Unidad 4 Estructuras no lineales

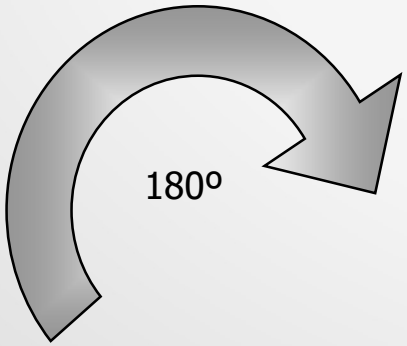
1. Árboles
2. Grafos

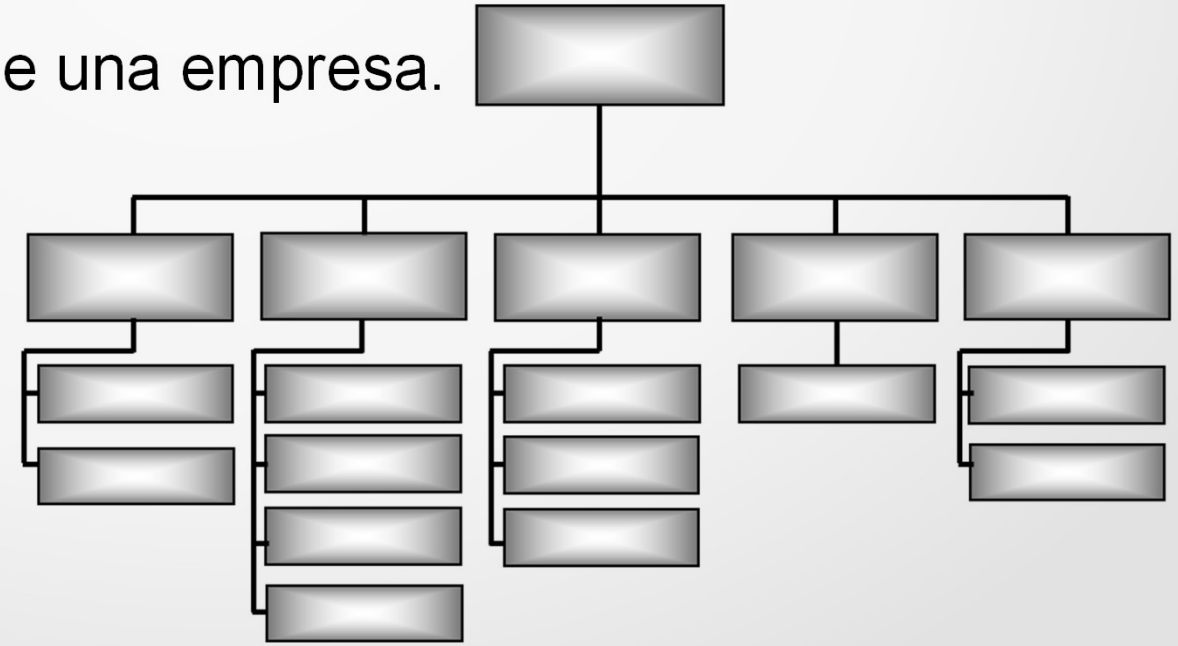
4.1.1 Clasificación de Árboles

Una estructura de árbol es una forma de representar la **JERARQUÍA** de ciertos objetos en una forma gráfica.

Se le llama **árbol** porque la gráfica se asemeja a un árbol natural, aunque, a diferencia de los arboles naturales, estas estructuras jerárquicas crecen hacia abajo, es decir, la raíz se encuentra arriba y las hojas en la parte inferior.

Ejemplo: El organigrama de una empresa.





4.1.1 Clasificación de Árboles

Clasificación de árboles.

En la teoría computacional, los árboles tienen diversos usos.

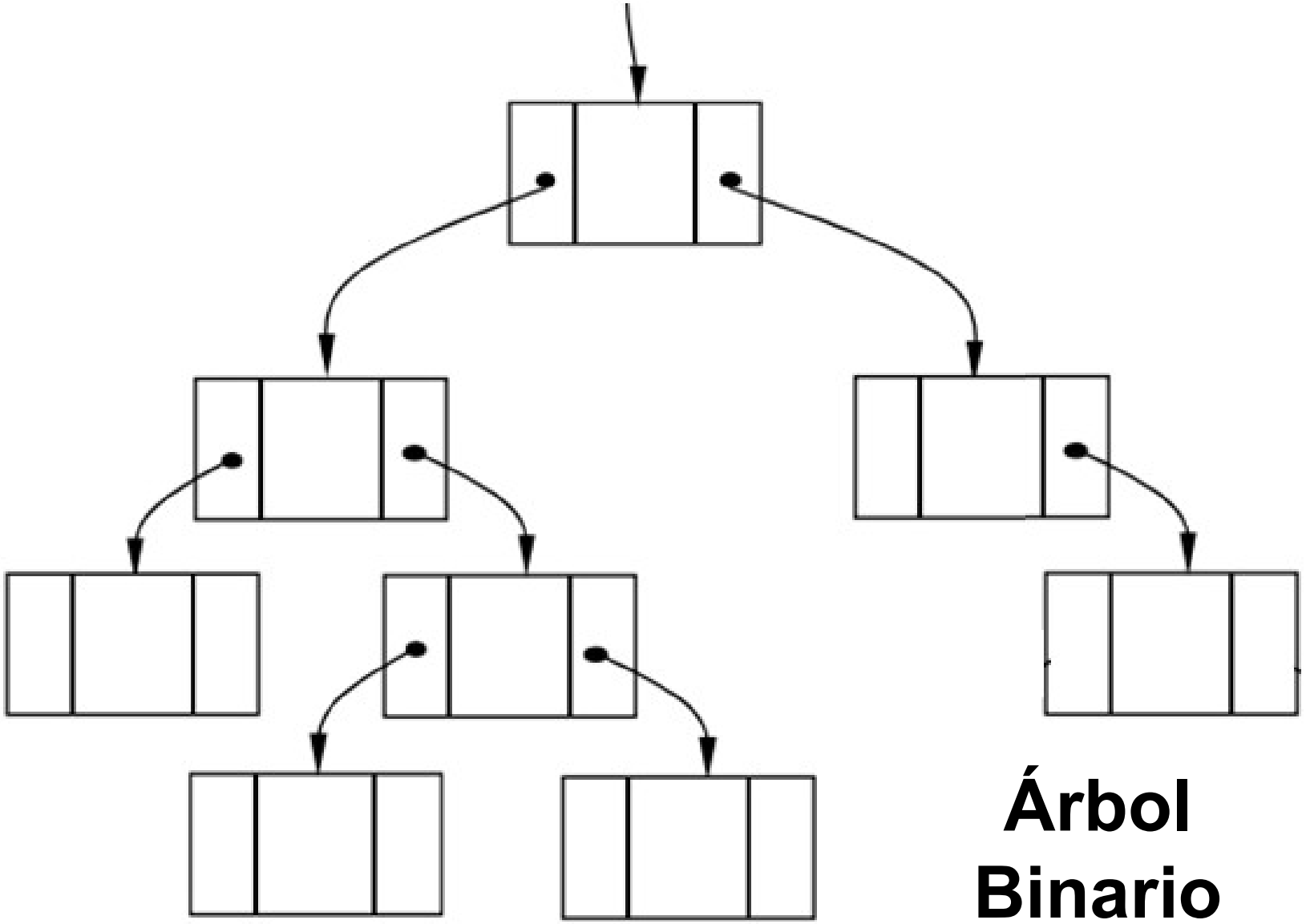
La clasificación que nos interesa, por ahora, es la relativa al número de nodos a que puede apuntar otro nodo y al grado de eficiencia que puede aportar un árbol para la búsqueda de datos almacenados en él.

