

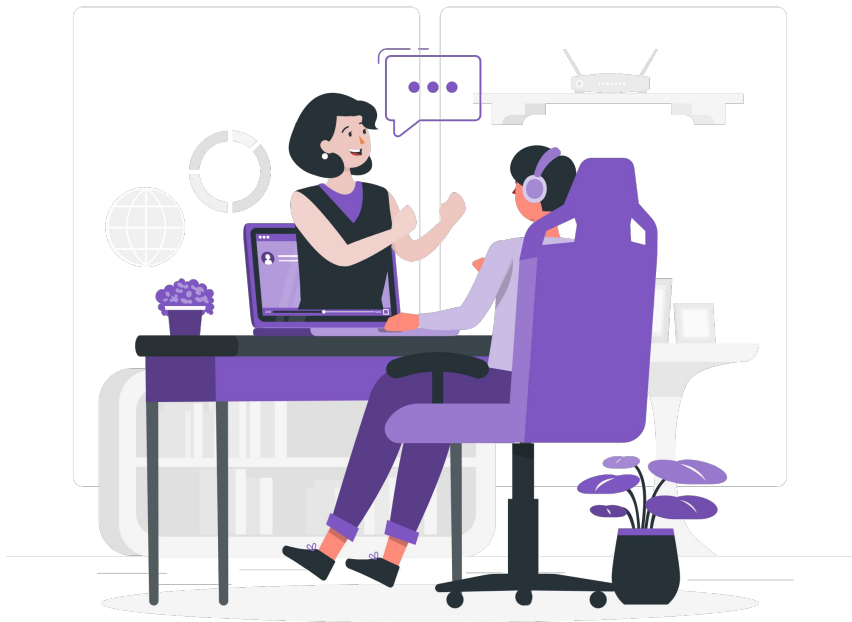
TABLAS HASH

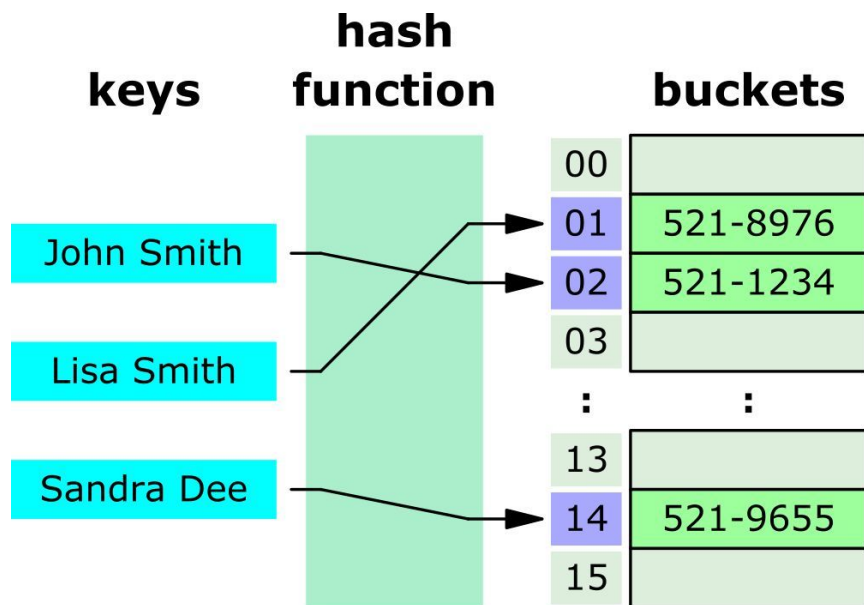
DEV.F
DESARROLLAMOS(PERSONAS);

dev

TABLAS HASH

Son una estructura de datos que permite crear una lista de valores emparejados, esto nos permite recuperar un determinado valor usando la llave de ese valor para el índice.






TABLAS HASH

Una tabla hash transforma una llave en un índice entero, usando una función hash, el índice decidirá dónde almacenar el par llave/valor en la memoria

HASH TABLE TIME COMPLEXITY IN BIG O NOTATION

Algorithm	Average	Worst case
Space	$O(n)$	$O(n)$
Search	$O(1)$	$O(n)$
Insert	$O(1)$	$O(n)$
Delete	$O(1)$	$O(n)$



El ejemplo más común de una tabla hash en JavaScript es el tipo de datos **Object**, donde se puede emparejar el valor de la propiedad del objeto, con una llave que encuentra esa propiedad.

Se le considera un tipo especial de implementación de la tabla hash por dos razones:

- Tiene propiedades añadidas por la clase **Object**. Pero las llaves que introduzcas pueden entrar en conflicto y sobrescribir las propiedades por defecto heredadas de la clase.
- El tamaño de la tabla hash no es rastreado. Es necesario contar manualmente cuantas propiedades son definidas por el programador en lugar de ser heredadas por el prototipo.

