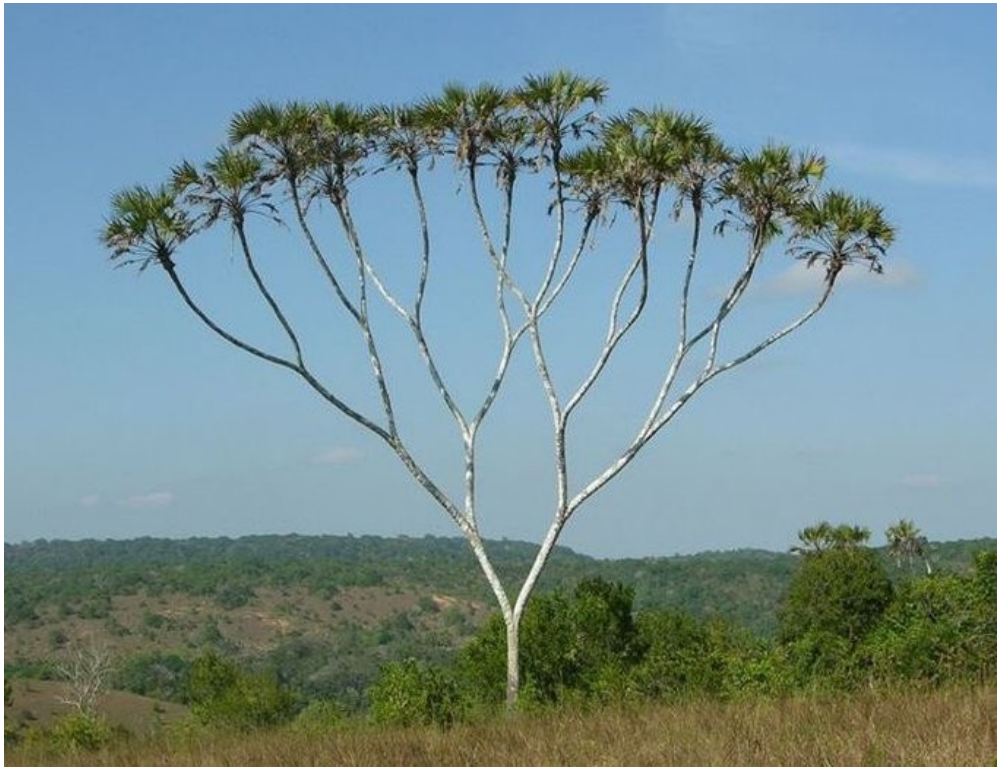
Algoritmos y Estructuras de Datos

Ingeniería en Sistemas de la Información

Árboles Binarios

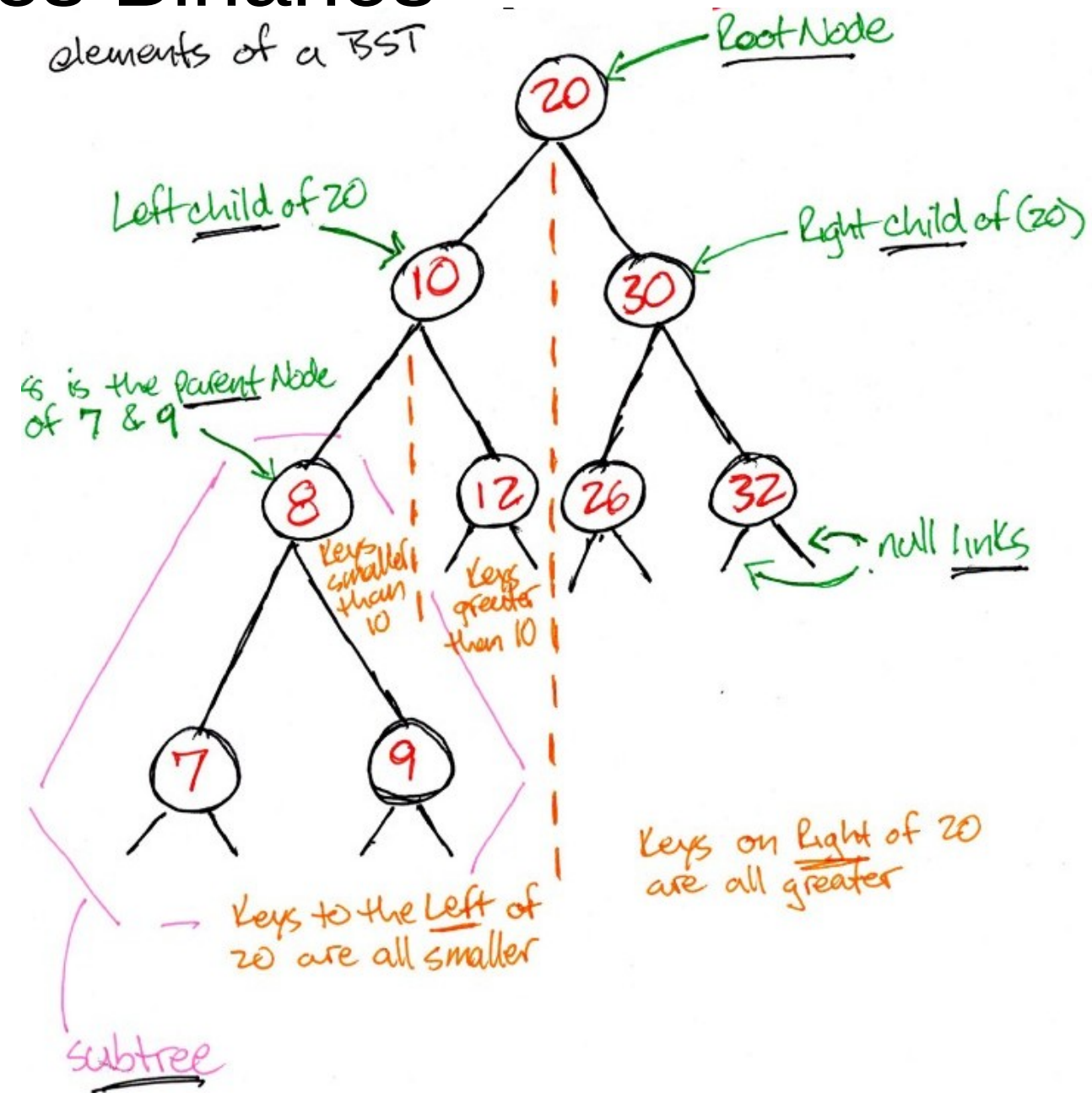
UAP | UNIVERSIDAD ADVENTISTA DEL PLATA

Árboles Binarios



Un árbol es un tipo abstracto de datos (TAD) ampliamente usado que imita la estructura jerárquica de un árbol, con un valor en la raíz y subárboles con un nodo padre, representado como un conjunto de nodos enlazados.

Árboles Binarios



Terminología

Raíz: El nodo superior de un árbol.

Hijo: Un nodo conectado directamente con otro cuando se aleja de la raíz.

Padre: La noción inversa de hijo.

Hermanos: Un conjunto de nodos con el mismo padre.

