Reporte: Torres de Hanoi - Recursividad vs Iteración

Introducción

Este reporte analiza y compara dos implementaciones del problema de las Torres de Hanoi: una recursiva y una iterativa. El objetivo es comprender las diferencias en complejidad, legibilidad y rendimiento.

Resultados del Benchmark

Se realizaron pruebas con n = 10, 15 y 20 discos. Los resultados obtenidos son:

n (Discos)	Movimientos	Tiempo Recursivo (ms)	Tiempo Iterativo (ms)
10	1,023	0.333	1.371
15	32,767	14.468	52.096
20	1,048,575	418.994	1059.279

Observaciones:

- La versión recursiva resultó ser más rápida en estas pruebas. Esto puede deberse a la sobrecarga de gestión de estructuras de datos (Diccionario, HashSet) en la implementación iterativa proporcionada, comparado con la gestión optimizada de la pila de llamadas del sistema.
- Ambas versiones muestran un crecimiento exponencial en el tiempo de ejecución, consistente con la complejidad O(2^n).

Reflexión

1. ¿Por qué crees que la recursión se siente "más natural" para este problema?

La recursión es natural porque la definición del problema es inductiva: para mover una torre de tamaño `n`, primero debes mover una torre de tamaño `n-1`. El algoritmo recursivo traduce literalmente esta definición: "Mueve n-1 a auxiliar, mueve disco grande a destino, mueve n-1 a destino".

2. ¿Qué otros problemas conoces que sean naturalmente recursivos?

- **Recorridos de árboles y grafos:** DFS (Depth First Search), recorridos in-order, pre-order, post-order.
- **Algoritmos de ordenamiento:** Merge Sort, Quick Sort.
- **Cálculos matemáticos:** Factorial, Fibonacci (aunque ineficiente sin memoización), Potenciación.
- **Fractales:** Triángulo de Sierpinski, Curva de Koch.

3. Si fueras a implementar esto en un sistema real, ¿qué versión elegirías y por qué?

Dependería de las restricciones:

- **Recursiva:** Si `n` es pequeño (ej. < 30) y la legibilidad/mantenimiento es prioridad. Es menos propensa a errores de implementación.

- **Iterativa:** Si existe riesgo de desbordamiento de pila (Stack Overflow) y `n` puede ser grande, o si el entorno tiene una pila muy limitada (sistemas embebidos). Sin embargo, para Hanoi, `n=64` es computacionalmente imposible de terminar, por lo que el Stack Overflow solo es riesgo si la pila es extremadamente pequeña.

4. ¿Cómo podrías optimizar la versión recursiva para manejar n más grandes?

- **Optimización de llamada de cola (Tail Call Optimization):** Aunque C# no siempre lo garantiza, reestructurar el código para que la última operación sea la llamada recursiva puede ayudar.
- **Híbrido:** Usar recursión hasta cierto nivel y luego iteración.
- **Evitar asignaciones:** En la implementación actual, se crean strings en cada paso. Escribir directamente a un buffer o contar movimientos reduciría la presión sobre el GC y mejoraría el rendimiento.

Conclusión

Aunque la versión iterativa evita el uso de la pila de llamadas, su implementación es significativamente más compleja y, en este caso particular, más lenta debido a las estructuras auxiliares. La recursión ofrece una solución elegante y eficiente para este problema específico, siempre que `n` se mantenga dentro de límites razonables.