HORA DEL CÓDIGO



```
[  
  [ "Spain", 110],  
  [ "France", 100]  
]
```

```
[  
  [  
    [ "Spain", 110 ],  
    [ "á", 192 ]  
  ],  
  [  
    [ "France", 100]  
  ],  
]
```

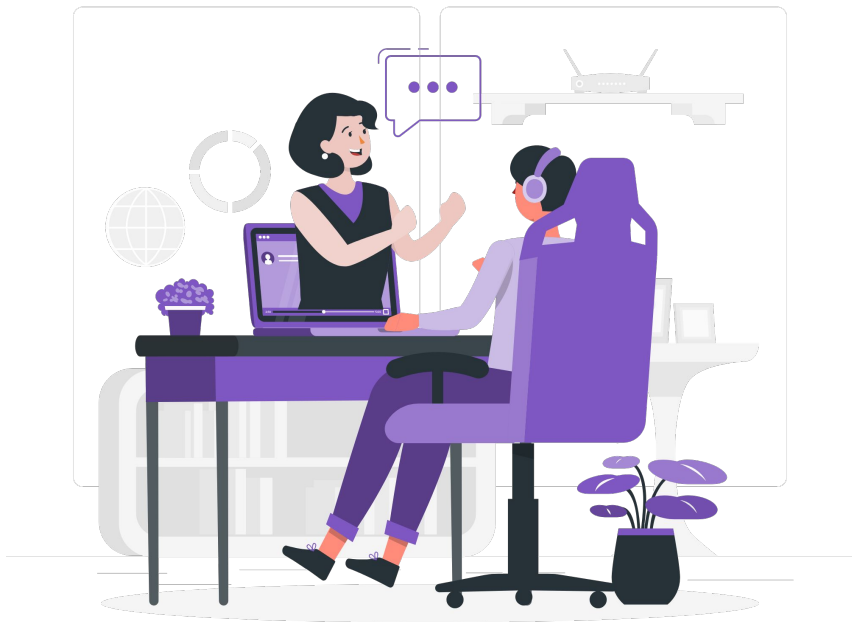
ÁRBOLES BINARIOS

DEV.F
DESARROLLAMOS(PERSONAS);

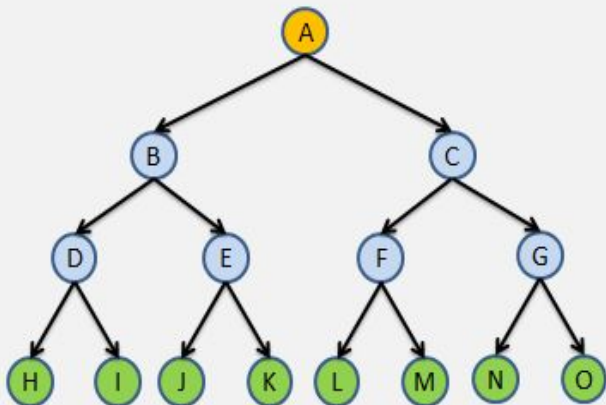
dev

ÁRBOLES BINARIOS

Los árboles binarios son una estructura de datos que nos ayuda a organizar información en memoria, una característica es que un árbol tiene la capacidad de almacenar nodos cuantos sean necesarios.



Estructura Jerárquica



V.S.

Estructura Lineal



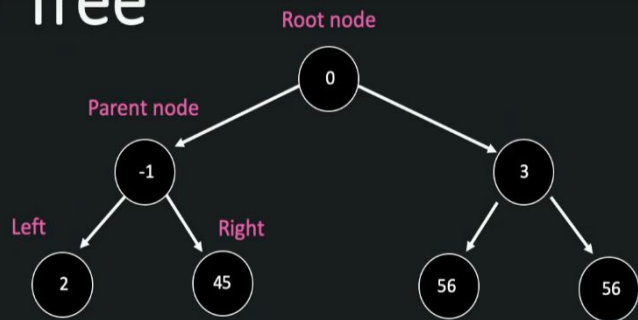
La imagen muestra la diferencia entre las estructuras de datos líneas y las no lineales como lo son los Árboles.

ARBOLES

Los árboles se caracterizan por almacenar sus nodos en forma jerárquica y no en forma lineal como lo vimos con las listas ligadas, colas y pilas.

Los nodos pueden almacenar nuestros elementos html, números, palabras, todo tipo de dato que se pueda almacenar

Binary Tree



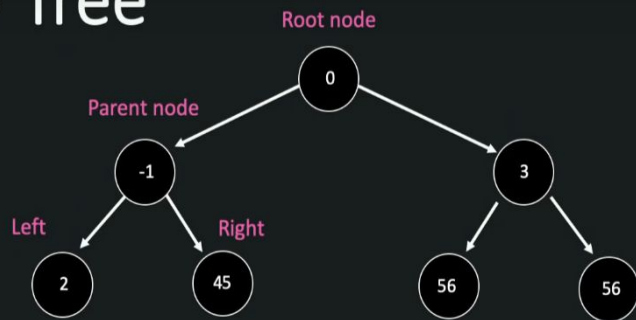
ÁRBOL BINARIO

HAY UNA REGLA QUE NOS PODRÁ
AYUDAR A ENTENDER CÓMO SE
TRABAJAN LOS ÁRBOLES BINARIOS

Y ES QUE DENTRO DE LOS NODOS
PADRES

SOLO ESTÁ PERMITIDO QUE TENGAN DOS
NODOS HIJOS.

