Árboles Binarios Ordenados

Estructuras de Datos

Andrea Rueda

Pontificia Universidad Javeriana
Departamento de Ingeniería de Sistemas

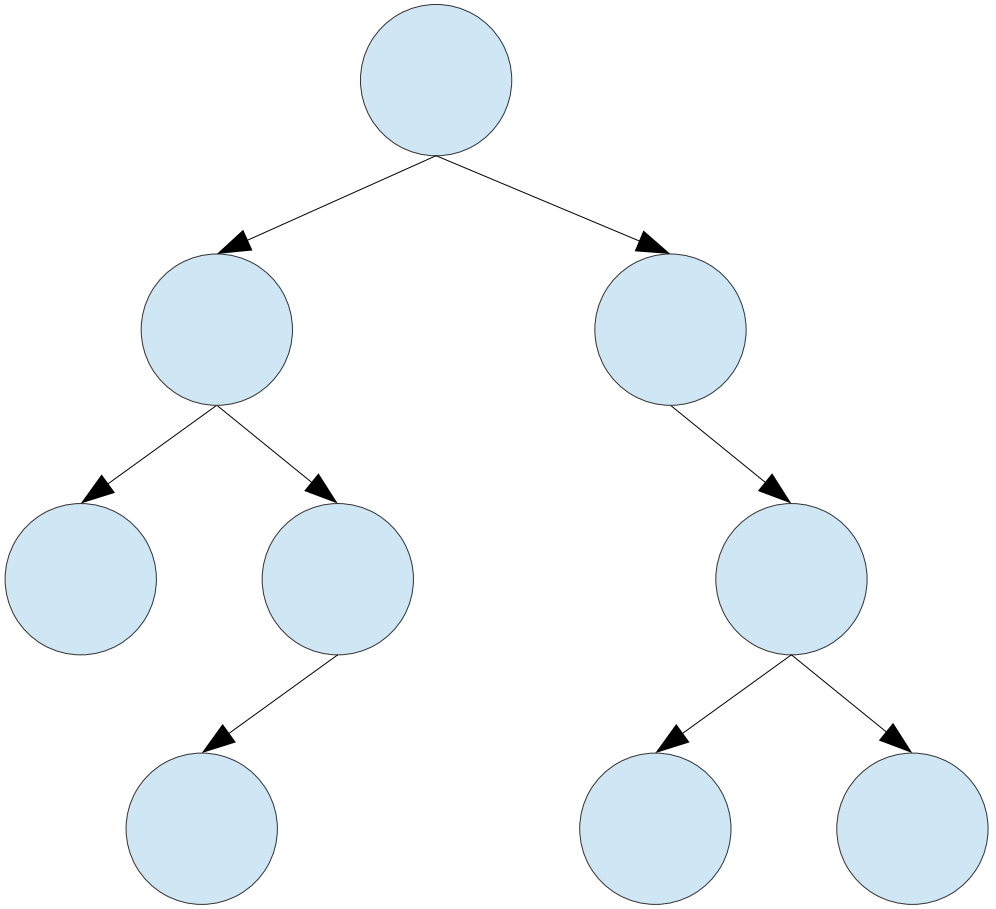
Árboles Binarios

Árbol Binario

Árbol binario (2-árbol): árbol en el cual cada uno de sus nodos puede tener no más de 2 hijos.

- Árbol con grado 2.
- Cada nodo puede tener 0, 1 o 2 hijos (subárboles).
- Descendiente de la izquierda es el hijo (subárbol) izquierdo.
- Descendiente de la derecha es el hijo (subárbol) derecho.

