Act 6.1 - Póster Argumentativo de Reflexión y Exposición de la Unidad de Formación TC1031

Alejandro Daniel González - A01570396
Agustín Blanco Oliver - A00828415

Estructuras de Datos (Diseño y Trabajo Equipo):

- Search and Sort Algorithms (Sequential Search y Bubble Sort)
- Linked List (Doubly Linked List)
- Binary Search Tree
- Graphs
- Hash Tables

Search and Sort Algorithms:

O(n^2)

- **Sequential Search:** Recorre el la estructura y va de manera secuencial buscando algo.
- **Bubble Sort:** Utiliza el método de intercambio.

- Esta actividad no tenía mucha dificultad, sin embargo nos fuimos con la finta y elegimos las 2 opciones más sencillas de aplicar sin tomar en cuenta la complejidad y eficiencia.

- Por lo tanto, decidimos que si se mejorara el proyecto utilizaríamos binary search e merge sort. Debido a que el worst case de merge sort es O(n log (n)) mientras que el de bubble sort es O(n^2).

- En vez de sequential search (O(n)) sería binary search (O(log n)), debido a que puede ser casi igual de fácil implementar y su complejidad es muy similar con algoritmos más complicados de implementar.

Linked List:

O(n)

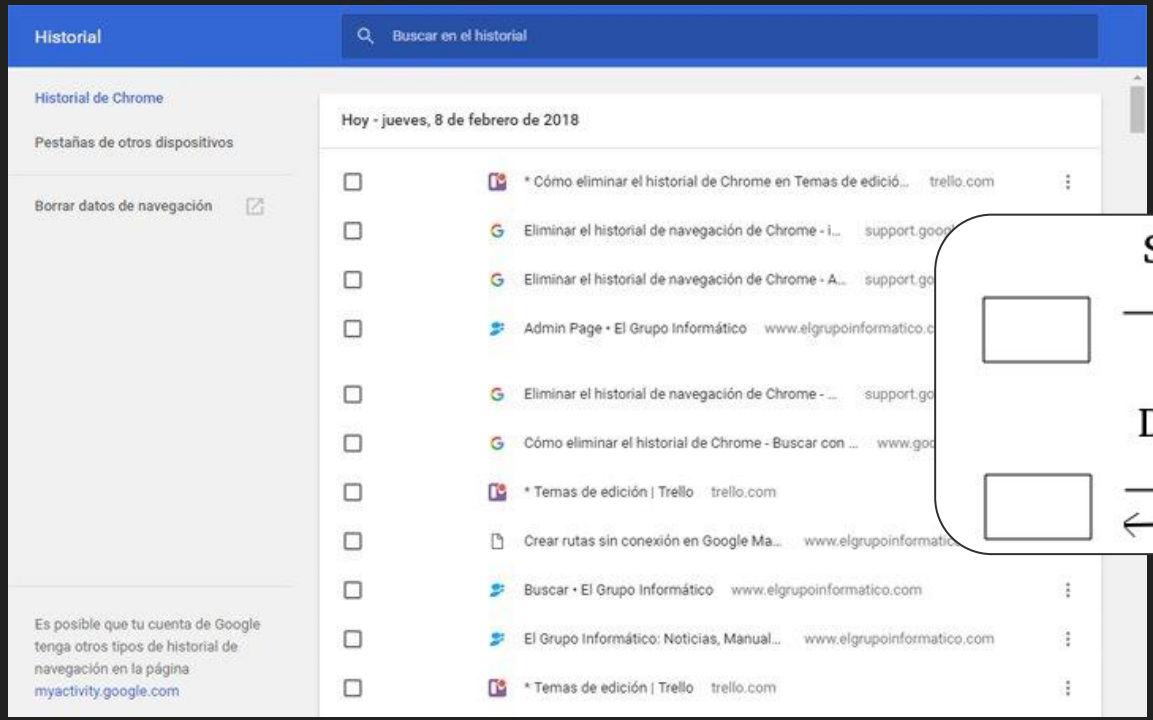
- **Single Linked List:** Recorre en una sola dirección.
- **Double Linked List:** Recorre en ambas dirección.