Binary Tree



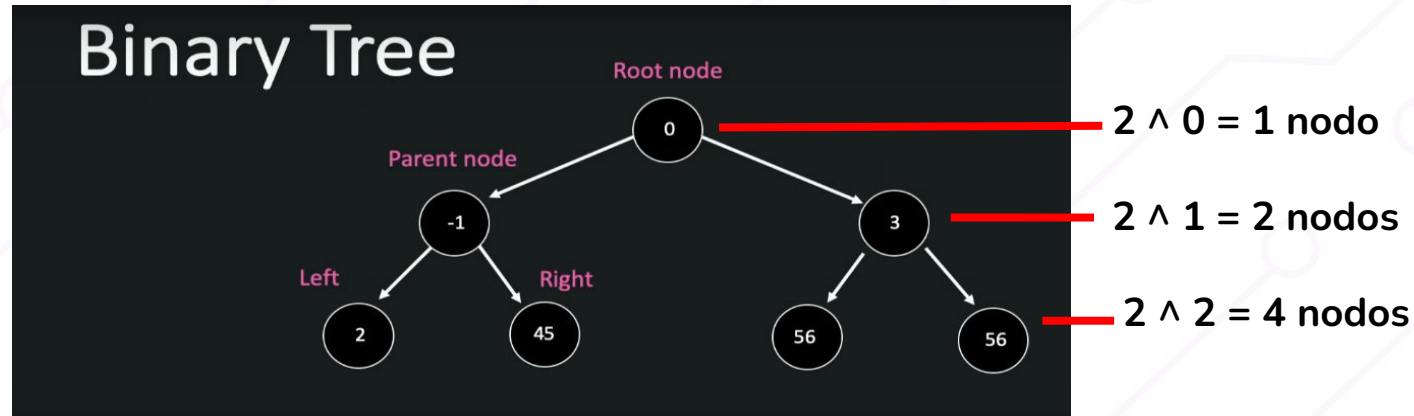
ÁRBOL BINARIO

El “nodo” es la estructura básica que usamos para construir un “árbol”.

Todos los elementos de un árbol son nodos. A su vez, cada nodo es un sub-árbol. Los nodos se caracterizan por tener un valor, y referencias a otros nodos.

NIVELES

Cómo dato interesante de los árboles binarios, los nodos son potencia de 2.



Nota: recuerda que solo tenemos dos nodos como máximo

Datos importantes de los Árboles

Para comprender mejor qué es un árbol comenzaremos explicando como está estructurado:

Nodos: Se le llama Nodo a cada elemento que contiene un Árbol.

Nodo Raíz: Se refiere al primer nodo de un Árbol, Solo un nodo del Árbol puede ser la Raíz.

Nodo Padre: Se utiliza este término para llamar a todos aquellos nodos que tiene al menos un hijo.

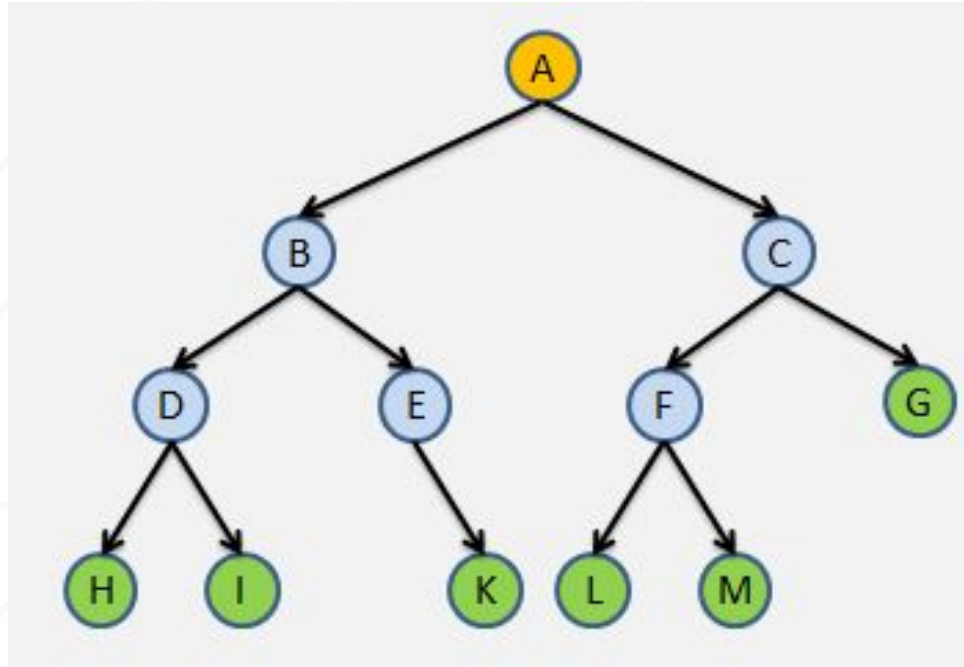
Nodo Hijo: Los hijos son todos aquellos nodos que tiene un padre.

