



1INF27

Algoritmia y Estructura de Datos

2024

Profesores:

Cueva, R. | Allasi, D. | Roncal, A. | Huamán, F.

0581

0582

0583

0584

OBJETIVOS

Resultados de Aprendizaje:

- Definir los árboles como estructuras de datos.
- Definir los términos asociados a los árboles.
- Analizar las implementaciones con árboles binarios.
- Explicar los métodos existentes para recorrer los árboles.
- Examinar un ejemplo de árbol binario.



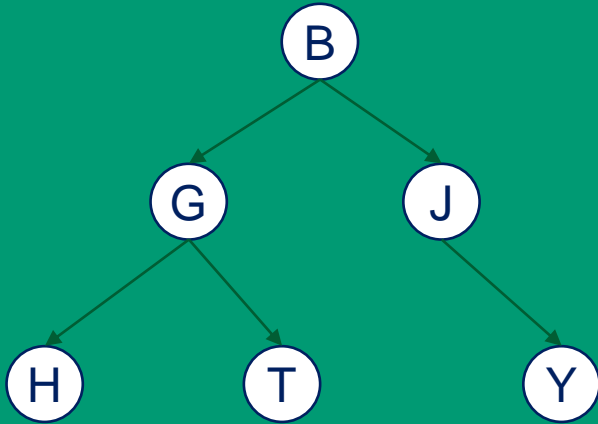
Capítulo 2

ÁRBOLES BINARIOS



Definición

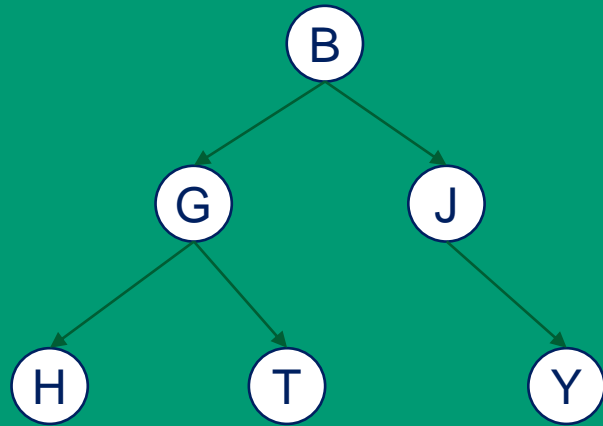
Las pilas, colas y listas son estructuras de datos lineales, lo que quiere decir que sus elementos están dispuestos en orden, uno después de otro. Un árbol es una estructura **no lineal** en la que los elementos están organizados en una **jerarquía**.



ÁRBOLES

Un árbol está compuesto de un conjunto de **nodos** en los que se almacenan los elementos y **aristas** que conectan los nodos con otros.





ÁRBOLES BINARIOS

Clasificación

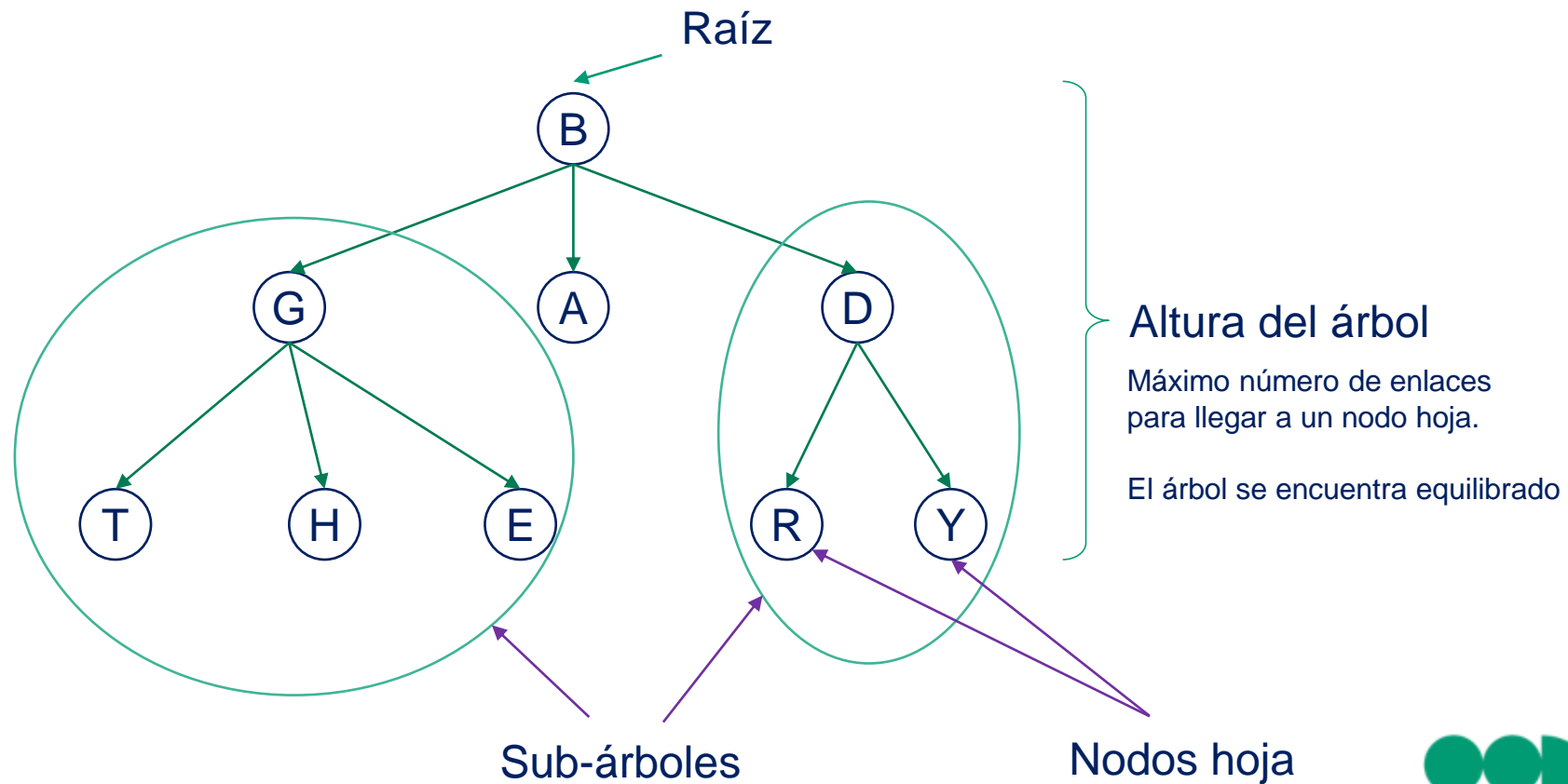
Los árboles en los que los nodos puedan tener *máximo dos hijos* se denominan **árboles binarios (AB)**.

