Universidad del Norte

Departamento de Ingeniería de Sistemas y Computación

Estructura de Datos II

Laboratorio 1

Prof. Sebastian Racedo

**Fecha de entrega límite: 6 de Septiembre, 12:00 (estandar de 24 horas)**

1. (2.0) Debe construir un árbol que se autobalancee cada vez que se realice una acción, conforme lo requiera dicha acción. Concretamente este árbol debe poder realizar las siguientes operaciones:
   1. Insertar un nodo (opcional, hacer validación).
   2. Eliminar un nodo.
   3. Buscar un nodo (Queda libre como devolver si el nodo existe).
   4. Recorrido en orden por nivel (De forma recursiva).
   5. La altura dada un nodo (Recordar que la altura de un nodo es definida como el número de aristas más larga desde el nodo hasta un nodo hoja)
   6. Encontrar el abuelo o tío de un nodo dado.

Nota: Puede asumir que no se ingresarán valores repetidos, pues estos implicarán una mayor complejidad en la solución. Además, puede asumir un árbol inicial o construirlo desde 0. El código debe estar documentado y debe ser de autoría propia de los integrantes del grupo.

Para puntaje extra: Investigar e implementar una interfaz gráfica donde se puedan apreciar los resultados.

1. (2.0) Investigar, plantear y resolver un problema en el que se use árboles binarios. No tiene que ser un problema de autoría propia (aunque sería bastante fructífero para ustedes). Puede ser un problema que encontraron y que les pareció de interés por sus intereses en la carrera. En este punto se deben realizar las siguientes actividades:
   1. Investigar, plantear y definir bien un problema en el que se use árboles binarios.
   2. Explicar porque este problema es de su interés.
   3. Resolverlo usando un lenguaje de programación. (Sin importar la autoría el código debe tener su documentación hecha por los integrantes del grupo)
2. (1.0) Investigar, plantear y resolver un ejemplo en el que se usen arboles B (o B+). El código debe estar documentado y el ejemplo debe ser de autoría propia de los integrantes del grupo.

Puede usar Java o Python para desarrollar este laboratorio. Mínimo dos integrantes, máximo tres.

Todo el código (ya sea un solo proyecto o múltiples proyectos) debe ser subido a un repositorio de GitHub (privado de preferencia).

**Citar cualquier cosa que no sea de su autoría según las reglas IEEE.**

**MAY THE FORCE BE WITH YOU!**

**SOLUCIÓN DE LA PARTE ESCRITA DEL LABORATORIO**

**Punto #2**

1. **Investigación de Arboles Binario**

**Definición**

El árbol binario se define como una estructura de datos de árbol con un máximo de 2 hijos. Como cada elemento de un árbol binario sólo puede tener 2 hijos, normalmente los llamamos hijo izquierdo y derecho.

**Representación de un árbol binario**

Un árbol binario está representado por un puntero al nodo superior del árbol. Si el árbol está vacío, el valor de la raíz es NULL.

El nodo del árbol binario contiene las siguientes partes

* Datos
* Puntero al hijo izquierdo
* Puntero al hijo derecho

**Operación básica en el árbol binario:**

* Insertar un elemento.
* Eliminación de un elemento.
* Búsqueda de un elemento.
* Recorrer un elemento.

