Proyecto 1

Arboles binarios

Investigación y reporte del proyecto 1:
Arboles binarios de búsqueda balanceados (AVL),
Arboles de expresiones aritméticas y Heaps

Alumnos:
Hernández Ruiz de Esparza Guillermo
Romero Pizano Christian Gustavo

Materia: Estructura de datos y algoritmos II. 1317
Profesor: Edgar Tista Gracia
Semestre 2023-1
5 de diciembre de 2022

Índice

1. Objetivo

Que el alumno implemente aplicaciones relacionadas con los árboles binarios y que desarrolle sus habilidades de trabajo en equipo y programación orientada a objetos.

2. Introducción

Como ya vimos en clase, los árboles binarios son estructuras de datos no lineales donde se nos permite almacenar información en nodos de manera ordenada. La ventaja más fuerte que tienen respecto a otras estructuras de datos es que los algoritmos de sus operaciones son demasiado eficientes.

Los árboles de búsqueda binarios son una estructura de datos fundamental utilizada para construir estructuras de datos más abstractas, como conjuntos, conjuntos múltiples y arreglos asociativos.

Podemos encontrar diversas variantes de este tipo de árbol, las cuales a su vez cuentan con diferentes propiedades que los distinguen del resto. Durante el desarrollo de este proyecto vamos a investigar un poco más sobre 3 variantes de árboles binarios de búsqueda.

3. Investigación

3.1. Árboles AVL

3.1.1. Introducción

Árbol AVL		
Estructura	Árbol	
Creación	1962	
Inventores	Georgy Andelson y Evgenii Landis	
Tiempo de complejidad en notación "Big O"		
	Promedio	**Peor Caso**
Recorrido	O(log n)	O(log n)
Búsqueda	O(log n)	O(log n)
Inserción	O(log n)	O(log n)
Eliminación	O(log n)	O(log n)

Los árboles AVL reciben este nombre en 1962 cuando los dos inventores soviéticos, G.M. Adelson-Velskii y E.M. Landis publicaron su artículo "Un algoritmo para la organización de la información".

Un árbol AVL es un árbol binario de búsqueda autobalanceado, y como tal fue la primera estructura de datos en ser inventada. En un árbol AVL, la altura de los dos hijos de la raíz (subárboles) difieren como máximo por uno, por lo que tras cada operación dentro del árbol se trendrá que verificar que siga balanceado. Tanto el recorrido, como la inserción y la eliminación tienen un tiempo de complejidad O(log n) para cualquier caso, donde n es la cantidad de nodos en el árbol que se encuentran involucrados en la operación. Puede que durante la inserción y la eliminación se requieran algunas 'rotaciones' para balancear al árbol.

Se define como el "factor de equilibrio" de un nodo al valor de la altura de su subárbol izquierdo menos la altura de su subárbol derecho; Se considera como balanceado a un nodo con un factor de equilibrio de 1, 0 o -1. Un nodo con un diferente factor de equilibrio se considera desbalanceado y requiere un reacomodo en el árbol a través de las ya mencionadas rotaciones.

Los árboles AVL a menudo se comparan con los árboles red-black por la similitud entre sus operaciones y su tiempo de ejecución. Debido a que los árboles AVL cuentan con un equilibrio más rígido, son más rápidos que los árboles red-black para aplicaciones de búsqueda intensiva, sin embargo los árboles red-black son más rápidos en la inserción y en la eliminación.

3.1.2. Operaciones

Las operaciones básicas para un árbol AVL involucran realizar las mismas acciones como si se tratase de un árbol binario desbalanceado, pero se añaden las operaciones conocidas como "rotaciones de árbol", que ayudan a restaurar el factor de equilibrio de los subárboles.

3.1.3. Búsqueda/Recorrido

La búsqueda en un árbol AVL se lleva a cabo exactamente igual que en un árbol de búsqueda desbalanceado. No es necesario realizar alguna acción en particular debido a que la estructura del árbol no se ve modificada durante la búsqueda.

Si cada nodo registra el tamaño de sus subárboles (incluyéndose a sí mismo) entonces los nodos pueden ser recuperados en un tiempo de O(log n) a través de índices. Una vez que un nodo haya sido encontrado dentro del árbol, sus nodos adyacentes se pueden recuperar bajo un tiempo de ejecución constante.