Nodo Hermano: Los nodos hermanos son aquellos nodos que comparte a un mismo padre en común dentro de la estructura.

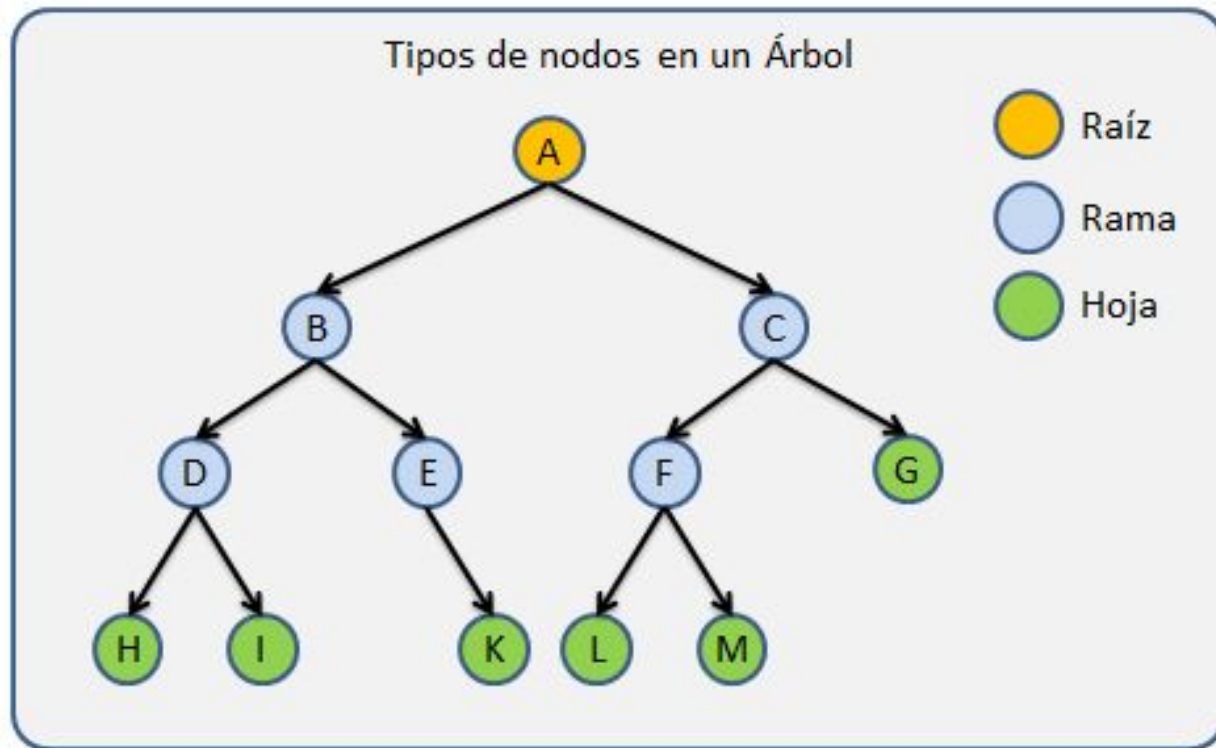
Nodo Hoja: Son todos aquellos nodos que no tienen hijos, los cuales siempre se encuentran en los extremos de la estructura.

Nodo Rama: Estos son todos aquellos nodos que no son la raíz y que además tiene al menos un hijo.

