Trabajo Integrador – Datos avanzados

Árbol binario con listas

Alumnos:

- Pedro Martin Torres – pedro.m.torres@gmail.com
- Agustin Trigo – a_trigo295@hotmail.com

Enlaces:

- Repositorio GitHub: https://github.com/pedro-m-torres/tp-p1-arboles
- Video: https://www.youtube.com/watch?v=AcZyNZrZymk

Índice:

Introducción:

Los árboles son una de las estructuras de datos más fundamentales y poderosas en informática. Son utilizados en una amplia variedad de aplicaciones del mundo real, como la organización de archivos en sistemas operativos, la gestión de bases de datos (por ejemplo, en árboles de búsqueda binaria), y en algoritmos de optimización.

El concepto de jerarquía o estructura ramificada que representan los árboles es común en diversas disciplinas, desde la biología (para representar la evolución de especies) hasta la teoría de redes (como en la organización de conexiones entre nodos).

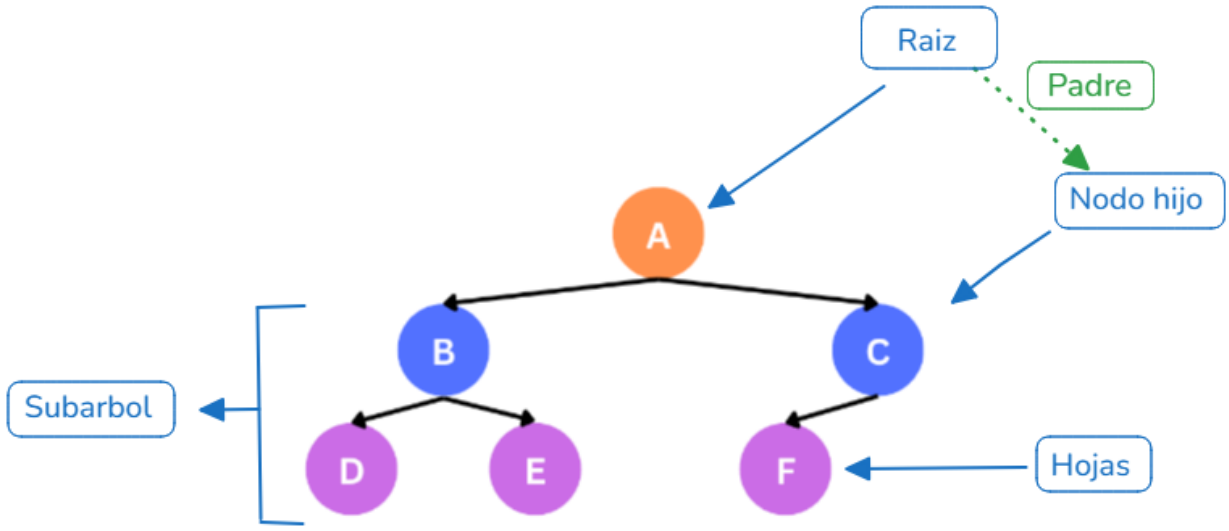
Entender cómo funcionan los árboles es esencial para mejorar la eficiencia de los algoritmos y la optimización de recursos, como el tiempo de ejecución y la memoria.

Marco teórico:

Elementos de un árbol:

Los árboles son representados, en general, a través de nodos conectados entre sí por ramas. Un nodo es una unidad básica de un árbol. Cada nodo contiene dos componentes principales: un valor y una rama a sus nodos hijos, si es que los tiene. Los nodos en un árbol están conectados entre sí mediante estas referencias, formando una estructura jerárquica.

- **Raíz:** es el punto e entrada a la estructura, generalmente ubicado en la parte más superior del árbol. No tiene un nodo padre y es único, es decir, cada árbol tiene un solo nodo raíz.
- **Hojas:** es cualquier nodo que no tenga hijos, es decir, un nodo terminal del árbol. Los nodos hoja están en los extremos de la estructura de un árbol y no tienen más ramificaciones debajo de ellos.
- **Altura:** número de niveles desde la raíz hasta la hoja más profunda.
- **Subárboles:** cualquier nodo junto a sus descendientes forma un subárbol.



Propiedades:

- **Longitud:** es el número de ramas que hay que transitar para llegar de un nodo a otro.
- **Profundidad:** es la longitud de camino entre el nodo raíz y un nodo.
- **Nivel:** es la longitud del camino que lo conecta al nodo raíz más uno. Un nivel puede contener uno o más nodos.
- **Altura:** es el máximo nivel del árbol.
- **Grado:** es el número de hijos que tiene dicho nodo.
- **Orden:** es la máxima cantidad de hijos que puede tener cada nodo.
- **Peso:** es el número total de nodos que tiene un árbol.

