Algoritmos de ordenamiento y búsqueda Act.1.3

Autor: David Román Velasco

Fecha: 10/09/2021

Con esta actividad me di cuenta de los algoritmos de ordenamiento y búsqueda son bastante esenciales para el desarrollo de programas que deben de manejar bases de datos o archivos con grandes cantidades de datos, por lo que al contar con estas herramientas facilitamos la modificación y acceso a información especifica entre todos los datos que se le ingresan al programa, porque si no resultaría imposible ir buscando los datos que necesitas de uno por uno en archivos de 16,000 líneas, y menos ordenarlos, de ahí a que estos algoritmos sean tan importantes, ya que nos permiten hacer esto de una forma rápida y eficiente, un claro ejemplo es con el desarrollo de la "Act 1.3 - Actividad Integral de Conceptos Básicos y Algoritmos Fundamentales (Evidencia Competencia)", sin los algoritmos de búsqueda y ordenamiento que vimos en clase, no se hubiera llegado a una solución factible y es muy probable que no se hubiera podido realizar correctamente, ya que se debe manejar una gran cantidad de datos.

Complejidad computacional de los diferentes algoritmos:

Búsqueda Secuencial

Según RuneStone (s.f.) esta búsqueda consiste en recorrer el array desde el primer elemento hasta el ultimo de manera lineal hasta encontrar el valor buscado.

Complejidad si el item no está presente:

- Mejor: O(1)
- Promedio: O(n)
- Peor: O(n)

Complejidad si el item está presente:

- Mejor: O(1)
- Promedio: O(n/2)
- Peor: O(n)

Complejidad de búsqueda secuencial de arreglos (es similar, pero se detiene la búsqueda cuando se encuentra un valor mayor al buscado) ordenados tanto si el item está o no presente:

- Mejor: O(1)
- Promedio: O(n/2)
- Peor: O(n)

Búsqueda Binaria

Según DelftStack (2021), consiste en analizar el valor central del vector ordenado, si el elemento buscado es mayor o menor se va recorriendo por uno de los dos tramos del vector.

Complejidad:

- Mejor: O(1)
- Promedio: O($\log_2 n$)
- Peor: O(log n)

Orden Burbuja

Según Hidalgo (2001), consiste en ciclar la lista comparando elementos de dos en dos, si el que le sigue es menor al anterior se intercambian

Complejidad:

- Mejor: O(n)
- Promedio: O(n^2)
- Peor: O(n^2)

Ordena Intercambio

Según Hidalgo (2001), consiste en agarrar un valor inicial e ir comparándolo con el resto del arreglo hasta encontrar su posición final. Eso se hace con cada valor del arreglo.

Complejidad:

- Mejor: O(n)
- Promedio: $O(n^2)$
- Peor: $O(n^2)$

Ordena Merge-Sort

Según CodeMyN (s.f.), divide el arreglo en subarreglos que se subdividirán para poder ordenar el arreglo.

Complejidad:

- Mejor: O(n log n)
- Promedio: O(n log n)
- Peor: O(n log n)

El algoritmo de ordenamiento usado fue el de burbuja, sin embargo, nos dimos cuenta de que usar el de merge-sort hubiera sido más factible debido a su menor grado de complejidad, ya que como son 16,000 líneas que ordenar se tarda bastante en ordenar las líneas con el de burbuja.

Referencias

CodeMyN. (s.f.). Métodos de Ordenamiento MergeSort en C++. Recuperado de: https://codemyn.blogspot.com/2019/07/metodos-de-ordenamiento-mergesort-en-c.html

DelftStack. (2021). Búsqueda binaria. Recuperado de: https://www.delftstack.com/es/tutorial/algorithm/binary-search/

Hidalgo, J. (2001). Algoritmos de ordenamiento. Recuperado de: http://conclase.net/c/orden/introduccion

RuneStone Academy. (s.f.). La búsqueda secuencial. Recuperado de: https://runestone.academy/runestone/static/pythoned/SortSearch/LaBusquedaSecuencial.html

Universidad de Granada. (s.f.). “Capitulo 5: Arrays y Cadenas.” Proyecto: Sistema De Ayuda Al C. Recuperado de: https://ccia.ugr.es/~jfv/ed1/c/cdrom/cap5/f_cap56.htm