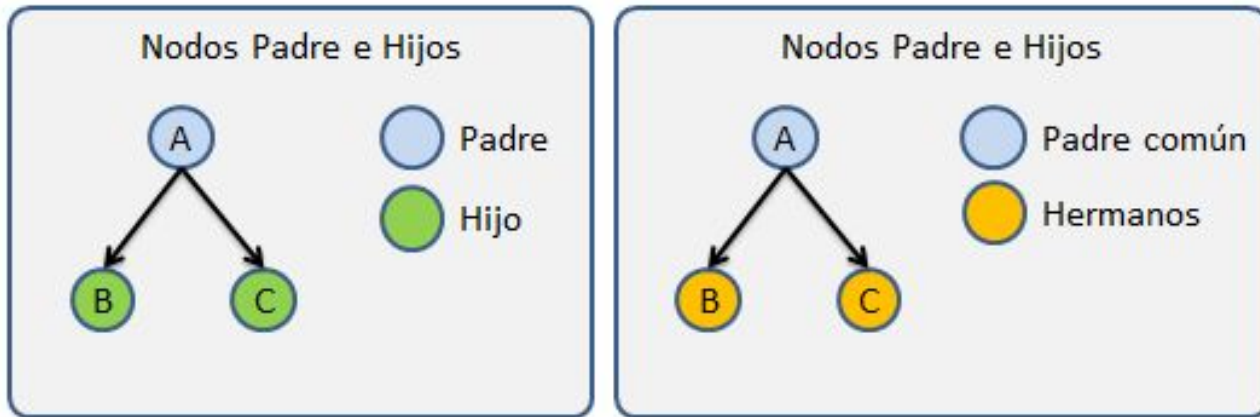
Nodos Gráficamente



Nodos Gráficamente



Relación entre Nodos



La siguiente imagen muestra de forma gráfica los
nodos Padre, Hijo y Hermanos



Dato interesante

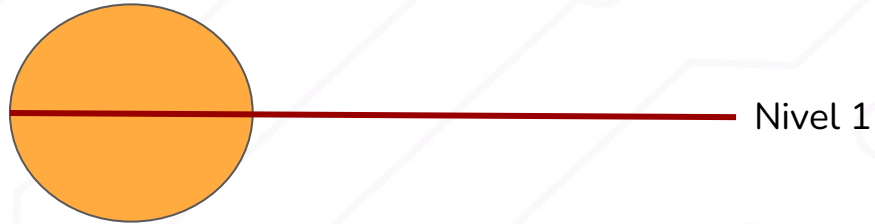
Los árboles además de los nodos tienen otras propiedades importantes que son utilizadas en diferentes ámbitos .

Nivel

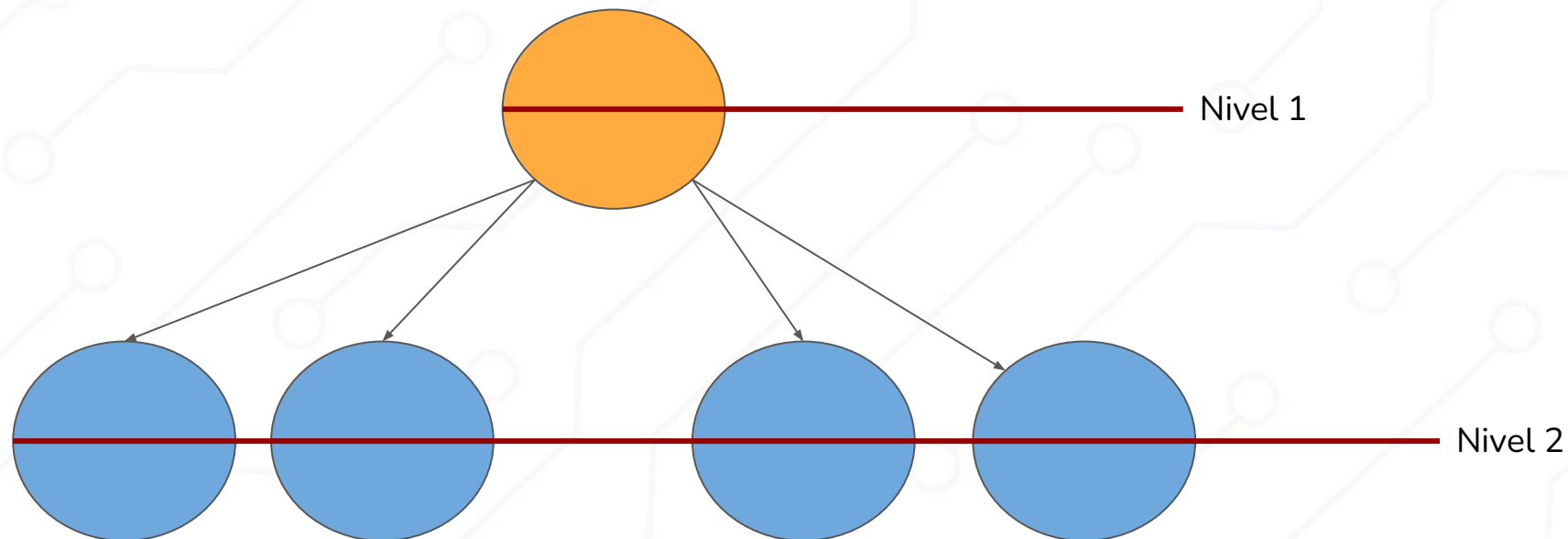
1. Nos referimos como nivel a cada generación dentro del árbol.
 2. Por ejemplo, cuando a un nodo hoja le agregamos un hijo, el nodo hoja pasa a ser un nodo rama pero además el árbol crece una generación por lo que el Árbol tiene un nivel más.
 3. Cada generación tiene un número de Nivel distinto que las demás generaciones.
- Un árbol vacío tiene 0 niveles
 - El nivel de la Raíz es 1
 - El nivel de cada nodo es calculado contando cuántos nodos existen sobre el, hasta llegar a la raíz + 1, y de forma inversa también se podría, contar cuántos nodos existes desde la raíz hasta el nodo buscado + 1.

