Práctica 02

DOCENTE	CARRERA	CURSO
Vicente Machaca Arceda	Maestría en Ciencias de la Computación	Algoritmos y Estructura de Datos

PRÁCTICA	TEMA	DURACIÓN
02	Estructura de datos	–

1. Integrantes

- Grupo N 2
- Integrantes:
  - EDER ALONSO, AMPUERO ATAMARI
  - HOWARD FERNANDO, ARANZAMENDI MORALES
  - JOSE EDISON, PEREZ MAMANI
  - HENRRY IVAN, ARIAS MAMANI

2. Repositorio GitHub

URL Github: Repositorio Práctica 2 AyED

3. Estructuras de Datos

3.1. AVL

El árbol AVL recibe su nombre de las iniciales de sus inventores, Georgii Adelson-Velskii y Yevgeniy Landis. Dieron a conocer esto mediante la publicación de un artículo en 1962, Ün algoritmo para organizar la información"(Ün algoritmo para organizar la información").

Un árbol AVL es un árbol de búsqueda binaria que tiene una altura equilibrada: para cada nodo x, las alturas de los subárboles izquierdo y derecho de x difieren en 1 como máximo. Para implementar un árbol AVL, mantenemos un atributo adicional en cada nodo: x.h es la altura del nodo x. Como para cualquier otro árbol binario de búsqueda T, asumimos que T.root apunta al nodo raíz.

El árbol AVL siempre está equilibrado de modo que para todos los nodos, la altura de la rama izquierda no difiera en más de una unidad de la altura de la rama derecha o viceversa. Gracias a esta forma de equilibrio, la complejidad de una búsqueda en uno de estos árboles se mantiene siempre en el orden de complejidad O(log n). El factor de equilibrio puede almacenarse directamente en cada nodo o calcularse a partir de la altura de los subárboles.

Para lograr este equilibrio, la inserción y eliminación de nodos debe realizarse de manera especial. Si la condición de equilibrio se rompe al realizar una operación de inserción o eliminación, se debe realizar una serie de rotaciones de nodos.

3.1.1. Resultados del experimento

- En la figura 7 mostramos la búsqueda del nodo 22 con resultado de no encontrado
- En la figura 7 mostramos la búsqueda del nodo 21 con resultado de "Se encontró el nodo".
- En la figura 7 mostramos el máximo y mínimo valor .
- En la figura 3 mostramos nuestro árbol inicial con 11 nodos.
- En la figura 4 mostramos el árbol luego de insertar un nodo nuevo.
- En la figura 5 mostramos luego de eliminar un nodo.
- En la figura 6 mostramos luego de eliminar el nodo raíz.

```
Buscando nodo 22
No encontrado
Buscando nodo 21
Se encontro el nodo
Resultado: 21
valor Mínimo: 6
valor Máximo: 63
```

Figura 1: Árbol AVL búsqueda variada

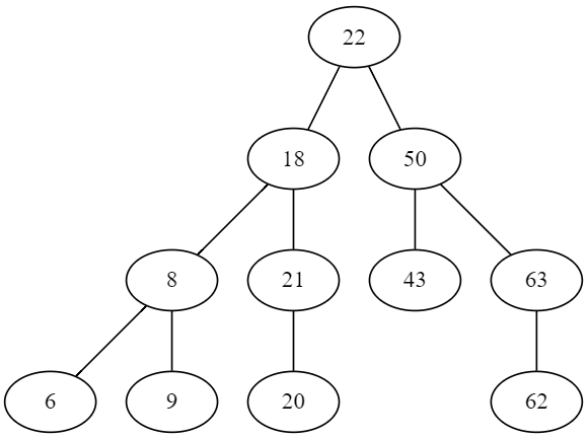

Figura 2: Árbol AVL inicial

3.2. B-Tree

La idea tras los árboles-B es que los nodos internos deben tener un número variable de nodos hijo dentro de un rango predefinido. Cuando se inserta o se elimina un dato de la estructura, la cantidad

