Tecnológico de Monterrey

Campus Monterrey

Programación de estructuras de datos y algoritmos fundamentales

Actividad Integral de Conceptos Básicos y Algoritmos Fundamentales

Profesor: Dr. Eduardo Arturo Rodríguez Tello

A01571226 Mariano Barberi Garza
A00833623 Iván Alberto Romero Wells

22 de Noviembre de 2022

Iván

Introducción

Los ataques cibernéticos se presentan cada vez con más frecuencia en nuestro mundo digital, en donde, la información de todas las personas conectadas por medio del internet, con el uso de múltiples dispositivos, se pone en riesgo. Un ataque cibernético puede simbolizar un gran riesgo para todos sus afectados, ya que, pone en riesgo la información de los usuarios, abriendo múltiples vulnerabilidades de alto nivel, como acceso a información personal, información bancaria, o "backdoors" que afectan de manera directa a personas y compañías.

El rastreo de los ataques cibernéticos nos permiten, no solo darnos cuenta de que estos mismos suceden, sino que también, nos permiten encontrar un origen y comportamiento del ataque, y dar un seguimiento al mismo. De este modo, el determinar acciones por realizar nos permitirán acabar o disminuir los daños por los ataques.

Marco Teórico

Arreglo

Se le conoce como arreglo a la manera indexada de organizar datos, en donde, para acceder a un dato en el arreglo, se requiere de su índice para poder acceder a él. Es una herramienta sumamente útil para poder guardar y acceder a una serie de datos y es una de las estructuras de datos más primitivas en la computación. Su implementación es muy simple, dados N datos de tipo T, se reserva en memoria N veces el espacio de almacenamiento que requiere cada

dato T. Las implementaciones modernas, como la de la estructura de datos *vector* de la librería estándar de c + +, abordan su implementación de manera dinámica, permitiendo que los N datos puedan cambiar conforme a su uso.

Algoritmos de Búsqueda

Al hacer uso de una estructura de datos lineal, como lo son los arreglos, se puede presentar la situación en donde se busca el lugar (índice) de un dato en específico dentro de la estructura. Para esto se hace uso de algoritmos de búsqueda para encontrar dicho dato en el arreglo. Los dos algoritmos de búsqueda son los siguientes:

Búsqueda lineal:

Esta búsqueda, como su nombre lo describe, consiste en una simple búsqueda de manera secuencial, en donde se comparan todos los elementos de la lista hasta encontrar el elemento buscado. Esta búsqueda tiene una complejidad de O(n).

Búsqueda binaria:

Dado un conjunto ordenado de datos, la búsqueda binaria, nos permite encontrar un elemento de forma muy eficiente. Esta búsqueda divide al conjunto por su mitad; esta mitad es evaluada con respecto al elemento a buscar, en caso de que el elemento buscado sea la mitad, se retorna este índice; por el contrario, si el dato es mayor o menor, se realiza nuevamente una búsqueda binaria en el conjunto correspondiente, ya sea de la mitad descendiendo o ascendiendo, dependiendo del caso. Esta búsqueda nos brinda la ventaja de un tiempo de ejecución de O(log n) dado que se hacen log2(n) comparaciones en el conjunto, cosa que nos brinda una gran ventaja en comparación de la búsqueda lineal.

Algoritmos de Ordenamiento

El problema de ordenar un conjunto de datos, es uno de los más importantes dentro de las ciencias computacionales. Por sus grandes aplicaciones son de los algoritmos más utilizados en la informática. Existen diversos tipos de algoritmos de Ordenamiento, entre los cuales se encuentran el ordenamiento de burbuja (Bubble Sort), ordenamiento rápido (Quicksort), ordenamiento por unión (Merge Sort), entre muchos otros. Cada uno de estos algoritmos ofrece diversas ventajas y desventajas, que se pueden emplear en diversas situaciones. A pesar de esto, es bien sabido que uno de los mejores algoritmos de ordenamiento, en términos de complejidad de tiempo y memoria, es el de unión (Merge Sort).

Merge Sort

Este algoritmo de ordenamiento tiene un funcionamiento ingenioso, que se basa en la unión de listas ordenadas. El Merge Sort funciona de la siguiente manera: Dada una lista, se separa en dos listas, la lista inferior y la lista superior, estas listas son ordenadas (se puede implementar con una llamada recursiva), posteriormente son unidas y con esto la lista estará ordenada. Este algoritmo tiene una complejidad de O(n log n) para tiempo y espacio.