Árbol binario:

Es un árbol en el que cada nodo puede tener, como máximo, 2 hijos. Dicho de otra manera, es un árbol grado dos. En un árbol binario el nodo de la izquierda representa al hijo izquierdo y el nodo de la derecha representa al hijo derecho.

Existen varias formas de recorrer un árbol. Las tres formas más comunes comunes son preorden, inorden y postorden.

Preorden

El recorrido preorden es útil cuando necesitas realizar alguna operación en el nodo antes de procesar a sus hijos. Un caso típico es cuando estás copiando un árbol o clonando su estructura.

Para recorrer un árbol binario no vacío en preorden se ejecutan los siguientes pasos:

1. Se comienza por la raíz.
2. Se baja hacia el hijo izquierdo de la raíz.
3. Se recorre recursivamente el subárbol izquierdo.
4. Se sube hasta el hijo derecho de la raíz.
5. Se recorre recursivamente el subárbol derecho.

Inorden

El recorrido inorden es especialmente útil para los árboles de búsqueda binaria, ya que garantiza que los nodos se visiten en orden ascendente.

Para recorrer un árbol binario no vacío en inorden se ejecutan los siguientes pasos:

1. Se comienza por el nodo hoja que se encuentre más a la izquierda de todos.
2. Se sube hacia su nodo padre.
3. Se baja hacia el hijo derecho del nodo recorrido en el paso 2.
4. Se repiten los pasos 2 y 3 hasta terminar de recorrer el subárbol izquierdo.
5. Se visita el nodo raíz.
6. Se recorre el subárbol derecho de la misma manera que se recorrió el subárbol izquierdo.

Postorden

El recorrido postorden es útil cuando quieres realizar alguna acción en los nodos hijos de un nodo antes de procesar el nodo mismo. Esto es común cuando se trabaja con estructuras jerárquicas donde necesitas procesar primero las subestructuras.

Para recorrer un árbol binario no vacío en postorden se ejecutan los siguientes pasos:

1. Se comienza por el nodo hoja que se encuentre más a la izquierda de todos.
2. Se visita su nodo hermano.
3. Se sube hacia el padre de ambos.
4. Si tuviera hermanos, se visita su nodo hermano.
5. Se repiten los pasos 3 y 4 hasta terminar de recorrer el subárbol izquierdo.
6. Se recorre el subárbol derecho, comenzando por el nodo hoja que se encuentre más a la izquierda y siguiendo el mismo procedimiento que con el subárbol izquierdo
7. Se visita el nodo raíz.

Árboles binarios de búsquedas

Los árboles binarios de búsqueda son árboles binarios donde para cada nodo del árbol se cumple que:

- Los valores de todos los nodos en su subárbol izquierdo son menores que el valor del nodo.
- Los valores de todos los nodos en su subárbol derecho son mayores que el valor del nodo.

Caso práctico:

Para este caso desarrollamos el concepto de árbol binario de búsqueda, pero utilizando listas anidadas.

Con valores numéricos ingresos por el usuario, por ejemplo, un determinado puntaje. Creamos un árbol dinámicamente a partir de estos datos para luego realizar las operaciones de búsqueda y visualización.

Además, el programa cuenta con la posibilidad de recorrer el árbol de las tres maneras posibles: preorden, postorden e inorden. Y extraer sus propiedades: altura, peso y profundidad.

Metodología:

- A partir de datos ingresados por usuario la función "*crear_arbol()"* va ingresando los valores a la lista vacía inicializada anteriormente.
- La función "*crear_arbol()*" ejecuta internamente a la función "*ingresar_datos()*" la cual determina si los valores deben ir hacia la izquierda o derecha según corresponda.
- Importamos el módulo que carga el árbol para luego, a partir de este generar las otras funcionalidades.
- Estas funcionalidades se manipulan a través de un menú claro con opciones.

Resultados obtenidos y conclusión:

El desarrollo nos permitió trabajar con esta estructura de datos de una manera más sencilla. Los conceptos tales como recursividad y listas fueron de gran utilidad al momento implementar las diferentes funcionalidades y para el entendimiento del tema en sí.

Bibliografía:

Material campus virtual - https://tup.sied.utn.edu.ar/course/view.php?id=12&section=67