Ejercicio: análisis del tiempo de ejecución y uso de memoria

Objetivos del ejercicio

- Evaluar el impacto en el tiempo de ejecución y el uso de memoria de diversos algoritmos.
- Comparar la eficiencia de algoritmos con diferentes complejidades.
- Analizar experimentalmente algoritmos de búsqueda y ordenación utilizando código en Python.

Planteamiento del ejercicio

Se les proporciona una serie de algoritmos implementados en Python (por ejemplo, búsqueda secuencial, búsqueda binaria, MergeSort y BubbleSort). Su tarea es:

1. Ejecutar cada algoritmo con entradas de diferentes tamaños.
2. Medir el tiempo de ejecución usando el módulo `time` o `timeit` de Python.
3. Medir el uso de memoria utilizando herramientas como `memory_profiler` o `tracemalloc`.
4. Comparar los resultados obtenidos y discutir las diferencias en relación a la complejidad teórica de cada algoritmo.

Ejemplo de medición de tiempo en Python

A continuación se muestra un ejemplo de cómo medir el tiempo de ejecución de una función (en este caso, una búsqueda secuencial):

```
import time

def linear_search(arr, x):
for i, item in enumerate(arr):
if item == x:
return i
return -1

# Generar una lista grande
arr = list(range(1000000))
x = 999999

start_time = time.time()
result = linear_search(arr, x)
end_time = time.time()

print("Resultado:", result)
print("Tiempo de ejecucion:", end_time - start_time, "
```

Ejemplo de medición de memoria en Python

También pueden medir el uso de memoria. Por ejemplo, usando `memory_profiler`:

```
        from memory_profiler import memory_usage

        def bubble_sort(arr):
        n = len(arr)
        for i in range(n):
        for j in range(0, n - i - 1):
        if arr[j] > arr[j+1]:
        arr[j], arr[j+1] = arr[j+1], arr[j]
        return arr

        # Medir el uso de memoria
        mem_usage = memory_usage((bubble_sort, ([64, 34, 25, 12, 22,
    11, 90],)))
        print("Uso de memoria:", mem_usage, "MB")
```

Preguntas para discusión

- ¿Cómo varían el tiempo de ejecución y el uso de memoria a medida que aumenta el tamaño de la entrada?
- ¿Qué algoritmo resulta más eficiente en tiempo y cuál consume menos memoria? ¿Por qué?
- ¿Se corresponden los resultados experimentales con la complejidad teórica de cada algoritmo?

Conclusión del ejercicio

- Analicen sus resultados y preparen una breve presentación en la que expliquen las diferencias observadas.
- Discutan posibles mejoras o alternativas para optimizar el rendimiento de los algoritmos.