TAREA 4 - DIFERENCIAS ENTRE ÁRBOLES AVL, B, B+ Y ROJO-NEGRO

Johan Araya González
e-mail: jarayag1@ucenfotec.ac.cr
Luis Fernando Solano Montoya
e-mail: lsolanom1@ucenfotec.ac.cr
Stephanie Quiros Hernández
e-mail: squirosh@ucenfotec.ac.cr
Jose Picado Zumbado
e-mail: jpicadoz@ucenfotec.ac.cr
Douglas Conejo Cascante
e-mail: dconejoc@ucenfotec.ac.cr

RESUMEN: En este documento se brinda una explicación sobre las estructuras de datos llamadas árboles, enfocándose en encontrar las diferencias de los tipos: AVL, B, B+ y Rojo y Negro.

1 INTRODUCCIÓN

Los árboles representan las estructuras no lineales y dinámicas de datos más importantes en computación . Dinámicas porque las estructuras de árbol pueden cambiar durante la ejecución de un programa. No lineales, puesto que a cada elemento del árbol pueden seguirle varios elementos.

La definición de árbol es la siguiente: es una estructura jerárquica aplicada sobre una colección de elementos u objetos llamados nodos; uno de los cuales es conocido como raíz. además se crea una relación o parentesco entre los nodos dando lugar a términos como padre, hijo, hermano, antecesor, sucesor, ancestro,entre otros... Formalmente se define un árbol de tipo T como una estructura homogénea que es la concatenación de un elemento de tipo T junto con un número finito de árboles disjuntos, llamados subárboles. Una forma particular de árbol puede ser la estructura vacía.

La figura siguiente representa a un árbol general.

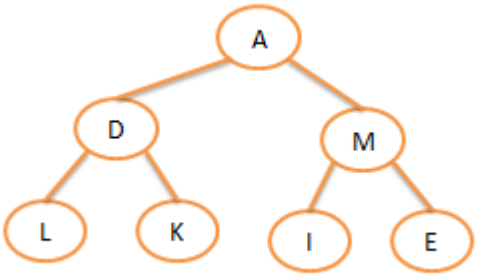

Figura #1 árbol general.

Se utiliza la recursión para definir un árbol porque representa la forma más apropiada y porque además es una característica inherente de los mismos.

Los árboles tienen una gran variedad de aplicaciones. Por ejemplo, se pueden utilizar para representar fórmulas matemáticas, organizar adecuadamente la información, construir un árbol genealógico, para el análisis de circuitos eléctricos además de numerar los capítulos y secciones de un libro.

Se abordará el análisis de las estructuras de datos e inserción de claves en los árboles AVL, B, B+ y Rojo-Negro indicando al final sus diferencias.

2 Arboles Binarios:

2.1 Árbol AVL

Árbol AVL: Un árbol AVL según Gurín, es un árbol binario de búsqueda que cumple que para todo nodo del árbol, la diferencia en la altura de sus subárboles es a lo mucho 1. Se denomina AVL por las iniciales de los creadores de esta estructura Adelson-Velskii y Landis.

Para poder implementar esta estructura es necesario hacer rotaciones en el árbol para poder mantener la condición que hace al árbol AVL. Estas rotaciones son de cuatro tipos, rotación simple a la izquierda, rotación simple a la derecha, rotación doble a la izquierda y rotación doble a la derecha. Estas rotaciones se realizan al insertar o eliminar un nodo del árbol y que este quede desbalanceado.

Se conoce como factor de equilibrio (FE) de un nodo la medida de la diferencia de alturas de sus subárboles, por cada nivel de altura del subárbol izquierdo se resta una unidad al FE y por cada nivel de altura del subárbol derecho se suma una unidad al FE, después de realizar las operaciones si el FE del nodo es 1,-1 o 0, entonces el nodo está equilibrado, en cualquier otro caso se debe realizar una rotación para equilibrar el nodo.