Árbol Binario

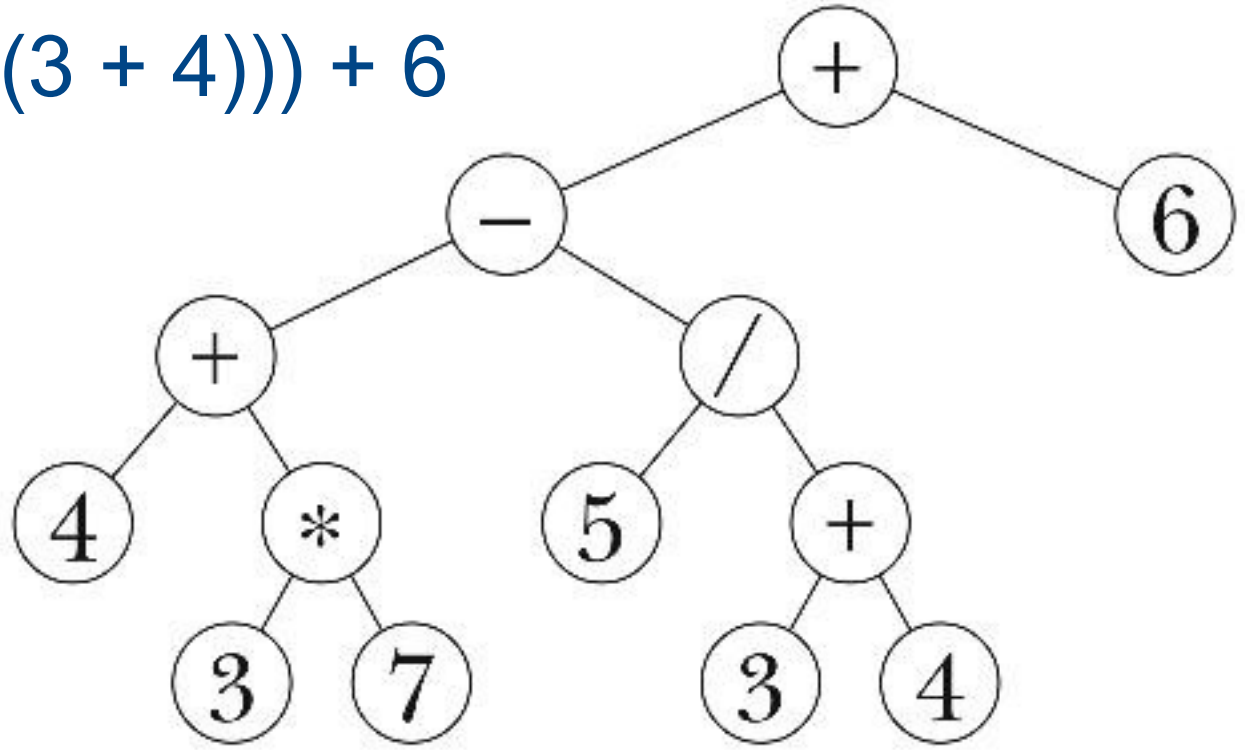


Árbol Binario

Aplicaciones:

- Evaluación de expresiones aritméticas (operandos/operadores).

