ESTRUCTURA DE DATOS

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN Y LA DECISIÓN

MONITOR: ALEJANDRO SALAZAR MEJÍA

alsalazarm@unal.edu.co

2023-2S

TALLER 9: ÁRBOLES Y ÁRBOLES BINARIOS DE BÚSQUEDA

Este taller tiene como objetivo reforzar tu comprensión de árboles, árboles binarios de búsqueda y los diferentes métodos para recorrerlos. Explorarás los árboles como una estructura de datos recursiva y practicarás la creación de algoritmos recursivos. **La recursión** es un enfoque poderoso en la resolución de problemas algorítmicos, y este taller te brindará la oportunidad de dominar esta técnica.

Los talleres son herramientas de estudio, y no tienen asignada una ponderación dentro de las evaluaciones del curso. Se recomienda desarrollar cada problema mediante la presentación de un seudocódigo, y en el caso de diseño de clase mediante diagramas de clase. Sin embargo, si el estudiante puede, solucionar cada problema empleando un lenguaje de programación de alto nivel.

NOTA: Para resolver los ejercicios usando seudocódigo, asuma que ya tiene la implementación de la estructura de datos con los métodos y características que considere necesarias.

1. El profesor George O'Jungle tiene un árbol binario de 27 nodos, en el que cada nodo está etiquetado con una única letra del alfabeto romano o el carácter &. Los recorridos preorden y postorden del árbol visitan los nodos en el siguiente orden:

- Preorder: I Q J H L E M V O T S B R G Y Z K C A & F P N U D W X
- Postorder: H E M L J V Q S G Y R Z B T C P U D N F W & X A K O I

Dibuje el árbol binario de George.

2. Recuerde, un árbol binario es ***completo*** si cada nodo interno tiene exactamente dos hijos

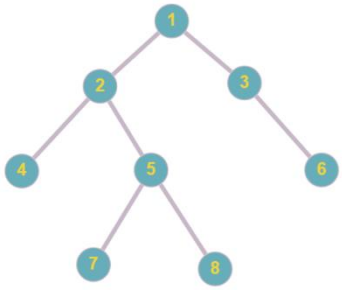
i. Describa y analice un algoritmo recursivo para reconstruir un árbol binario **completo** arbitrario, dadas las secuencias de sus nodos en preorden y postorden como entrada.
ii. Demuestre que no existe ningún algoritmo para reconstruir un árbol binario **arbitrario** a partir de sus secuencias de nodos en preorden y postorden. (***hint:*** basta dar un contraejemplo).

3. Dados dos nodos *U* y *V* de un árbol binario, determine si *U* es ancestro de *V* recursivamente.

4. Convierta un Árbol Binario dado en una Lista Doblemente Enlazada.

5. El algoritmo de **Recorrido Por Niveles** se define como un método para recorrer un árbol de forma que todos los nodos presentes en el mismo nivel se recorran por completo antes de recorrer el siguiente nivel. Este método también es conocido como *Búsqueda a lo Ancho (BFS).* Por ejemplo,

`Input:`



`Output: 1, 2, 3, 4, 5, 6, 7, 8`

Implemente dicho algoritmo. *(**hint:** use una cola como se muestra en la Lectura 11 para hallar el padre de un nodo)*

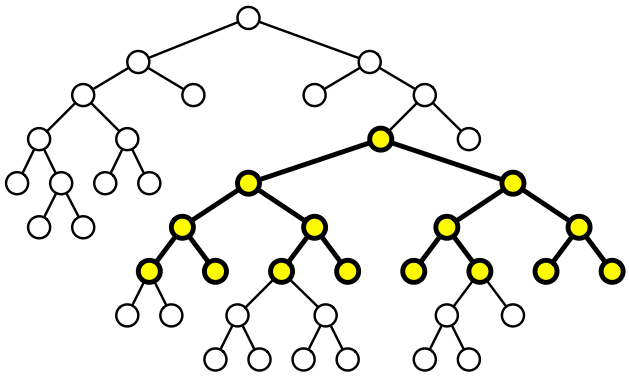
6. Dé un ejemplo de un Árbol Binario de Búsqueda *T* y de una llave *k* tal que `T.find(k)` corra en tiempo $\Omega(n)$, donde n es la cantidad de nodos en el árbol. Es decir, muestre que existen casos en los que el método `find` no es eficiente y recorre todo el árbol.