2.1.1 Características

- Debido a estar balanceado su eficiencia es de a lo sumo orden O(log(n)) lo que presenta una mejora a un árbol binario de búsqueda que tiene como peor caso orden O(n).
- El árbol siempre está equilibrado
- Los subárboles de cada nodo tienen una diferencia de a lo sumo 1 en su altura.

Una representación gráfica de un árbol que está AVL es la siguiente:

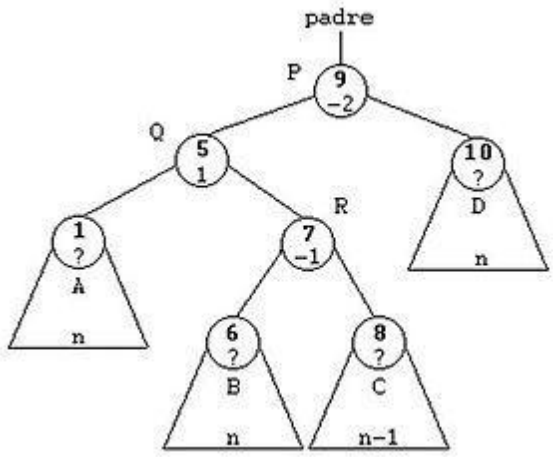

Figura 2. Árbol AVL.

En caso contrario un ejemplo de un árbol que no es AVL es el siguiente:

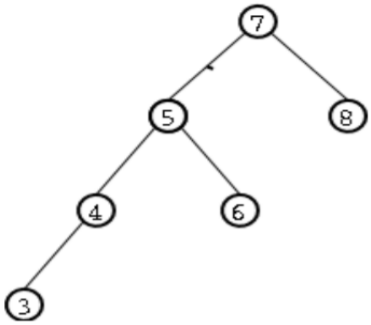

Figura 3. Árbol que no es AVL.

En la imagen anterior se puede observar que este árbol no es AVL pues el nodo con el dato 7 tiene un subárbol izquierdo de altura 3 y un subárbol derecho de altura 1, por lo que su diferencia es 2, lo que incumple la condición para que el árbol sea AVL.

En el caso de las inserciones y eliminaciones en un árbol AVL son iguales a los de un árbol binario de búsqueda, lo que cambia es que durante una inserción o eliminación, si se desequilibra el árbol se realiza una rotación, por lo que se van a exponer las rotaciones mencionadas anteriormente.

2.1.2 Rotación simple a la izquierda

Este tipo de rotación se utiliza cuando el FE de un nodo es de +2, es decir que el subárbol derecho del nodo tiene dos niveles más de altura que el izquierdo. Esta rotación, se basa primero en tomar como nuevo subárbol derecho del nodo, el subárbol izquierdo del hijo derecho original. Después, al hijo derecho original se le asigna como como hijo izquierdo el nodo original. Y finalmente se asigna como nuevo nodo el subárbol derecho original. Gráficamente se vería así:

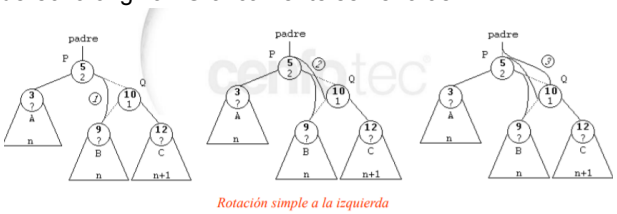

Figura 4. Rotación simple a la izquierda.

2.1.3 Rotación simple a la derecha

Esta rotación es simétrica a la anterior y se da cuando el FE del nodo es de -2, es decir cuando el subárbol izquierdo tiene dos niveles más que el subárbol derecho. Y su algoritmo es tomar como nuevo subárbol izquierdo del nodo, el subárbol derecho del hijo izquierdo original. Seguidamente, se toma como nuevo subárbol derecho del hijo izquierdo original el nodo y finalmente queda como nuevo nodo el hijo izquierdo original. Lo que gráficamente se ve así:

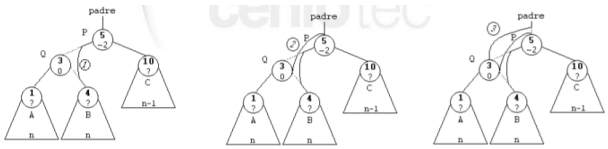

Figura 5. Rotación simple a la derecha.