4.1.1 Clasificación de Árboles

Arboles

- **Binarios**
  Cada nodo solo puede apuntar a otros 2 nodos como máximo
  - **Balanceados**
    Garantiza cierto grado de eficiencia durante su operación
  - **Desbalanceados**
- **No Binarios**
  Cada nodo puede apuntar a cualquier cantidad de nodos
  - **Balanceados**
    Garantiza cierto grado de eficiencia durante su operación
  - **Desbalanceados**

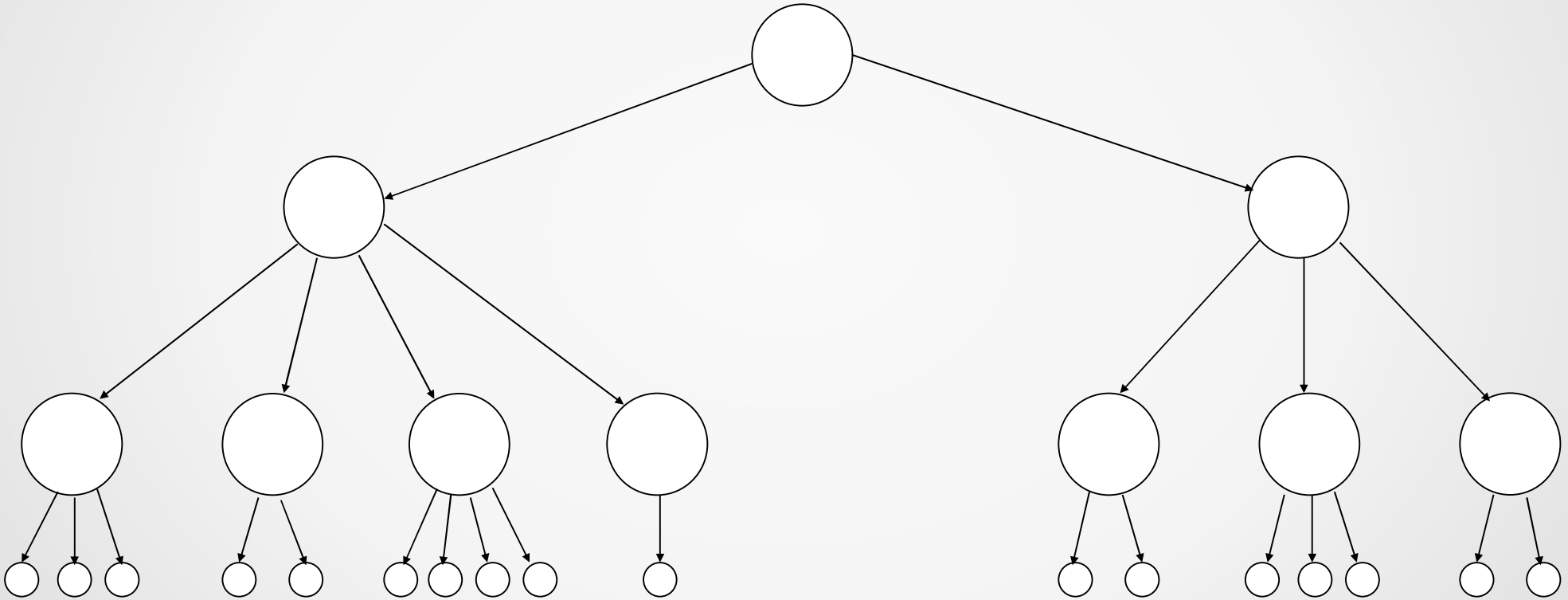
4.1.1 Clasificación de Árboles



Árbol Binario

4.1.1 Clasificación de Árboles

Árbol No Binario



4.1.2 Operaciones Básicas sobre AB

4.2 Arboles Binarios. Operaciones Básicas.

Si se desea localizar cierto dato en una lista lineal ordenada MUY GRANDE, el proceso puede tomar mucho tiempo.

Las estructuras NO LINEALES incluyen más de una ruta para organizar los datos.

Por lo tanto las búsquedas sobre estructuras NO LINEALES requieren menos tiempo.

Los Árboles Binarios son estructuras NO LINEALES, y si se implementan en memoria dinámica, son NO SECUENCIALES.

4.1.2 Operaciones Básicas sobre AB

Una aplicación de los árboles, es la Búsqueda.

Significa que se conservan en los nodos del árbol datos únicos (Llaves de Identificación), para fines, entre otros, de localización rápida.

Los nodos de un Árbol Binario de Búsqueda pueden apuntar a otros dos, uno que le precede en la lista y otro que le sucede.

LISTA ENCADENADA DOBLE ORDENADA

- **Apuntador al Nodo Predecesor Inmediato (El dato menor más cercano de toda la lista).**
- **Apuntador al Nodo Sucesor Inmediato (El dato mayor más cercano de toda la lista).**