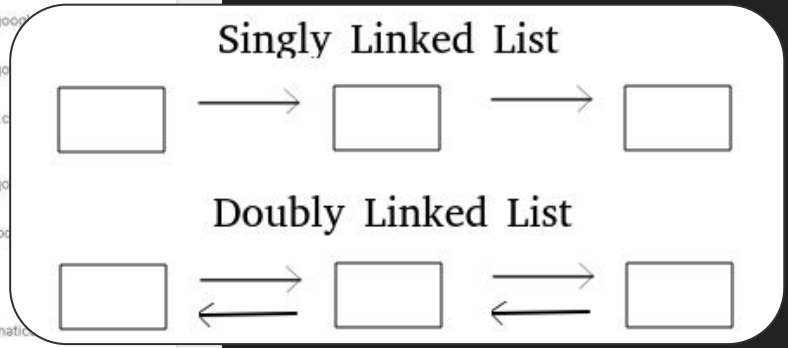
- Ya que la Single y Double Linked List tienen la misma complejidad, la Double Linked List es la superior ya que le da la capacidad de recorrer en ambas direcciones, lo cual es completamente una mejora a la Single Linked List.

- Para el caso del ejercicio integral fue un apoyo ya que requerimos ordenar los elementos, por lo tanto el conocer el elemento anterior nos permitió mejorar la eficiencia del sort y del search.

Linked List:

O(n)





Binary Search Tree:

O(log(n))

- Los arboles binarios de búsqueda son bastante útiles, ya que son un tipo de estructura de datos simples de visualizar y tienen muchas formas almacenar datos.

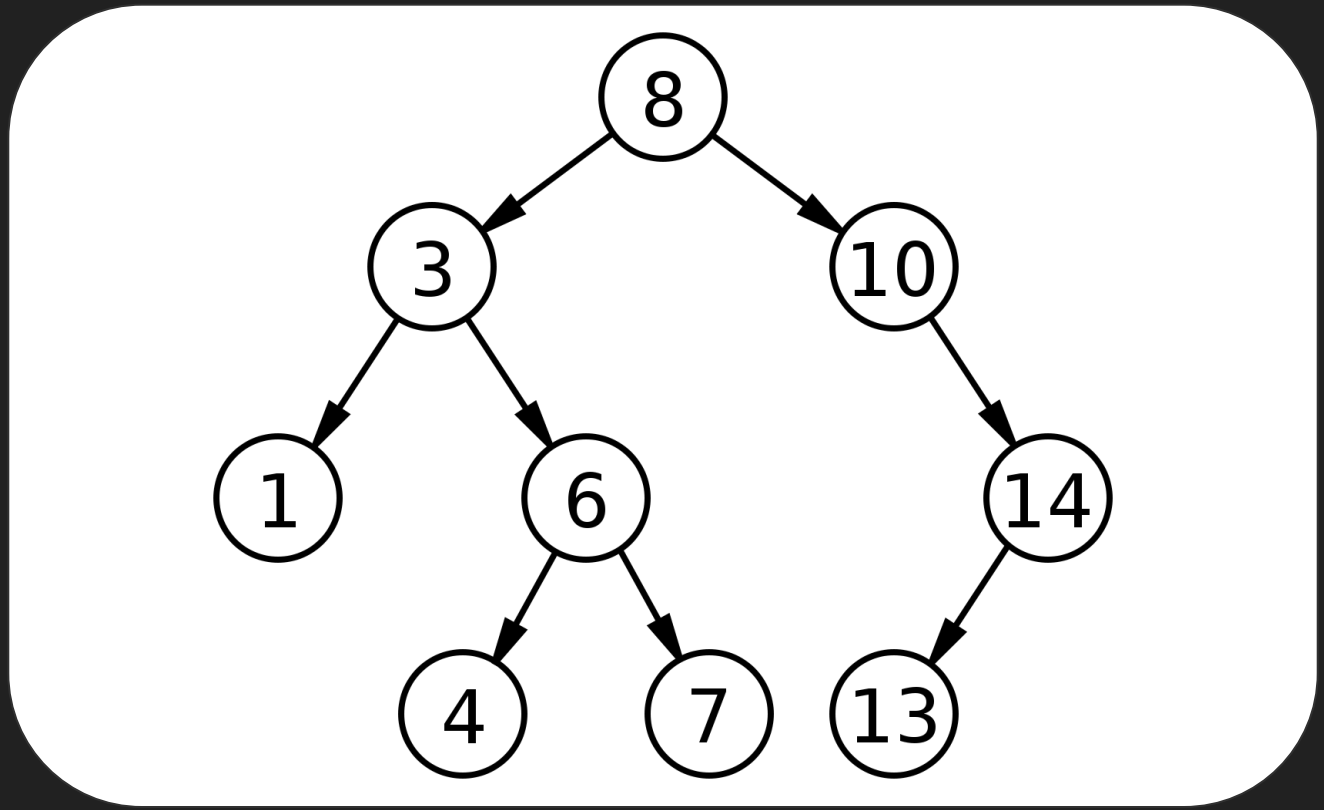
- La forma general en la que esta ordenado un árbol binario de búsqueda es que las llaves del lado izquierdo del nodo actual son menores a la llave raíz y las llaves del lado derecho del nodo actual son mas grandes que la llave raíz.