2.1.4 Rotación doble a la izquierda

Esta rotación se da cuando el FE del nodo es de +2 y además que la raíz del subárbol derecho de dicho nodo tenga un FE de -1. Y el algoritmo es primero realizar una

rotación simple del subárbol derecho a la derecha y después una rotación simple del nodo a la izquierda. Y de manera gráfica se representa de la siguiente manera:

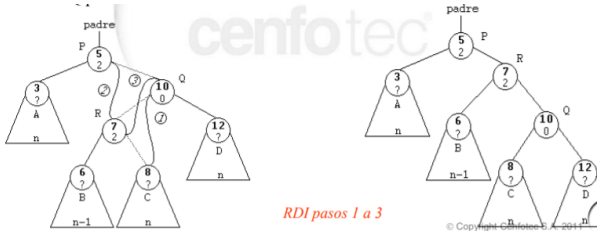

Figura 6. Primeros 3 pasos de la rotación doble a la izquierda.

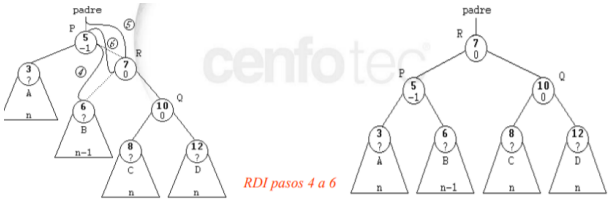

Figura 7. Últimos 3 pasos de la rotación doble a la izquierda.

2.1.5 Rotación doble a la derecha

Esta rotación similar a los casos anteriores, es simétrica a la rotación doble a la izquierda pero cuando el FE del nodo es -2 y que la raíz del subárbol izquierdo del nodo tenga un FE de +1. Y similar a su contraparte el algoritmo es primero realizar una rotación simple a la izquierda del subárbol izquierdo del nodo y después una rotación simple del nodo a la derecha. Y en este caso su representación gráfica es la siguiente:

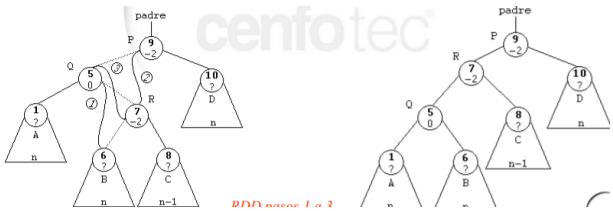

Figura 8. Primeros 3 pasos de la rotación doble a la derecha.

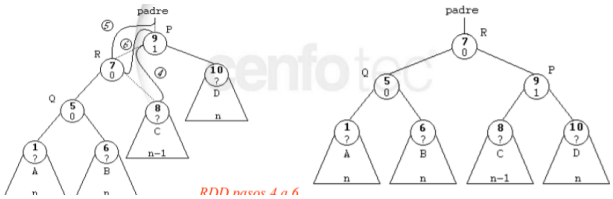

Figura 9. Últimos 3 pasos de la rotación doble a la derecha.

2.2 Árbol B

Árbol B: Se utilizan para manejar archivos que contengan gran cantidad de información, los cuales son usados como método de búsqueda externa.

Fueron propuestos por Bayer McCreight en 1970.

Cuando se inserta o se elimina un dato de la estructura, la cantidad de nodos hijo varía dentro de un nodo.

Los árboles B no necesitan re balancearse tan frecuentemente como los árboles binarios de búsqueda auto balanceables.

