# Tecnológico de Monterrey

Campus Monterrey

Programación de estructuras de datos y algoritmos fundamentales

## Actividad Integral de Datos Lineales

Profesor: Dr. Eduardo Arturo Rodríguez Tello

A01571226 Mariano Barberi Garza
A00833623 Iván Alberto Romero Wells

26 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

*Listas Ligadas*

Las listas ligadas son una estructura de datos que consiste en encadenar nuestros datos en una lista de manera lineal, en donde cada nodo de información de nuestro conjunto, contendrá en su información al siguiente nodo (y en el caso de las listas doblemente ligadas, al nodo previo). Esto nos permite acceder a la información e iterar por medio de nuestros nodos de manera sencilla. Esta estructura, a diferencia de otras estructuras de datos, como los arreglos

indexados, tiene tiempos de consulta cuya complejidad es de O(n), con lo cuál, es importante a considerar para determinar de manera correcta su uso, con la problemática adecuada.

*Algoritmos de Búsqueda*

Al hacer uso de una estructura de datos lineal, como lo son las listas ligadas, 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). La ventaja de esta búsqueda es que no se requiere de un conjunto ordenado, sino que se puede realizar en cualquier momento.

Búsqueda binaria:

Dado un conjunto ordenado de datos ordenados, 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 dato; del contrario, se evalua si este dato está en el conjunto superior o inferior con respecto a nuestra mitad, y con esto, se realiza la búsqueda nuevamente en un conjunto de mitad de tamaño. La desventaja de utilizar la búsqueda binaria, dentro de una lista ligada, es que, dado a la naturaleza de la estructura de datos, no es posible acceder a un elemento, sin haber iterado por otros elementos previamente. Es por esto que, en las mejores implementaciones de búsqueda binaria, se sigue teniendo una complejidad temporal de ejecución de O(n), pero nos brinda la ventaja de disminuir considerablemente las

comparaciones a O(log n), cosa que nos da una ventaja considerable, comparado a la búsqueda secuencial.

*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. Este algoritmo de ordenamiento, es el mejor en funcionamiento para la implementación en listas ligadas, ya que dado a la dificultad de una consulta indexada comparado a otras estructuras, este algoritmo nos permite abordad el manejo de ordenamiento de manera distinto y bastante eficiente.

## Desarrollo

Para el desarrollo de esta actividad integral, se organizaron nuestros registros introducidos en una lista doblemente ligada. Considerando esto, al ordenar nuestros datos, se hizo uso del algoritmo de ordenamiento de Merge Sort. Una vez teniendo una lista ordenada, para obtener el rango introducido por el usuario, se implementó una búsqueda binaria, que nos permitía encontrar los nodos límites de nuestro rango de manera eficiente.

## Conclusión

El uso de estructuras de las listas ligadas, puede llegar a ser sumamente útil para ciertas aplicaciones, pero también es un factor a tener en cuenta en la implementación de diversos métodos y funciones a aplicar para la solución de la problemática en cuestión. Por ejemplo, fue fundamental tomar en cuenta la naturaleza de la estructura de datos para determinar el algoritmo de ordenamiento a emplear sin ver afectaciones en tiempos de ejecución. Por esto, es sumamente importante conocer nuestra problemática y adecuar las mejores herramientas para poder solucionar este problema

## Referencias

*Linked List Data Structure - GeeksforGeeks. (2015). GeeksforGeeks.* https://www.geeksforgeeks.org/data-structures/linked-list/

*Binary Search on Singly Linked List - GeeksforGeeks. (2018, March 21). GeeksforGeeks.* https://www.geeksforgeeks.org/binary-search-on-singly-linked-list/

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

## Mariano

### Importancia de uso de Doubly Linked List sobre Linked List

Primero debemos entender que es lo que puede hacer una Doubly Linked List que una Linked List no pueda, y esto es que la DLL puede viajar para adelante y para atrás, por lo cual hace que sea más eficaz, haciendo que en la LL hacer una inserción o borrado de cualquier nodo en una posición dada la complejidad es de O(n), en cambio en una DLL es de O(1), gracias a esto en bases de datos enormes es mucho más fácil y rápido utilizar una DLL aunque consuma más memoria.

### Inserción y Borrado

Sabiendo que en la Doubly Linked List se puede viajar para adelante y atrás, al momento de querer hacer una inserción o un borrado en cierta posición es de gran ayuda, ya que no tiene que recorrer toda la base de datos en caso de que la posición querida sea la última, este mejor desempeño al insertar y borrar datos no impacta mucho en esta solución, ya que la actividad no nos pide insertar ni borrar datos, pero emplear esta solución si nos ayuda en otras aplicaciones.

### Búsqueda

La complejidad de la búsqueda es O(n) en todos los casos, lo cual podría ser mejor en una LL en algún caso, pero cuando queramos implementar ordenamientos la DLL será mucho mejor y es por eso que usamos la DLL.

**Merge Sort**

Este algoritmo siendo un poco complicado de implementar, por sus dos funciones (en este caso tres [split, merge, merge Sort]) 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.

Esto es clave para la implementación de esta solución, ya que para dar un rango de fechas debemos primero tenerlas ordenadas para así poder verificar el momento del ataque más fácilmente, es de suma importancia que la complejidad de esto no sea enorme, ya que si tenemos una mala implementación de la ordenación nuestro programa al lidiar con tantos datos puede durar mucho tiempo, y si pensamos en incrementar la base de datos exponencialmente sería casi imposible con una computadora normal ordenar todos los datos si su complejidad es mayor.

**Bibliografía:**

Sharma, A. (2022, November 15). *Difference between a singly linked list and a doubly linked list*. PrepBytes Blog. Retrieved November 26, 2022, from https://www.prepbytes.com/blog/linked-list/difference-between-a-singly-linked-list-and-a-doubly-linked-list/#:~:text=Accessing%20elements%20in%20a%20Doubly,list%20is%20O(n).