Marlon Yahir Martínez Chacón – A01424875

Importancia y eficiencia del uso de los diferentes algoritmos de ordenamiento y búsqueda

Para esta actividad se nos proporcionó con un caso en el cual unos servidores intentaron ser hackeados por alguien externo por lo cual se había conseguido mucha información respecto a los ataques, por ejemplo, el mes y el día, además de la hora exacta en que sucedió dicho intento de hackeo, también nos da la información de la dirección ip, el puerto y el motivo de la falla que dio lugar a que se creara ese reporte. Esta información es muy valiosa para los encargados de dicha tecnología para saber si es que se modificó algo debido a la falla que surgió por lo que es importante conocer todo lo que existe dentro del registro generado, pero esto puede llegar a ser complicado debido a que es demasiada información la que se tiene por ejemplo, el archivo .txt que se nos dio constaba de más de 16 mil registros con la información antes mencionada; lo más importante al tener tal cantidad de información es poder realizar búsquedas dentro de los registros por las fechas que se nos proporciona para que sea más sencillo y práctico para solicitar la información.

Pero antes de poder realizar estas búsquedas de información dentro de los registros es necesario procesar toda esa información en nuestro programa y ordenarla para aplicar los algoritmos de búsqueda más eficientes para grandes cantidades de información por lo que es necesario utilizar un algoritmo de ordenamiento, existe muchos algoritmos de ordenamiento, entre menos comparaciones se realicen en la información el algoritmo será más eficiente y tardará menos en darnos la información ordenada; los algoritmos de ordenamiento fueron el selection sort y quick sort, el primero menos eficiente que el segundo, por lo que pude observar durante la ejecución de mi código el selection sort a diferencia del quick sort genera muchas comparaciones en este caso fueron 141,212,415, a diferencia del quick sort que fueron 295,333 comparaciones; es una diferencia abismal el número de comparaciones realizadas entre un algoritmo y otro para ordenar la información por lo que el quick sort es el algoritmo más eficiente para poder ordenar grandes cantidades de información.

Tabla: Complejidad de los algoritmos de ordenamiento.

Algoritmo	Mejor	Promedio	Peor	Estable	Espacio
Swap sort	$O(n^2)$	$O(n^2)$	$O(n^2)$	Sí	$O(1)$
Selection sort	$O(n^2)$	$O(n^2)$	$O(n^2)$	No	$O(1)$
Buble sort	$O(n)$	$O(n^2)$	$O(n^2)$	Sí	$O(1)$
Insertion sort	$O(n)$	$O(n^2)$	$O(n^2)$	Sí	$O(1)$
Merge sort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$	Sí	$O(n)$
Quick sort	$O(n \log n)$	$O(n \log n)$	$O(n^2)$	Sí	$O(\log n)$

Marlon Yahir Martínez Chacón – A01424875

Ya tenía previsto que el claro ganador de los algoritmos de ordenamiento sería el quick sort, debido a su complejidad computacional que es la más baja de todas ya que el tiempo que puede llegar a tomar se calcula mediante un logaritmo por lo que es la mejor función, a diferencia del selection sort que es el tamaño de los datos al cuadrado, que pudimos comprobar mediante el programa dándonos la mitad del cuadrado del tamaño; también quick sort es el mejor en cuanto a espacio debido a que se expresa mediante un logaritmo.

Una vez que ya se tuvieron los datos ordenados, se utilizó un algoritmo de búsqueda, también el más eficiente para que realizará la menor cantidad de comparaciones y se obtuviera el índice de la fecha que se está buscando entre tal cantidad de información lo más rápido posible, para este caso se utilizó el algoritmo de búsqueda binaria que realiza la menor cantidad de comparaciones y por lo tanto, es el más eficiente de los demás algoritmos.

Tabla: Complejidad de los algoritmos de búsqueda.

Algoritmo	Mejor	Promedio	Peor
Búsqueda secuencial	$O(1)$	$O(n)$	$O(n)$
Búsqueda secuencial en arreglos ordenados	$O(1)$	$O(n)$	$O(n/2)$
Búsqueda binaria	$O(1)$	$O(\log_2 n)$	$O(\log n)$

Esto lo podemos comprobar gracias a la complejidad computacional que tienen los algoritmos como se pueden observar el mejor, como había mencionado, es la búsqueda binaria que su tiempo se encuentra expresado con un logaritmo por lo que es el que tiene menor tiempo de entre los 3, y por lo tanto el más eficiente para utilizar para encontrar información dentro de grandes cantidades de datos.

Con esta actividad me di cuenta que es importante la elección de un algoritmo de ordenamiento y de búsqueda, ya que se tiene que escoger el más eficiente de todos los que existen para que los procesos no duren mucho tiempo, ni se ocupe tanto espacio para que tus programas no presenten alguna clase de error en el futuro y no sobrecargue el procesador cuando se necesiten comparar una gran cantidad de datos.

Referencias

- Algoritmos de ordenamiento. (s. f.). http://lwh.free.fr/pages/algo/tri/tri_es.htm
- Platzi: Cursos online profesionales de tecnología. (s. f.-c). https://platzi.com/tutoriales/1832-ordenamiento/9229-algoritmos-de-ordenamiento/
- iNGENET Bitácora | La importancia de los algoritmos (Parte 1). (s. f.). http://bitacora.ingenet.com.mx/2012/12/ctinla-importancia-de-los-algoritmos/

- Búsqueda binaria (artículo) | Algoritmos. (s. f.). Khan Academy. https://es.khanacademy.org/computing/computer-science/algorithms/binary-search/a/binary-search
- Campos, O. (2011b, junio 14). Implementando el algoritmo QuickSort. Genbeta. https://www.genbeta.com/desarrollo/implementando-el-algoritmo-quicksort
- E. (2017, 19 diciembre). Búsqueda binaria - Emmita. Medium. https://medium.com/@Emmitta/b%C3%BAsqueda-binaria-c6187323cd72
- GeeksforGeeks. (2023, 19 marzo). QuickSort. https://www.geeksforgeeks.org/quick-sort/
- GeeksforGeeks. (2023c, marzo 14). Selection Sort Algorithm. https://www.geeksforgeeks.org/selection-sort/