Descendiente: Un nodo accesible por descenso repetido de padre a hijo.

Ancestro: Un nodo accesible por ascenso repetido de hijo a padre.

Terminología

Hoja: Un nodo sin hijos.

Nodo interno: Un nodo con al menos un hijo.

Grado: Número de subárboles de un nodo.

Brazo: La conexión entre un nodo y otro.

Camino: Una secuencia de nodos y brazos conectados con un nodo descendiente.

Terminología

Nivel: El nivel de un nodo se define por 1 + (el número de brazos entre el nodo y la raíz).

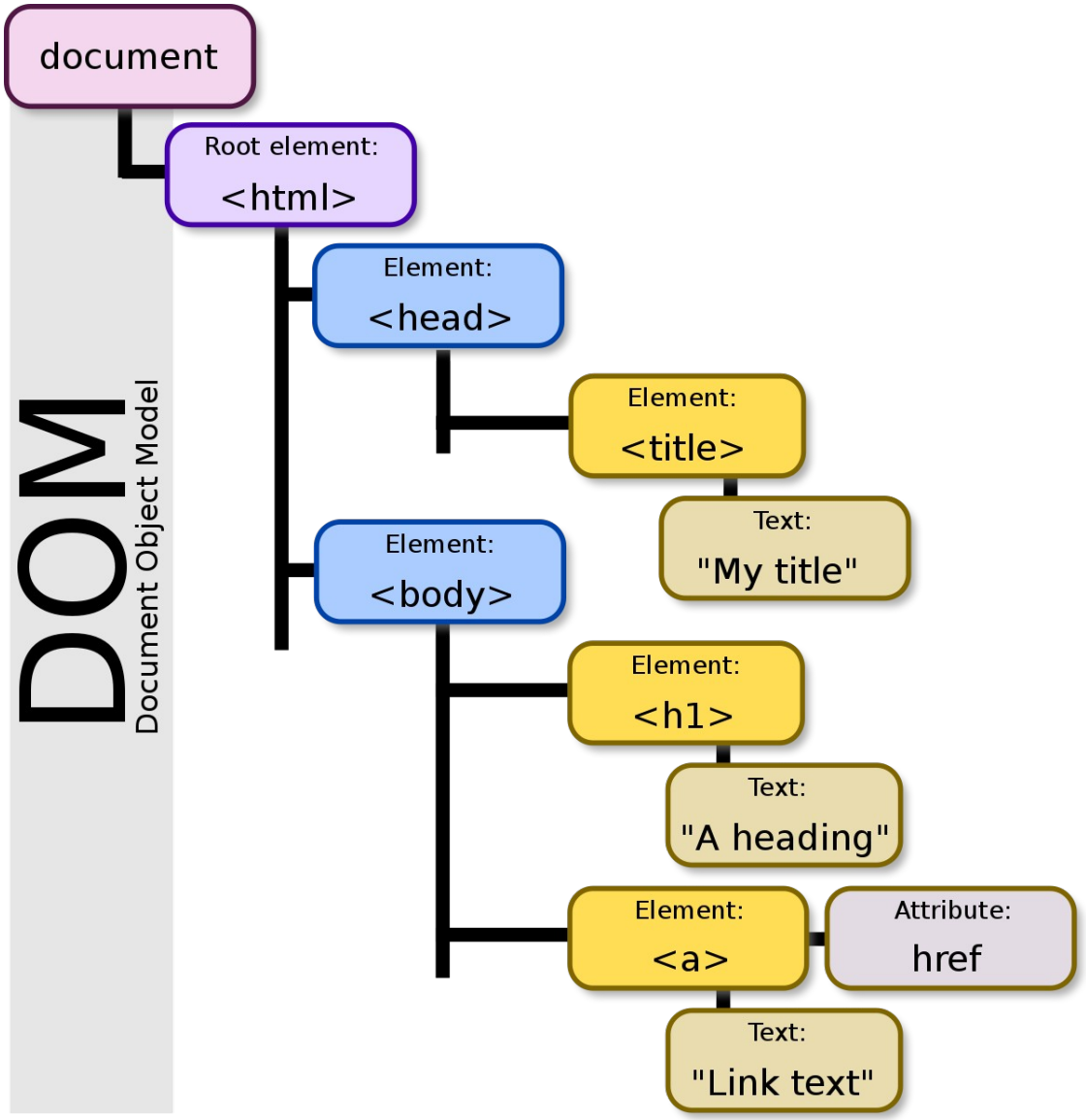
Altura de un nodo: La altura de un nodo es el número de brazos en el camino más largo entre ese nodo y una hoja.