Un árbol se considera equilibrado si todas las hojas del árbol se encuentran en el mismo nivel o, como máximo, con un nivel de diferencia unas respecto a otras.



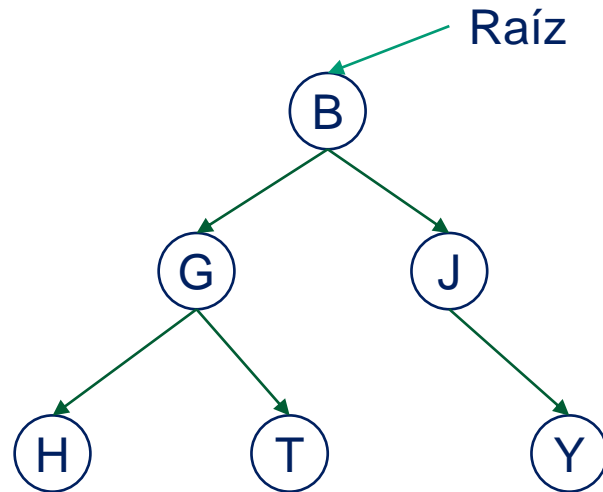
ÁRBOLES

Partes:



ÁRBOLES BINARIOS

Terminología: los nodos situados en los niveles inferiores del árbol son los **hijos** de los nodos ubicados en el anterior nivel.



En la figura:

Los nodos **G** y **J** son los hijos de **B**.
El nodo **G** es el padre de **H** y **T**

Cada nodo solo puede tener un padre
y a lo más dos hijos.



Operaciones con árboles binarios

- Crear árbol
- Plantar árbol
- Verificar si el árbol está vacío
- Recorrer los elementos del árbol
- Determinar la altura del árbol
- Determinar hijo derecho
- Determinar hijo izquierdo
- Finalizar un árbol



Recorrido de un árbol

- Muchas operaciones se basan en recorrer los elementos de un árbol.
- Como no es una estructura lineal (como los arreglos o las listas), se necesita de algoritmos diferentes para recorrerlos.
- Tenemos tres formas de recorrer un árbol
 - Pre-orden
 - En-orden
 - Post-orden

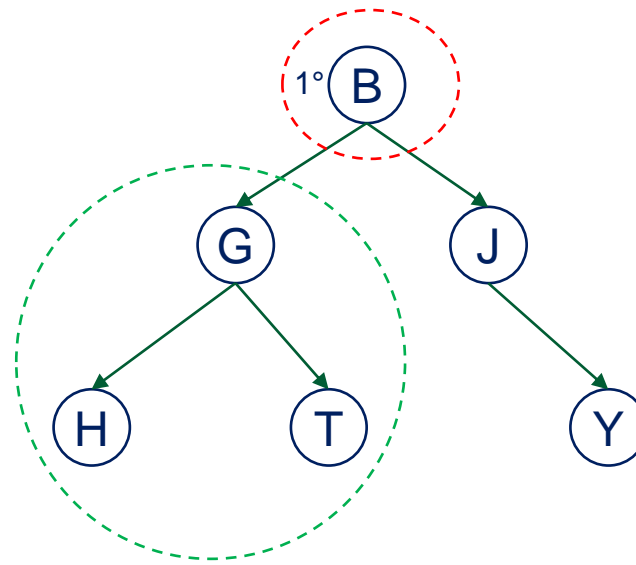


Pre - orden



Pre-Orden:

se visita **primero el nodo raíz**, y después los subárboles **izquierdo en *pre-orden*** y **derecho en *pre-orden***.