1. 2.2.1 Características:

- El árbol B crece hacia arriba.
- El orden (m) de un árbol B es el número máximo de ramas que pueden partir de un nodo.
- Si n es el número de ramas que parten de un nodo de un árbol b, el nodo contendrá n-1 claves.
- El árbol está ordenado.
- Todos los nodos terminales, (nodos hoja), están en el mismo nivel.
- Todos los nodos intermedios, excepto la raíz, deben tener entre m/2 y m ramas no nulas.
- El máximo número de claves por nodo es m-1
- El mínimo número de claves por nodo es (m/2)-1.
- La profundidad (h) es el número máximo de consultas para encontrar una clave.
- Para insertar una clave se verifica si la página donde le corresponde ingresar ordenado tiene espacio, si hay espacio se pueden reordenar las claves en la página. Si no hay espacio se parte la página y se crean dos páginas. Y el elemento central se promociona y sube a la página superior. Si en la página superior hay espacio se ordenan las claves de lo contrario si está lleno se parte la página en dos y se promociona el valor intermedio creándose una nueva raíz.

En el ejercicio se desea añadir la clave 68

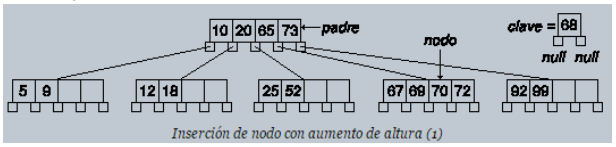

Figura 10. Insertar clave 68 en árbol B

El valor 68 va entre 67 y 69, entonces se debe partir la página en dos páginas y se promueve el valor intermedio, en este caso el 69 hacia la página superior.

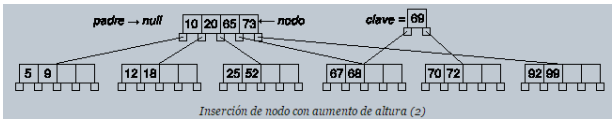

Figura 11. La página se parte en dos páginas

El 69 es promovido pero como la raíz está llena, se divide la raíz en dos páginas y se promueve el 65 y se crea una nueva raíz con el 65.

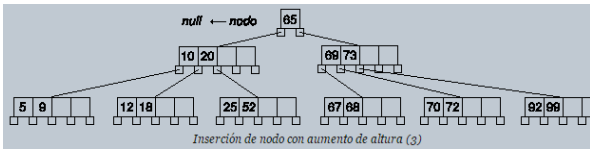

Figura 12. La página raíz está llena y se divide

2.3 Árbol B +

Árbol B +: según la Universidad de Granada, Los árboles B+ son una mejora sobre los árboles B, debido a que conservan la propiedad de acceso aleatorio rápido y implementan un recorrido secuencial rápido.En un árbol B+ todas las claves se encuentran en hojas,replicándose en la raíz y nodos interiores aquellas que resulten necesarias para definir los caminos de búsqueda.Para facilitar el recorrido secuencial rápido las hojas se pueden vincular, consiguiendo una trayectoria secuencial para recorrer las claves del árbol.

Características:

- Cada una de las páginas, a excepción de la raíz, tiene entre [m/2] y m descendientes.
- Cada una de las páginas, a excepción de la raíz, contiene entre [m/2] –1 y m –1 elementos.
- La página que corresponde a la raíz, o es hoja o tiene al menos 2 hijos.
- Las páginas hojas se encuentran todas al mismo nivel.
- Todas las claves están en las páginas hojas.
- Las claves de las páginas raíz e interiores se utilizan como índices.
- Las hojas del árbol están enlazadas

según la Universidad de Granada, La representación de un Árbol B+ es la siguiente:

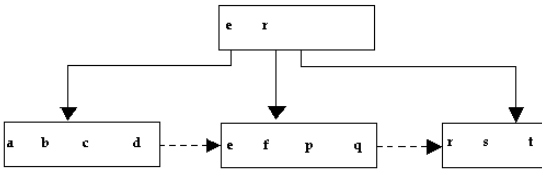

Figura 13. Árbol B+.

2. 2.3.1 INSERCIÓN EN UN ÁRBOL B+

La diferencia con la inserción en árboles B está en que cuando se inserta una nueva clave en una página llena, ésta se divide en otras dos,pero ahora la primera contendrá m/2 claves y la segunda 1+m/2, y lo que subirá a la página anterior será una copia de la clave central.