3.1.4. Inserción

Después de insertar un nodo es necesario verificar que cada uno de sus "ancestros" mantengan la estructura de un árbol AVL. Si el factor de equilibrio se mantiene dentro

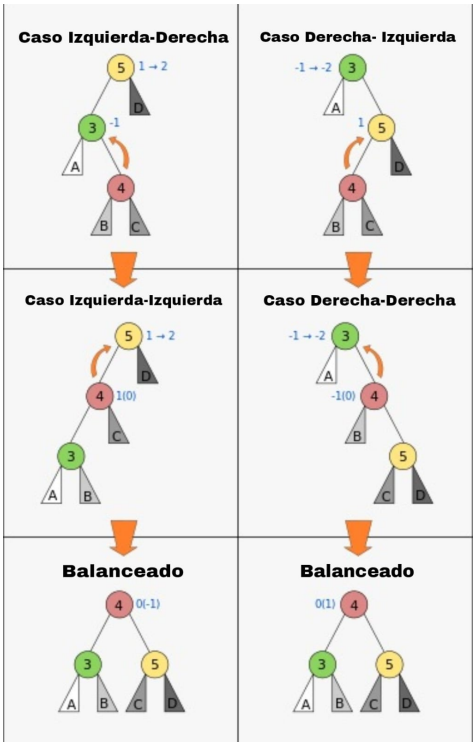


Figura 1: Descripción gráfica de las rotaciones.

del rango [-1,1] para cada nodo registrado, entonces no es necesaria ninguna rotación. Sin embargo, en caso de que el factor se convierta en ±2, entonces el subárbol enlazado a este nodo se encuentra desbalanceado.

Si las inserciones se realizan en serie, después de cada inserción, como máximo uno de los siguientes 4 casos necesita ser aplicado para restaurar el árbol AVL, donde dos de ellos son simétricos a los otros dos. Sea P la raíz del árbol desbalanceado, con R y L que se refieren a los hijos izquierdo y derecho respectivamente, entonces

Casos “Izquierda-Izquierda” e “Izquierda-Derecha”:

-Si el factor de equilibrio de P es +2, entonces el subárbol izquierdo pesa más que el subárbol derecho del nodo dado, por lo que debemos checar el factor de equilibrio de L. Es requerida la rotación a la derecha con P.

-Caso izquierda-izquierda: Si el factor de equilibrio de L es+1, se realiza una rotación a la derecha.

-Caso izquierda-derecha: Si el factor de equilibrio de L es -1, se requieren 2 rotaciones diferentes. La primera es una rotación izquierda con L como la raíz, y la segunda es una rotación derecha con P como la raíz.

Casos “Derecha-Derecha” y “Derecha-Izquierda”:

-Si el factor de equilibrio de P es -2, entonces el subárbol derecho pesa más que el subárbol izquierdo del nodo dado, por lo que es necesario checar el factor de equilibrio de R. Es necesaria la rotación a la izquierda con P.

-Caso derecha-derecha: Si el factor de equilibrio de R es -1, se requiere una rotación.

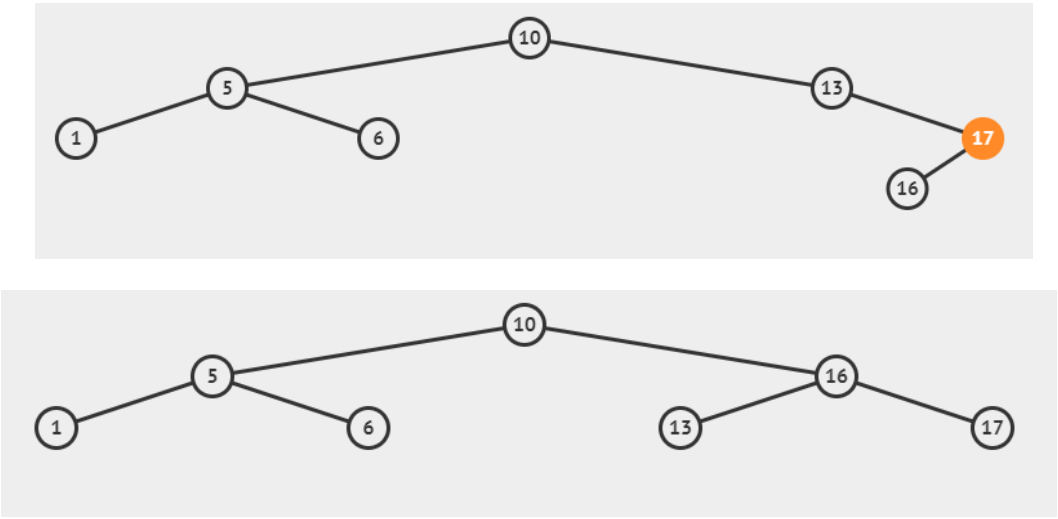


Figura 2: Ejemplo del caso Izquierda-Derecha

-Caso derecha-izquierda: Si el factor de equilibrio de R es +1, se requieren 2 rotaciones diferentes La primera es una rotación derecha con R como la raíz. La segunda es una rotación izquierda con P como la raíz.

