Recorrido de arboles binarios

Arturo Daza, Sebastian Samaniego, Giovanny Gonzalez

I. Introducción

Los árboles binarios son estructuras de datos fundamentales en ciencias de la computación y programación. Para comprender y manipular eficazmente los datos almacenados en estos árboles, es crucial entender los métodos de recorrido. El recorrido inorden explora primero el subárbol izquierdo, seguido del nodo actual y luego el subárbol derecho. En el recorrido preorden, el nodo actual se visita antes de los subárboles izquierdo y derecho, mientras que en el postorden, los subárboles izquierdo y derecho se visitan antes del nodo actual. Estos recorridos son esenciales para ordenar datos, evaluar expresiones matemáticas y realizar diversas operaciones en algoritmos y estructuras de datos. En este contexto, exploraremos cómo estos recorridos nos permiten entender y manipular los árboles binarios de manera eficiente.

II. Resultados

II-A. Ejercicios parte A

Ejercicio 1: Recorrido en Inorden - Supongamos que tenemos el siguiente recorrido en inorden: 4, 2, 5, 1, 6, 3, 7. Para construir el árbol a partir de este recorrido, primero debemos entender cómo funciona el recorrido en inorden. El recorrido en inorden visita el subárbol izquierdo, luego el nodo actual y, finalmente, el subárbol derecho. Para reconstruir el árbol, podemos seguir estos pasos:

1. El primer número en el recorrido inorden (4) será la raíz del árbol.

Correción - La raiz vendria siendo el numero 1 ya que es la mitad de la lista y segun la definición.

2. Los números antes de 4 (2 y 5) pertenecen al subárbol izquierdo de 4.

Corrección - Se toma la mitad del lado izquierdo de una que vendria siendo 2 por ende, ese es el indice del subarbol izquierda, el valor a la izquierda va a la izquierda y el de la derecha a la derecha.

3. Los números después de 4 (1, 6, 3 y 7) pertenecen al subárbol derecho de 4.

Corrección - Se toma la mitad del lado derecho de una que vendria siendo 3 por ende, ese es el indice del subarbol derecho, el valor a la izquierda va a la izquierda y el de la derecha a la derecha.

¿Cual sera el arbol resultante?:

Ejercicio 2: Recorrido en Preorden - Supongamos que tenemos el siguiente recorrido en preorden: 1, 2, 4, 5, 3, 6, 7. Para construir el árbol a partir de este recorrido, el proceso es similar:

1. El primer número en el recorrido preorden (1) será la raíz del árbol.

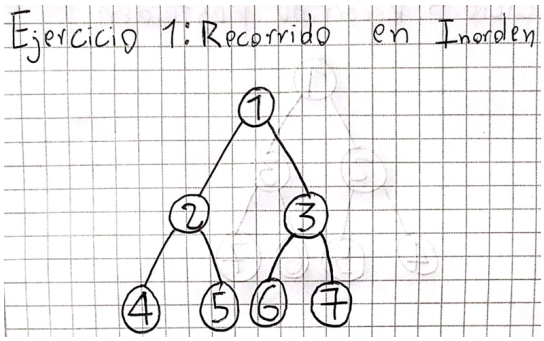

Figura 1: Gráfica de recorrido en inorden.

2. Los números después de 1 (2, 4 y 5) pertenecen al subárbol izquierdo de 1.

3. Los números después de 5 (3, 6 y 7) pertenecen al subárbol derecho de 1.

¿Cual sera el arbol resultante?:

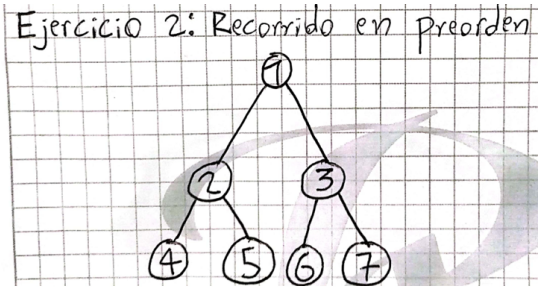

Figura 2: Gráfica de recorrido en preorden.

Ejemplo 3: Recorrido en Postorden Supongamos que tenemos el siguiente recorrido en postorden: 4, 5, 2, 6, 7, 3, 1. Para construir el árbol a partir de este recorrido, sigue estos pasos:

1. El último número en el recorrido postorden (1) será la raíz del árbol.

2. Los números antes de 1 (4, 5, 2, 6, 7, 3) se dividen en dos partes: los que pertenecen al subárbol izquierdo y los que pertenecen al subárbol derecho.

3. Los números después de 1 (3, 7, 6, 2, 5, 4) son los que pertenecen al subárbol izquierdo.

¿Cual sera el arbol resultante?:

II-B. Ejercicios Parte B

Supongamos que tenemos el siguiente árbol binario:

Entregue las respuestas de recorrido preorden, postorden e inorden teniendo como apoyo el siguiente codigo, Se suguiere el completar la implementacion para validar la respuesta.