7. Un **subárbol** de un Árbol Binario se entiende como cualquier subgrafo conectado. En otras palabras, es como si tomaras un "trozo" más pequeño de un árbol binario, que a su vez, también es un árbol binario en sí mismo.

Un Árbol Binario es **perfecto** si cada nodo interno tiene dos hijos y cada hoja tiene exactamente la misma profundidad.

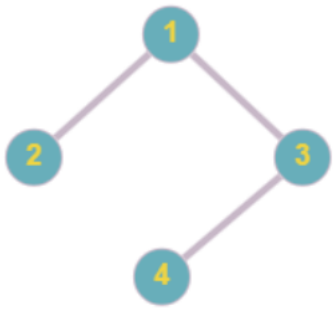
Describa y analice un algoritmo recursivo para calcular el subárbol perfecto más grande de un árbol binario dado. Su algoritmo debe devolver tanto la raíz como la profundidad de este subárbol.

Considere la siguiente figura que ilustra el subárbol perfecto más grande de un árbol dado:

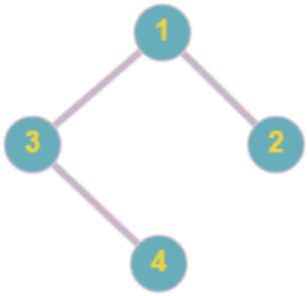


8. Dado un Árbol Binario, diseñe y analice un algoritmo recursivo para convertir dicho árbol binario en su **Árbol Espejo**. Imagine como si pusiera el Árbol Binario al frente de un espejo. Nos interesa conseguir la imagen del árbol en el espejo. Por ejemplo,

`Input:`

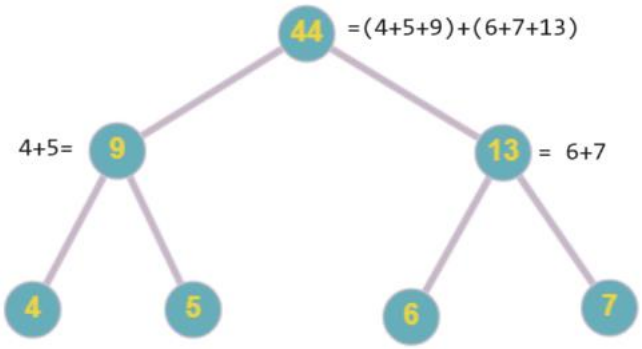


`Output:`



9. Dado un Árbol Binario, diseñe y analice un algoritmo recursivo que identifique si el árbol es un **Árbol de Sumas** o no. En un Árbol de Sumas, el valor de cada nodo interno es igual a la suma de todos los elementos presentes en su subárbol izquierdo y derecho. El valor de un nodo hoja puede ser cualquier cosa y el valor de un nodo hijo ausente se considera 0. Por ejemplo,

`Input:`



`Output: True`

10. Como he mencionado en numerosas ocasiones, la recursión es el método **más poderoso** para diseñar algoritmos. Lastimosamente, en la práctica puede llegar a ser bastante ineficiente. Por suerte, cualquier algoritmo recursivo se puede replantear de forma iterativa (aunque lograr esto puede ser difícil).

i. Dado un Árbol Binario, escriba un algoritmo iterativo que imprima el recorrido Preorden del árbol binario dado. ***(hint:*** *use una pila****)***
ii. Consulte algoritmos iterativos para imprimir los recorridos Inorden y Postorden de un Árbol Binario.