un número**  
  n!=n×(n−1)!n! = n \times (n-1)!n!=n×(n−1)!
* **Serie de Fibonacci**  
  F(n)=F(n−1)+F(n−2)F(n) = F(n-1) + F(n-2)F(n)=F(n−1)+F(n−2)  
  *(aunque este enfoque es ineficiente sin optimización)*
* **Búsqueda binaria recursiva**  
  Divide un arreglo ordenado en mitades y busca en una de ellas según el valor objetivo.

**Complejidad y eficiencia**

Aunque los algoritmos recursivos pueden ser simples y elegantes, a veces no son los más eficientes. Por ejemplo:

* La versión recursiva del Fibonacci tiene **complejidad exponencial** si no se usa **memorización** (técnica de programación dinámica).
* La recursión consume **espacio en la pila de llamadas**, lo que puede ser un problema en estructuras profundas.

El capítulo menciona técnicas para optimizar recursión como:

* **Recursión de cola**
* **Transformación a versión iterativa**

**Retroceso (Backtracking)**

El **retroceso** es una técnica que se utiliza cuando se necesita explorar múltiples configuraciones posibles para encontrar una o todas las soluciones válidas.

**Características del retroceso:**

* Se construye una solución **parcial**, paso a paso.
* Si se detecta que una solución parcial **no puede conducir a una solución válida**, se **retrocede** (backtrack) y se intenta otra opción.
* Es fundamental en problemas donde hay que explorar un **espacio de búsqueda grande**, pero se pueden **podar ramas** que no cumplen con las condiciones.