RESUMEN – RECURSIVIDAD Y BACKTRACKING

1. Recursividad

La **recursividad** es una técnica de programación en la que una función se llama a sí misma para resolver un problema.
La idea fundamental es dividir un problema grande en versiones más pequeñas del mismo problema hasta llegar a un **caso base**, que detiene la recursión y devuelve un resultado concreto.

Se utiliza en:

- Procesamiento de estructuras jerárquicas (árboles, grafos).
- Problemas matemáticos (factorial, Fibonacci, potencias).
- Problemas de exploración y búsqueda.

Características principales:

- **Caso base:** condición que detiene la recursión.
- **Caso recursivo:** llamada a la función con una versión reducida del problema.
- **Pila de llamadas:** cada llamada recursiva se almacena temporalmente hasta completarse.

---

2. Backtracking

El **backtracking** es una técnica de búsqueda sistemática que explora posibles soluciones y **retrocede** cuando una opción no lleva a un resultado válido.
Es una extensión natural de la recursividad.

Se usa para problemas donde se deben explorar múltiples caminos o combinaciones, como:

- Laberintos
- Sudoku
- N-reinas
- Permutaciones y combinaciones
- Problema de la mochila (Knapsack)

Cómo funciona:

1. Avanza tomando una decisión parcial.
2. Si la decisión no es válida, **retrocede** (backtrack).
3. Prueba una alternativa diferente.
4. Se repite el proceso hasta encontrar una solución o agotar todas las opciones.

---

3. Ejemplo clásico: Resolver un laberinto

El algoritmo intenta moverse en diferentes direcciones desde una posición inicial.
Si encuentra un obstáculo, retrocede y prueba otra ruta.

Proceso:

- Si llega a la salida → éxito.
- Si llega a un punto inválido → retrocede.
- Marca los caminos visitados para no repetirlos.

Este comportamiento refleja perfectamente la naturaleza del backtracking.

---

4. Selección óptima (Backtracking para optimización)

El backtracking puede ser usado para encontrar la **mejor solución posible** evaluando todas las combinaciones válidas.

Ejemplo:
Problema de la mochila (Knapsack)
El algoritmo evalúa incluir o excluir cada objeto y calcula cuál combinación maximiza el valor total sin exceder el peso permitido.

---

5. Ventajas y desventajas

Ventajas:

- Permite explorar exhaustivamente todas las soluciones posibles.
- El código suele ser más claro usando recursividad.
- Ideal para problemas combinatorios.

Desventajas:

- Puede consumir mucha memoria debido a la pila de llamadas.
- Puede ser lento en problemas con muchas combinaciones.
- No siempre es la técnica más eficiente sin optimización.

Conclusión

La **recursividad** permite descomponer un problema en partes más simples y manejables.
El **backtracking**, por su parte, explora rutas posibles y retrocede cuando una de ellas no funciona.

Ambas técnicas son fundamentales en el diseño de algoritmos que requieren exploración, decisión y optimización. Su uso adecuado permite resolver problemas complejos de forma clara, estructurada y flexible.