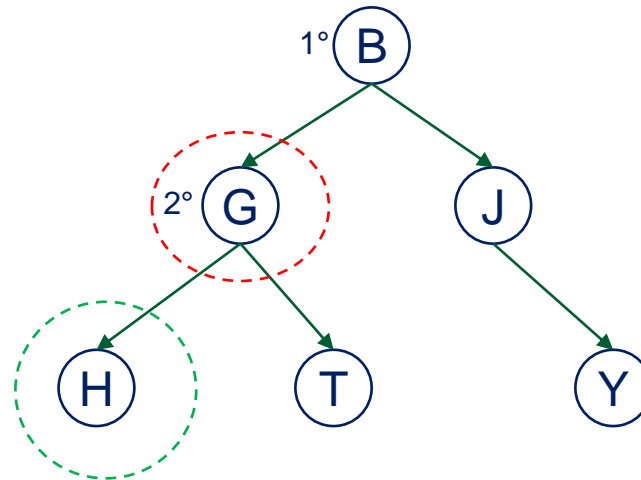


Orden de visita: B



Pre-Orden:

se visita **primero el nodo raíz**, y después los subárboles **izquierdo en *pre-orden*** y **derecho en *pre-orden***.

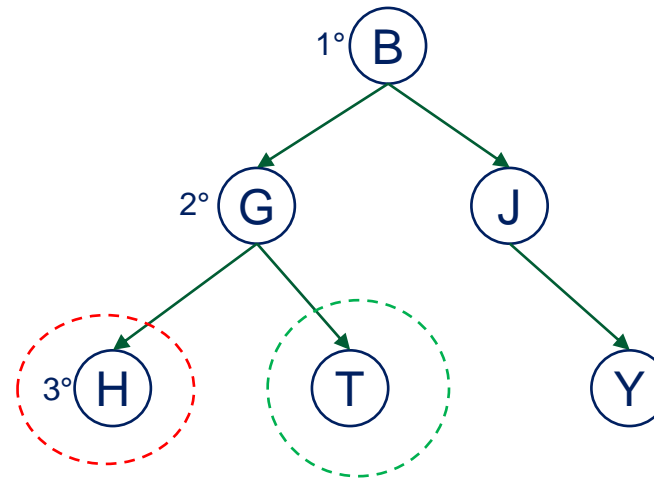


Orden de visita: B G



Pre-Orden:

se visita **primero el nodo raíz**, y después los subárboles **izquierdo en *pre-orden*** y **derecho en *pre-orden***.

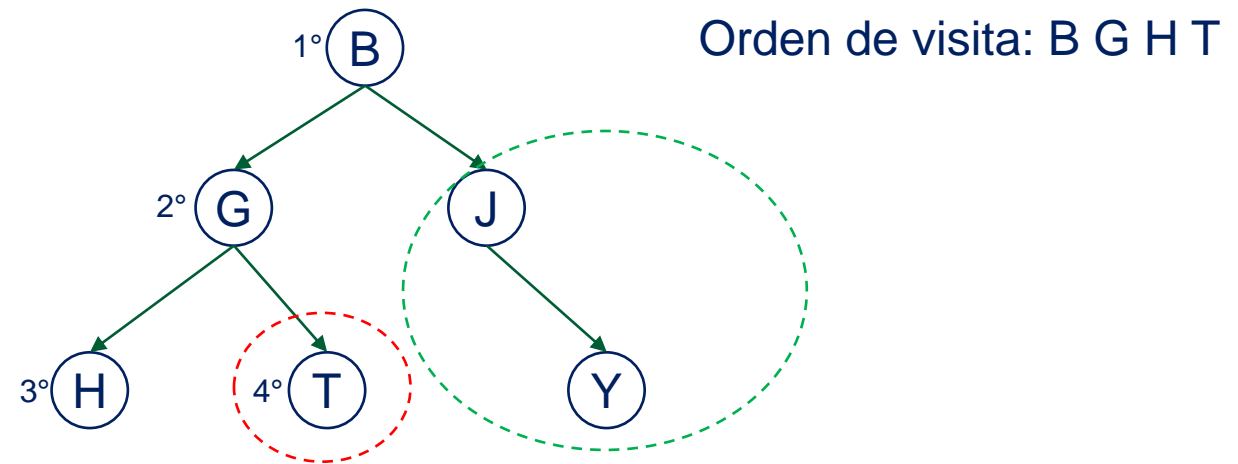


Orden de visita: B G H



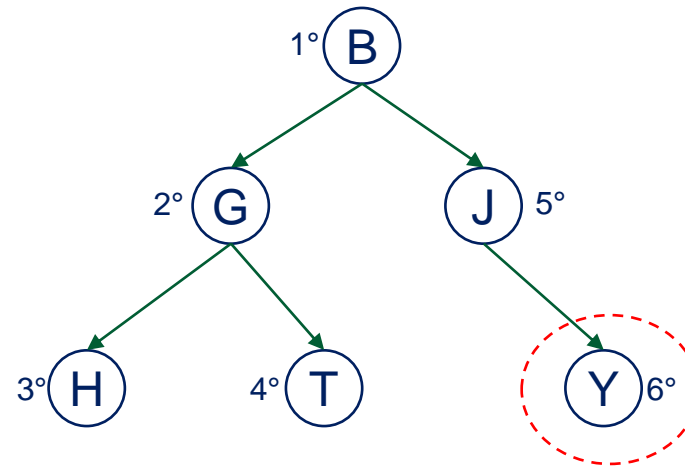
Pre-Orden:

se visita **primero el nodo raíz**, y después los subárboles **izquierdo en *pre-orden*** y **derecho en *pre-orden***.