3.1.5. Eliminación

Si el nodo es una hoja o tiene únicamente a un hijo, simplemente se elimina. De otro modo,se reemplaza ya sea con el nodo más hasta la derecha del subárbol izquierdo. o con el nodo más a la izquierda del subárbol derecho. Después de la eliminación, se van analizando los nodos desde el nodo que reemplazó al nodo eliminado hasta la raíz, mientras se van ajustando los factores de equilibrio.

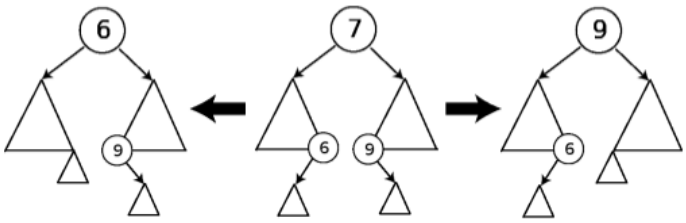


Figura 3: Eliminación del nodo 7.

Si el factor de equilibrio del árbol es 2 y el del subárbol izquierdo es 0, se debe realizar una rotación en P. Lo mismo ocurre en un caso “inverso”.

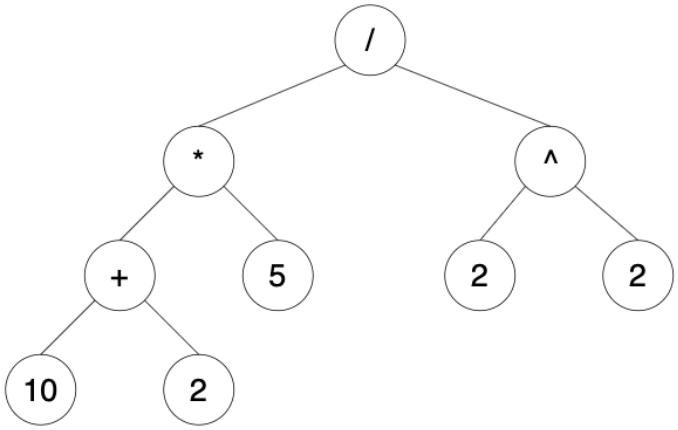
El recorrido para el análisis de puede evitar si el factor de balance es un ± 1, indicando que la altura del subárbol se ha mantenido intacta. Si el factor es 0, entonces la altura del subárbol disminuyó en uno y se necesita seguir con el análisis.

Si el factor se convierte en un ± 2, entonces el subárbol se encuentra desbalanceado por lo que es necesaria una rotación para arreglarlo. Si después de la rotación el factor es un 0, entonces el recorrido hasta la raíz continúa.

3.2. Árbol de expresión aritmética

La implementación de arboles binarios para resolver expresiones aritméticas surge debido al problema que representa trabajar con paréntesis para un algoritmo. Se busca representar una expresión como un árbol binario manteniendo la jerarquía de operaciones. Para cumplir con la jerarquía cada sub-árbol debe de ser evaluado por completo antes de evaluar al nodo. Cada nodo intermedio representa una operación y las hojas son los valores que serán evaluados, en estos arboles se pueden representar operaciones binarias (+, -, $\cdot$, /, $\wedge$) y de un solo operando (funciones trigonométricas, log, ln, $\sqrt[n]{x}$). En todos los árboles de expresiones aritméticas la raíz es la ultima operación que se va a realizar, esto aplica también para los nodos intermedios que tengan operaciones como hijos.

Ejemplo: la expresión $(10 + 2 \cdot 5)/\,(2^2)$ se representaría con el siguiente árbol:



Primero se debe realizar 2^2 y almacenar su resultado en B, después $2 \cdot 5$ y almacenar su resultado en C, posteriormente realizar $10 + C$ y almacenar su resultado en A como ultimo paso se realiza A/B y este sera el resultado de la expresión.