Altura de un árbol: La altura de un árbol es la altura de su nodo raíz.

Profundidad: La profundidad de un nodo es el número de brazos desde la raíz del árbol hasta un nodo.

Rama: Una ruta del nodo raíz a cualquier otro nodo.

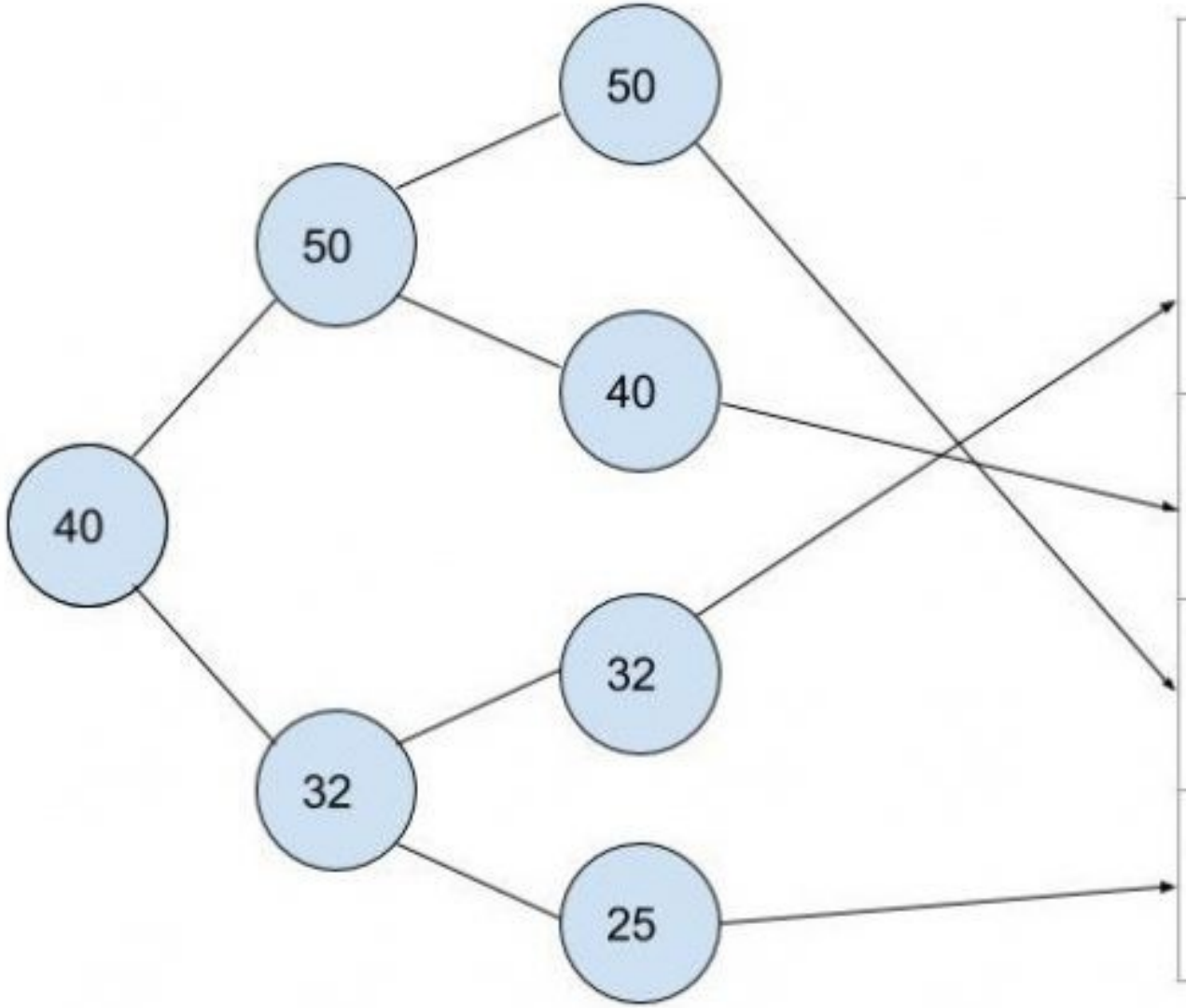
Árboles – Aplicaciones - WEB



Árboles – Aplicaciones
Representación de datos


```
{
    "curso": [
        {
            "id": "01",
            "materia": "JSON",
            "nivel": "básico",
            "author": "José Manuel"
        },
        {
            "id": "07",
            "materia": "Liferay",
            "nivel": "medio",
            "author": "José Manuel"
        }
    ]
}
```