**Aplicaciones del árbol binario:**

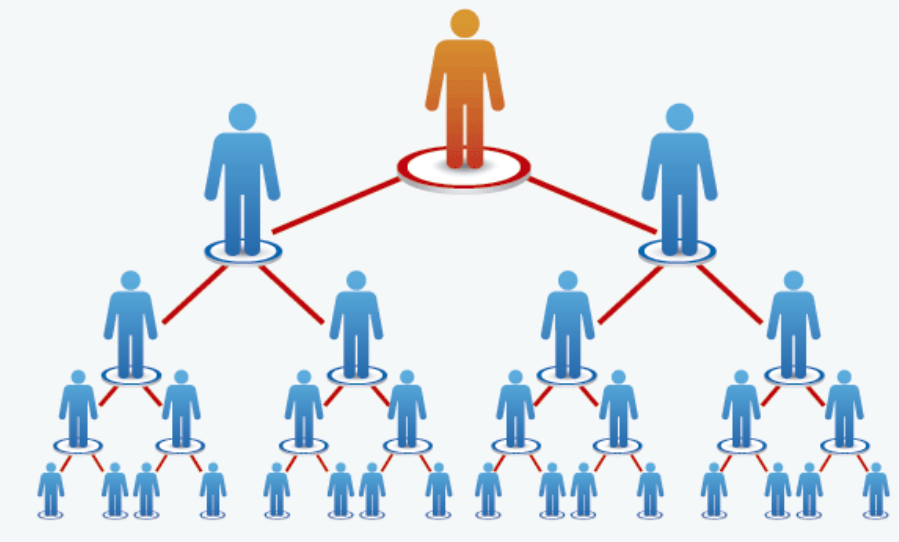
* En los compiladores se utilizan árboles de expresión, que es una aplicación del árbol binario.
* La cola de prioridad es otra aplicación del árbol binario que se utiliza para buscar el máximo o el mínimo en complejidad de tiempo O(1).

**Recorridos del árbol binario:**

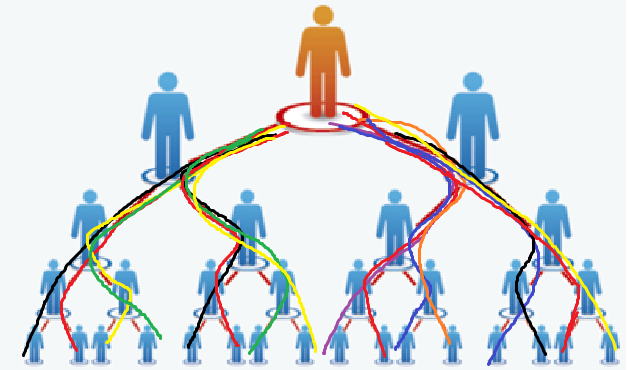
* **Pre-Order Traversal:** Aquí, el recorrido es: raíz - hijo izquierdo - hijo derecho. Esto significa que primero se recorre el nodo raíz, luego su hijo izquierdo y finalmente el hijo derecho.
* **InOrder Traversal:** Aquí, el recorrido es: hijo izquierdo - raíz - hijo derecho. Esto significa que primero se recorre el hijo izquierdo, luego el nodo raíz y finalmente el hijo derecho.
* **PostOrder Traversal:** Aquí, el recorrido es: hijo izquierdo - hijo derecho - raíz. Significa que primero se recorre el hijo izquierdo, luego el derecho y finalmente su nodo raíz.

**Planteamiento y definición del problema**

Una empresa multinivel organiza su sistema de personal en forma de árbol binario, de la siguiente forma:



Existe un vendedor supremo (color naranja) el cual distribuye e informa acerca del negocio a las demás personas, por ende, cada persona puede informar e incitar a lo más a dos personas para unirse a la compañía. Cada vendedor tiene la ruta de su “influencia”; por ejemplo: las rutas de influencia del vendedor que está encerrado en un círculo (en la imagen siguiente) son 8



La empresa desea buscar la ruta más influyente (ruta más larga) de un vendedor en específico; esto lo hará con respecto a su código de empleado. Además, necesita saber en qué nivel se encuentra el vendedor, los niveles inician en 0, siendo este el nivel de alta gama.