Pre-Orden:

se visita **primero el nodo raíz**, y después los subárboles **izquierdo en *pre-orden*** y **derecho en *pre-orden***.



Orden de visita: B G H T J Y



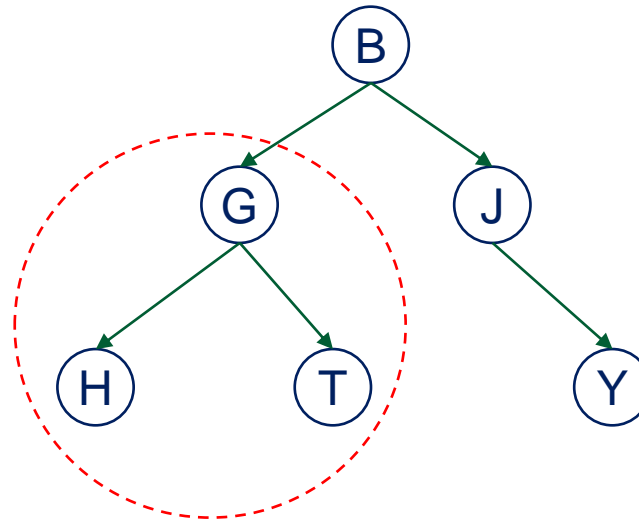
En - orden



En-orden:

Se visita primero **subárbol izquierdo** en **en-orden**, el **nodo raíz** y el **subárbol derecho** en **en-orden**.

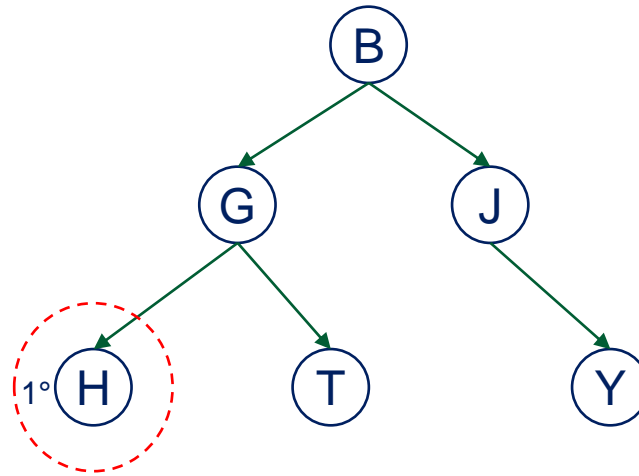
Orden de visita:



En-orden:

Se visita primero **subárbol izquierdo** en **en-orden**, el **nodo raíz** y el **subárbol derecho** en **en-orden**.

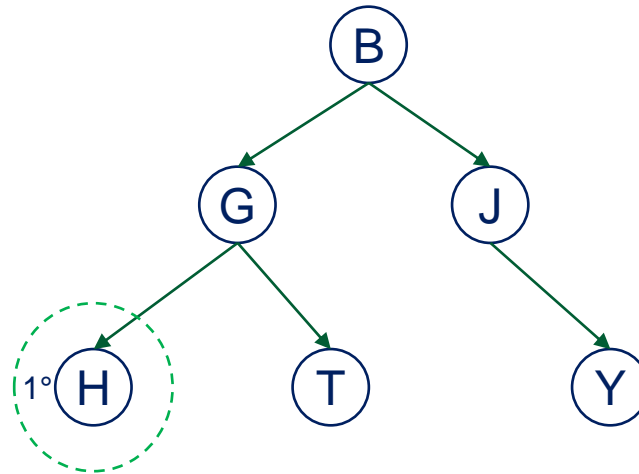
Orden de visita:



En-orden:

Se visita primero **subárbol izquierdo** en **en-orden**, el **nodo raíz** y el **subárbol derecho** en **en-orden**.

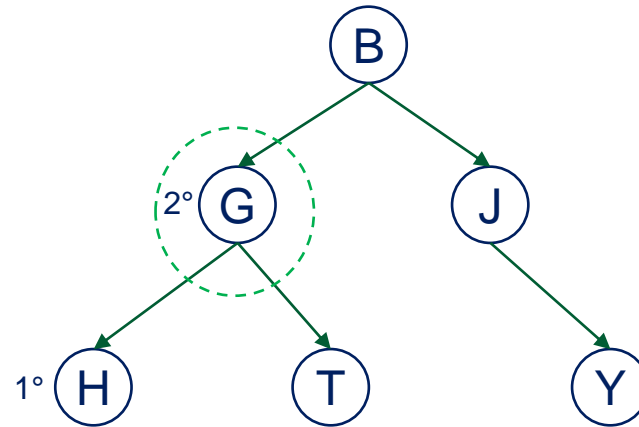
Orden de visita: H



En-orden:

Se visita primero **subárbol izquierdo** en **en-orden**, el **nodo raíz** y el **subárbol derecho** en **en-orden**.