2.3.2 BORRADO EN UN ÁRBOL B+

En una operación de borrado se debe considerar:

Si al eliminar la clave el número de claves es mayor o igual a m/2 el proceso finaliza. Las claves de las páginas raíz o internas no son modificadas aunque sean una copia de las eliminadas,pues siguen siendo un separador válido entre las claves de las páginas descendientes.

Si al eliminar la clave el número de las mismas en la página es menor que m/2 será necesaria una fusión y redistribución de las mismas tanto en las páginas hojas como en el índice.

2.3.3. BÚSQUEDA EN UN ÁRBOL B+

En la búsqueda no debe parar cuando se encuentre la clave en la página raíz o en una página interior,si no que se debe proseguir en la página apuntada por la rama derecha de dicha clave.

2.4 Árbol Rojo Negro

Árbol Rojo-Negro: Es un árbol binario de busqueda con un bit extra por el nodo (el color) y contiene las siguientes características:

- La raíz es negra.
- Cada nodo tiene un estado rojo o negro.
- Los nodos hojas (nulos) son negros.
- Un nodo rojo tiene 2 hijos negros.
- Todo camino de la raíz a cualquier hoja pasa por el mismo número de nodos negros.

La notación de un Árbol Rojo-Negro es la siguiente:

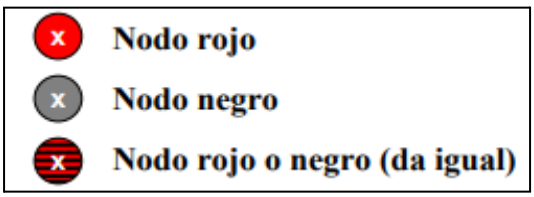

Figura 14. Notación de un Árbol Rojo-Negro

Se define el concepto de altura negra de un nodo, se le denomina por la letra “H” para diferenciarla de la altura normal “h”. Si n es un nodo se puede calcular su altura negra así:

H(n) = { máx(H(n.izdo), H(n.dcho)) + 1 si n es negro
máx(H(n.izdo), H(n.dcho)) si n es rojo

La condición de equilibrio de este árbol es que el camino más largo desde la raíz hasta una hoja (rama) no puede ser más largo que el doble del camino más corto, calculado desde la raíz hasta la hoja más corta.

Y precisamente por la forma de la estructura, permite el doble de diferencia, y eso explica porque su eficiencia se duplica con respecto a la del árbol AVL.

Se deben cumplir las siguientes propiedades:

1. Cambiar un nodo de rojo a negro,(la altura negra incrementa en todos los nodos ascendientes).
2. Cambiar un nodo de negro a rojo, si el padre o alguno de los hijod es rojo,(la altura negra se decrementa en todos los nodos ascendienetes).
3. Si como resultado de una operación la raiz paasa a ser rojo, se puede cambiar a negro directamente sin afectar las condiciones.
4. Para borrar un nodo rojo no afecta, pero borrar un nodo negro, la altura negra decrece en los decendientes.

2.4.1 .INSERCIÓN DE NODOS.

La inserción se realiza inicialmente como en un árbol binario de búsqueda. Al nuevo nodo se le asigna el color rojo.

Dependiendo del caso se realizan una serie de operaciones, donde “x” representa el nodo que se está comprobando, (x) es un nodo rojo, su padre (y) existe y es rojo, y su abuelo (z) existe pero no puede ser raíz de un nodo rojo. El nodo hermano del padre se denomina tío (t).

Caso 1: Tío rojo, nodo (x) izquierdo o derecho:

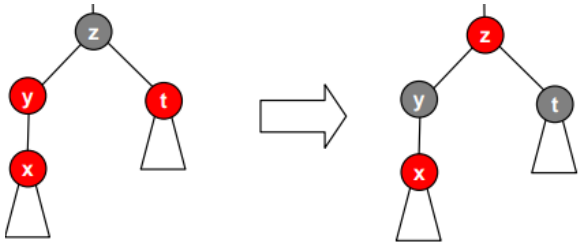

Figura 15. Caso de inserción #1.