- Entonces cuando nos pidió encontrar las IPs más afectadas, al estar el árbol completo utilizamos el inOrder traversal que nos permite almacenar de manera ordenada de mayor a menor sin utilizar sort.

- Pero sin duda alguna la mayor ventaja es que el in-order traversal, así como los métodos de búsqueda y los algoritmos de ordenamiento son muy eficientes.

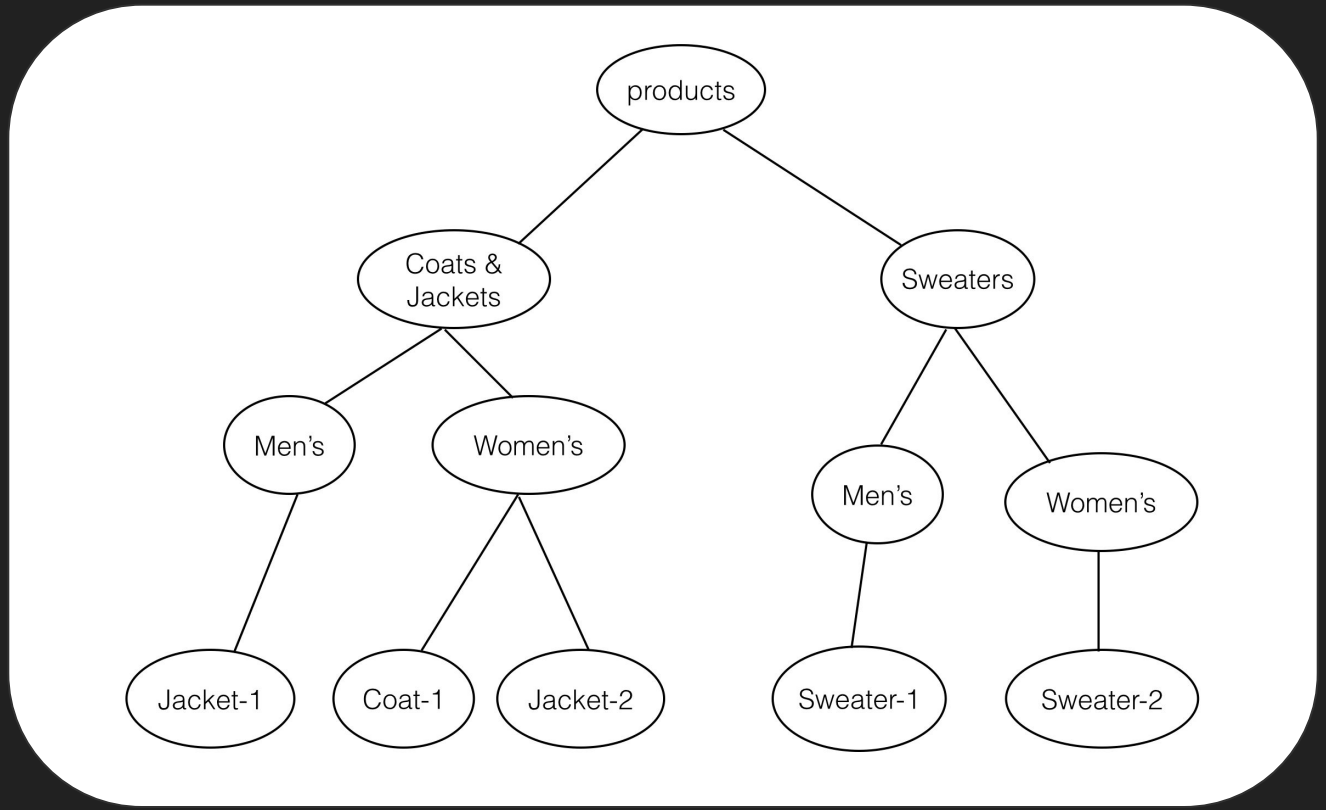
Binary Search Tree:

O(log(n))



Binary Search Tree:

O(log(n))



Graphs:

O(n)

- Los grafos son sumamente eficientes ya que a diferencia de otros tipos de estructuras de datos los nodos se pueden conectar a más de un nodo a su vez por medio de los arcos.