Árboles – Aplicaciones – Índices en BD

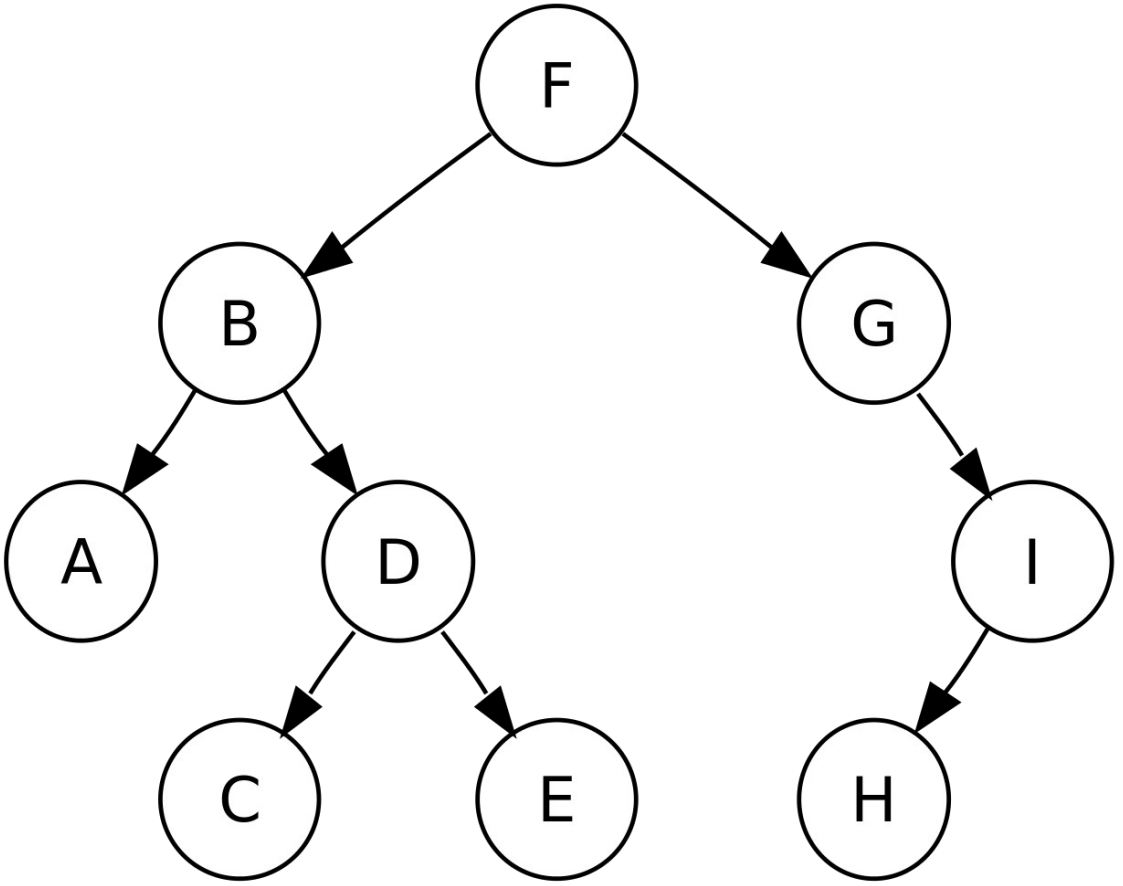


Id	Name	Age
1	Alice	32
2	Bob	40
3	Carol	50
4	Dave	25

Tipos de recorrido

Profundidad primero: visitar todos los nodos desde la raíz hasta las hojas, cuando ya no quedan más nodos hijo por visitar, volver atrás (backtracking) y seguir con otros hermanos de nodo ya procesado.

Hay 3 formas de hacerlo:

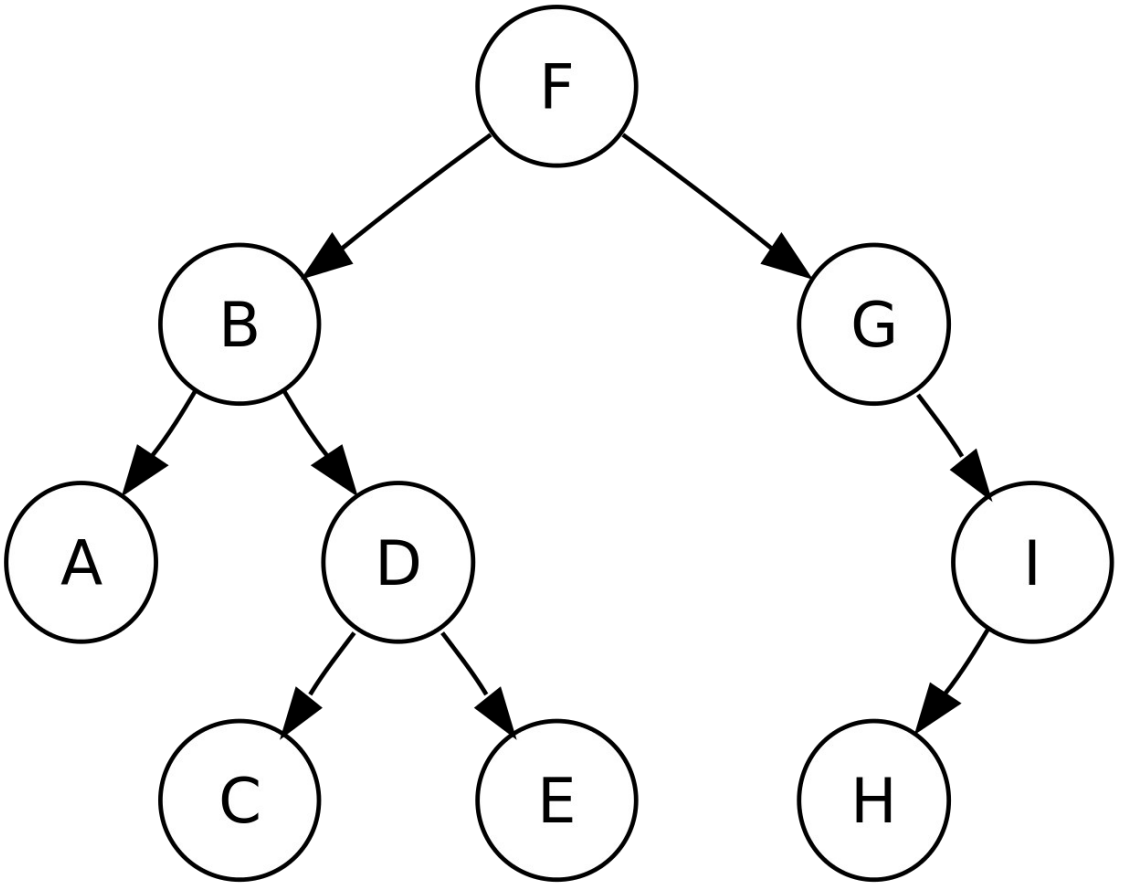


Tipos de recorrido - Preorden

Recursivamente y en orden hacer:

1) Visitar el nodo
2) Pasar por el nodo izquierdo
3) Pasar por el nodo derecho

F B A D C E G I H

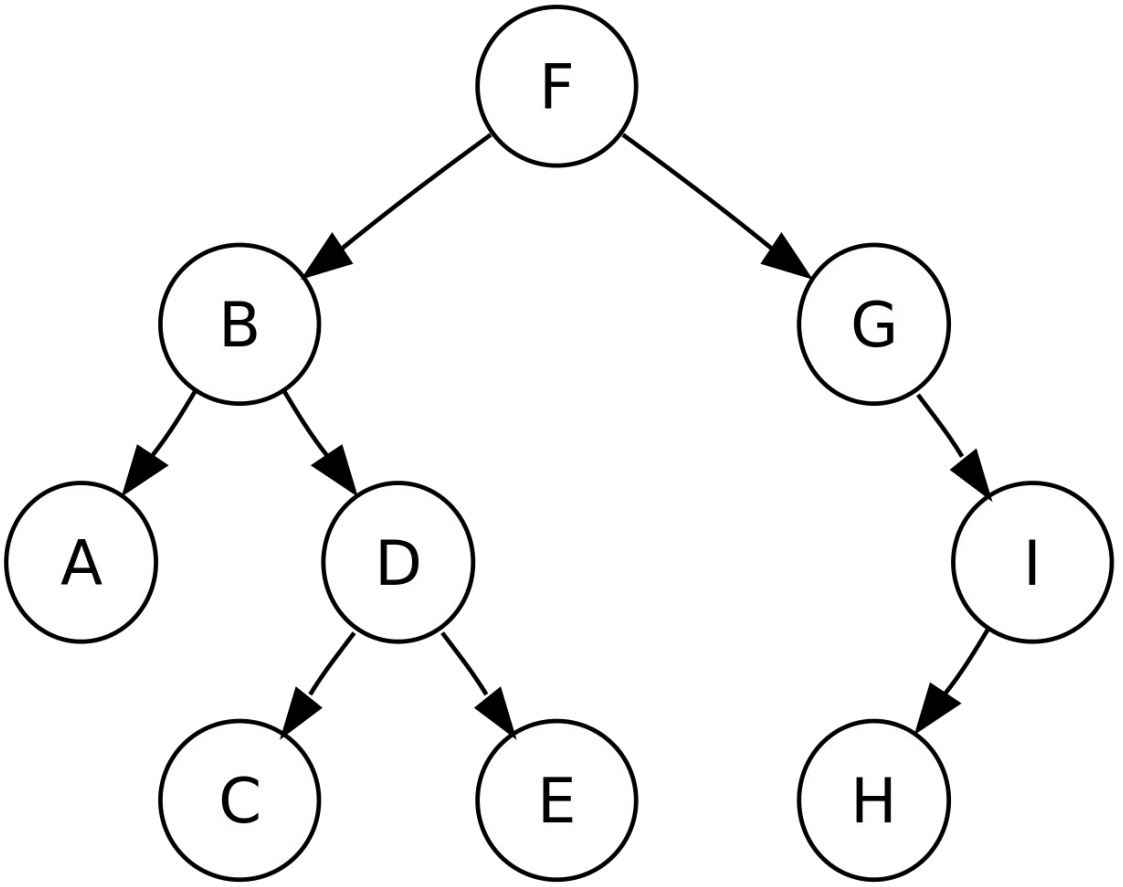


Tipos de recorrido - Inorden

Recursivamente y en orden hacer:

1) Pasar por el nodo izquierdo
2) Visitar el nodo
3) Pasar por el nodo derecho

A B C D E F G H I

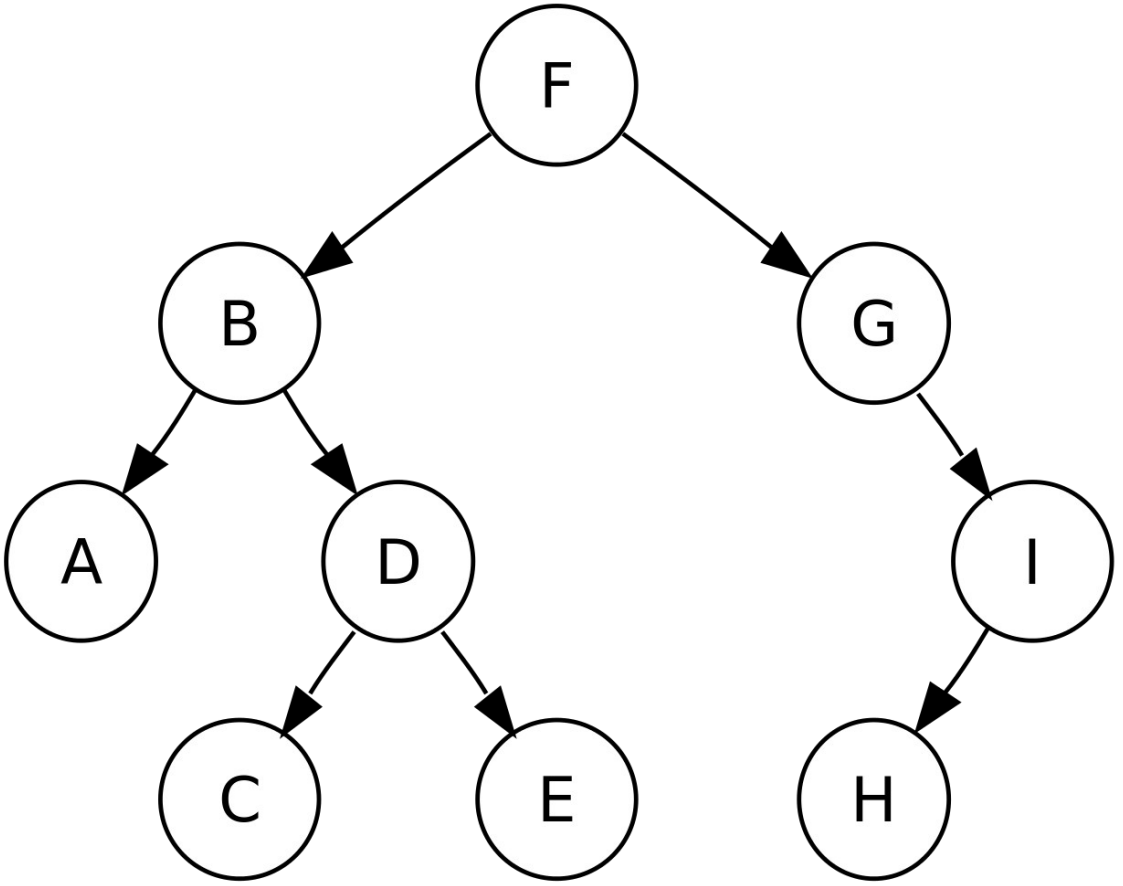


Tipos de recorrido - Postorden

Recursivamente y en orden hacer:

1) Pasar por el nodo izquierdo
2) Pasar por el nodo derecho
3) Visitar el nodo

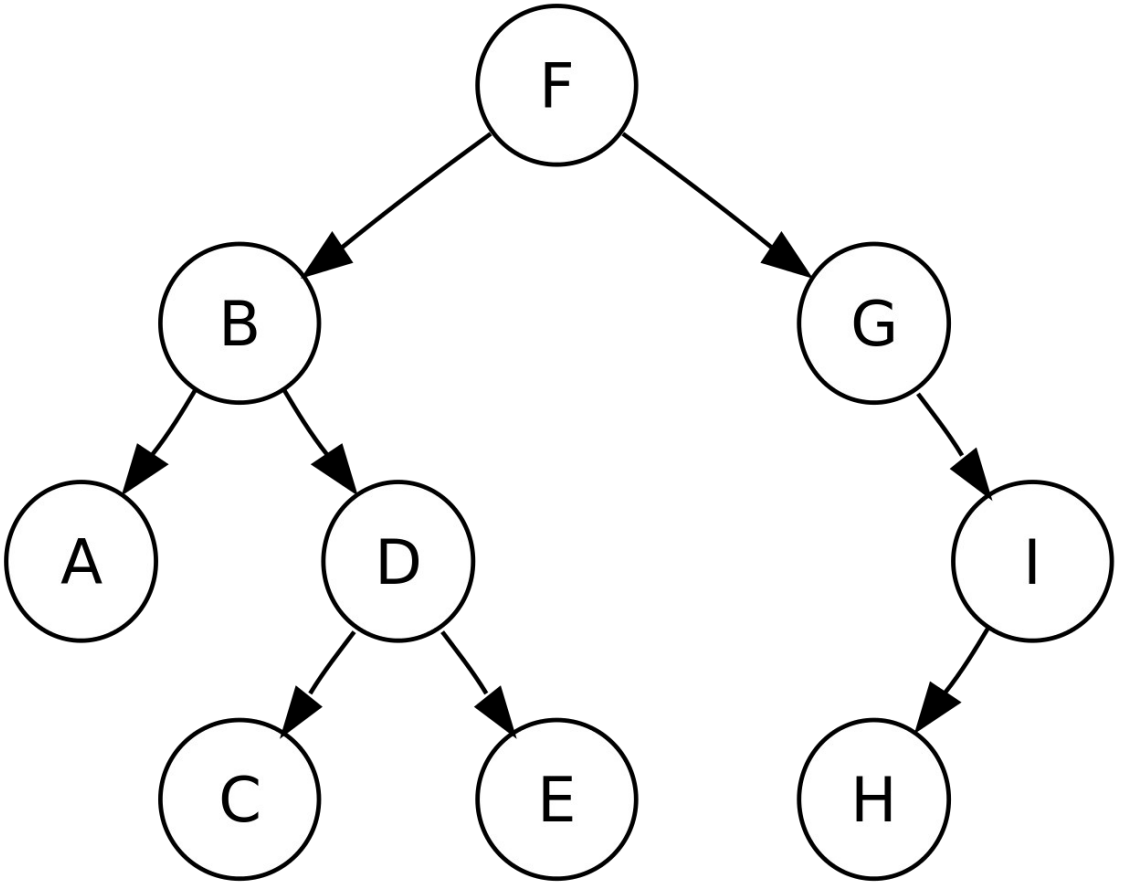
A C E D B H I G F



Tipos de recorrido

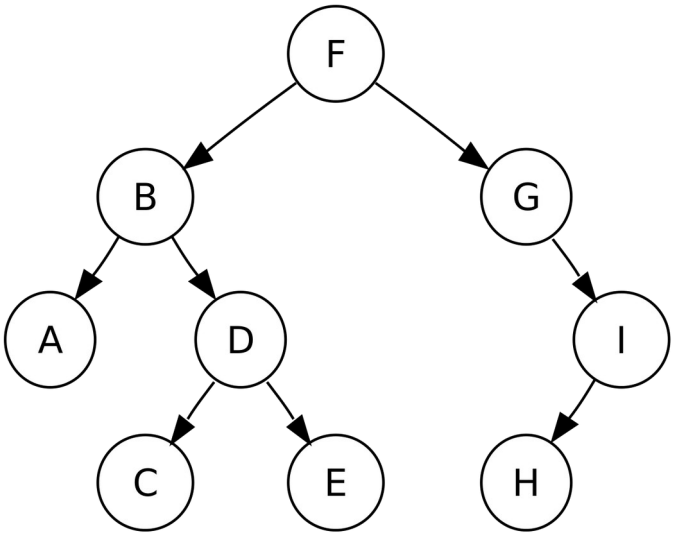
Profundidad en anchura primero: visitar todos los nodos por nivel, una vez concluido se pasa al siguiente nivel hasta el último.