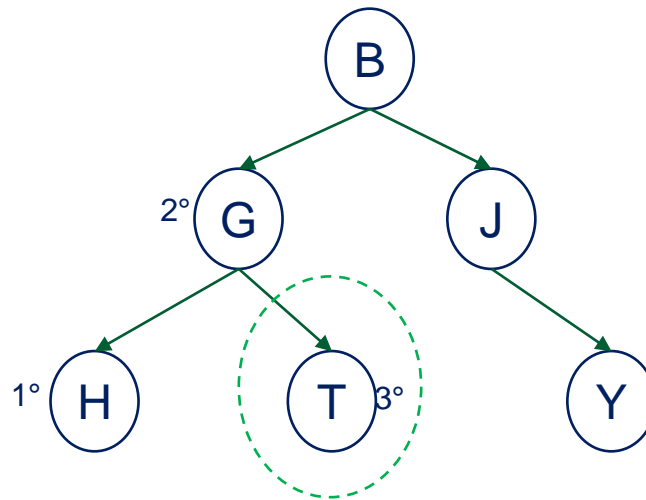
Orden de visita: H, G



En-orden:

Se visita primero **subárbol izquierdo** en **en-orden**, el **nodo raíz** y el **subárbol derecho** en **en-orden**.

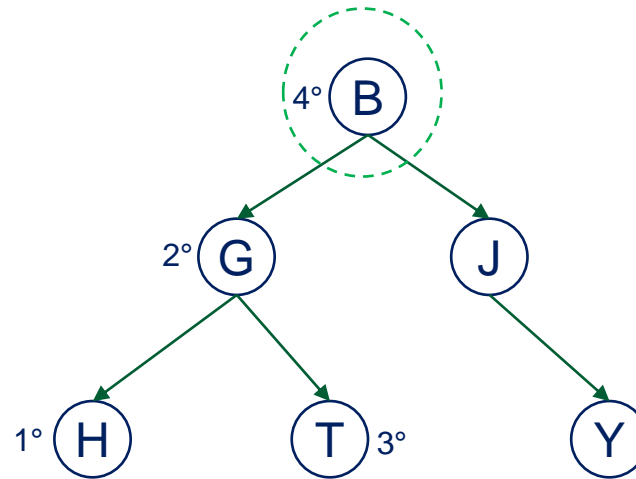
Orden de visita: H, G, T



En-orden:

Se visita primero **subárbol izquierdo** en **en-orden**, el **nodo raíz** y el **subárbol derecho** en **en-orden**.

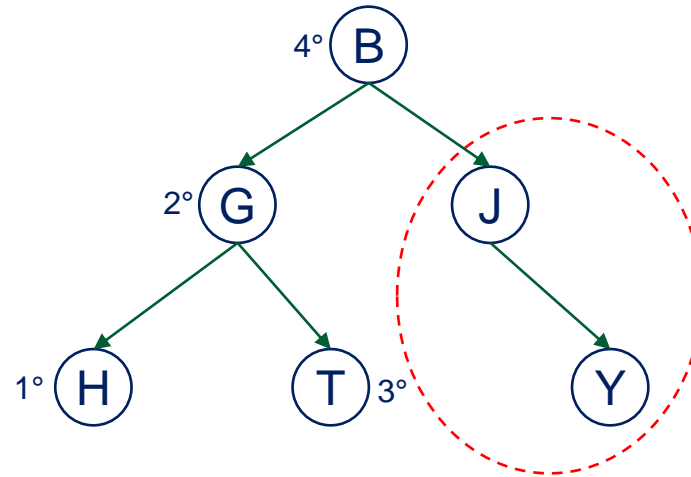
Orden de visita: H, G, T, B



En-orden:

Se visita primero **subárbol izquierdo** en **en-orden**, el **nodo raíz** y el **subárbol derecho** en **en-orden**.

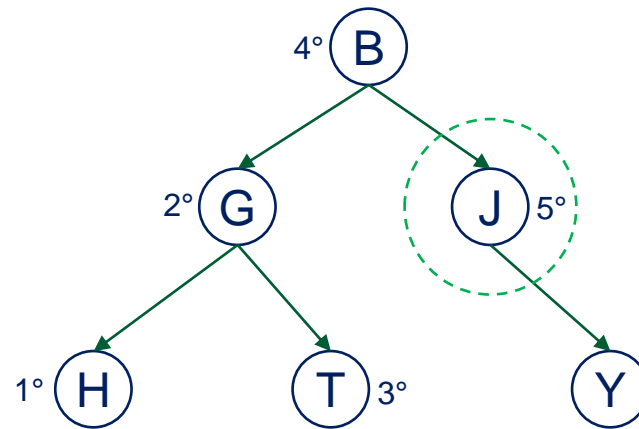
Orden de visita: H, G, T, B



En-orden:

Se visita primero **subárbol izquierdo** en **en-orden**, el **nodo raíz** y el **subárbol derecho** en **en-orden**.

