Instituto Tecnológico y de Estudios Superiores de Monterrey

Programación de estructuras de datos y algoritmos fundamentales (Gpo 602)

Reflexión Evidencia 1

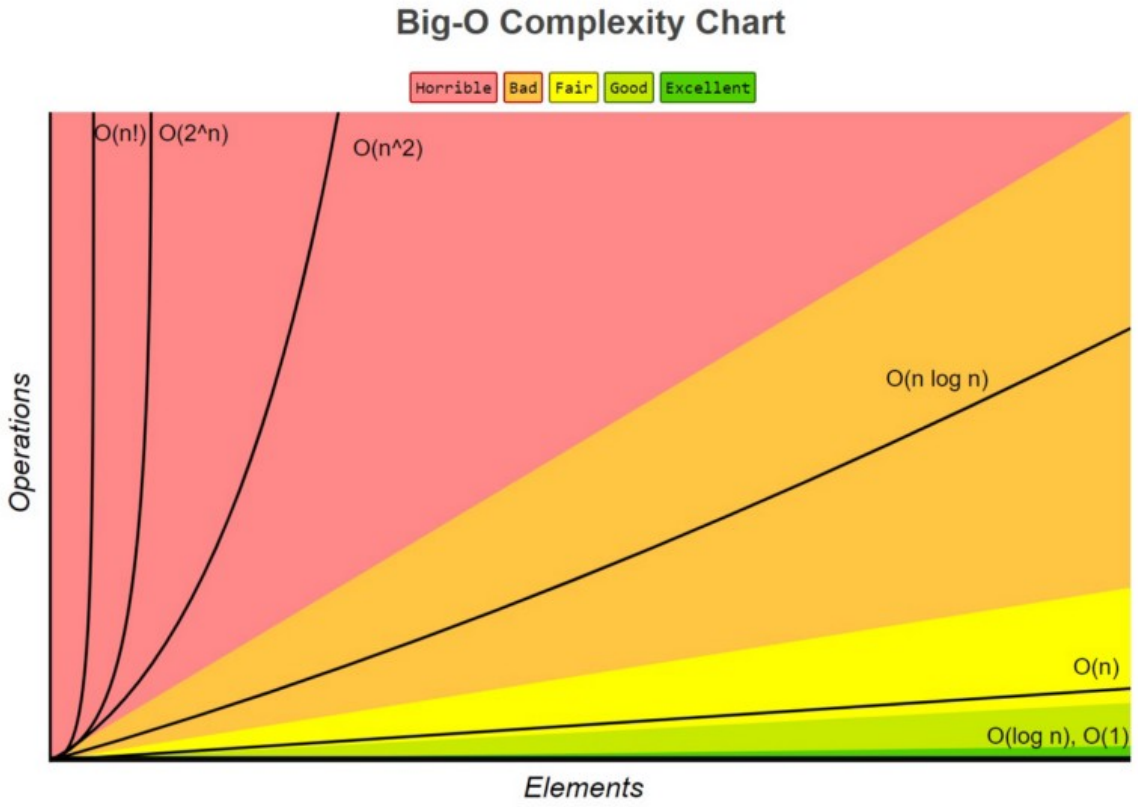
Nombre de los profesores
David Alonso Cantú Delgado

Hecho por:
Hugo Ochoa López Portillo A00835999

Domingo 10 de septiembre del 2023

Los algoritmos de ordenamiento son muy importantes en la programación ya que estos nos permiten manejar una gran cantidad de datos de manera eficiente al ordenarlos. Cuando tu coleccionas algo como las estampitas del mundial es fácil saber que repetidas tienes después de abrir cinco sobres, pero después de doscientos es muy difícil saber que tarjetas extras tienes. Tu puedes tener interés en saber cuales son tus estampas extras ya que así puedes intercambiarlas, por lo tanto es normal que las órdenes por número de estampa. Los algoritmos de ordenamiento hacen eso, ordenar largas cantidades de datos, para poder hacer lo que gustes con ellos.

Existen muchas maneras de decirle a la computadora que ordene un grupo de datos, uno como programador busca que el algoritmo que use sea el que menos recursos de la computadora use, por motivos de rapidez y eficiencia. Para decir que tan bien rinde un algoritmo se tiene la "Notación Big-O" esta nos indica la complejidad de un algoritmo cuando se ejecuta.



Big-O Notation (freeCodeCamp.org, 2020)

El algoritmo de ordenamiento que se usó para este proyecto ya que manejaba miles de datos fue "Quick Sort". Este algoritmo consiste en dividir el arreglo en dos particiones, una que contenga los elementos menores a un elemento pivote, y otra que contenga los elementos mayores al pivote. Después se ordenan ambas particiones, y automáticamente se tiene todo el arreglo ordenado. El pivote lo escoge libremente el programador, en este caso es el último elemento. Se escogió este algoritmo porque me resultó intuitivo y es válido porque en su mejor caso tiene una notación es O($n\,log(n)$), sin embargo en este código la VCC (contador,i) no se multiplica sólo se suma, por lo tanto es O(n^2), lo cual resulta en la peor versión posible de este algoritmo, este es un aspecto de mejora del código.

Otros algoritmos de ordenamiento a considerar:

- Swap Sort ($O(n^2)$): Se compara el elemento inferior de la lista con todos los restantes, se efectúa el intercambio de posiciones cuando el orden resultante no sea correcto y este proceso se repite hasta que todos los elementos estén ordenados.
- Bubble Sort($O(n^2)$): Este algoritmo de ordenamiento consiste en comparar cada elemento del arreglo con el siguiente elemento del arreglo, si el elemento es mayor que el otro, se intercambian. El objetivo de este proceso es dejar el valor más grande al final del arreglo. Se repite el proceso hasta que no sea posible ningún intercambio más.
- Selection Sort($O(n^2)$): Busca en el arreglo el valor más grande (también se puede con el menor) e intercambia este valor por el valor que se encuentre en la última (o primera) posición. Este proceso se repite con los valores restantes del arreglo (ya no se toman en cuenta los valores que ya se acomodaron).
- Insertion Sort($O(n^2)$): Consiste en comparar cada elemento del arreglo con los que se encuentra en posiciones anteriores, si resulta que el elemento con el que se está comparando es mayor, se recorre a la derecha el, sin embargo si el elemento con el que se está comparando es menor, se detiene el proceso debido a que ya se encontró la posición del elemento.
- Merge Sort $O(n\ log_2 n)$: Se divide el vector o arreglo en dos listas secundarias, luego se ordenan ambas listas secundarias luego se comparan valores entres listas secundarias para encontrar la lista principal ordenada.
- Shell Sort ($O(n^2)$): Este algoritmo de ordenamiento consiste en dividir el arreglo o vector en bloques de varios elementos para organizarlos después por medio de Insertion Sort. Este proceso se repite, pero con intervalos cada vez más pequeños, para que al final se haga un intervalo de una sola posición.

En conclusión los algoritmos de ordenamiento son muy importantes ya que nos permiten manejar grandes cantidad de datos de manera sencilla e intuitiva, además es importante saber que algoritmo de ordenamiento escoger para tener así el mejor programa posible.

Referencias

freeCodeCamp.org. (2020). All you need to know about “Big O Notation” to crack your next coding interview. freeCodeCamp.org. https://www.freecodecamp.org/news/all-you-need-to-know-about-big-o-notation-to-crack-your-next-coding-interview-9d575e7eec4/

González, G. I. C. (s. f.). Ordenamiento por método Shell. http://132.248.48.64/repositorio/moodle/pluginfile.php/1472/mod_resource/content/1/contenido/index.html