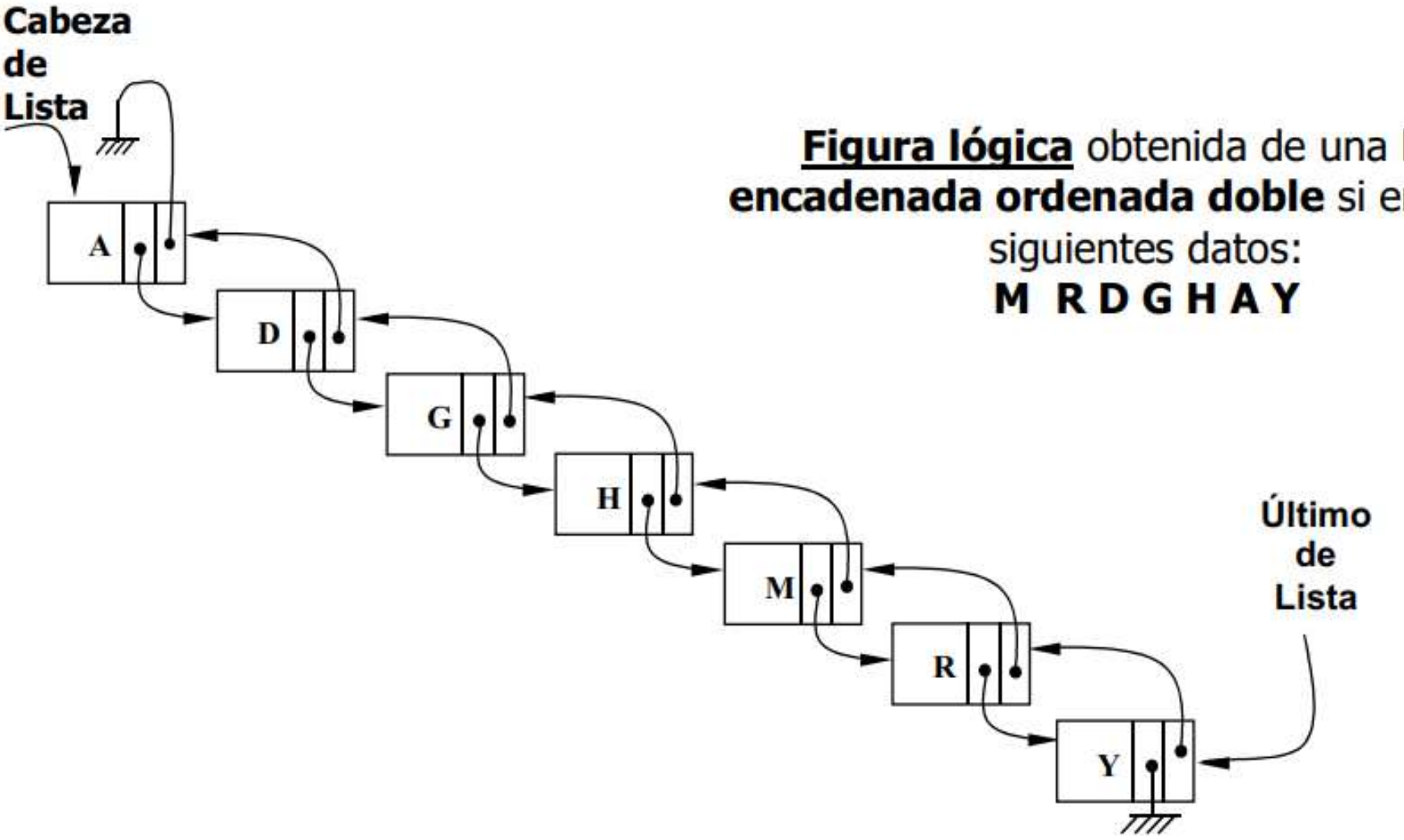
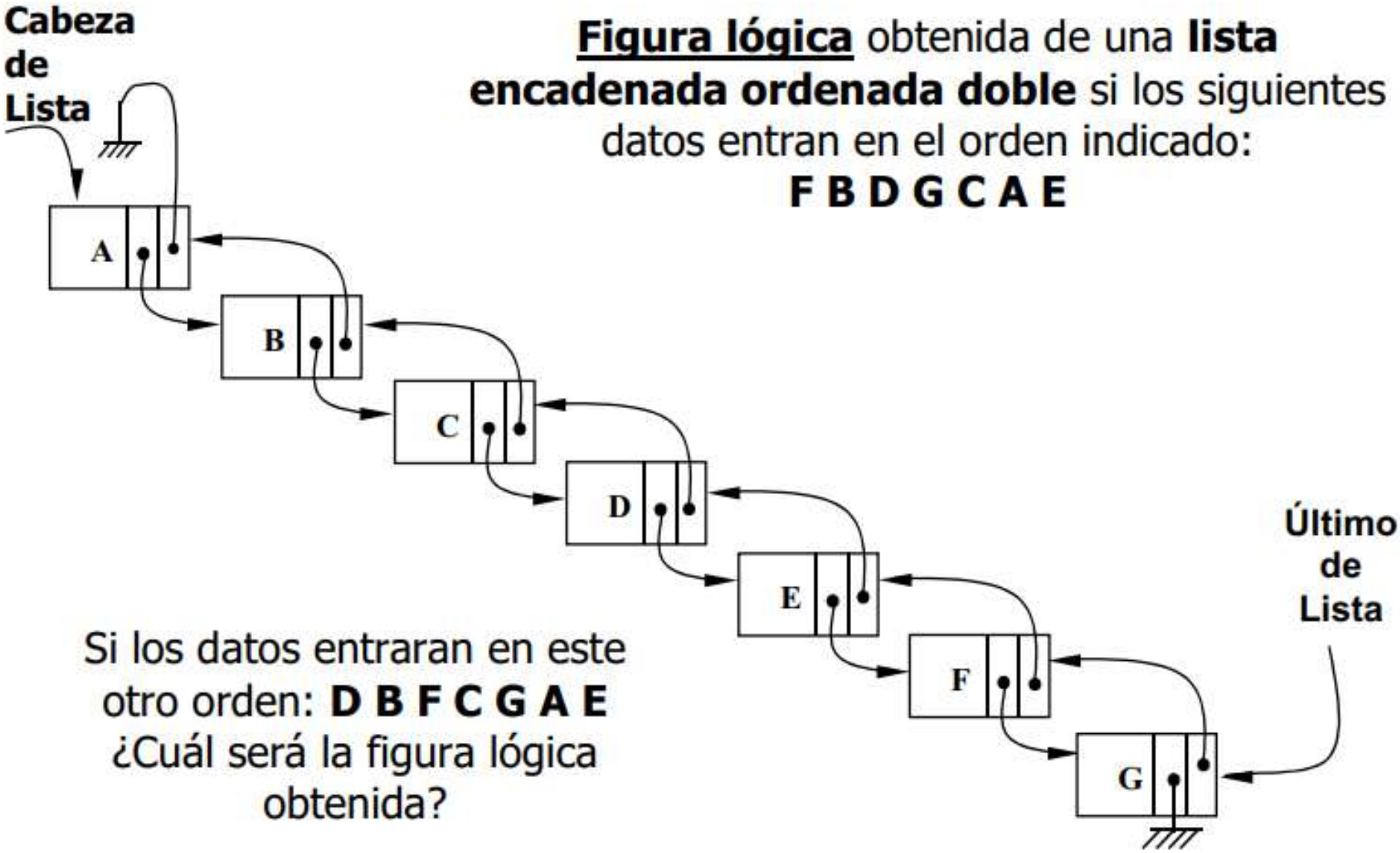


Figura lógica obtenida de una **lista encadenada ordenada doble** si entran los siguientes datos:

M R D G H A Y

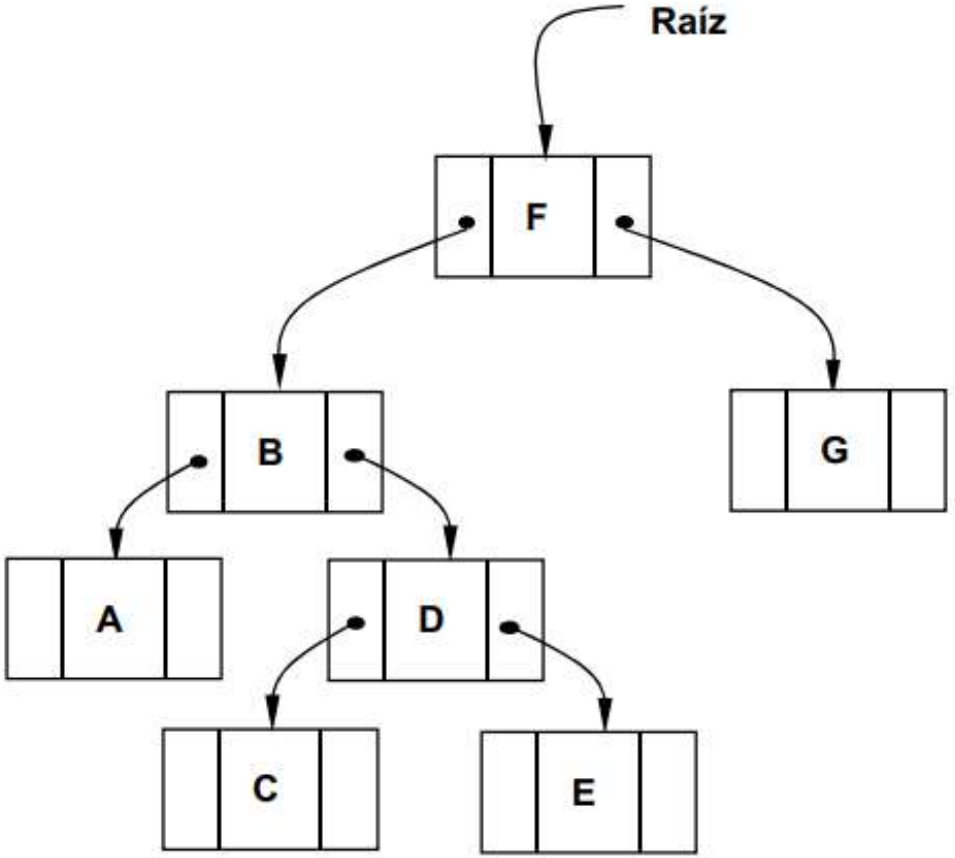
4.1.2 Operaciones Básicas sobre AB



ARBOL BINARIO DE BUSQUEDA

- **Apuntador al Nodo IZQUIERDO (Cualquiera de los datos de la lista que sea Menor).**
- **Apuntador al Nodo DERECHO (Cualquiera de los datos de la lista que sea Mayor).**

