Luis Angel Zempoalteca Serrano

ReflexAct1.3: Investigación y Reflexión sobre Algoritmos de Ordenamiento y Búsqueda

En el análisis de problemas de gestión de datos y procesamiento de información, la selección adecuada de algoritmos de ordenamiento y búsqueda es crucial para garantizar la eficiencia y la efectividad del sistema.

Algoritmos de Ordenamiento

- QuickSort: es un algoritmo de ordenamiento basado en la técnica de divide y vencerás. Selecciona un elemento como pivote y particiona el arreglo en dos sub-arreglos, uno con elementos menores que el pivote y otro con elementos mayores. Luego, aplica el mismo proceso a los sub-arreglos.

  Ventajas: Eficiente en promedio con una complejidad temporal de O(n log n). Requiere menos espacio adicional comparado con otros algoritmos como MergeSort.
  Desventajas: En el peor de los casos (cuando el pivote es el elemento más grande o más pequeño), la complejidad puede llegar a $O(n^2)$.No es estable.

- MergeSort: este también utiliza la técnica de divide y vencerás. Divide el arreglo en mitades hasta que cada sub-arreglo tenga un solo elemento y luego fusiona los sub-arreglos ordenadamente.
  Ventajas: Tiene una complejidad temporal garantizada de O(n log n) en el peor caso.
  Es estable, lo que significa que preserva el orden relativo de elementos iguales.
  Desventajas: Requiere espacio adicional O(n) para las fusiones, lo que puede ser significativo para arreglos grandes.

Algoritmos de Búsqueda

- Búsqueda Secuencial: examina cada elemento del arreglo uno por uno hasta encontrar el objetivo.
  Ventajas: Simple de implementar, no requiere que el arreglo esté ordenado.
  Desventajas: Ineficiente para arreglos grandes con una complejidad de O(n).
- Búsqueda Binaria: divide repetidamente el arreglo ordenado en mitades, comparando el objetivo con el elemento central y ajustando el rango de búsqueda según corresponda.
  Ventajas: Muy eficiente con una complejidad de O(log n) en arreglos ordenados.
  Desventajas:Requiere que el arreglo esté ordenado.

Luis Angel Zempoalteca Serrano

Reflexión

La selección del algoritmo de ordenamiento y búsqueda es fundamental para garantizar la eficiencia y efectividad en el procesamiento de grandes volúmenes de datos y depende en gran medida de las características del problema en cuestión, como el tamaño del conjunto de datos y los requisitos de eficiencia. La implementación y evaluación de estos algoritmos en situaciones prácticas, como el procesamiento de registros de eventos, muestran la importancia de elegir el algoritmo adecuado para mejorar el rendimiento y la eficacia del sistema.

En el contexto de nuestro problema, en el que trabajamos con registros de eventos, fue crucial elegir un enfoque que ofreciera un rendimiento óptimo tanto en tiempo como en recursos. Optamos por utilizar QuickSort debido a su notable eficiencia en la mayoría de los casos prácticos. Este algoritmo destaca por su capacidad para manejar grandes cantidades de datos de manera rápida y efectiva, siendo especialmente adecuado para archivos como el que procesamos, donde el volumen de información podía crecer considerablemente. A diferencia de otros algoritmos de ordenamiento, QuickSort no solo se comporta bien en promedio, sino que también aprovecha eficientemente la memoria al no requerir espacio adicional significativo.

Referencias

BlackeyeB. (2023, May 1). Algoritmos de ordenación explicados con ejemplos en JavaScript, Python, Java y C++. freeCodeCamp.org. https://www.freecodecamp.org/espanol/news/algoritmos-de-ordenacion-explicados-con-ejemplos-en-javascript-python-java-y-c/

iNGENET Bitácora | *La importancia de los algoritmos (Parte 1)*. (n.d.). https://bitacora.ingenet.com.mx/2012/12/ctinla-importancia-de-los-algoritmos/