3.2.1. Notación polaca inversa

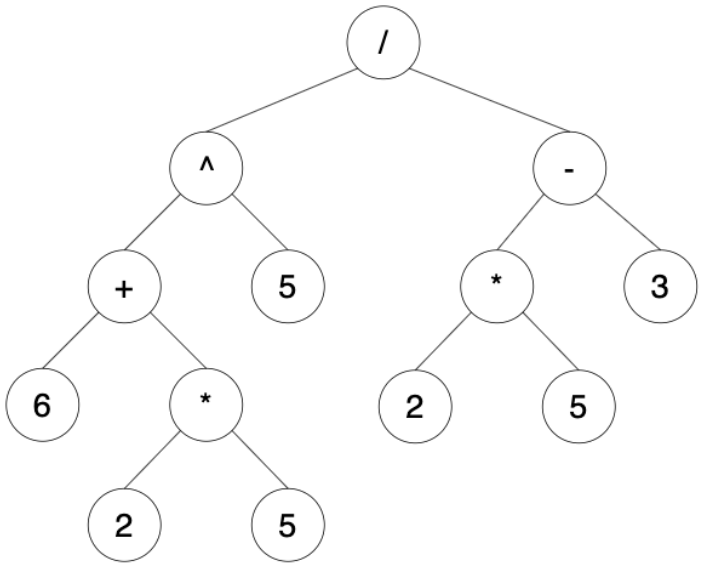
La notación polaca inversa es una variación de la notación polaca, propuesta en 1920 por Jan Lucasiewicz, esta variante fue desarrollada por Charles L. Hamblin en 1950. En la notación polaca los operandos van después del operador, mientras que en la notación polaca inversa los operandos van antes que el operador. La representación de la operación $a + b$ en notación polaca es: $+ab$, en notación polaca inversa: $ab+$. Una característica de la notación polaca inversa es que no necesita paréntesis, por esta razón es tan utilizada dentro de sistemas computacionales. La expresión: $(3 + 4) \cdot 5$ se representa como 3 4 + 5 $\cdot$.

Para obtener la notación polaca inversa a partir de un árbol de expresión aritmética se debe realizar el recorrido en post orden del árbol, ya que se termino el recorrido el resultado se debe leer de derecha a izquierda para obtener la notación polca inversa.

Para evaluar una expresión que esta representada en notación polaca inversa es necesario utilizar una pila para almacenar los números. Se debe visitar cada posición de la

notación polaca empezando desde el índice 0 hasta el ultimo elemento, dependiendo del valor se realizaran distintas operaciones. Si el valor es un número se agrega a la pila de valores, si es un operador se extrae la cantidad necesaria de números de la pila para evaluarlos y el resultado se agrega a la pila.

Ejemplo: la expresión $((6 + 2 \cdot 5)^5/(2 \cdot 5 - 3)$ se representaría con el siguiente árbol:



Al recorrer en post orden el árbol se obtiene la expresión: / - 3 · 5 2 ∧ 5 + · 5 2 6 esto se debe leer de derecha a izquierda para obtener la notación polaca inversa el resultado es: 6 2 5 · + 5 ∧ 2 5 · 3 - /. La evaluación de esta expresión se realiza de la siguiente manera:

6 2 5 · + 5 ∧ 2 5 · 3 - / · + 5 ∧ 2 5 · 3 - / + 5 ∧ 2 5 · 3 - / 5 ∧ 2 5 · 3 - /

Pila → | **Pila** | → | **Pila** | → | **Pila** | →

6
2
5

6
10

16

∧ 2 5 · 3 - / 2 5 · 3 - / · 3 - / 3 - / - /

Pila
16
5

→ | **Pila** | → | **Pila** | → | **Pila** | → | **Pila** | →

1048576

Pila
1048576
2
5

Pila
1048576
10

Pila
1048576
10
3

/ Resultado

Pila
1048576
7

→ | **Pila** |
|---|
| 149796.5714 |

3.2.2. Construcción del árbol de expresión aritmética

Para pasar de la notación infija de una expresión (representación tradicional) a un árbol se debe recorrer la expresión y seguir los siguientes pasos:

1. Crear un árbol con raíz vacía y posicionarse en la raíz
2. Si el valor es un paréntesis de apertura agregar al nodo actual un hijo izquierdo y moverse a ese hijo
3. Si el valor es un operador asignar al nodo actual el operador, agregar un hijo derecho y moverse a ese hijo
4. Si el valor es un numero asignar al nodo actual en número y regresar al padre, si el padre no es un operador binario regresar al padre del operador
5. Si el valor es un paréntesis de cierre regresar al padre, si el padre no es un operador binario regresar al padre del operador

Para verificar que el árbol se construyo correctamente al terminar de leer la expresión el nodo actual debe ser null.

3.3. Algoritmos de heaps

Describe what you did in order to use the data for analysis.

4. Implementación

4.1. Arboles AVL

Para la implementación de este tipo de árbol fue necesario modificar las operaciones básicas de un árbol binario de búsqueda normal, esto debido a que son estructuras demasiado similares, sin embargo ahora se tienen que tomar en cuenta las rotaciones que hace el árbol para mantenerse balanceado.