Se debe tener en cuenta lo siguiente:

* Suponga que el árbol ya está creado
* Cada nodo (persona) contiene la información de su código de empleado
* Usted decide como implementar la clase nodo, pero debe tener la información de su código como el data
* No asuma que es un perfect binary tree (como la imagen del ejemplo)

***Def:*** un árbol binario perfecto es un tipo de árbol binario en el que cada nodo interno tiene exactamente dos nodos hijes y todos los nodos hoja están en el mismo nivel

1. **¿Por qué este problema es de nuestro interés?**

Durante la investigación de árboles binarios y su estructura, nos dimos cuenta de las muchas aplicaciones que tienen los arboles binarios en la vida cotidiana y que muchas empresas lo usan para organizar una información determinada. Este problema nos pareció de nuestro interés porque es algo que es bastante comprensible para todos, ya que en algún momento de nuestra vida alguien nos ha incitado a unirnos a una empresa multinivel o empresa piramidal. Nos pareció que una empresa piramidal implementada en un árbol binario tiene muchas aplicaciones que son entendibles a simple vista; además, un algoritmo como el de “buscar las rutas de un nodo” sirve para desarrollar aún más una situación de interés para cualquier empresa multinivel.

1. **Solución**

La solución de este problema se resolvió en el lenguaje de programación java. En el Github se encuentra el codigofuente de este.

<https://github.com/EmilyRoldan/Lab1_EDDII.git>

**Punto #3**

1. **Investigación de Arboles B**

A menudo se usan árboles binarios de búsqueda para ordenar listas de valores, minimizando el número de lecturas, y evitando tener que ordenar dichas listas.

Pero este tipo de árboles tienen varias desventajas:

1. Es difícil construir un árbol binario de búsqueda perfectamente equilibrado.
2. El número de consultas en el árbol no equilibrado es impredecible.
3. Y además el número de consultas aumenta rápidamente con el número de registros a ordenar.

Para evitar estos inconvenientes se usan árboles-B, sobre todo cuando se ordenan ficheros, donde se ha convertido en el sistema de indexación más utilizado

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 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.

Un árbol-B de orden M (el máximo número de hijos que puede tener cada nodo) es un árbol que satisface las siguientes propiedades:

* + - Cada nodo tiene como máximo M hijos.
    - Cada nodo (excepto raíz y hojas) tiene como mínimo M/2 hijos.
    - La raíz tiene al menos 2 hijos si no es un nodo hoja.
    - Todos los nodos hoja aparecen al mismo nivel.
    - Un nodo no hoja con k hijos contiene k-1 elementos almacenados.
    - Los hijos que cuelgan de la raíz (r1, ···, rm) tienen que cumplir ciertas condiciones:

1.  El primero tiene valor menor que r1.

2.  El segundo tiene valor mayor que r1 y menor que r2, etc.

3.  El último hijo tiene valor mayor que rm.

1. **Planteamiento del problema**

Una empresa textil requiere organizar los pendientes de sus empleados para una mejor organización del personal. El personal se divide en los siguientes niveles: **nivel superior, nivel medio y nivel inferior,** los cuales contienen lo siguiente:

* En el nivel superior se encuentra la gerencia
* En el nivel medio se encuentran los supervisores
* En el nivel inferior se encuentran los empleados

Cada nivel contiene las tareas de cada clasificación del personal (gerencia, supervisores y empleados), las cuales aumentan de izquierda a derecha, con el fin de organizar quienes tienen más pendientes por hacer que otros. La empresa desea buscar cual es el empleado que tiene más pendientes por hacer, para darle un bono de descanso por su eficiencia laboral

Cabe destacar que la información se encuentra almacenada en un árbol B, teniendo en cuenta que:

* El grado mínimo del árbol es de 2, por ende, cada nodo soporta 4 apuntadores a sus descendientes y 3 claves, las cuales representan las tareas de cada grupo de personal (gerencia, supervisores y empleados)
* Cada nodo vendría siendo un grupo de personas, **correspondiendo a su nivel**. Es decir, si hay un nodo en el nivel medio, sería un nodo de un grupo de supervisores. Puede haber varios grupos (del mismo tipo) en cada nivel.
* Suponga que el árbol ya está creado

1. **Solución**

La solución de este problema se resolvió en el lenguaje de programación java. En el Github se encuentra el codigofuente de este.

<https://github.com/EmilyRoldan/Lab1_EDDII.git>

**Referencias**

<https://www.geeksforgeeks.org/binary-tree-set-1-introduction/>, 05/09/22

<https://sites.google.com/site/clasearbolesb/arboles-b>, 05/09/22