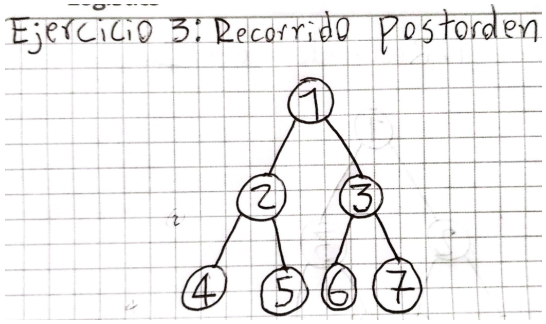

Figura 3: Gráfica de recorrido en postorden.

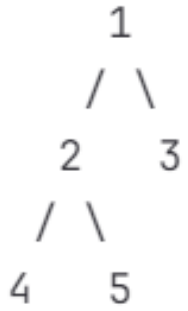

Figura 4: Arbol binario parte B.

Recorrido en Preorden

Recorrido en Preorden (Raíz - Izquierda - Derecha): 1, 2, 4, 5, 3

Recorrido en Inorden

Recorrido en Inorden (Izquierda - Raíz - Derecha): 4, 2, 5, 1, 3

Recorrido en Postorden

Recorrido en Postorden (Izquierda - Derecha - Raíz): 4, 5, 2, 3, 1

III. Analisis

1. Ordenamiento y Búsqueda: Los recorridos en árboles binarios son fundamentales para el ordenamiento y la búsqueda eficiente de datos. Por ejemplo, el recorrido inorden en un árbol binario de búsqueda produce una secuencia ordenada de elementos. Esto se utiliza en algoritmos de ordenamiento y para realizar búsquedas eficientes.

2. Evaluación de Expresiones Matemáticas: Los recorridos en árboles binarios son esenciales para evaluar expresiones matemáticas complejas. Al convertir una expresión matemática en un árbol de expresión y luego realizar un recorrido preorden (también conocido como notación polaca), podemos evaluar la expresión de manera eficiente.

3. Optimización de Algoritmos: En muchos algoritmos y estructuras de datos, el orden en que se acceden los datos es crucial para la eficiencia. Los recorridos en árboles binarios se utilizan para organizar datos de manera que se puedan acceder y manipular de manera eficiente, lo que es esencial en aplicaciones que requieren un alto rendimiento.

```
def preorden(node):
    if node is not None:
        print(node.value)  # Visita el nodo actual
        preorden(node.left)  # Recorre el subárbol izquierdo
        preorden(node.right)  # Recorre el subárbol derecho

# Llamamos a la función de recorrido en preorden desde la raíz del árbol
arbol = Nodo(1)
arbol.left = Nodo(2)
arbol.right = Nodo(3)
arbol.left.left = Nodo(4)
arbol.left.right = Nodo(5)

preorden(arbol)
```

Figura 5: Codigo de recorrido en preorden.

```
def inorden(node):
    if node is not None:
        inorden(node.left)  # Recorre el subárbol izquierdo
        print(node.value)  # Visita el nodo actual
        inorden(node.right)  # Recorre el subárbol derecho

# Llamamos a la función de recorrido en inorden desde la raíz del árbol
inorden(arbol)
```

Figura 6: Codigo de recorrido en inorden.

IV. Conclusiones

En base a los análisis de los ejercicios de la parte A y B, podemos concluir lo siguiente:

1. Los recorridos en árboles binarios, ya sean inorden, preorden o postorden, son herramientas cruciales en el campo de la informática para entender y manipular estructuras de datos complejas. A través de los ejercicios realizados, hemos obtenido resultados notables.

2. En los Ejercicios de la Parte A, hemos observado un fenómeno interesante: los tres tipos de recorridos dieron como resultado el mismo árbol binario. Esto subraya la propiedad única de los recorridos en árboles binarios: aunque se utilicen diferentes estrategias para visitar los nodos, el orden específico de visitar los nodos determina la estructura del árbol final. Este hecho es esencial en diversas aplicaciones, desde la evaluación de expresiones matemáticas hasta la construcción de estructuras de datos complejas.

3. En los Ejercicios de la Parte B, hemos observado que los recorridos preorden, inorden y postorden han producido árboles binarios diferentes. Este resultado resalta otra característica fundamental de los recorridos en árboles binarios: el orden en el que se visitan los nodos afecta directamente la estructura resultante del árbol. Los árboles obtenidos en estos ejercicios ilustran cómo diferentes secuencias de recorrido pueden generar árboles con organizaciones distintas.

V. Referencias

1. Recorrido de árboles", Wikipedia, La enciclopedia libre. [En línea]. Disponible: https://es.wikipedia.org/wiki/Recorrido$_de$

```
def postorden(node):
    if node is not None:
        postorden(node.left)  # Recorre el subárbol izquierdo
        postorden(node.right)  # Recorre el subárbol derecho
        print(node.value)  # Visita el nodo actual

# Llamamos a la función de recorrido en postorden desde la raíz del árbol
postorden(arbol)
```

Figura 7: Codigo de recorrido en postorden.

2. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Ïntroduction to Algorithms", 3rd ed., The MIT Press, 2009.