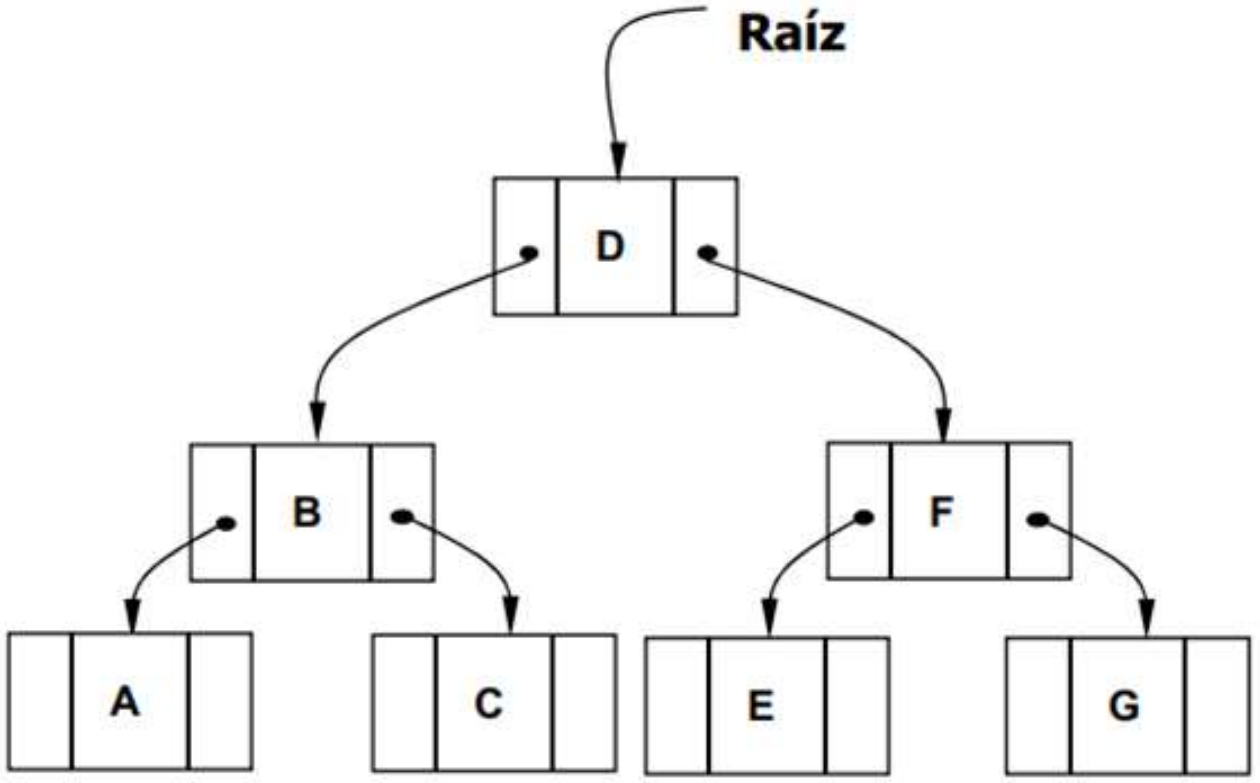
Si los datos **" F B D G C A E "** se guardan en un **árbol binario de BUSQUEDA**, la figura lógica resultante sería como la siguiente:



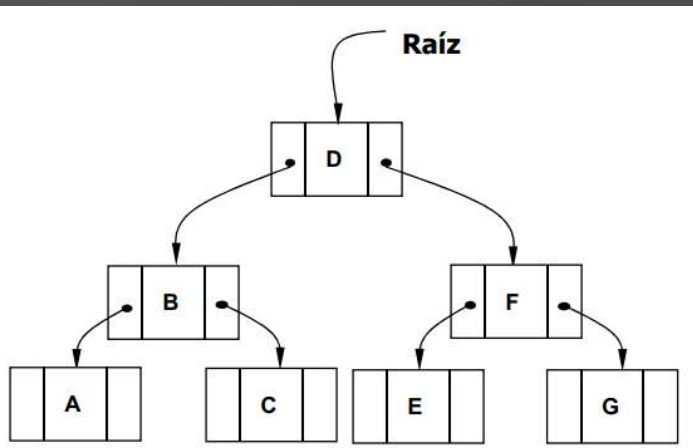
- **Cada nodo apunta a un dato menor por la izquierda.**
- **Cada nodo apunta a un dato mayor por la derecha.**
- **Se requiere un apuntador al primer nodo de la lista (RAIZ).**

4.1.2 Operaciones Básicas sobre AB

Si los datos entrasen en el siguiente orden: " **D B F C G A E** " la forma lógica resultante sería:



¿Por qué un árbol binario teóricamente es mejor que una lista encadenada?



¿Cuántos nodos se deben visitar para encontrar el dato "G"?

En un árbol: 3

En una lista encadenada: 7

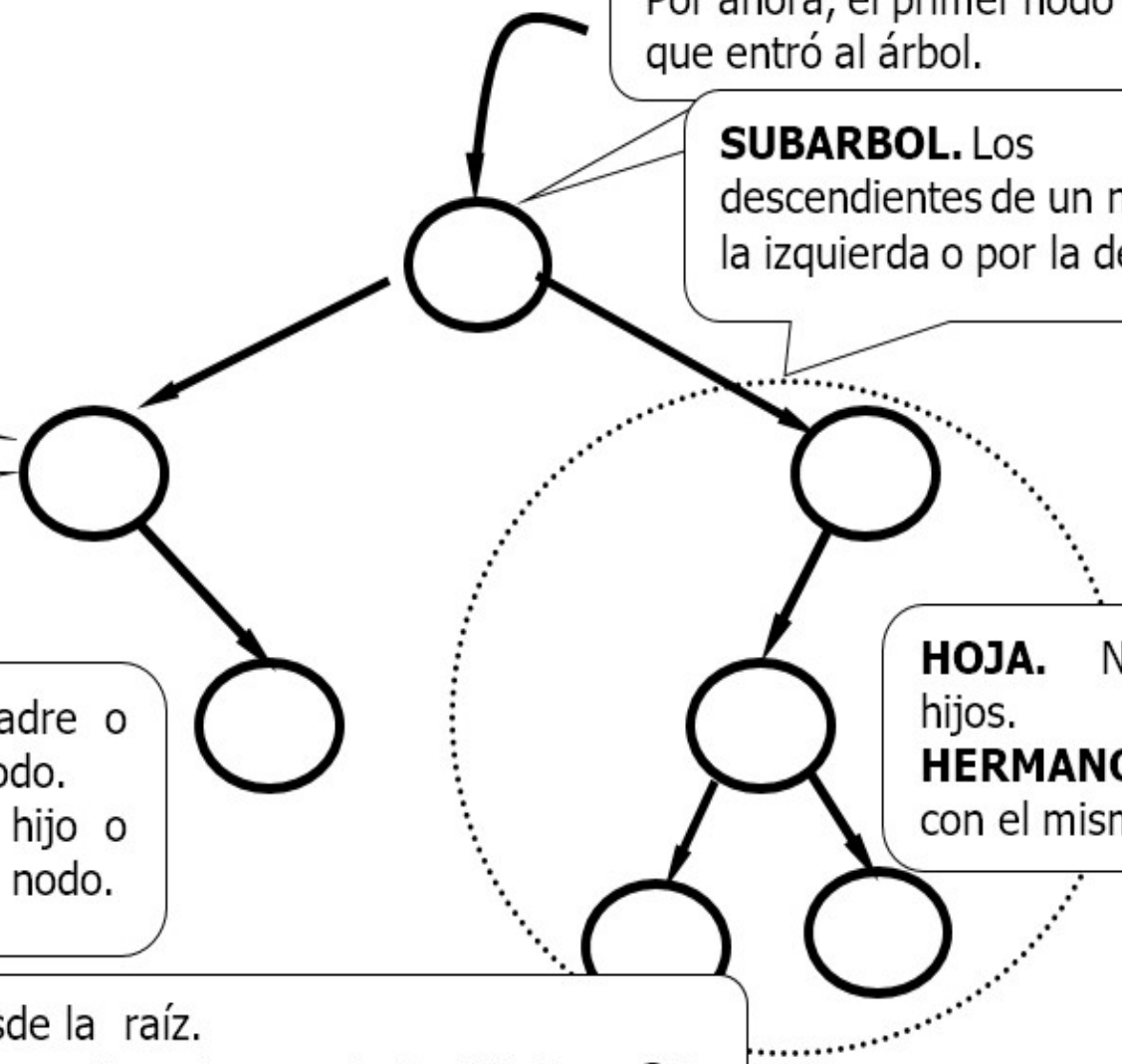
Si se construye un árbol COMPLETO (todos los niveles con el número máximo de nodos) con 1023 datos .....