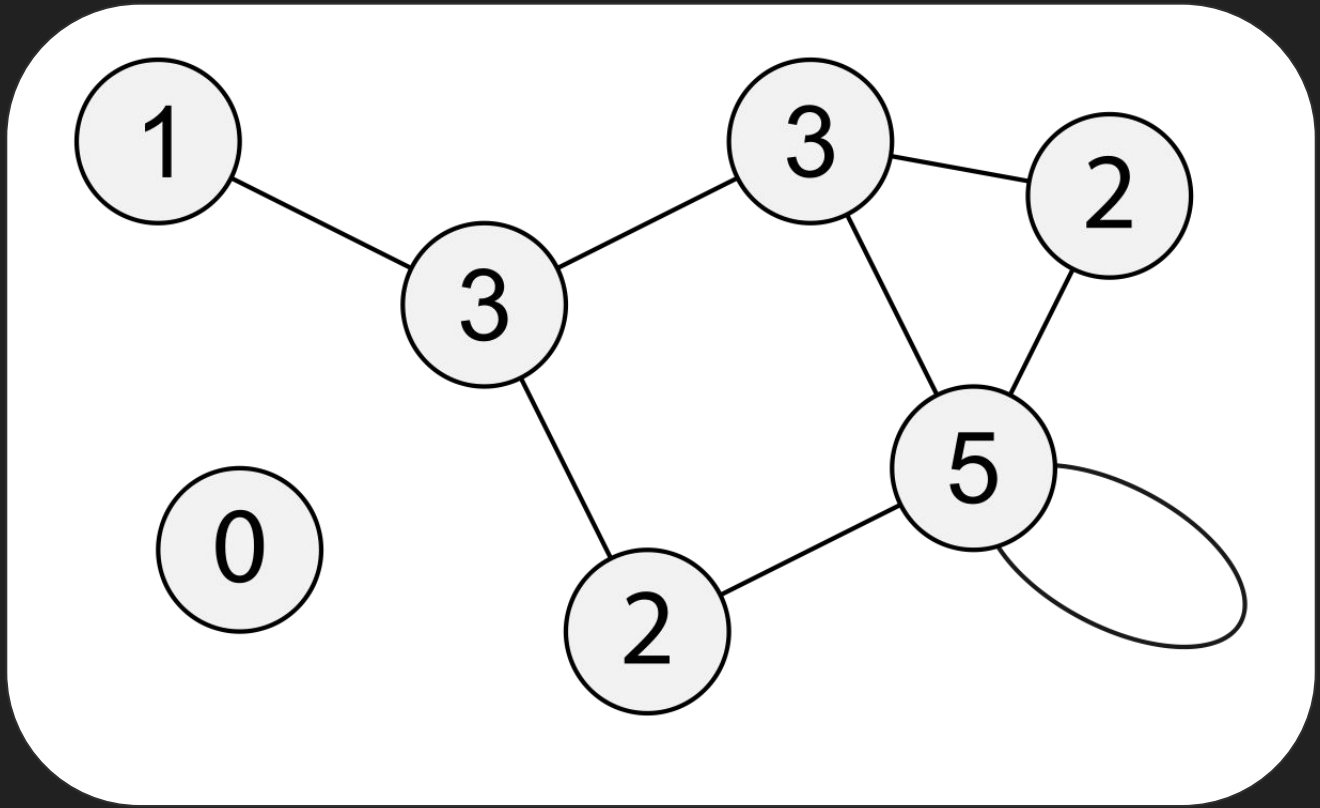
- Una característica importante que tienen los grafos es que estos pueden ser dirigidos o no dirigidos, lo que permite que los nodos sigan una trayectoria especifica o un poco libre dependiendo del tipo de este.

- Los grafos representan las interrelaciones entre elementos, por esto son muy prácticos y se pueden utilizar en casi cualquier programa en el que se requiera algún tipo de interacción.

- Esta actividad fue bastante retadora, ya que requirió del uso de lista de adyacencias para la visualización de los grafos y puede llegar a ser muy compleja cuando se intenta programar.

