Tecnológico de Monterrey

Actividad 3.4

Evidencia competencia

Actividad Integral de árboles AVL

Equipo:

Arturo Carballo Arias - A01662245

a01662245@tec.mx

José Aram Méndez Gómez - A01657142

a01657142@tec.mx

Juan Francisco García Rodríguez - A01660981

a01660981@tec.mx

Programación de estructuras de datos y algoritmos fundamentales Gpo 631

David Alejandro Escarcega Centeno

Noviembre de 2022

Realizar en forma individual una investigación y reflexión de la importancia y eficiencia del uso de AVL en una situación problema de esta naturaleza. ¿Cómo podrías determinar si una red está infectada o no?

Actualmente vivimos en una era con demasiada tecnología, y se maneja una enorme cantidad de datos en distintos ámbitos, por lo que siempre se tienen que encontrar formas muy eficientes para manipular y ordenar estos datos. Mediante la programación se han encontrado distintos algoritmos y estructuras para llevar a cabo un orden ante tal cantidad de datos, como lo pueden ser los arreglos, las listas, o árboles de búsqueda binaria. Una de las estructuras de datos con la que hemos estado trabajando últimamente son los árboles de búsqueda binaria y los árboles AVL, las cuales han resultado tener numerosas ventajas con respecto a otras estructuras de datos; pero en este trabajo utilizamos árboles AVL, que básicamente son árboles de búsqueda binaria que se encuentran balanceados.

Un árbol AVL es un árbol de búsqueda binaria de altura equilibrada. Cada nodo está asociado a un factor de equilibrio que se calcula como la diferencia entre la altura de su subárbol izquierdo y el subárbol derecho. El árbol AVL lleva el nombre de sus inventores Adelson, Velskii y Landis, y fue publicado en 1962 en su artículo “Un algoritmo para la organización de la información”. Para que el árbol esté equilibrado, el factor de equilibrio de cada nodo debe estar entre -1 y 1; de lo contrario, el árbol estará desequilibrado y habrá necesidad de balancearlo.

El árbol AVL controla la altura de un árbol de búsqueda binaria y evita que se infecte. Cuando un árbol binario se infecta, en el peor de los casos la complejidad se hace de Big O(n) para todas las operaciones. Pero al usar el factor de equilibrio, el árbol AVL impone un límite en el árbol binario y, por lo tanto, mantiene todas las operaciones en Big O(log n). El factor de equilibrio mencionado anteriormente se debe calcular en cada operación de inserción o borrado de un nodo, el cual si es < -1 o > 1, necesitamos rotar el árbol hacia la izquierda o hacia la derecha para que no se desequilibre; hay cuatro técnicas de rotación que usamos para equilibrar los árboles AVL: rotación izquierda, rotación derecha, rotación izquierda-derecha, y rotación derecha-izquierda.

Esta estructura de datos tiene cierta algunas ventajas importantes, entre las cuales se encuentran:

- La altura del árbol AVL siempre está equilibrada, lo que significa que la altura nunca supera el O (log n), donde n es el número total de nodos del árbol.
- Brinda una mayor complejidad en el tiempo de búsqueda en comparación con los árboles de búsqueda binaria simples.
- Los árboles AVL tienen capacidades de auto equilibrio.

De igual manera, es importante mencionar que los árboles AVL se han convertido en una de las herramientas más utilizadas en cuanto al ordenamiento de datos se refiere. En el mundo real, estas son aplicadas en diversas situaciones, como por ejemplo:

- Se utilizan principalmente para tipos de conjuntos y diccionarios en memoria.
- También se usan ampliamente en aplicaciones de bases de datos en las que las inserciones y eliminaciones son menores pero hay búsquedas frecuentes de los datos requeridos.
- Se utiliza en aplicaciones que requieren búsquedas mejoradas aparte de las aplicaciones de bases de datos.

Para esta evidencia implementamos diversas funciones que nos ayudaron a insertar o borrar datos del árbol, a obtener el factor de equilibrio, a balancearlo, a recorrer el árbol, etc. La mayoría de las funciones utilizadas para programar el árbol AVL son de baja complejidad comparada con otras estructuras de datos O (log n), lo que realmente facilita trabajar con ellos; y de hecho en el proyecto lo pudimos observar, donde a pesar de la gran cantidad de datos que tenemos, la ejecución del programa es relativamente rápida..

Aram: La evidencia me ayudó a fortalecer la implementación de árboles binarios y árboles AVL. Considero que son una estructura de datos sumamente útil, ya que debido a su naturaleza logarítmica, es sumamente eficiente para hacer operaciones como insertar, eliminar o buscar. Y si hablamos de árboles AVL, son aún más eficientes, ya que mantienen la diferencia entre alturas en no mayor a 1 debido al balanceo que hacen. Creo que todas las estructuras de datos son útiles y dependiendo la problemática, unas serán más útiles que otras. Pero mi top 3 hasta ahora son listas ligadas, mapas hash y árboles binarios. Las demás, aunque sí tienen sus usos, no las considero tan eficientes en comparación a estas.

Francisco: En esta evidencia pude darme cuenta de la eficiencia que tienen los árboles binarios y AVL al momento de ordenar y realizar búsquedas con grandes cantidades

de datos. Realmente el tiempo de ejecución es bastante rápido, en comparación con estructuras de datos como los mapas hash; y en situaciones de esta naturaleza, si conviene totalmente tener un árbol de búsqueda binaria balanceado por su sencilla complejidad. Por último, quiero agregar que me pareció sencilla la implementación de funciones para esta estructura de datos, ya que tampoco requirió un nivel de programación tan elevado, e incluso pienso que es más fácil de entender que estructuras como las listas ligadas.

Arturo: Fue muy interesante poder observar como es la implementación de los árboles binarios en datos que se podrían utilizar en la vida real, además de observar cómo funciona un árbol AVL para el ordenamiento de datos. Es útil tener más algoritmos para el orden de datos.