¿Cuántos nodos en promedio se visitarán suponiendo que se hacen 1023 búsquedas, cada vez un dato distinto y todos existen?

En una lista encadenada: 512

En un árbol: 9.01

Vocabulario Básico



RAIZ
Por ahora, el primer nodo que entró al árbol.

SUBARBOL. Los descendientes de un nodo por la izquierda o por la derecha.

HIJO
Nodo apuntado por otro. Si está apuntado por la izquierda es hijo izquierdo, si es por la derecha, es hijo derecho.

PADRE
Nodo que apunta a uno o dos nodos.

HOJA. Nodo sin hijos.
HERMANOS. Nodos con el mismo padre.

ANTEPASADO. Nodo que es padre o ANTEPASADO del padre de otro nodo.
DESCENDIENTE. Nodo que es hijo o DESCENDIENTE de un hijo de otro nodo.

NIVEL. Distancia de un nodo desde la raíz.
Nivel de la raíz = 0. Número máximo de nodos en el **nivel "n"** es $\mathbf{2^n}$.

4.1.2 Operaciones Básicas sobre AB

Creación e Inserción en Arboles Binarios de Búsqueda.

- Todos los árboles inician vacíos.
- Para crear un árbol solo se requiere una variable apuntador que señale a un árbol nulo (FIN DE LISTA).
- Para añadir nuevos nodos a un ABB hay que hacer una búsqueda del lugar que ocupará cada nodo nuevo.
- La búsqueda en un ABB consiste en mover un apuntador a la izquierda o a la derecha hasta encontrar el lugar que debe ocupar el nuevo nodo.
- La única consideración especial que se debe hacer es cuando el árbol está vacío.

4.1.2 Operaciones Básicas sobre AB

El algoritmo siguiente es un proceso de búsqueda para localizar cierto valor dentro de un árbol. Establezcamos que en un árbol binario de búsqueda, a la izquierda de cualquier nodo se guardan valores menores o iguales y a la derecha los mayores.

```
Temp ← RAIZ
MIENTRAS Temp != NULO & INFO(Temp) != VALOR
        SI VALOR > INFO(Temp)
            Temp ← DER(Temp)
        DE LO CONTRARIO
             Temp ← IZQ(Temp)
        FIN SI
FIN MIENTRAS
```

4.1.2 Operaciones Básicas sobre AB

Este algoritmo deberá adecuarse a las características de cada proceso de búsqueda particular:

- Para dar de alta un nuevo dato.
- Cuando en el nodo se guarda información muy completa y se quiere consultar.
  - Puede haber muchos datos en un nodo, correspondientes a una persona por ejemplo, sin embargo el que determina su lugar en el árbol, solo es uno de ellos y se le llama LLAVE.
- Para eliminar un nodo.

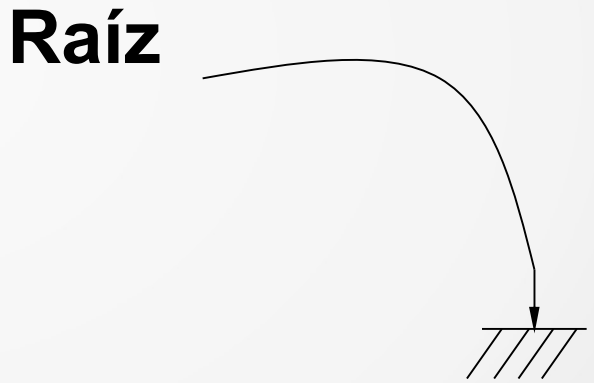
4.1.2 Operaciones Básicas sobre AB

En un ABB cada nuevo nodo se añade como una HOJA, por lo tanto hay que adecuar el algoritmo de búsqueda que se vio anteriormente para llegar a un nodo con un NULO en uno de los campos de apuntador mientras se viaja por él.

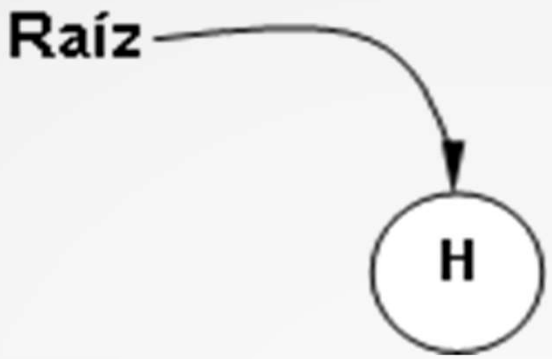
Antes de analizar los algoritmos de creación, primero se debe aprender a crear ABB. Se ejercitará con las llaves siguientes en el orden indicado: H Q M K A F O.

