Tecnológico de Monterrey

Act 1.3 - Actividad Integral de Conceptos Básicos y Algoritmos Fundamentales (Evidencia Competencia)

Héctor Calderón Reyes | A01350637

Noreth Sofia Villalpando Saldaña | A01368579

Profesor:

Dr. Eduardo Arturo Rodríguez Tello

Fecha de entrega:

8 de Julio del 2023

Programación de estructuras de datos y algoritmos fundamentales (Gpo 570)

Reflexión Actividad 1.3 Hector

En informática, los algoritmos de ordenamiento y búsqueda son cruciales. Estos algoritmos son esenciales para el procesamiento de datos y la resolución de problemas en una variedad de aplicaciones, incluida la búsqueda de información en la web, la organización de bases de datos y la realización de análisis científicos.

En el caso de la evidencia, se utilizó una bitácora de eventos de seguridad para ilustrar cómo los algoritmos de ordenamiento y búsqueda pueden ser aplicados en un contexto real. Se implementaron dos algoritmos de ordenamiento, Swap Sort y Merge Sort, y se comparó su eficiencia.

El algoritmo Swap Sort, es un algoritmo de ordenamiento simple que funciona intercambiando repetidamente elementos adyacentes si están en el orden incorrecto. Aunque es fácil de entender e implementar, Swap Sort es ineficiente para listas grandes, con una complejidad temporal de O(n^2) en el peor de los casos.

Por otro lado, Merge Sort es un algoritmo de ordenamiento más sofisticado que divide la lista en mitades, las ordena por separado y luego las combina. Merge Sort es mucho más eficiente que Swap Sort, con una complejidad temporal de O(n log n) en el peor de los casos.

En la actividad, Merge Sort realizó significativamente menos comparaciones que Swap Sort, lo que indica que es el algoritmo de ordenamiento más eficiente de los dos. Esto es consistente con el análisis de la complejidad temporal de estos algoritmos.

Además de los algoritmos de ordenamiento, también se utilizó un algoritmo de búsqueda binaria para encontrar registros específicos en la bitácora. La búsqueda binaria es un algoritmo eficiente que funciona dividiendo repetidamente la lista en mitades hasta que encuentra el elemento buscado. Sin embargo, la búsqueda binaria requiere que la lista esté ordenada, lo que subraya la importancia de los algoritmos de ordenamiento.

En conclusión, los algoritmos de ordenamiento y búsqueda son herramientas esenciales en la informática que permiten procesar y analizar datos de manera eficiente. La elección del algoritmo adecuado depende de las características específicas del problema, como el tamaño de los datos y si están ordenados o no. En el caso de la actividad, Merge Sort y la búsqueda binaria demostraron ser soluciones eficientes para el análisis de la bitácora.

Referencias:

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to algorithms. MIT press.
Knuth, D. E. (1997). The art of computer programming: sorting and searching (Vol. 3). Pearson Education.

Reflexión Actividad 1.3 Noreth

La importancia y eficiencia de los algoritmos de ordenamiento y búsqueda en la situación problema recae en su capacidad para optimizar el procesamiento y la búsqueda de manera rápida y eficiente. Al utilizar algoritmos de ordenamiento eficientes podemos organizar la bitácora de manera rápida y reducir el tiempo de ejecución. Asimismo, la búsqueda binaria permite una búsqueda eficiente por fechas en la bitácora ordenada, mejorando la respuesta y la experiencia del usuario. Estos algoritmos contribuyen a una mayor eficiencia en la manipulación y búsqueda de eventos, lo que es crucial en entornos de seguridad donde el tiempo y la precisión son fundamentales.

Para los algoritmos de búsqueda, se investigó sobre la búsqueda binaria, la búsqueda secuencial y la búsqueda secuencial en vectores ordenados. Se encontró que el algoritmo de binary search es particularmente eficiente para buscar en una lista ordenada, gracias a su complejidad de tiempo O (log n). Por esta misma razón lo utilizamos para nuestra actividad.

Para la implementación de nuestro proyecto, decidimos que lo mejor era usar dos algoritmos de ordenamiento, uno que fuera de los más eficientes y otro que no, para así comprar su eficiencia. Uno de los algoritmos que utilizamos es el algoritmo de ordenamiento Swap Sort, que implica comparar elementos adyacentes e intercambiarlos si están en el orden incorrecto. Observamos que Swap Sort es relativamente simple de entender e implementar. Sin embargo, también notamos que su complejidad de tiempo es O(n^2), lo que indica que su rendimiento podría deteriorarse significativamente para conjuntos de datos más grandes. Esto sugiere que Swap Sort puede no ser la opción más eficiente para clasificar una cantidad considerable de datos, así como nosotros usamos fechas, ips y mensajes de fallo.

A su vez, implementamos Merge Sort, un algoritmo que divide recursivamente la lista en sublistas más pequeñas, las clasifica individualmente y luego las vuelve a fusionar para obtener una lista ordenada. La complejidad temporal de Merge Sort es O(n log n), lo que sugiere que debería funcionar mejor que Swap Sort para conjuntos de datos más grandes. Esto se alinea con nuestra expectativa de que Merge Sort sea más eficiente en términos de tiempo de ejecución.

Según el análisis de la complejidad temporal y al contrastar el numero de comparaciones que lleva a cabo cada uno de los algoritmos, es razonable concluir que Merge Sort es la mejor opción para clasificar fechas en esta situación problemática. Su complejidad de tiempo O(n log n) indica una mejor escalabilidad y eficiencia, particularmente para conjuntos de datos más grandes. Al aprovechar los algoritmos apropiados, podemos manejar de manera eficiente el conjunto de datos, reducir los recursos computacionales y mejorar la efectividad general de la solución.

Referencias

Zaveri, M. (2022, May 4). An intro to Algorithms: Searching and Sorting algorithms. *Medium*.

https://codeburst.io/algorithms-i-searching-and-sorting-algorithms-56497dbaef20