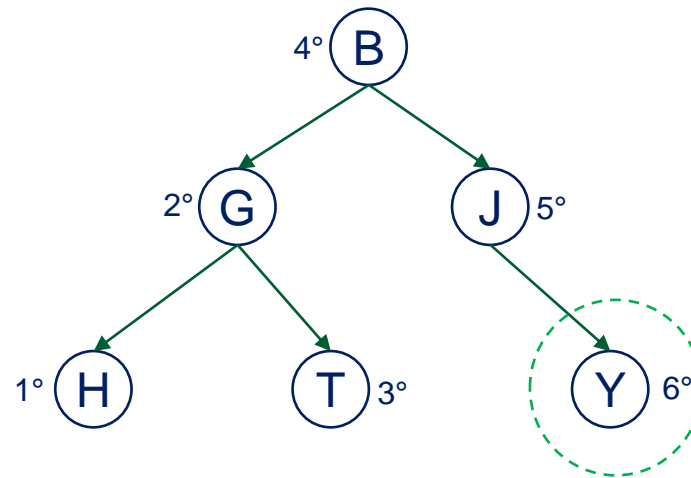
Orden de visita: H, G, T, B, J



En-orden:

Se visita primero **subárbol izquierdo** en **en-orden**, el **nodo raíz** y el **subárbol derecho** en **en-orden**.

Orden de visita: H, G, T, B, J, Y



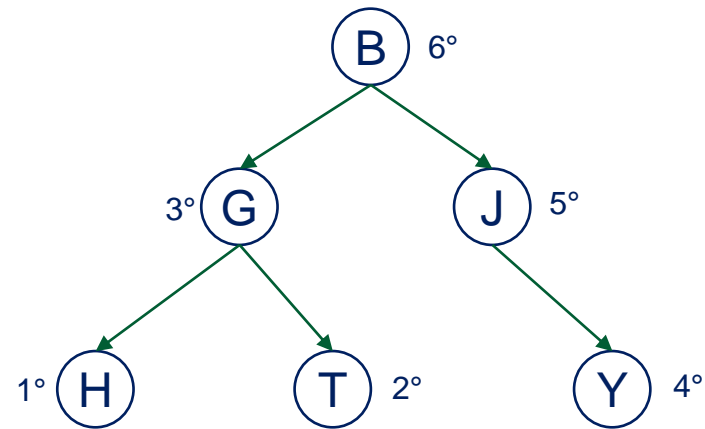
Post - orden



Post-orden:

Se visita primero **subárbol izquierdo** en post-orden, el **subárbol derecho** en post-orden y el **nodo raíz**

Orden de visita: H, T, G, Y, J, B



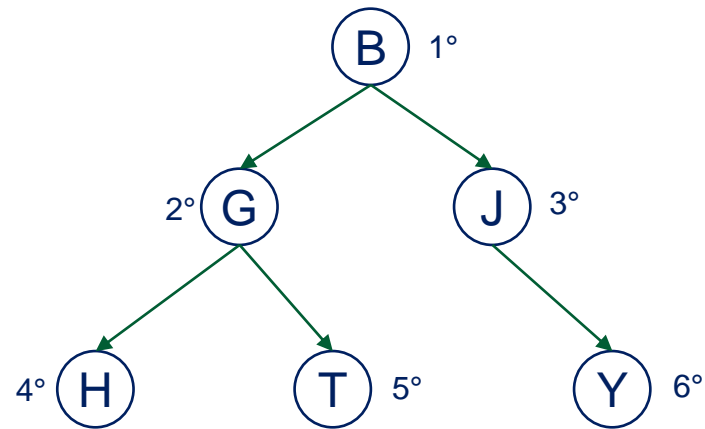
En amplitud



En amplitud

Recorre el árbol nivel por nivel comenzando por la raíz.

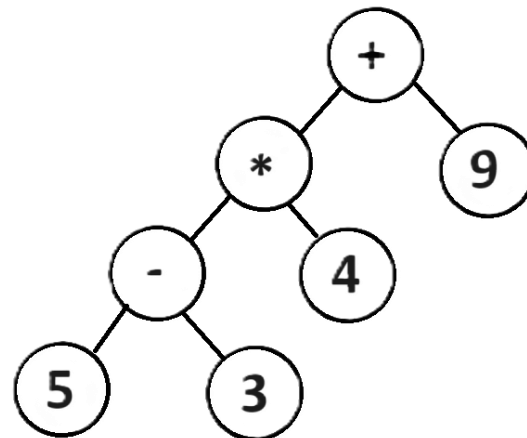
Orden de visita: B, G, J, H, T, Y



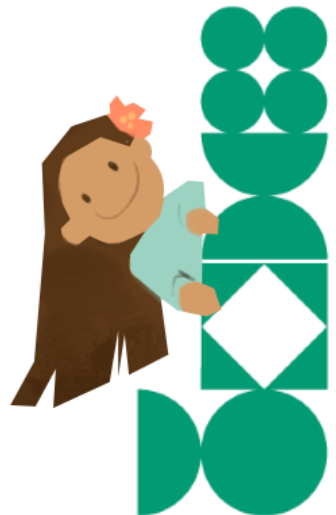
Ejercicio

Construir árboles de expresión

Usando el concepto de expresiones postfijas, vamos a construir un árbol de expresión. Dónde: la raíz y todos los nodos internos de un árbol de expresión contienen operaciones y que todas las hojas contienen operandos. Estos árboles se evalúan de abajo hacia arriba, recorriéndolos en-orden.



