Tecnológico de Monterrey

ACT 1.3 Actividad Integral de Algoritmos

Rogelio Guzman Cruzado, A01639914

Estructura de datos

28/03/2022

Reflexión individual

A lo largo del primer periodo de este curso pudimos aprender el funcionamiento y la aplicación de varios algoritmos de ordenamiento y de búsqueda, esto, en parte fue de gran ayuda para poder formar una imágen más clara sobre la importancia de la implementación de estos algoritmos en proyectos de computación y me hizo pensar en lo revolucionario que es el concepto del algoritmo, ya que estos conceptos matemáticos y lógicos han ascendido al humano otro nivel y a la luna, logrando que procesos que tardarían una infinidad de tiempo si fueran realizados a mano sean resueltos en cuestión de milisegundos, así aumentando drásticamente la velocidad a la que hemos progresado como sociedad.

Debido a la naturaleza de este proyecto, tendremos que utilizar a nuestro favor algoritmos tanto de búsqueda como de ordenamiento, debido a que parte esencial de esta situación problema es poner a prueba nuestros conocimientos ordenando una set de datos muy grande de menor a mayor, y buscando un rango de información muy específica en esta misma lista ordenada, algo que no sería posible sin la ayuda de estos algoritmos.

Pero entre todos estos algoritmos que han sido ideados y testeados a lo largo de la historia, cada uno tiene un uso específico, y entre estos que sirven un mismo propósito, existen unos más optimizados que otros, haciendo su trabajo de manera más eficiente y utilizando menos tiempo y memoria. Y aunque es importante conocerlos todos y su funcionamiento lógico para poder tener un entendimiento profundo de la lógica detrás de un algoritmo y sus usos específicos, para este proyecto decidí elegir los que se consideran los más óptimos.

Como mi algoritmo de ordenamiento seleccionado me decidí por quicksort, ya que es el algoritmo más rápido conocido corriendo a una complejidad de **O(n log n)** en su caso promedio y aunque su peor caso sube su complejidad a un **O(n^2)** es muy improbable debido a que el pivote tiene que ser el último o primer dato y la lista tiene que estar sorteada en reversa. Así mismo, como el set de datos es grande decidí implementar quicksort para asegurarme que corra rápidamente. Otra opción muy estable y veloz pudo haber sido el Merge sort, pero me incliné por quicksort debido a que es un algoritmo más amigable con la memoria y el cache local debido a que no tiene que crear un array temporal extra.

Por el lado de mi algoritmo de búsqueda, utilice el binary search, y aunque gran parte de la toma de esta decisión fue porque fue explícitamente ordenado que utilizara este método, también estuve bastante contento con esta órden, debido a que búsqueda binaria también es el mejor algoritmo de búsqueda para sets de datos ordenados, gracias a que su big o notation en el caso promedio y en el peor es la misma, está siendo de **O(log n)**, y en su mejor caso de **O(1)**, postrándose así como más veloz y más eficiente que el otro método mostrado, la búsqueda secuencial, debido a que cuenta con una complejidad de O(n) haciendo que el número de pasos que calcula sean proporcionales a la n del arreglo, haciendo que sea claramente más grande que logaritmo de n, que termina siendo una fracción de los pasos de O(n).