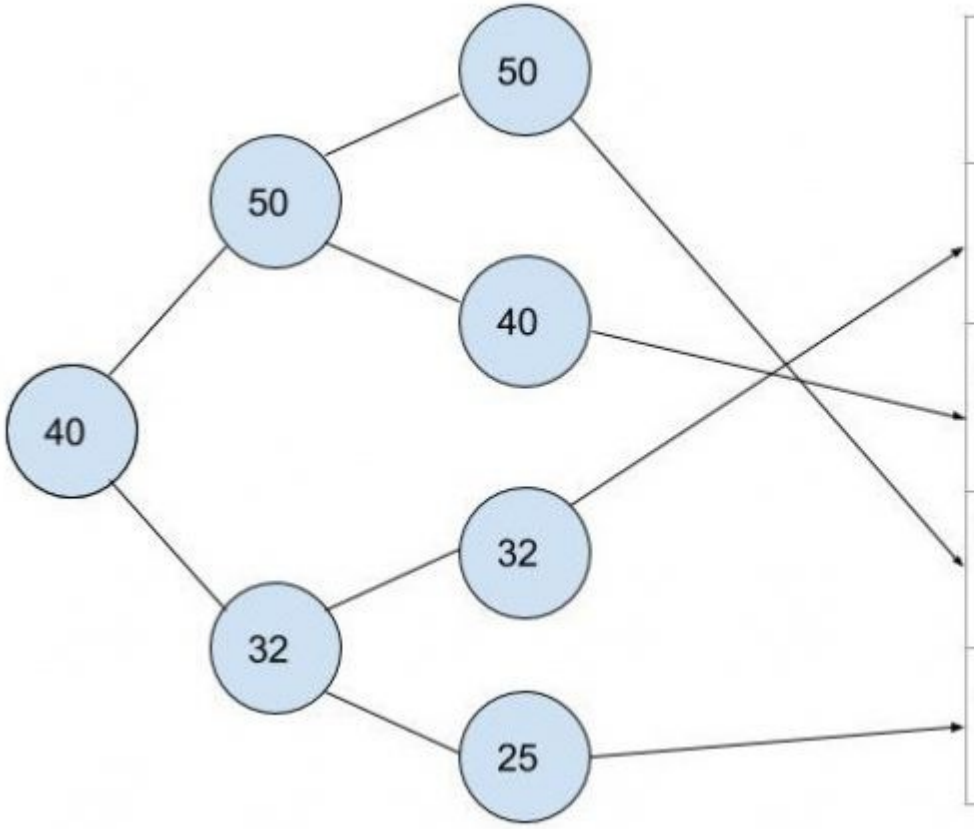
F B G A D I C E H



Inorden – caso especial 1

A B C D E F G H I

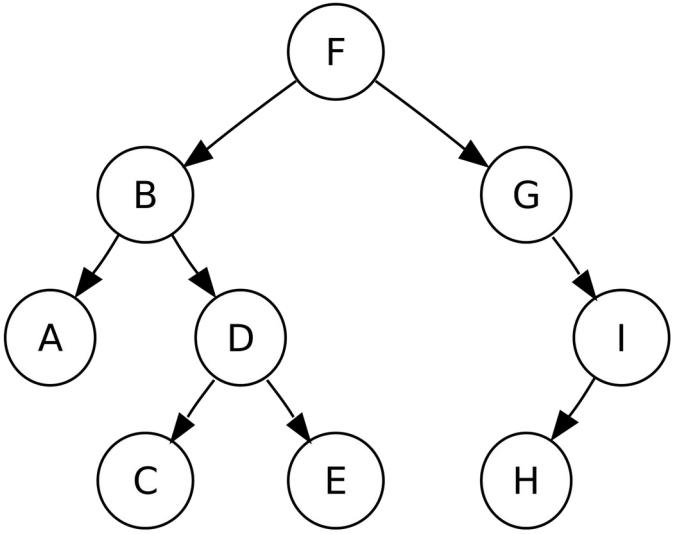


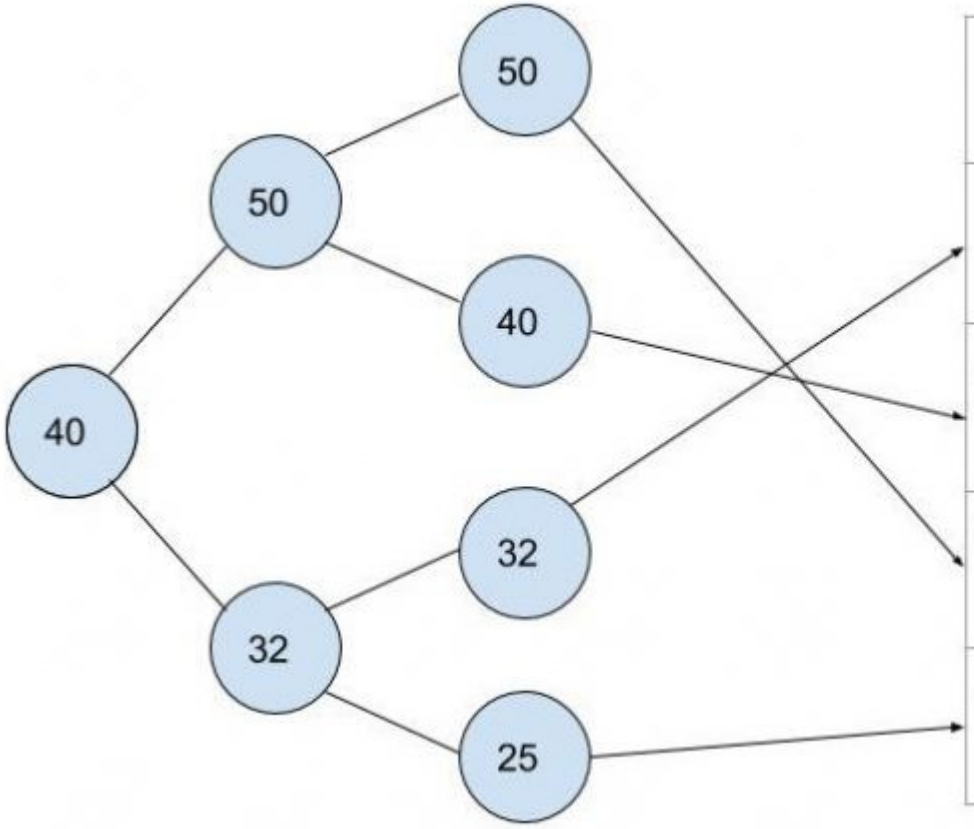


Id	Name	Age
1	Alice	32
2	Bob	40
3	Carol	50
4	Dave	25

Inorden – caso especial 2

Búsqueda binaria





Id	Name	Age
1	Alice	32
2	Bob	40
3	Carol	50
4	Dave	25

Implementación en Python


```
1 class Node:
2       valor = ""
3       left = None
4       right = None
5       def __init__(self, valor):
6           self.valor = valor
7
```