4.1.2 Operaciones Básicas sobre AB

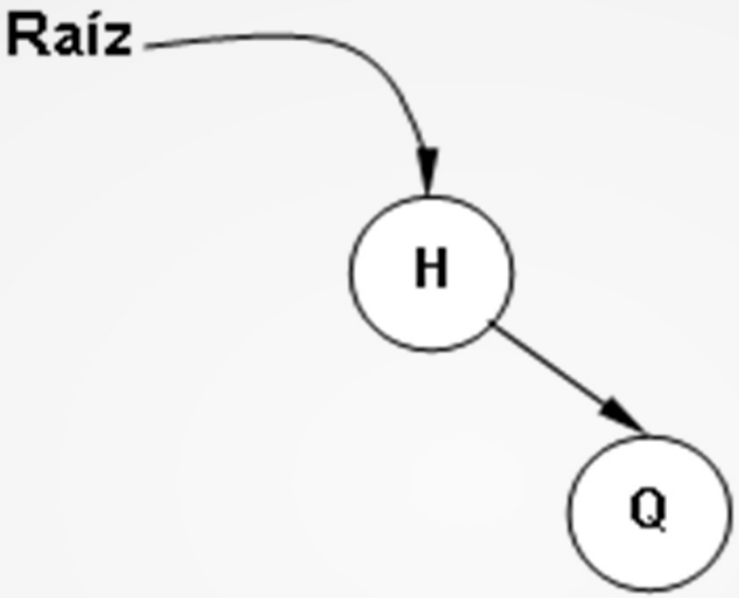
Estado Inicial



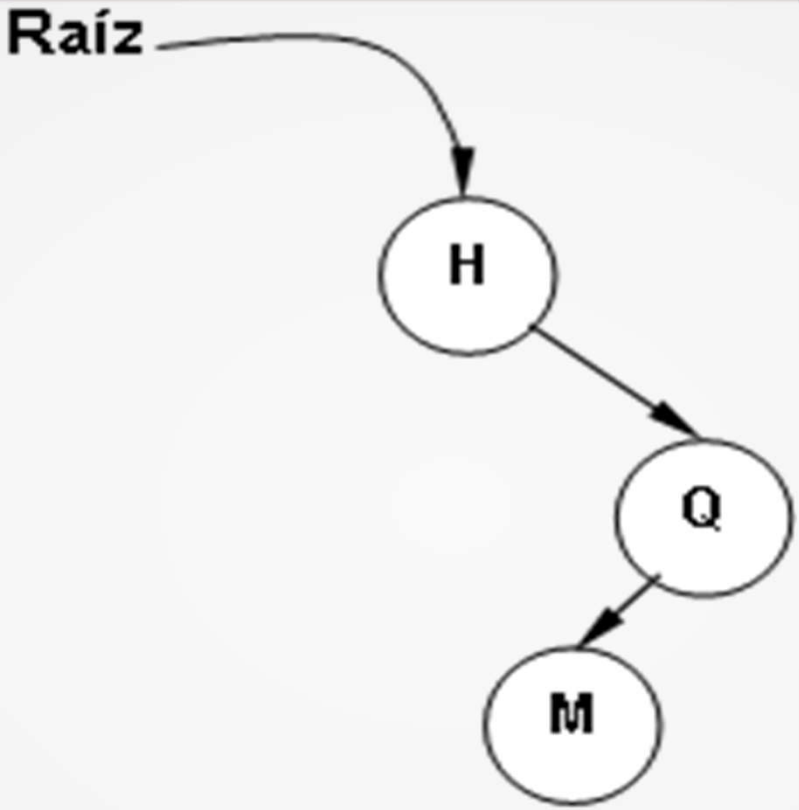
4.1.2 Operaciones Básicas sobre AB



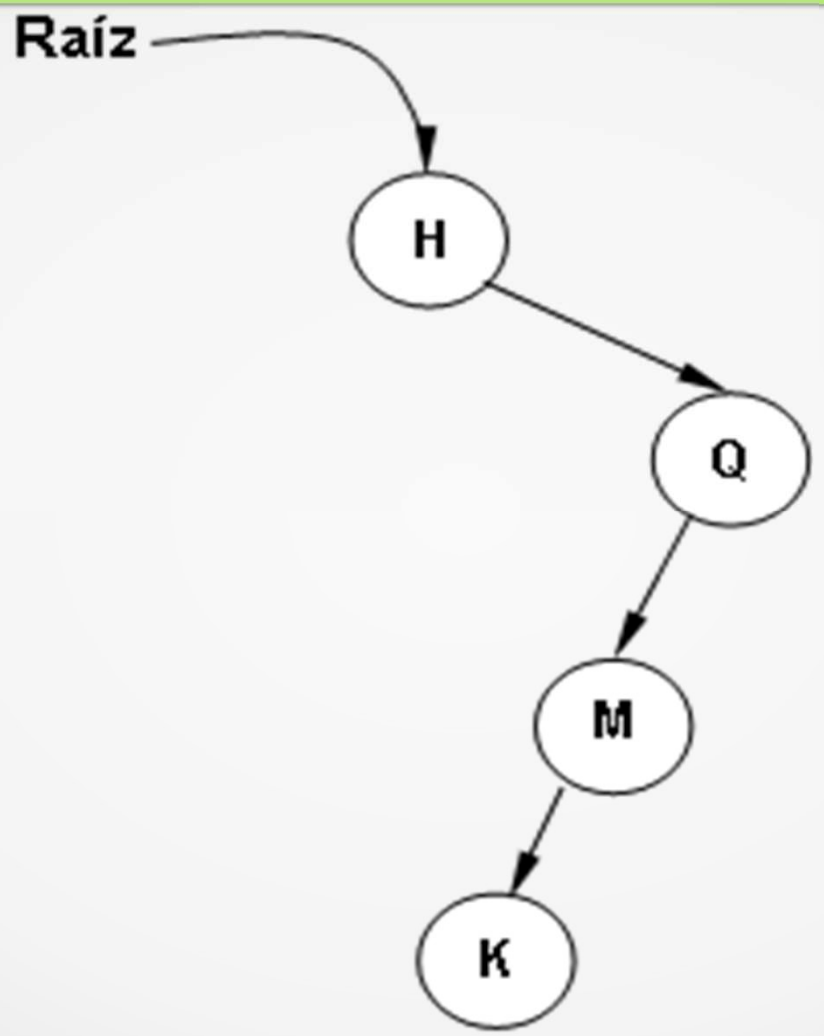
4.1.2 Operaciones Básicas sobre AB



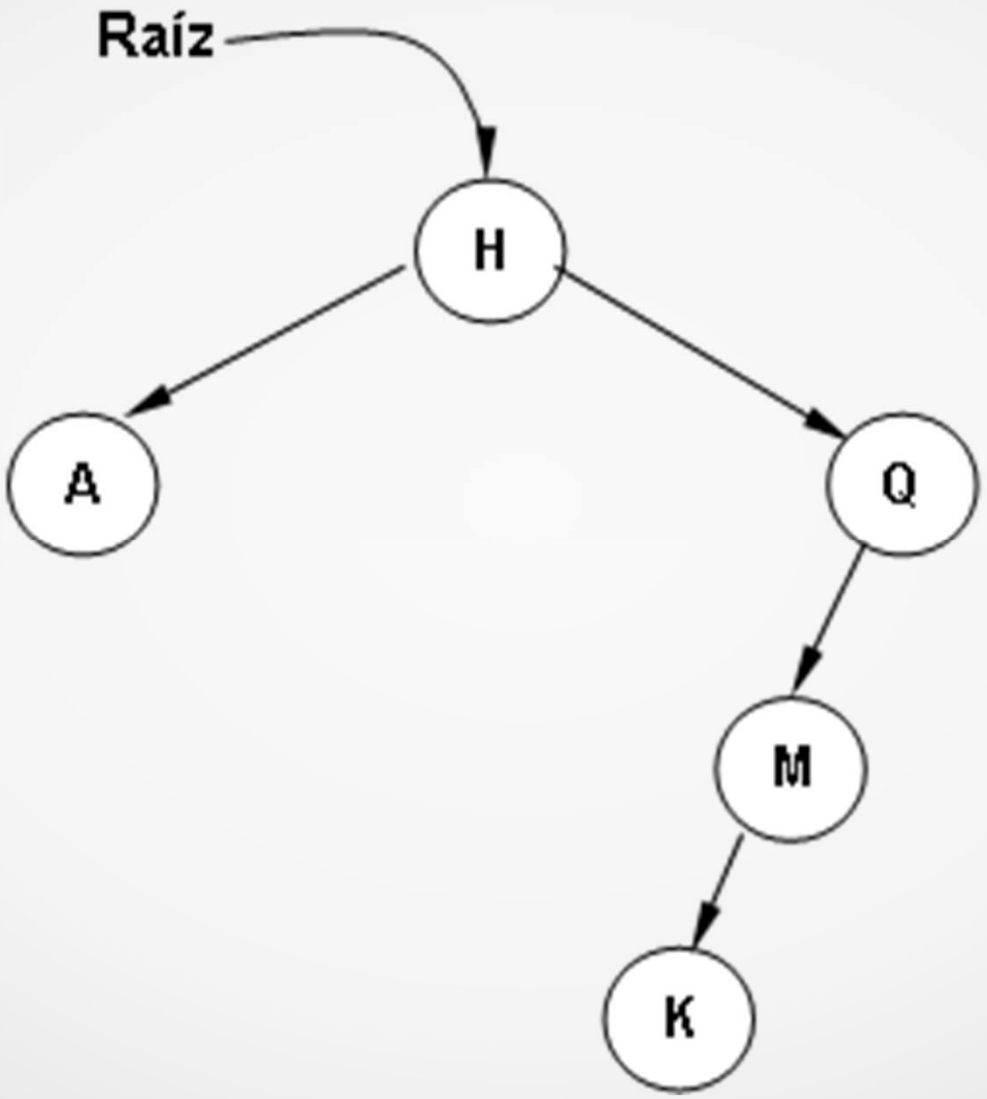
4.1.2 Operaciones Básicas sobre AB



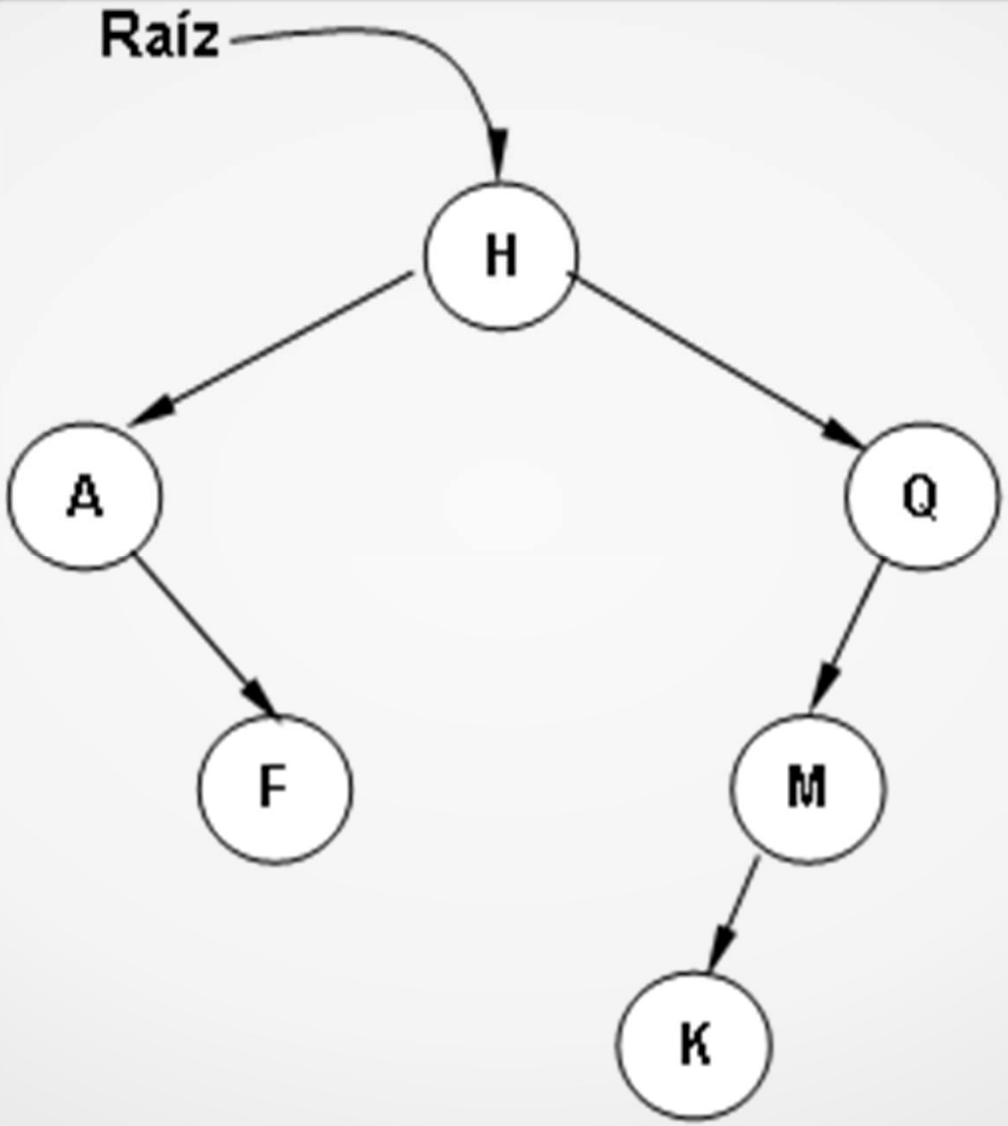
4.1.2 Operaciones Básicas sobre AB



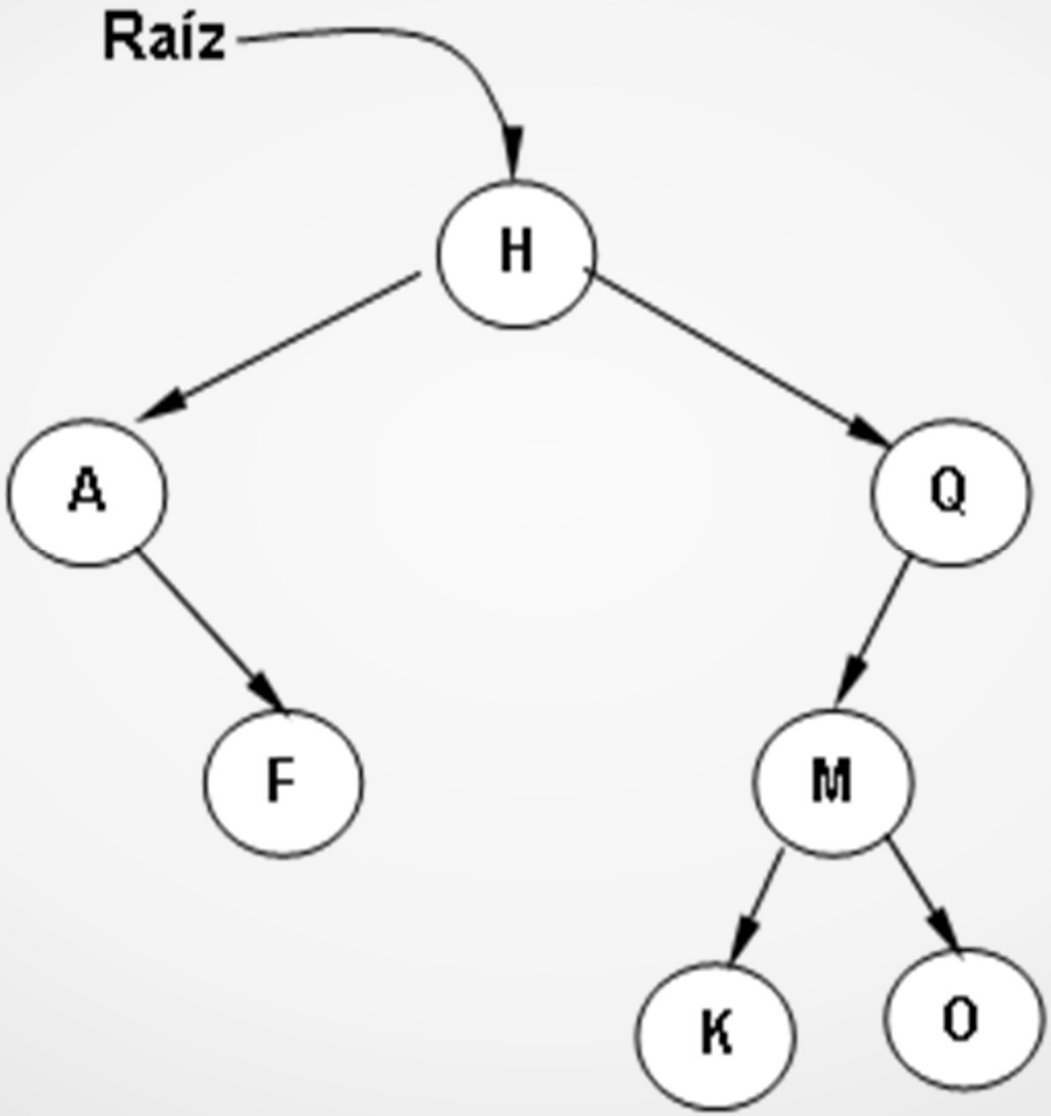
4.1.2 Operaciones Básicas sobre AB



4.1.2 Operaciones Básicas sobre AB

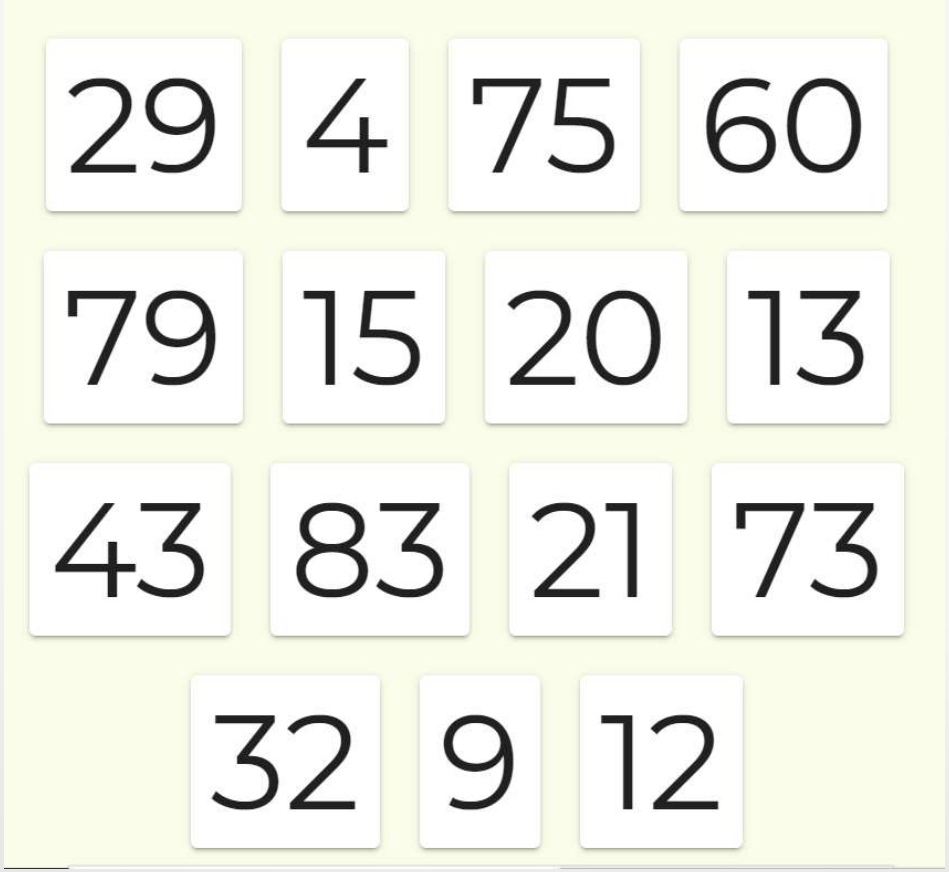


4.1.2 Operaciones Básicas sobre AB



4.1.2 Operaciones Básicas sobre AB

Construir un árbol a partir de los siguientes datos en el orden indicado:



4.1.2 Operaciones Básicas sobre AB

Algoritmos iterativos para añadir datos a un Árbol Binario de Búsqueda.

Procesos a efectuar durante altas de datos a un ABB:

1. Crear un nodo para el nuevo dato.
2. Buscar el lugar donde insertarlo.
3. Colocar el apuntador al nuevo nodo.

4.1.2 Operaciones Básicas sobre AB

1

NuevoNodo ← Dirección otorgada por el Sist Op

IZQ(NuevoNodo) ← NULO

DER(NuevoNodo) ← NULO

INFO(NuevoNodo) ← VALOR

4.1.2 Operaciones Básicas sobre AB

2

```
Temp ← RAIZ