Como se observa en la figura 15 (y) y (t) son rojos, lo cual infringe en la condición. La solución es cambiar de color los nodos y,z,t. Pero se da la condición de que el padre (z) es rojo y ahí estaría violando de nuevo la primera condición.

Caso 2: Tío negro, nodo (x) es hijo derecho:

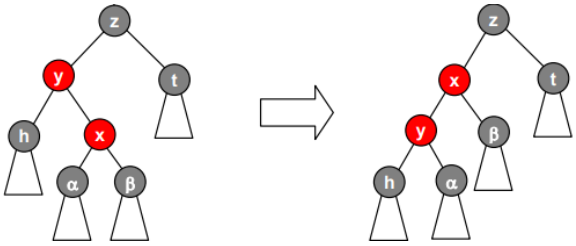

Figura 16. Caso de inserción #2.

Como se visualiza en la figura 16 el tío de (x) es negro, Se debe hacer una rotación x-y. Eso no soluciona el problema pero ahora el nodo que hay que comprobar tiene que ser un hijo izquierdo, El nodo que se comprueba ahora es el (y), y pasa al caso 3.

Caso 3: Tío negro, nodo (x) es hijo izquierdo:

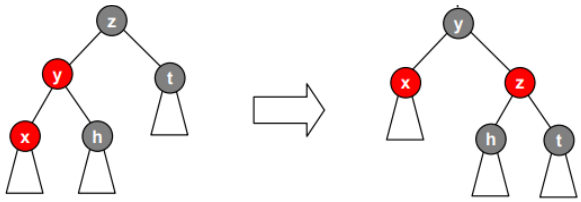

Figura 17. Caso de inserción #3.

Tomando como referencia la figura # se debe realizar una rotación z-y o rotación simple derecha, luego se le cambia el color a (z) y (y), esto resuelve el problema y el árbol cumple con todas las condiciones.

2.5 DIFERENCIAS ENTRE ÁRBOLES

2.5.1 FORMA DE CRECIMIENTO

AVL y Rojo Negro: crecen hacia arriba
B y B+: crecen hacia arriba

2.5.2 UBICACIÓN DE CLAVES

AVL y Rojo Negro: los menores que la raíz se insertan a la izquierda y los mayores a la derecha.

B y B +: Las claves se insertan en la página en forma ordenada ascendente.

2.5.3 INSERCIÓN DE CLAVES

AVL: Si el FE >= 2, se debe rebalancear, utilizando: RSD,RSI,RDD,RDI.

Rojo Negro: El nodo raíz que se inserta siempre tiene el color negro.

Se debe comparar si los tíos son del mismo color y cambiarlos con respecto a las condiciones explicadas en el punto 3.4.1

B: Si la página tiene espacios disponibles, se inserta de forma ordenada. Si la página está llena, ordenada.

Si la página está llena, se divide en 2 y se promueve a nivel superior el valor medio.

B+: Si la página tiene espacios disponibles, se inserta de forma ordenada. Si la página está llena, se divide en 2 y se hace una copia de la clave en el nivel superior.

2.5.4 FORMA DE ALMACENAMIENTO DE CLAVES

AVL y Rojo Negro: cada clave es un nodo.

B y B +: La página puede contener diferentes claves.

2.5.5 EFICIENCIA

- AVL es más rápido en la busqueda pero menos profundo que el Rojo Negro
- Rojo Negro es más rápido en eliminación e inserción por tener menos rotaciones.
- B + es más eficiente en la búsqueda que B por tener una copia de la clave en el nivel superior y en las hojas.

3 REFERENCIAS

[1]

(https://ccia.ugr.es/~jfv/ed1/tedi/cdrom/docs/arb_B.htm, 2023)

[2] (Gurin, 2023)

[3] (Estructura de Datos, 2023)

[4] Fuente especificada no válida.

[5] (Blancarte, 2023)

[6]
https://www.infor.uva.es/~cvaca/asigs/doceda/rojonegro.pdf