Act 1.3 - Actividad Integral de Conceptos Básicos y Algoritmos Fundamentales

Instituto Tecnológico y de Estudios Superiores de Monterrey

Programación de estructuras de datos y algoritmos fundamentales

12/09/2021

Profesores:

Luis Ricardo Peña Llamas

Elaborado por:

Pablo Agustín Ortega Kral A00344664

Act 1.3 - Actividad Integral de Conceptos Básicos y Algoritmos Fundamentales

1.- Marco Teórico

Un algoritmo es la parte fundamental sobre la cuál se construyen las ciencias computacionales, siendo este un número finito de pasos que se deben seguir para resolver un cierto problema. Concretamente, cuando hablamos de un algoritmo computacional este tiene una entrada de datos, una salida y parámetros que refieren a su efectividad y complejidad en términos de espacios y tiempo. En el presente trabajo se analizan los algoritmos de búsqueda y ordenamiento, y las aplicaciones de los mismos.

En primera instancia, veamos los algoritmos de búsqueda, que tienen la tarea de recorrer estructuras de datos para encontrar un cierto valor. La variación que existe en estos algoritmos refiere a la manera en la que navega por la estructura de datos; de este modo, existen dos clasificaciones generales: búsqueda secuencial, en la que se recorren los datos elemento por elemento hasta encontrar el valor buscado, y búsqueda interválica, que es práctica para listas ordenadas, pues permite dividir la estructura de datos repetidamente a la mitad para buscar en secciones más pequeñas (*GeeksforGeeks,* 2021).

Por otro lado, los algoritmos de ordenamiento, como indica su nombre, tienen la consigna de ordenar los elementos de la estructura de datos de menor a mayor. El mecanismo que utilizan estos argumentos para realizar esto es a través de la comparación de los elementos; lo que diferencia los distintos algoritmos es la manera en que realizan dicha comparación. Algunos de los algoritmos de ordenamiento más utilizados son:

- *Insertion Sort:* algoritmo que construye el *array* ordenado un elemento a la vez, en el que se toma un valor (*key)* y se recorre la estructura de datos comparando los valores con el *key* para asignarles la posición correspondiente. Si bien es fácil de implementar, es ineficiente con listas grandes teniendo una complejidad cuadrática O(n^2). (*Zaveri, M., 2018*)
- *Bubble Sort:* va recorriendo la estructura de datos en pares e intercambiándolos según es necesario; es decir, encontrar el valor más grande e ir llevándolo al final de la agrupación. Al igual que insertion sort, es fácil de implementar pero es altamente ineficiente, presentando una complejidad cuadrática O(n^2). (*Zaveri, M., 2018*)
- *Merge Sort:* algoritmo recursivo que divide el array a ordenar en dos partes, las ordena por separado utilizando el mismo método y al final junta los subarreglos ordenados. Este algoritmo es más sofisticado y como tal requiere más trabajo para

implementarse, pero es eficiente en listas grandes, con una complejidad de $O(nlog(n))$.(*GeeksforGeeks,* 2021)

2.- Reflexión sobre utilidad de algoritmos

Los algoritmos de búsqueda y ordenado son fundamentales en la programación pues permite leer y manipular estructuras de datos, siendo que la información es el recurso clave de cualquier código. Ya sea un ingeniero de software accediendo a una base de datos de usuarios o bien un ingeniero en robótica y sistemas accediendo a un arreglo de información registrada por un sensor, la manipulación de grandes cantidades de datos es esencial para cualquier programa.Ahora bien, la implementación de los algoritmos depende del problema a resolver, siendo que se busca un balance entre una eficiencia temporal proveniente de una baja complejidad y la facilidad de utilizar el algoritmo dentro del código.

En la situación problema planteada actualmente, se empleó estos algoritmos para procesar un registro de errores con 16,807 entradas. Para ello, se creó una clase de *Error* con los atributos fecha y registró y se instanció la información de la bitácora como un objeto *Error*, almacenando los en un vector. Posteriormente, se utilizó el *Merge Sort* para ordenar los registros de falla según el atributo de su fecha; se optó por utilizar dicho algoritmo dado el gran tamaño del vector a ordenar. Finalmente, se adaptó el algoritmo de búsqueda lineal para desplegar los elementos entre un rango de fechas especificados; si bien dicho algoritmo no es tan eficiente, teniendo una complejidad de $O(n)$, se optó por utilizarlo por la facilidad de implementación, considerando que al tener que desplegar los elementos dentro de un determinado rango aún se tenía que realizar una iteración a través de la lista para hacer la impresión a consola.

En conclusión, los algoritmos de búsqueda y ordenamiento son poderosas herramientas para el procesamiento y manipulación de estructuras de datos, usándose frecuentemente de manera conjunta. Su correcta aplicación permite realizar programas que utilizen grandes cantidades de información (ya sea de una base de datos o almacenada localmente)y resulta en códigos más eficientes, organizados y robustos.

3.- Referencias

- Insertion Sort - GeeksforGeeks. (2021). Retrieved 12 September 2021, from https://www.geeksforgeeks.org/insertion-sort/
- Merge Sort - GeeksforGeeks. (2021). Retrieved 12 September 2021, from https://www.geeksforgeeks.org/merge-sort/

- Searching Algorithms - GeeksforGeeks. (2021). Retrieved 12 September 2021, from https://www.geeksforgeeks.org/searching-algorithms/
- Sorting Algorithms - GeeksforGeeks. (2021). Retrieved 12 September 2021, from https://www.geeksforgeeks.org/sorting-algorithms/
- Zaveri, M. (2018). An intro to Algorithms: Searching and Sorting algorithms. Retrieved 12 September 2021, from https://codeburst.io/algorithms-i-searching-and-sorting-algorithms-56497dbaef20

Link a código: https://github.com/PAOK-2001/ErrorSorter