Recorrido de Árboles Taller 11

Andrés Salcedo Vera[1], Eder David Barrero Castañeda[2]
Fundacion Universitaria Konrad Lorenz
[1]jaimea.salcedov@konradlorenz.com, [2]ederd.barreroc@konradlorenz.edu.co

***Resumen*—Los árboles binarios desempeñan un papel fundamental en diversas aplicaciones contemporáneas, abarcando desde su aplicación en redes sociales hasta su utilidad en circuitos eléctricos y la comprensión del entorno que nos rodea. Por ende, resulta imperativo comprender a fondo su estructura y funcionalidad en el contexto actual.**

Este trabajo se centrará en el tema de los recorridos en árboles binarios. Se examinarán diversos ejemplos de estos recorridos, acompañados de explicaciones detalladas sobre su ejecución y funcionamiento. Por último, se presentarán algunos algoritmos implementados en Python relacionados con este tema, brindando una perspectiva práctica y aplicada.

I. Recorrido en árboles

Las estructuras de datos en forma de árboles presentan una amplia gama de variantes que parecen casi infinitas. En este vasto universo, existen miles de algoritmos, algunos más funcionales que otros, algunos más complejos y otros más sencillos. Esta diversidad tiene como objetivo proporcionar la mejor utilidad posible para diversos casos de uso. Además, los árboles adoptan innumerables formas y métodos de manipulación, extendiéndose más allá de los binarios para abarcar dimensiones y capacidades adicionales. No obstante, en este contexto, nos centraremos exclusivamente en los árboles binarios y sus recorridos, como el in-order, post-order y pre-order.

I-A. Recorridos In-Order

Los recorridos ïn-order.[en] árboles binarios se refieren a una estrategia específica para explorar y visitar los nodos del árbol. En este enfoque, primero se realiza el recorrido del subárbol izquierdo, luego se visita el nodo actual y, finalmente, se explora el subárbol derecho. Este método garantiza que los nodos se visiten en orden ascendente según su valor, lo que resulta especialmente útil en árboles de búsqueda binaria. La secuencia ïn-order"proporciona una forma sistemática de acceder a los elementos del árbol, facilitando su análisis y manipulación.

un ejemplo puede ser el siguiente:

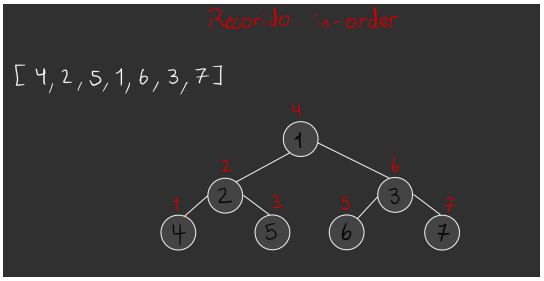

Figura 1. Enter Caption

En la figura, se destaca en rojo la secuencia de recorrido que se sigue para construir el árbol, mientras que en negro se indican los valores asignados a cada nodo. De esta manera, se completa la creación de nuestro primer árbol utilizando el método in-order. Este enfoque visual facilita la comprensión del proceso de formación del árbol, mostrando claramente cómo se asignan los valores a los nodos durante el recorrido especificado.

I-B. Recorridos Post-Order

Los recorridos "post-order.[en] árboles binarios constituyen un método de exploración que sigue una secuencia específica. En este enfoque, la visita de los nodos se realiza de manera recursiva, primero explorando el subárbol izquierdo, luego el subárbol derecho y finalmente visitando el nodo actual. Esta estrategia garantiza que los nodos hoja se visiten antes que sus nodos padres, y la secuencia resultante suele ser útil en operaciones donde se requiere procesar los nodos más bajos antes de los superiores. Los recorridos "post-order"son valiosos para realizar acciones como la liberación de memoria en estructuras de datos dinámicas y la evaluación de expresiones aritméticas.

Aquí puedes ver un ejemplo:

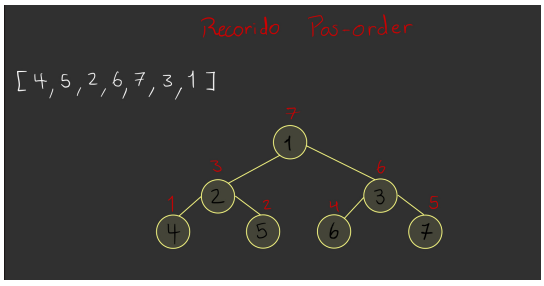

Figura 2. Enter Caption

Este tiene la misma logica que el anterior

I-C. Recorridos Pre-Order

Los recorridos "pre-order.[en] árboles binarios representan una técnica de exploración que sigue una secuencia específica. En este enfoque, la visita de los nodos se inicia por el nodo actual, luego se explora recursivamente el subárbol izquierdo y, finalmente, se explora el subárbol derecho. Este método asegura que los nodos se visiten en un orden que comienza desde la raíz y progresa hacia los nodos más bajos. Los recorridos "pre-order"son valiosos en situaciones donde se desea

realizar acciones antes de explorar los nodos secundarios, como la creación de expresiones prefijas para evaluar fórmulas matemáticas o para imprimir la estructura del árbol antes de realizar operaciones específicas en los nodos.