Raíz

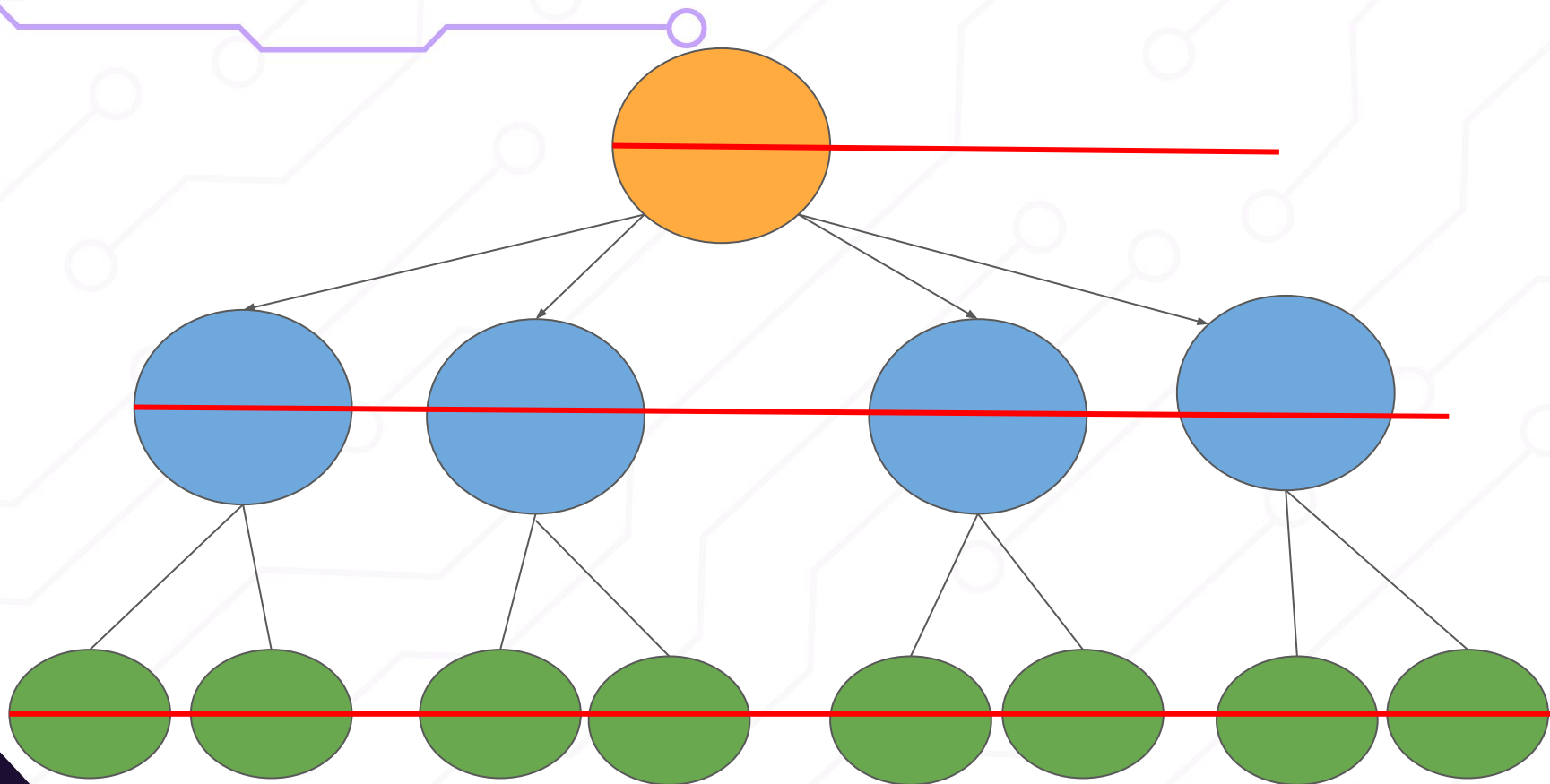
- Un árbol vacío tiene 0 niveles
- El nivel de la Raíz es 1