Desarrollo

Para el desarrollo de esta actividad integradora, se hizo uso de un arreglo, con su implementación dinámica en c++ de vector, para poder guardar todos nuestros registros. Para poder ordenar estos registros, se contemplaron múltiples algoritmos de ordenamiento, pero dado a su consistencia, se escogió utilizar el Merge Sort. De este modo, para obtener un rango de entre nuestros registros, se implementó una búsqueda binaria, que nos permitiese encontrar en tiempo logarítmico nuestros límites introducidos por el usuario.

Conclusión

La correcta implementación de la correcta estructura de datos y algoritmos para poder realizar modificaciones y análisis en esta, nos permitió lograr dar con una solución eficiente para la problemática dada y lograr de esta manera poder delimitar registros a lo largo del tiempo.

Referencias

Mishra, A. D., & Garg, D. (2008). Selection of best sorting algorithm. International Journal of intelligent information Processing, 2(2), 363-368.

Olaya Oliveros, A. (2021). Ataques cibernéticos.

Mariano

Importancia y Eficiencia del Uso de los Diferentes Algoritmos de Ordenamiento y Búsqueda

Algoritmos de Búsqueda

Búsqueda Secuencial:

La búsqueda secuencial consiste en recorrer de inicio a fin un array desde él y comprobar si alguno de los elementos es el vector buscado.

Esto lo hace de complejidad O(n), y en un archivo con muchos elementos a buscar haría el proceso muy lento y tedioso.

Búsqueda Binaria:

Consiste en analizar, en primer lugar el elemento central del vector, y elige el lado superior o inferior dependiendo de lo que se busque, después de cada comparación elimina la mitad de los elementos en el arreglo bajo búsqueda.

Esto lo hace que sea una complejidad de O(1) haciéndolo una forma de buscar muy eficaz y rápida, así agilizando cualquier proceso que quieras hacer dentro de tu programa que requiera buscar algún valor.

Algoritmos de Ordenamiento

Bubble Sort:

El bubble sort funciona de manera en la que empieza desde un lado y verifica si el valor siguiente es mayor o menor, si el programa quiere que lo cambie lo cambia, y eso hace con cada uno de los valores hasta verificar todo sin mover

nada, significando que habrán muchos swaps en este algoritmo, pero aunque es fácil de implementar no es muy eficiente con su complejidad de O(n^2),

Selection Sort:

A diferencia del bubble sort, este algoritmo de ordenamiento no hace tantos swaps, la gran diferencia entre selection y bubble es solo eso, envés de cambiar siempre solo selecciona el dato hasta encontrar uno menor y luego lo cambia, aunque es una mejor del bubble sort sigue teniendo complejidad de O(n^2).

Insertion Sort:

Este algoritmo tiene un marcador que va moviendo mediante avanza así marcando lo que aún no se ordena, selecciona el valor más cercano a ese marcador y va comparando los valores que ya están ordenados para insertarlo en donde debería. O(n^2).

Merge Sort:

Este algoritmo siendo más difícil de implementar, por sus dos funciones necesarias vale mucho la pena ya que su complejidad de O(nlogn) crea un programa muy ágil, y como lo hace es que divide el array en dos y luego hace lo mismo con las dos divisiones hasta que queden valores individuales y en ese momento empieza a hacer comparaciones para ordenar los valores.

Quick Sort:

Tenemos un pivote en este algoritmo lo cual hace posible que cualquier número que sea mayor al pivote se mueva a la derecha de él y cualquier valor que sea menor se queda ahí, esto se repite cambiando de pivote hasta que todo quede ordenado teniendo una complejidad de O(nlogn)

Bibliografía:

S, J. I. (n.d.). CAPIT1. Retrieved November 22, 2022, from https://ccia.ugr.es/~jfv/ed1/c/cdrom/cap5/cap56.htm#:~:text=La%20búsqueda%20secuencial%20consiste%20en,array%20con%20el%20valor%20buscado

Florez, S. A. (2018, November 27). Análisis Comparativo Algoritmos de Ordenamiento. Pereira Tech Talks. Retrieved November 22, 2022, from https://pereiratechtalks.com/analisis-de-algoritmos-de-ordenamiento/