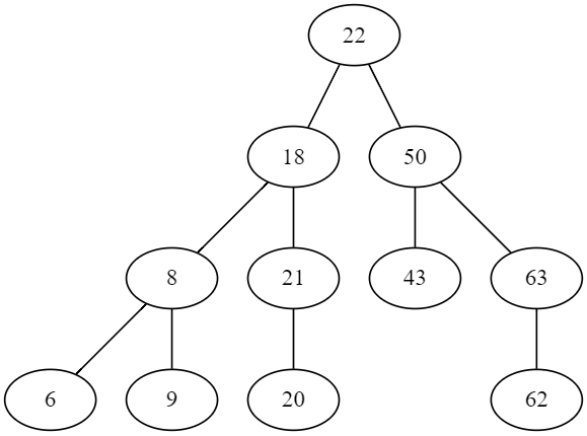

Figura 3: Árbol AVL inicial

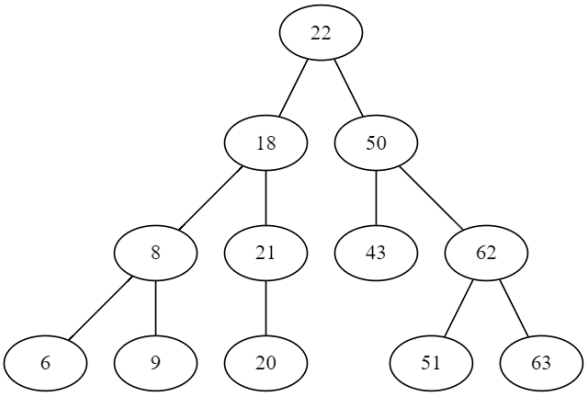

Figura 4: Árbol AVL con inserción

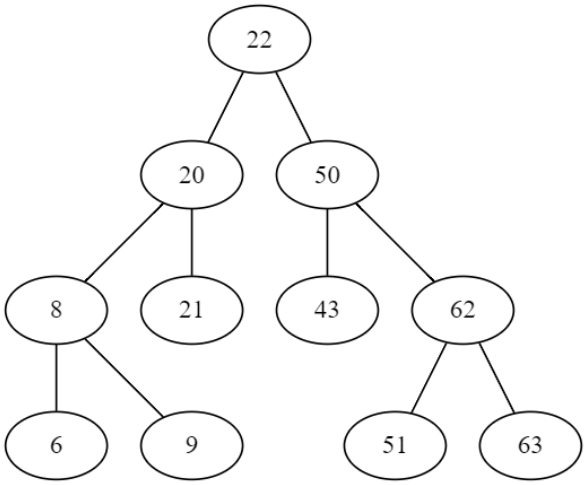

Figura 5: Árbol AVL con eliminación nodo

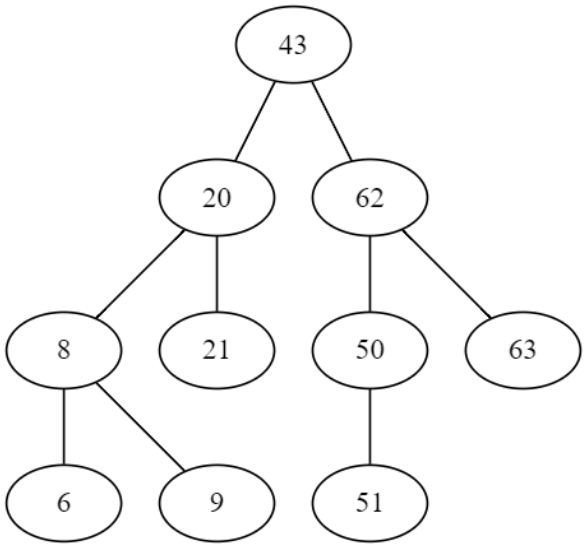

Figura 6: Árbol AVL con eliminación de la raíz

de nodos hijo varía dentro de un nodo. Para que siga manteniéndose el número de nodos dentro del rango predefinido, los nodos internos se juntan o se parten. Dado que se permite un rango variable de nodos hijo, los árboles-B no necesitan rebalancearse tan frecuentemente como los árboles binarios de búsqueda auto-balanceables. Pero, por otro lado, pueden desperdiciar memoria, porque los nodos no permanecen totalmente ocupados. Los límites (uno superior y otro inferior) en el número de nodos hijo son definidos para cada implementación en particular Un árbol-B se mantiene balanceado porque requiere que todos los nodos hoja se encuentren a la misma altura. Operaciones Básicas **Buscar:** La