Raiz y Padres

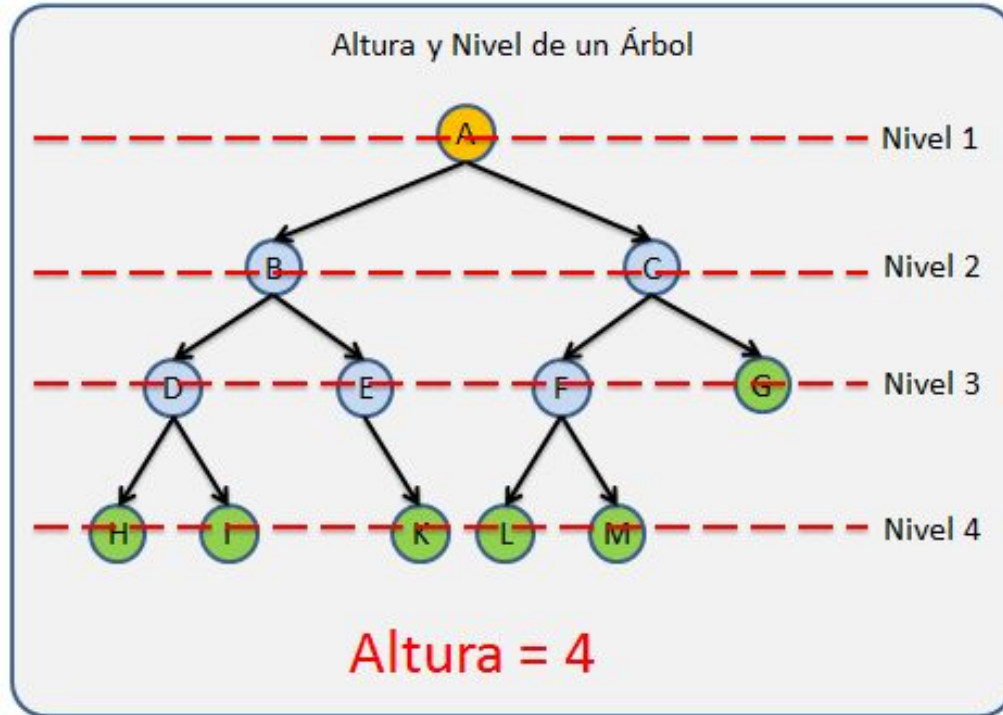


Raiz, Padres e Hijos



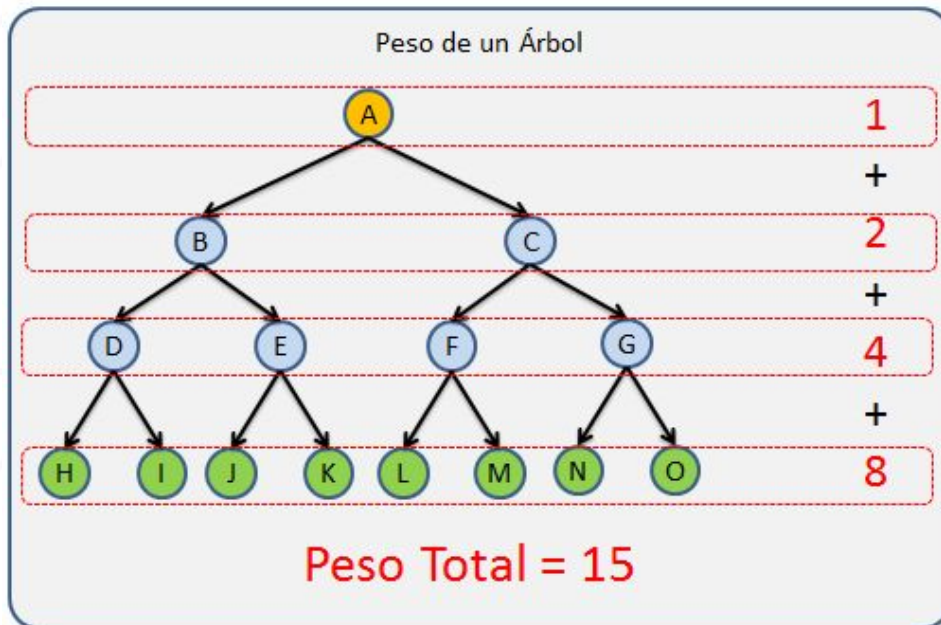
Altura y nivel de un árbol

Altura: Le llamamos Altura al número máximo de niveles de un Árbol.



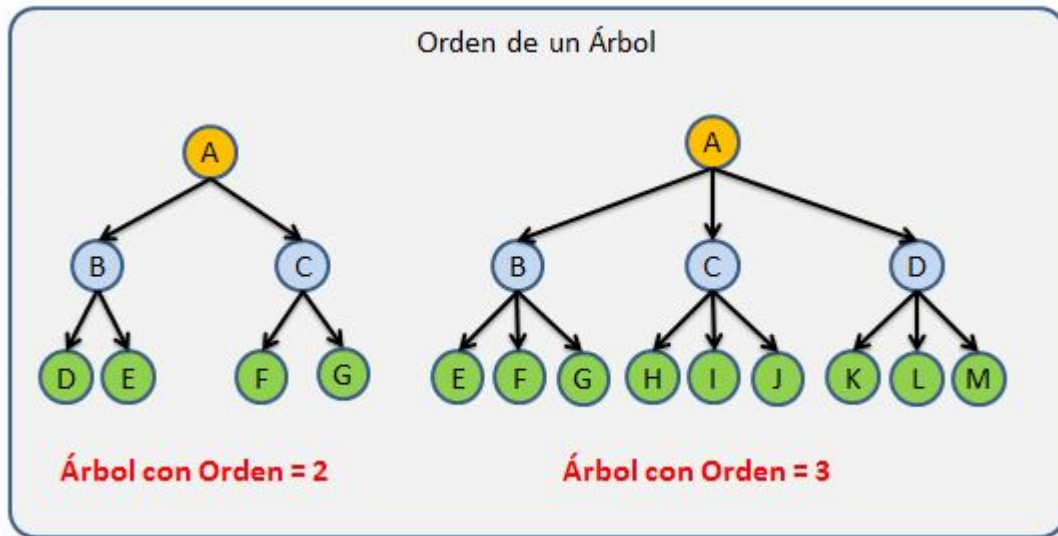
Peso de un Árbol

Peso: Conocemos como peso a el número de nodos que tiene un Árbol. Este factor es importante por que nos da una idea del tamaño del árbol y el tamaño en memoria que nos puede ocupar en tiempo de ejecución (Análisis de algoritmos.)