Graphs:

O(n)



Graphs:

O(n)

Puno
110
60
80
65
160
120
LEYENDA
Torre de Distribucion
Viviendas
Ruta del Cableado
Cable Torre - Casa

Hash Tables:

O(n)

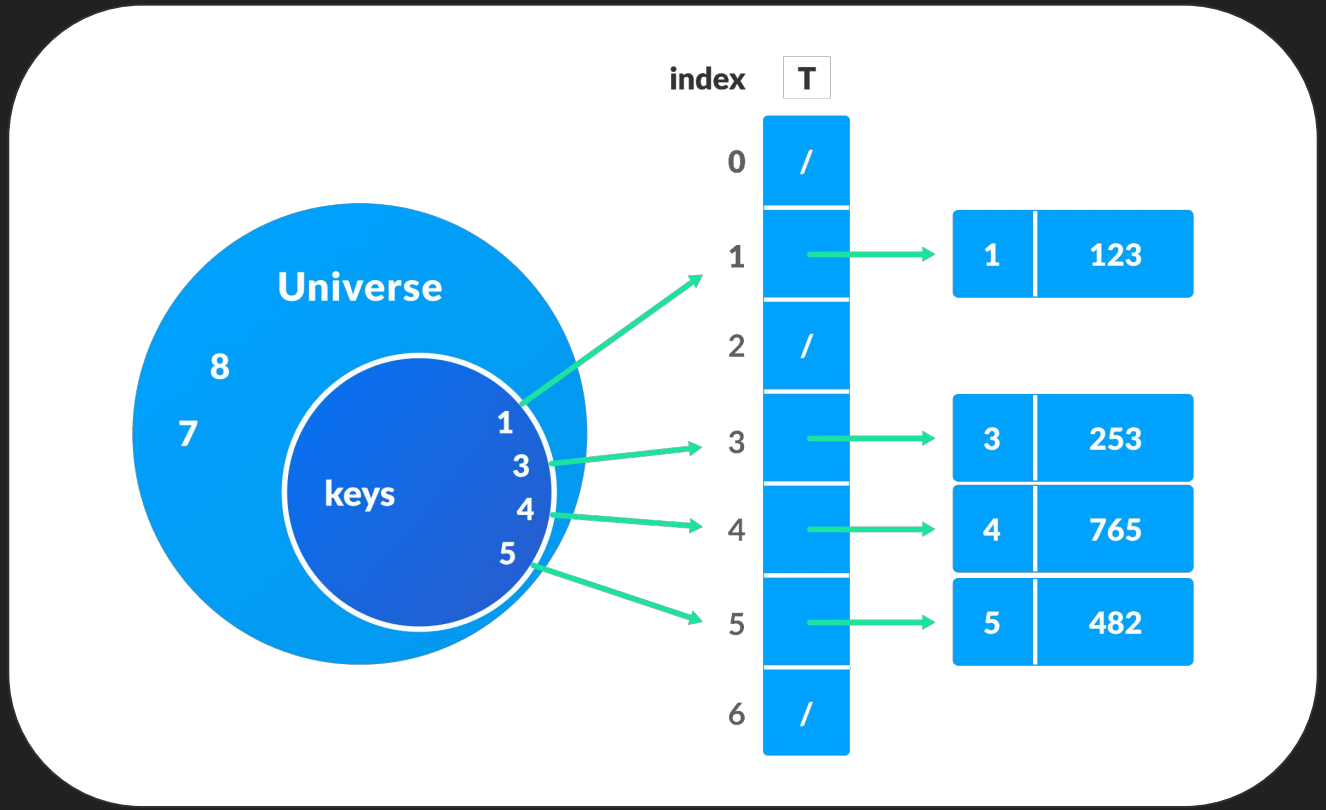
- Las Hash Tables tienen la capacidad de relacionar una llave con un valor, lo cual permite que no haya valores repetidos.

- La ventaja principal de una tabla hash en la entrega es que el acceso a los datos es extremadamente rápido siempre y cuando se cumpla lo siguiente: que no se acerque al 75% del tamaño limite de la tabla y que se distribuyan uniformemente las llaves.

- Consideramos que este fue uno de los más fáciles de implementar, además siendo hashtable uno de las estructuras más eficientes, permitió que nuestro código corriera en complejidad lineal. La implementación de un unordered_map permite que el código fuera todavía más eficiente ya que no requerimos que los datos estuvieran ordenados. El mismo unordered_map los fue agrupando en base a su llave.

Hash Tables:

O(n)



Conclusión:

En este curso vimos varias estructuras de datos, las cuales existen con la intención de optimizar código. Generalmente las estructuras de datos existen para buscar un dato, agregar datos, eliminar datos y/o ordenar una serie de datos de una forma específica.

Estas tienen ciertas características que las hacen especiales a cada una de ellas para utilizarse en diferentes situaciones de acuerdo con la que sea mas optima para esa situación específica.

¡GRACIAS POR SU TIEMPO!

Alex

Agustín