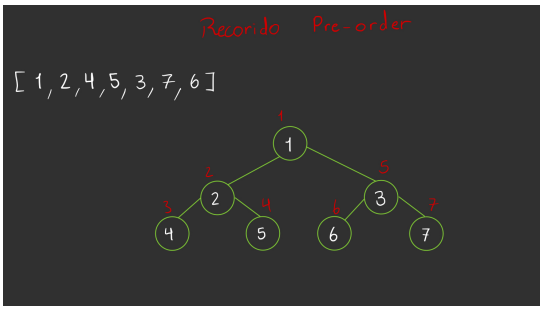

Figura 3. Enter Caption

II. Recorridos en Python

Ahora procederemos a implementar estos recorridos en Python. De esta manera, obtendremos una comprensión práctica de su funcionamiento, y los códigos correspondientes estarán disponibles en el repositorio para su uso en cualquier momento que sea necesario.

Esté sería el árbol que usáremos para los algoritmos:

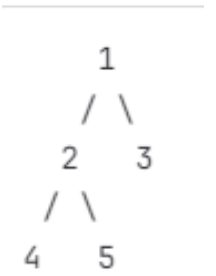

Figura 4. Enter Caption

II-A. Integración Árboles

Para dar inicio, necesitaremos la función que nos permita establecer un sistema de árboles. Para lograr esto, comenzaremos creando una clase que genere los nodos, definiendo las características de sus nodos secundarios. Dado que estamos tratando con un sistema binario, cada nodo tendrá, en su máximo, dos nodos secundarios.

```python
class Nodo:
    def __init__(self, value):
        self.value = value
        self.left = None
        self.right = None
```

Figura 5. Enter Caption

II-B. Recorrido in-Order

Para el recorrido in-order, desarrollaremos una función que explore primero el nodo izquierdo, seguido por el valor o raíz y, finalmente, el nodo derecho. La estructura se plantea de la siguiente manera:

```python
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 raiz del árbol
inorden(arbol)
```

```
✓ 0.0s
4
2
5
1
3
```

Figura 6. Enter Caption

Dandonos como resultado [4,2,5,1,3]

II-C. Recorrido Post-Order

Para el recorrido Post-Order, desarrollaremos una función que explore primero el nodo izquierdo, seguido por el nodo derecho y, finalmente, el valor o raíz. La estructura se plantea de la siguiente manera:

```python
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 raiz del árbol
postorden(arbol)
```

```
✓ 0.0s
4
5
2
3
1
```

Figura 7. Enter Caption

Dandonos como resultado [4,5,2,3,1]

II-D. Recorrido Pre-Order

Para el recorrido Post-Order, desarrollaremos una función que explore primero el nodo valor o raíz, seguido por el nodo izquierdo y, finalmente, el nodo derecho. La estructura se plantea de la siguiente manera:

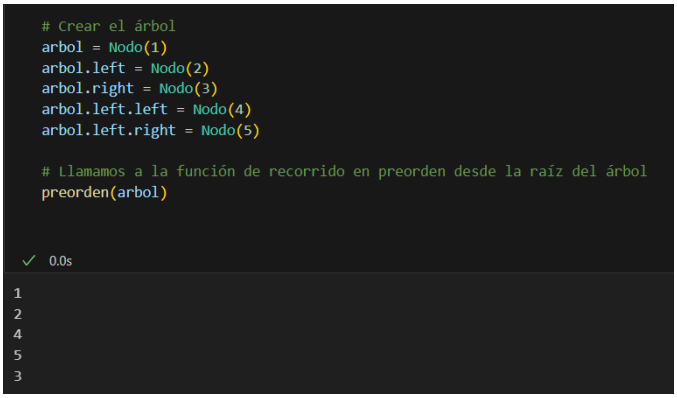

Figura 8. Enter Caption

Dandonos como resultado [1,2,4,5,3]

III. BIBLIOGRAFIA

OpenAI. (2021). ChatGPT: A large language model.

Andrés Salcedo Vera. (2023). Arboles y Recorridos. Eder David Barrero Castañeda. (2023). Arboles y Recorridos.