PadreNN ← NULO
MIENTRAS Temp <> NULO
        PadreNN ← Temp
        SI INFO(NuevoNodo) > INFO(Temp)
                Temp ← DER(Temp)
        DE LO CONTRARIO
                Temp ← IZQ(Temp)
        FIN SI
FIN MIENTRAS
```

*PadreNN=Padre del Nuevo Nodo

4.1.2 Operaciones Básicas sobre AB

3

```
SI PadreNN = NULO
        RAIZ ← NuevoNodo
DE LO CONTRARIO
        SI INFO(NuevoNodo) > INFO(PadreNN)
                DER(PadreNN) ← NuevoNodo
        DE LO CONTRARIO
                IZQ(PadreNN)  ← NuevoNodo
        FIN SI
FIN SI
```

4.1.2 Operaciones Básicas sobre AB

EJERCICIO 1:

Modificar el programa que está en el sitio felipealanis.com para que:

- En la opción “consulta”, indique si el nodo encontrado es hijo izquierdo o derecho de su padre.
- Indique el nivel del nodo.
- Además, debe mostrarse que valores contiene cada uno de los hijos que en todo caso tiene.

4.1.2 Operaciones Básicas sobre AB

EJERCICIO 2:

Escriba un programa, usando manejo dinámico de memoria, que guarde en cada nodo de un árbol, el Número de Control y el nombre de un alumno, usando el siguiente algoritmo:

Mostrar un menú con las siguientes opciones (apéguese al algoritmo descrito):

4.1.2 Operaciones Básicas sobre AB