((4 + (3 * 7)) – (5 / (3 + 4))) + 6

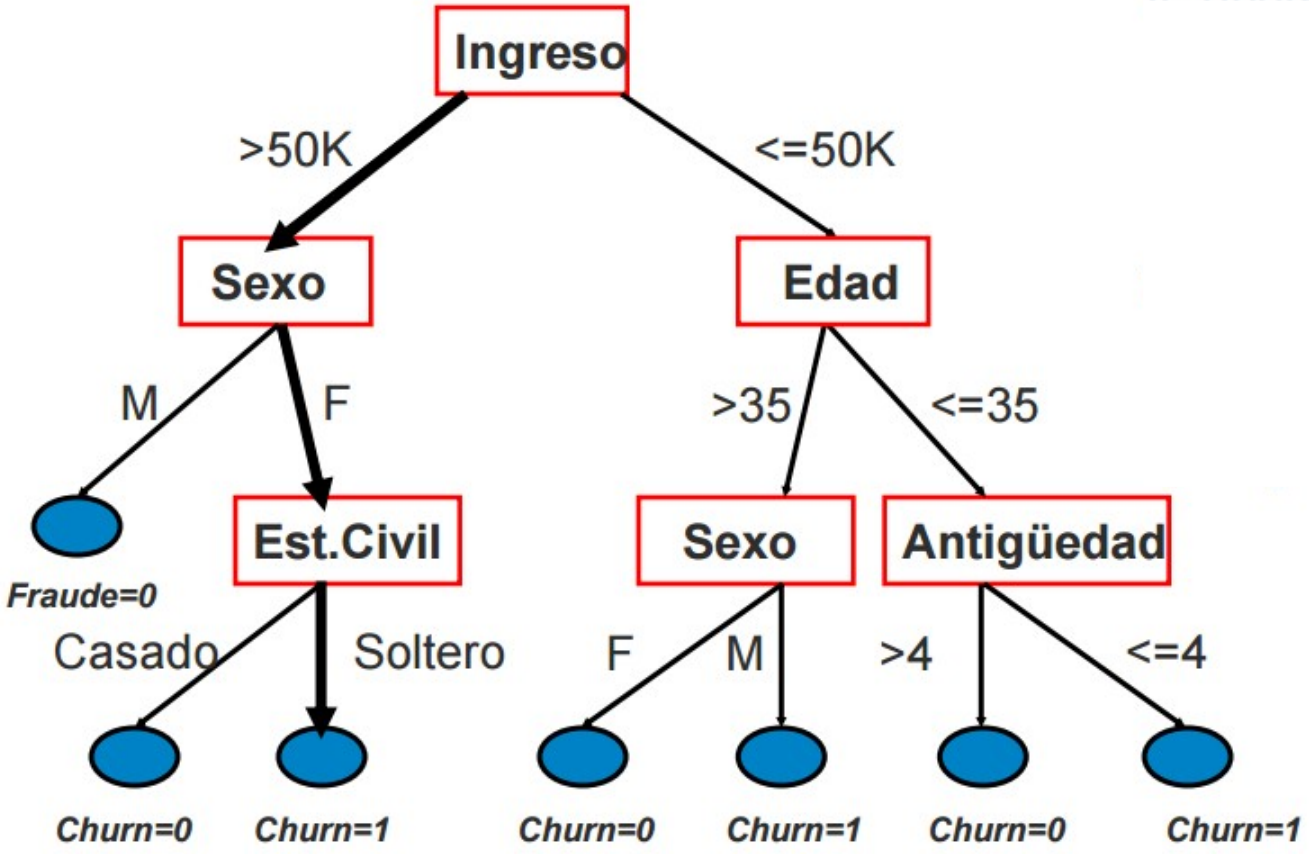


www.eecs.berkeley.edu/~bh/ssch18/trees.html

Árbol Binario

Aplicaciones:

- Procesos de decisión (preguntas con respuesta binaria).

