

Arboles

a01

¿Qué son los árboles binarios?

Los árboles binarios son una estructura de datos fundamental que consiste en nodos organizados jerárquicamente, donde cada nodo puede tener como máximo dos hijos (izquierdo y derecho).

¿Qué tipos de árboles existen?

Existen diferentes tipos de árboles binarios, como los árboles de búsqueda, árboles AVL, árboles rojo-negro y árboles B.

¿En qué tipos de aplicaciones se usan los árboles binarios?

Los árboles binarios se utilizan en diversas aplicaciones, incluyendo bases de datos, compresión de datos, compiladores, redes, juegos y aplicaciones gráficas, ya que permiten organizar, buscar y manipular datos de manera eficiente.

a02

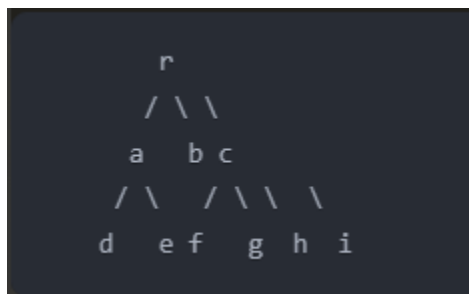
r: Es la raíz del árbol binario.

a, b, c: Son los nodos hijos de la raíz 'r'.

d, e: Son los nodos hijos del nodo 'a'.

f, g, h, i: Son los nodos hijos del nodo 'c'.

En esta representación, la raíz 'r' tiene tres hijos: 'a', 'b' y 'c'. El nodo 'a' tiene dos hijos: 'd' y 'e'. El nodo 'c' tiene cuatro hijos: 'f', 'g', 'h' e 'i'.



a03

```
r
a
d
e
b
f
g

Recorrido Inorden (Izquierda-Raíz-Derecha):
d
a
e
r
f
b
g

Recorrido Postorden (Izquierda-Derecha-Raíz):
d
e
a
f
g
b
r
PS C:\Users\jorge\OneDrive\Escritorio\Sem4ULSA>
```

a04


```
Estructura de datos > arboles > a04.py > ...
1  import matplotlib.pyplot as plt
2  import networkx as nx
3  from networkx.drawing.nx_pydot import graphviz_layout
4
5  # Crear un gráfico dirigido
6  G = nx.DiGraph()
7
8  # Agregar los nodos y las aristas al gráfico
9  G.add_edge(1, 2)
10 G.add_edge(1, 3)
11 G.add_edge(2, 4)
12 G.add_edge(4, 7)
13 G.add_edge(4, 8)
14 G.add_edge(3, 5)
15 G.add_edge(3, 6)
16 G.add_edge(5, 9)
17 G.add_edge(9, 11)
18 G.add_edge(6, 10)
19
20 # Dibujar el gráfico
21 pos = graphviz_layout(G, prog='dot')
22 nx.draw(G, pos, with_labels=True, arrows=False)
23
24 # Mostrar el gráfico
25 plt.show()
```

a05

1. Recorrido en Orden (Inorden): 1-2-3-4-5-6-7

Este recorrido nos indica que el orden correcto de los nodos es:

markdown


 Copy code

```
      4
     / \
    2   6
   / \ / \
  1  3 5  7
```

2. Recorrido Preorden: 1-2-3-4-5-6-7

Siguiendo este recorrido, el árbol se ve así:

markdown

 Copy code


```
      4
     / \
    2   6
   / \ / \
  1  3 5  7
```

3. Recorrido Postorden: 1-2-3-4-5-6-7

El árbol generado según este recorrido sería el mismo que el del recorrido preorden.

Por lo tanto, el árbol binario para estos recorridos dados es el mismo en todos los casos:

markdown

 Copy code

```
      4
     / \
    2   6
   / \ / \
  1  3 5  7
```

a06

- **Siguiente nodo en preorden del nodo 16**

17

- **Siguiente nodo en inorden del nodo 16**

1

- **Siguiente nodo en postorden del nodo 16**

15

- **¿Si estamos en el nodo 17, cuál es el siguiente nodo en preorden?**

15

- **¿Si estamos en el nodo 17, cuál es el siguiente nodo en inorden?**

No hay

- **¿Si estamos en el nodo 17, cuál es el siguiente nodo en postorden?**

15

a07

En base al árbol anterior y el siguiente recorrido: 12-8-4-13-16-14-9-5-2-10-6-17-15-11-7-3-1. Que tipo de recorrido es?

Postorden, El recorrido sigue el patrón de visitar primero los subárboles izquierdo y derecho de un nodo antes de visitar el nodo en sí.