1.- Altas

- Pedir al usuario que escriba el Número de Control de un alumno.
- En caso que exista en el árbol ese Número de Control, el programa debe mostrar un mensaje indicando que no se puede dar de alta de nuevo (se debe mostrar el nombre de la persona a la que corresponde ese número de control).
- Si no existe, el programa pedirá al usuario que escriba el nombre del alumno que se va a dar de alta.
- Ya con ambos datos, se busca el lugar adecuado para el nuevo nodo en el árbol (de acuerdo a los algoritmos estudiados) y se enlaza.

4.1.2 Operaciones Básicas sobre AB

2.- Consultas

- Pedir al usuario que escriba el Número de Control de un alumno.
- Se buscará dentro del árbol ese Número de Control.
- En caso de que no exista en el árbol, se indicará con un mensaje.
- En caso de que se encuentre, el programa debe mostrar el nombre del alumno al que corresponde el Número de Control.

3.- Terminar

4.1.2 Operaciones Básicas sobre AB

EJERCICIO 3:

- Escriba un programa, para crear un árbol binario, usando manejo dinámico de memoria.
- El tipo de datos que guardará es ***int***.
- Debe mostrar un menú con las siguientes opciones:
  1. Añadir dato.
  2. Mostrar el árbol.
  3. Salir