Orden de un árbol

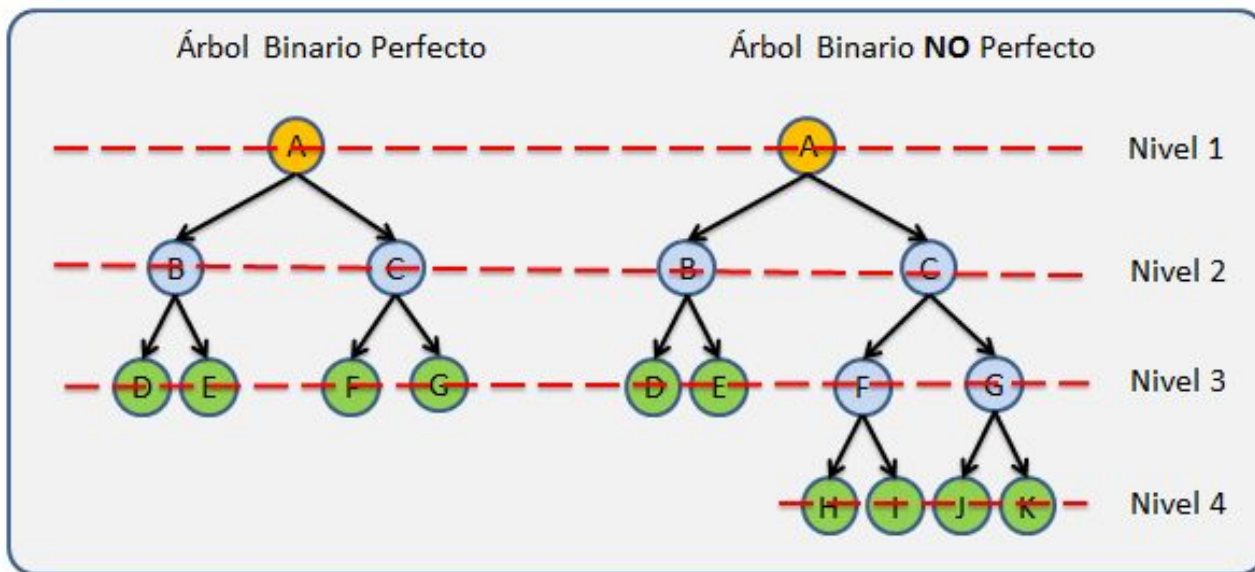
Orden: El Orden de un árbol es el número máximo de hijos que puede tener un Nodo.



Árbol binario perfecto

Árbol binario perfecto: Es un Árbol lleno en donde todas las Hojas están en el mismo Nivel.

Como podemos apreciar el árbol de la izquierda tiene todas sus hojas al mismo nivel y que además está lleno, lo que lo convierte en un árbol binario perfecto.



Sin embargo, del lado derecho podemos ver que aunque el árbol está lleno no tiene todas las hojas al mismo nivel lo que hace que no sea un árbol binario perfecto pero si lleno.

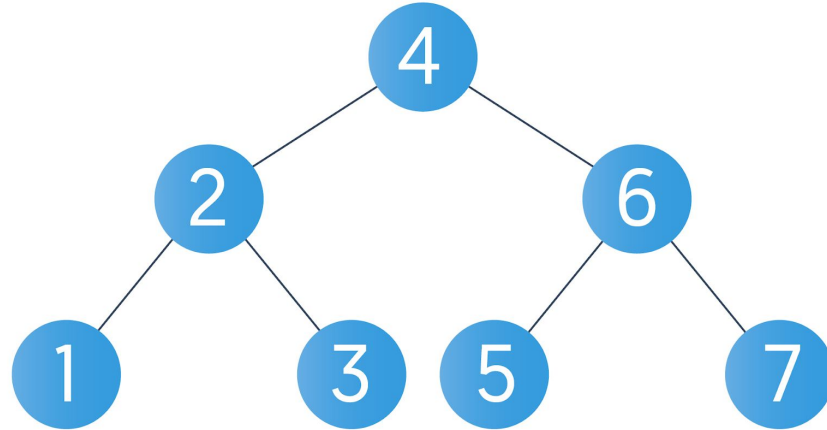
HORA DEL CÓDIGO



En el orden **preorden** se recorre de la siguiente manera: raíz, subárbol izquierdo, subárbol derecho.

En el orden **inorden** se recorre de la siguiente manera: subárbol izquierdo, raíz, subárbol derecho.

En el orden **postorden** se recorre de la siguiente manera: subárbol izquierdo, subárbol derecho, raíz.



In order traversal- 1 2 3 4 5 6 7

Pre order traversal- 4 2 1 3 6 5 7

Post order traversal- 1 3 2 5 7 6 4