$$(5 - 3) * 4 + 9$$

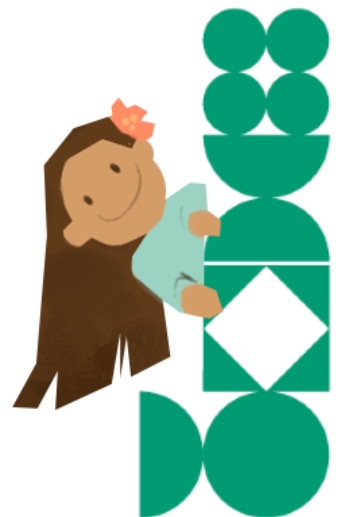
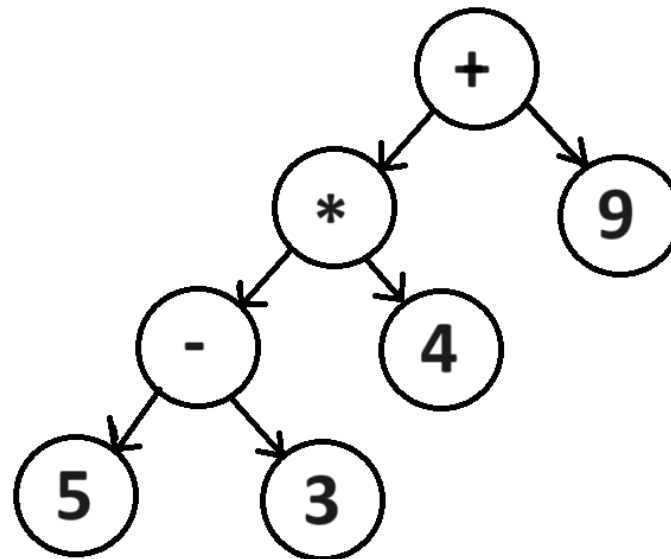


Ejercicio

Ejemplo de un árbol de expresión



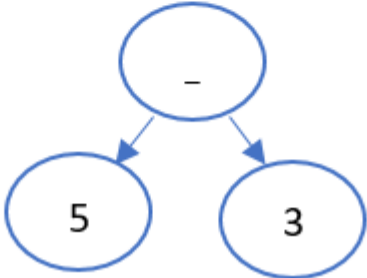
Expresión postfija de entrada: **5 3 - 4 * 9 +**

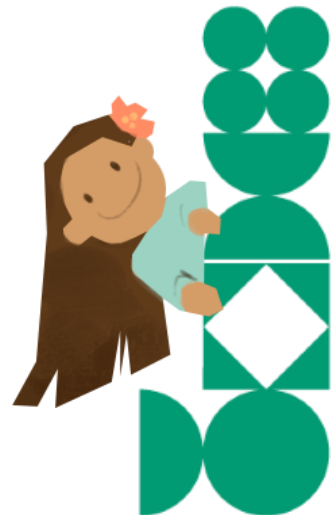
En el siguiente caso, se evalúa primero $(5 - 3)$, lo que da como resultado 2. Este resultado se multiplica por 4, lo que nos da 8. Finalmente, el resultado de dicho término se suma con 9, lo que nos da 17.



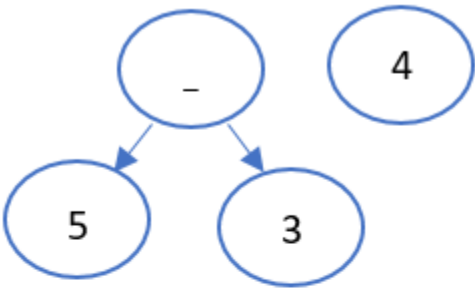
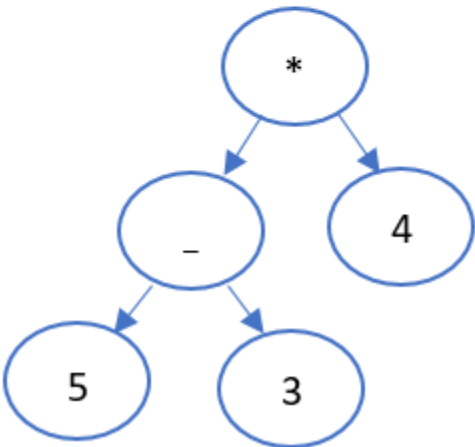
Ejercicio

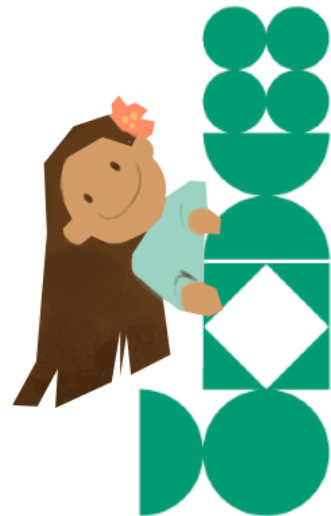
Expresión postfija de entrada: 5 3 – 4 * 9 +

Ident.	Pasos de procesamiento	Pila de árbol de expresión (cima a la derecha)
5	plantarArbolBinario(arbol, null, 5, null); apilar(pila, arbol.raiz);	
3	plantarArbolBinario(arbol, null, 3, null); apilar(pila, arbol.raiz);	
-	operador2 = desapilar(pila) operador1 = desapilar(pila) plantarArbolBinario(arbol, operador1, -, operador2); apilar(pila, arbol.raiz);	