Tanto para la inserción como para la eliminación, se van descartando los diferentes casos con los que nos podemos encontrar en un árbol binario de búsqueda (Ya sea que el nodo que vamos a insertar/eliminar sea raíz, subárbol u hoja además de sus hijos), y una vez determinado el factor de equilibrio de cada nodo verificamos que el árbol se encuentre balanceado, de lo contrario se realizará la rotación correspondiente según sea el caso. Esta parte de la implementación fue la más extensa debido a la enorme cantidad de casos que nos podemos encontrar durante las operaciones.

Para el recorrido y la búsqueda se utilizó la búsqueda por expansión debido a su fácil implementación y efectividad.

4.2. Árbol de expresión aritmética

Para la implementación de este árbol utilice la clase nodo que vimos en clase con algunas modificaciones, la clase ExpAritmetica hereda de ArbolBin. Para manejar la

expresión cree una nueva clase donde se lee, almacena y verifica la expresión ingresada por el usuario.

Para la construcción del árbol seguí los pasos mostradas en la investigación, su implementación esta hecha con pilas. Para evaluar la expresión se almacenan los datos de la notación polaca inversa en una pila. Todas las posibles excepciones están controladas y tienen un mensaje determinado para el usuario donde se especifica el error. Al momento de recorrer la notación se puede encontrar un valor numérico que se almacena en la pila o es una expresión que se evalúa utilizando un switch.

5. Conclusiones

Hernández Ruiz de Esparza Guillermo

Desarrollar este proyecto represento un reto complicado, el planear un trabajo en equipo siempre conlleva dificultades. La complejidad de las actividades estuvo al nivel que hemos trabajado tanto en practicas de laboratorio como en clase. En mi caso me dedique a desarrollar el modulo de árboles de expresiones aritméticas, hacer esto fue muy interesante por el proceso de investigación que fue necesario. En este caso decidí implementar operaciones no binarias, esto fue un reto ya que no encontré bibliografía al respecto sobre la forma de codificarlo.

Estoy convencido que de haber tenido una mejor organización pudimos haber entregado un mejor trabajo. También me llevo como experiencia negativa la situación con el compañero que abandono el equipo y esto me enseño que uno debe de tener en cuenta que estas situaciones pueden ocurrir.

Utilizar LaTeXfue un reto muy agradable y me gustaría seguir aprendiendo a usarlo.

En cuanto al aprendizaje sobre temas relacionados al proyecto manejar distintos tipos de arboles me ayudo a acentar los conocimienntos que tenia sobre este tema.

Romero Pizano Christian Gustavo

El desarrollo de este proyecto fue demasiado complejo y la dificultad de las actividades propuestas fue un tanto elevada, sin embargo cumplen su función de servir como complemento a los temas vistos en clase debido a que se siente como una extensión del tema de árboles.

El proyecto pudo haber sido llevado de una forma más tranquila con una mejor organización dentro del equipo, pero a pesar de las adversidades que nos fuimos encontrando la mayoría de las partes del proyecto quedaron terminadas en su totalidad y funcionan adecuadamente.

Visto desde un punto de vista más teórico, los temas analizados durante este proyecto fueron más sencillos de entender, teniendo un contraste con la parte de implementación del código puesto que fue la parte más extensa y complicada del proyecto.

Como resultado aprendimos a trabajar con otros tipos de árboles con características peculiares que pueden ser aplicados en casos específicos pero que cada uno cuenta con

sus particulares ventajas.

6. Referencias

-Robert Sedgewick, Algoritmos, Addison-Wesley, 1983, ISBN 0-201-06672-6, página 199, capítulo 15: Arboles Balanceados.

-Adelson-Velskii, G.; E. M. Landis (1962). Ün algoritmo para la organización de la información". Procedente de la Academia de Ciencias de la USSR. (Ruso) Traducción al inglés por Myron J. Ricci, 1962.

-Pfaff, Ben (Junio 2004). .ªnálisis de rendimiento de los árboles binarios de búsqueda en software del sistema"(http://www.stanford.edu/ blp/papers/libavl.pdf) (PDF). Stanford University.

- Redziejowski, R. R. (1969). On arithmetic expressions and trees. Communications of the ACM, 12(2), 81-84. https://doi.org/10.1145/362848.362859

- Reverse Polish Notation – from Wolfram MathWorld. (s. f.). https://mathworld.wolfram.com/Reverse

- Universitat Politècnica de València - UPV. (2018, 28 marzo). S4.11 Notación polaca. — — UPV [Vídeo]. YouTube. https://www.youtube.com/watch?v=uFRRCWZUItI