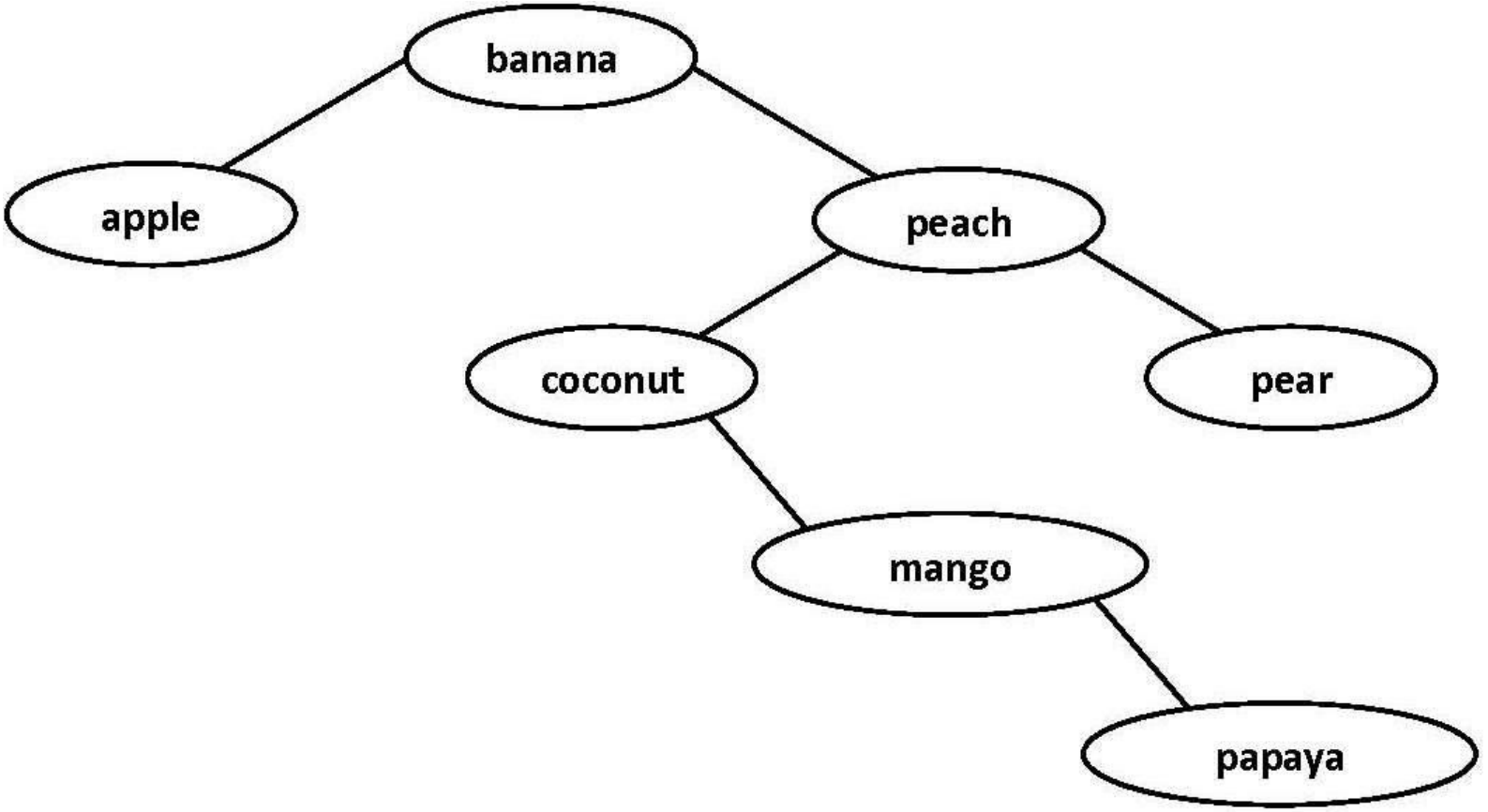


http://sasybi.blogspot.com.co/2015/05/arboles-de-decision-en-sas.html

Árbol Binario

Aplicaciones:

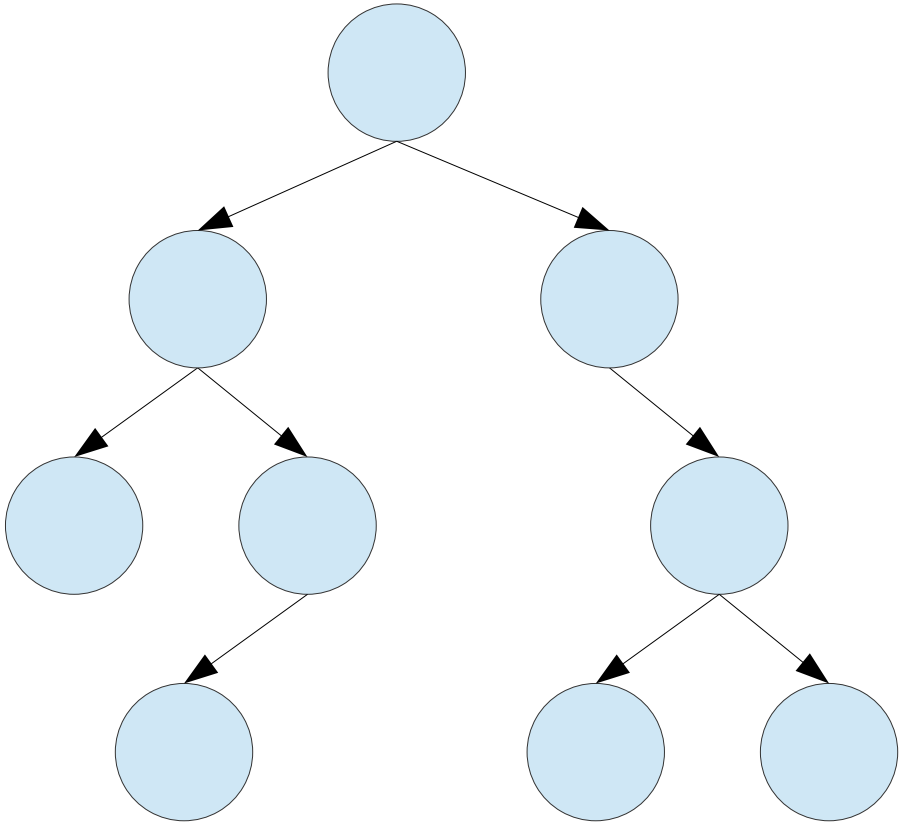
- Búsqueda (nodos ordenados).



