Cuestionario de Estudio: Complejidad Computacional (Nivel Básico)

Parte 1: Conceptos básicos

1. ¿Qué significa 'complejidad computacional'?
2. ¿Por qué es importante analizar la eficiencia de un programa además de que funcione correctamente?
3. Explica con tus palabras la diferencia entre tiempo de ejecución y uso de memoria.
4. ¿Qué es un 'algoritmo eficiente'? Da un ejemplo.
5. ¿Qué diferencia hay entre medir el tiempo real de un programa y analizar su complejidad?

Parte 2: Complejidad en el tiempo

1. ¿Qué significa decir que un algoritmo es de tiempo constante (O(1))? Da un ejemplo sencillo.
2. ¿Qué significa que un algoritmo tenga tiempo lineal (O(n))?
3. ¿Por qué un programa que revisa uno por uno los elementos de una lista suele considerarse O(n)?
4. Si un algoritmo compara todos los pares posibles de elementos en una lista de tamaño n, ¿qué complejidad aproximada tendría?
5. ¿Cuál crees que es más rápido: un algoritmo O(n) o uno $O(n^2)$, si la entrada tiene un tamaño muy grande?

Parte 3: Ejercicios prácticos

1. Supongamos que tienes una lista con 10 números. ¿Cuántas operaciones hace un algoritmo que: a) Solo imprime el primer número. b) Recorre toda la lista y los imprime uno por uno. c) Compara cada número con todos los demás.
2. ¿Qué complejidad tendría este pseudocódigo?: for i en lista: print(i)
3. ¿Y este otro?: for i en lista: for j en lista: print(i, j)
4. Si un algoritmo tarda 1 segundo en procesar 1,000 datos, ¿aproximadamente cuánto tardaría en procesar 2,000 datos si es O(n)? ¿Y si fuera $O(n^2)$?

Parte 4: Preguntas para reflexionar

1. ¿Por qué no siempre el algoritmo más sencillo es el más eficiente?
2. ¿Crees que siempre es necesario usar el algoritmo más rápido posible? ¿Por qué sí o por qué no?
3. ¿Qué pasaría si ignoramos la complejidad computacional y solo pensamos en que el programa 'funcione'?