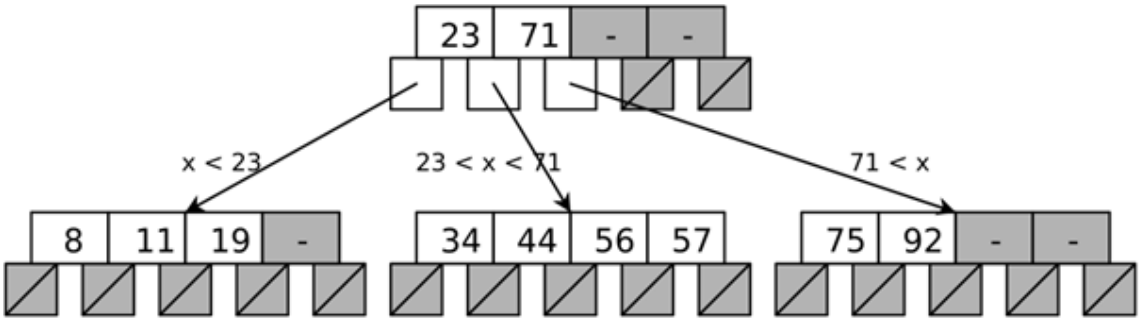

Figura 7: Gráfica B-tree

búsqueda es similar a la de los árboles binarios. Básicamente, empieza comprobando el nodo raíz, y si este es el que se busca, termina la operación. Si no lo es, mueve el puntero a su hijo izquierdo o al derecho, dependiendo de si el dato que se busca es menor o mayor del que contiene el nodo padre. De esta forma continua hasta encontrar el dato o terminar de recorrer el árbol sin encontrarlo. Este proceso es recursivo, puesto que el puntero se mueve hasta un nodo hijo, y podemos considerar a este como el nodo raíz de un nuevo árbol. El costo computacional es O($Log_2 n$)) **Insertar:** Las inserciones se hacen en los nodos hoja.

- Realizando una búsqueda en el árbol, se halla el nodo hoja en el cual debería ubicarse el nuevo

elemento.

- Si el nodo hoja tiene menos elementos que el máximo número de elementos legales, entonces hay lugar para uno más. Inserte el nuevo elemento en el nodo, respetando el orden de los elementos.
- De otra forma, el nodo debe ser dividido en dos nodos.

Eliminar: La eliminación de un elemento es directa si no se requiere corrección para garantizar sus propiedades. Hay dos estrategias populares para eliminar un nodo de un árbol B.

- localizar y eliminar el elemento, y luego corregir, o
- hacer una única pasada de arriba abajo por el árbol, pero cada vez que se visita un nodo, reestructurar el árbol para que cuando se encuentre el elemento a ser borrado, pueda eliminarse sin necesidad de continuar reestructurando

Se pueden dar dos problemas al eliminar elementos. Primero, el elemento puede ser un separador de un nodo interno. Segundo, puede suceder que al borrar el elemento número de elementos del nodo quede debajo de la cota mínima

3.2.1. Resultados del experimento

- Árbol Principal
- Insertar dos nodos
- Borrar un nodo con dos hijos
- Borrar nodo raíz
- Búsqueda de un nodo
- Búsqueda del mínimo
- Búsqueda del máximo

4. Conclusiones

- Un árbol AVL se mantiene ordenado, pero hay mas rotaciones en las inserciones que en las eliminaciones, su costo para buscar, eliminar, insertar es de O(Log n).
- Funciona de manera casi instantánea incluso con grandes volúmenes de datos. Ello se debe principalmente a la característica equilibrada del árbol, lo que permite tener acceso a todos los elementos con el mismo número de etapas, y en segundo lugar, al crecimiento logarítmico de la profundidad del árbol. Eso significa que la profundidad del árbol crece lentamente en comparación al número de hojas. Hay índices reales con millones de registros que tienen una profundidad de cuatro o cinco. Es poco común encontrar una profundidad de seis

5. Referencias

1. González, A. H. (2013). Operaciones sobre árboles.
2. Weiss, M. A., Jorge tr Lozano Moreno, Andoni colab. téc Eguíluz, Inés colab. téc Jacob. (1995). Estructuras de datos y algoritmos.
3. Gutiérrez, X. F. (2002). Estructuras de datos: especificación, diseño e implementación. Univ. Politèc. de Catalunya.

4. Brassard, G., Bratley, P., Giner, R. G. B. (1997). Fundamentos de algoritmia (Vol. 3, No. 5.1). eMadrid Madrid: Prentice Hall.
5. Binary search tree
6. Graph Playground
7. Visualización animada estructuras datos
8. Graphviz
9. Visualgo
10. Geeksforgeeks
11. Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Clifford Stein (2009). Introduction to Algorithms (3er edición). MIT Press and McGraw-Hill.