cslearners.blogspot.com/2011/06/binary-search-tree.html

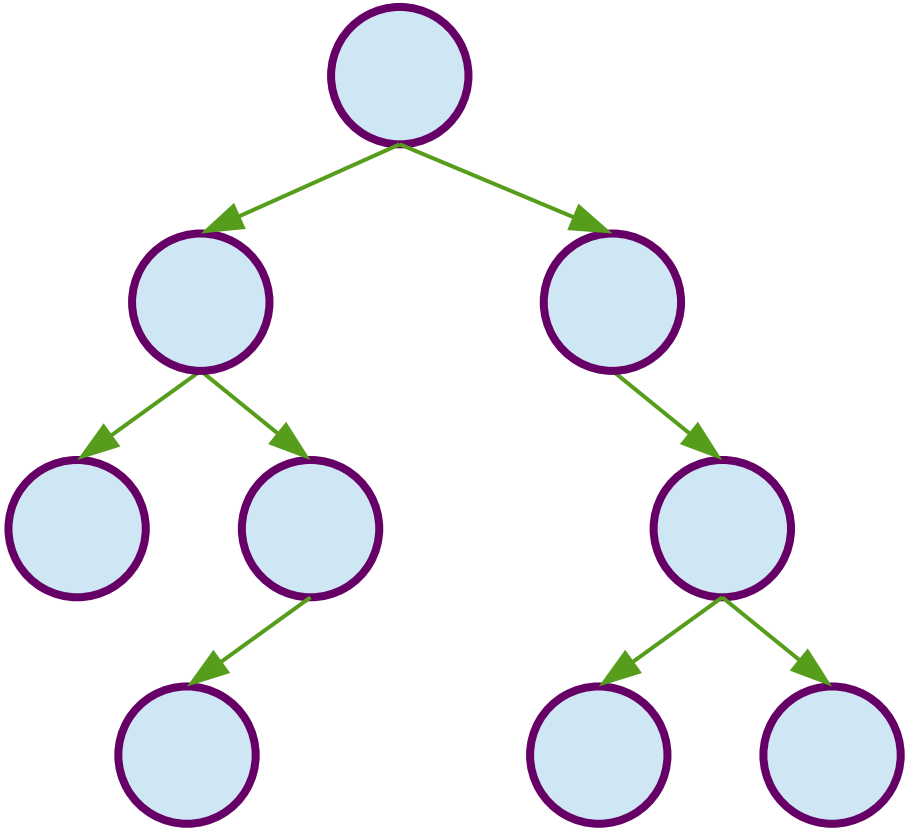
Árbol Binario

- Propiedades
  - Un árbol binario con n nodos, contiene exactamente $n-1$ aristas o conexiones



Árbol Binario

- Propiedades
  - Un árbol binario con n nodos, contiene exactamente $n-1$ aristas o conexiones