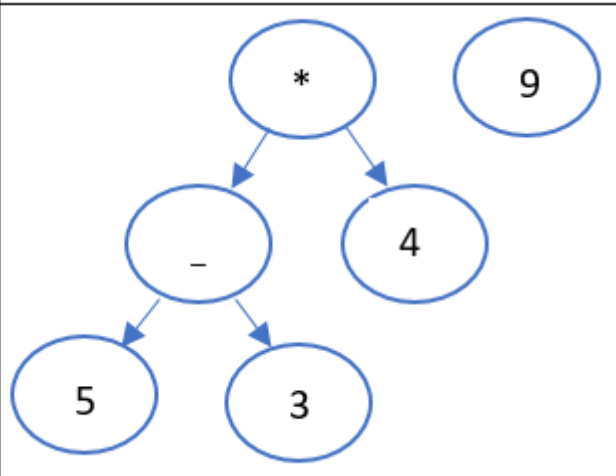


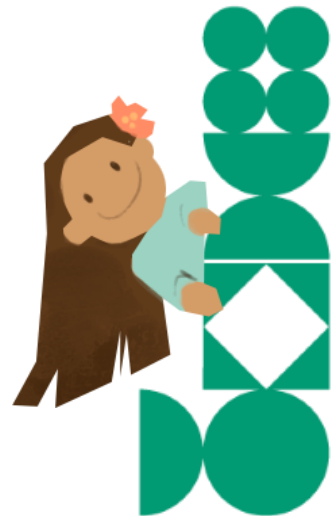
Ejercicio

4	<pre>plantarArbolBinario(arbol, null, 4, null); apilar(pila, arbol.raiz);</pre>	 <pre>graph TD; A["-"] --> B["5"]; A --> C["3"]; D["4"]</pre>
*	<pre>operador2 = desapilar(pila) operador1 = desapilar(pila) plantarArbolBinario(arbol, operador1, *, operador2); apilar(pila, arbol.raiz);</pre>	 <pre>graph TD; A["*"] --> B["-"]; A --> C["4"]; B --> D["5"]; B --> E["3"]</pre>

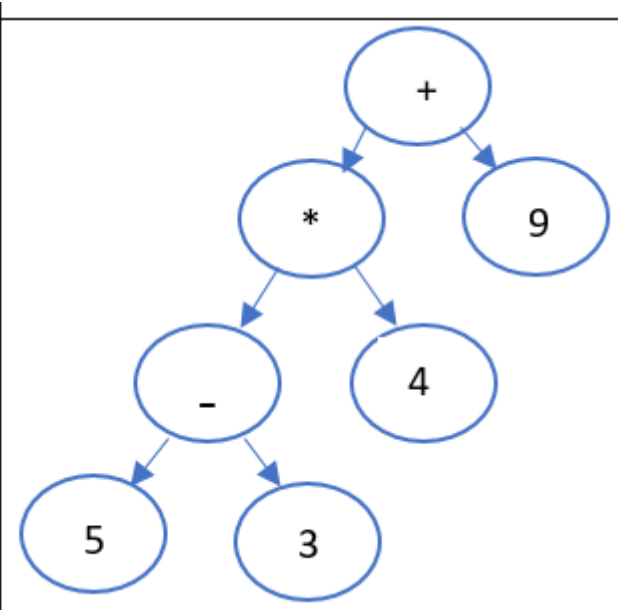


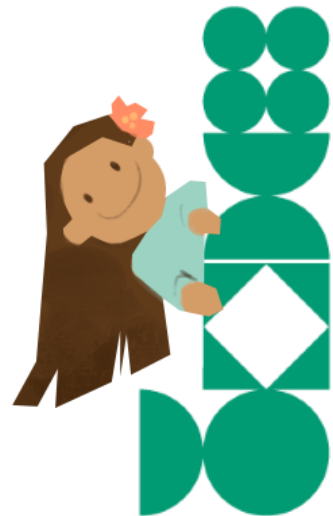
Ejercicio

9	<pre>plantarArbolBinario(arbol, null, 9, null); apilar(pila, arbol.raiz);</pre>	 <pre>graph TD; A((*)) --> B((-)); A --> C((4)); B --> D((5)); B --> E((3)); F((9))</pre>
---	---	--



Ejercicio

+	<pre>operador2 = desapilar(pila) operador1 = desapilar(pila) plantarArbolBinario(arbol, operador1, +, operador2); apilar(pila, arbol.raiz);</pre>	 <pre>graph TD Plus((+)) --> Mult((*)) Plus --> Nine((9)) Mult --> Minus((-)) Mult --> Four((4)) Minus --> Five((5)) Minus --> Three((3))</pre>
---	---	---



ÁRBOLES BINARIOS - Resumen

Resumen

- Un árbol es una estructura no lineal y dinámica cuyos elementos están organizados en una jerarquía.
- Se observa que en árboles binarios no se usan métodos de insertar ni de eliminar dado que faltan criterios para estos casos.



BIBLIOGRAFÍA

Cueva, R. (2022, Agosto 21). Capítulo 2. *Estructuras de Datos (TAD) y sus implementaciones. [PowerPoint slides]*. Facultad de Ciencias e Ingeniería. PUCP.

Hernández, R., Lázar, J. C., Dormido, R. y Ros, S. (2001). *Estructuras de Datos y Algoritmos*. Pearson Education, S.A., Madrid

Lewis, J. y Chase, J. (2006) *Estructuras de datos con Java diseño de estructuras y algoritmos*. Pearson Education S.A., Madrid