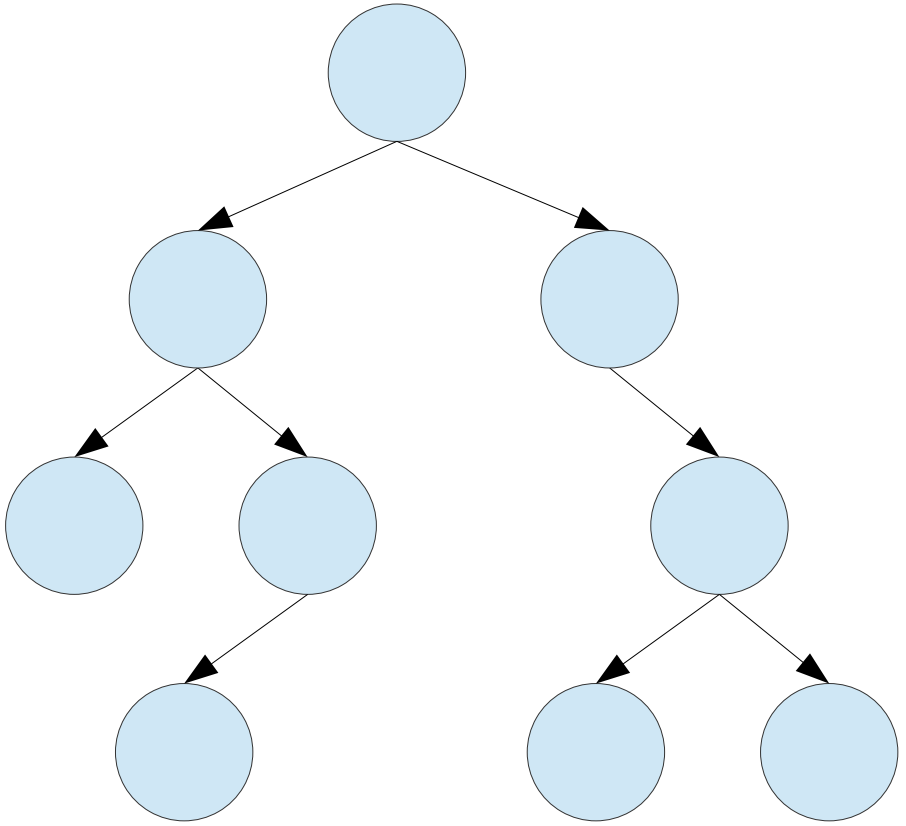


9 nodos

8 aristas

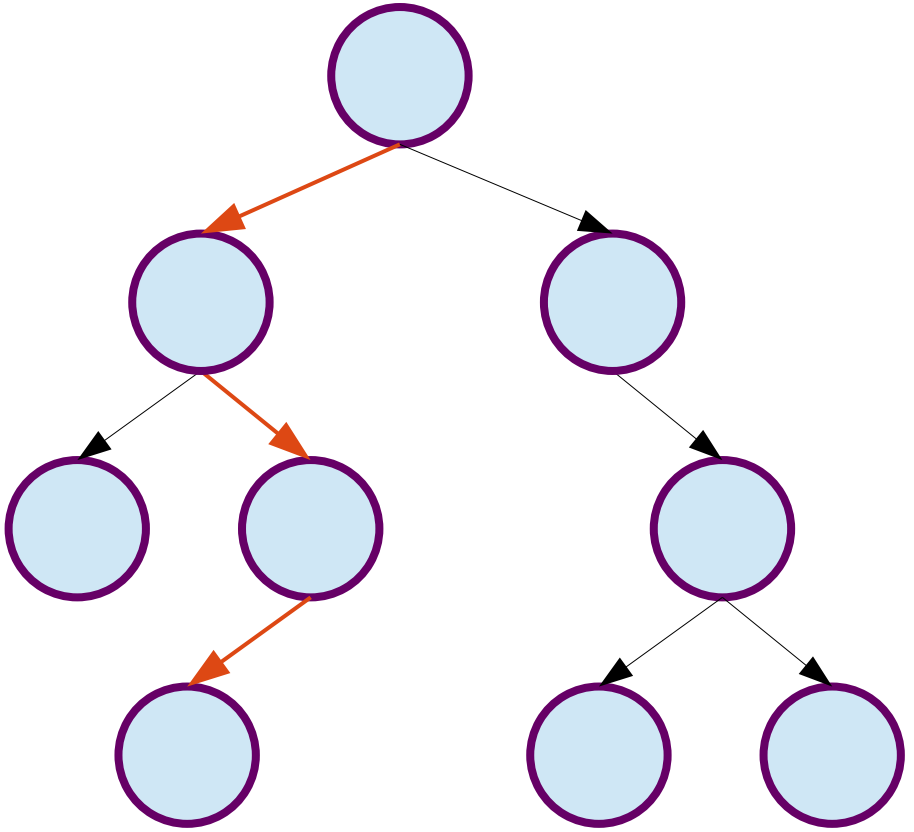
Árbol Binario

- Propiedades
  - Un árbol binario de altura h, tiene una cantidad de nodos mayor o igual a $h+1$ y menor o igual a $2^{h+1}-1$



Árbol Binario

- Propiedades
  - Un árbol binario de altura h, tiene una cantidad de nodos mayor o igual a $h+1$ y menor o igual a $2^{h+1}-1$

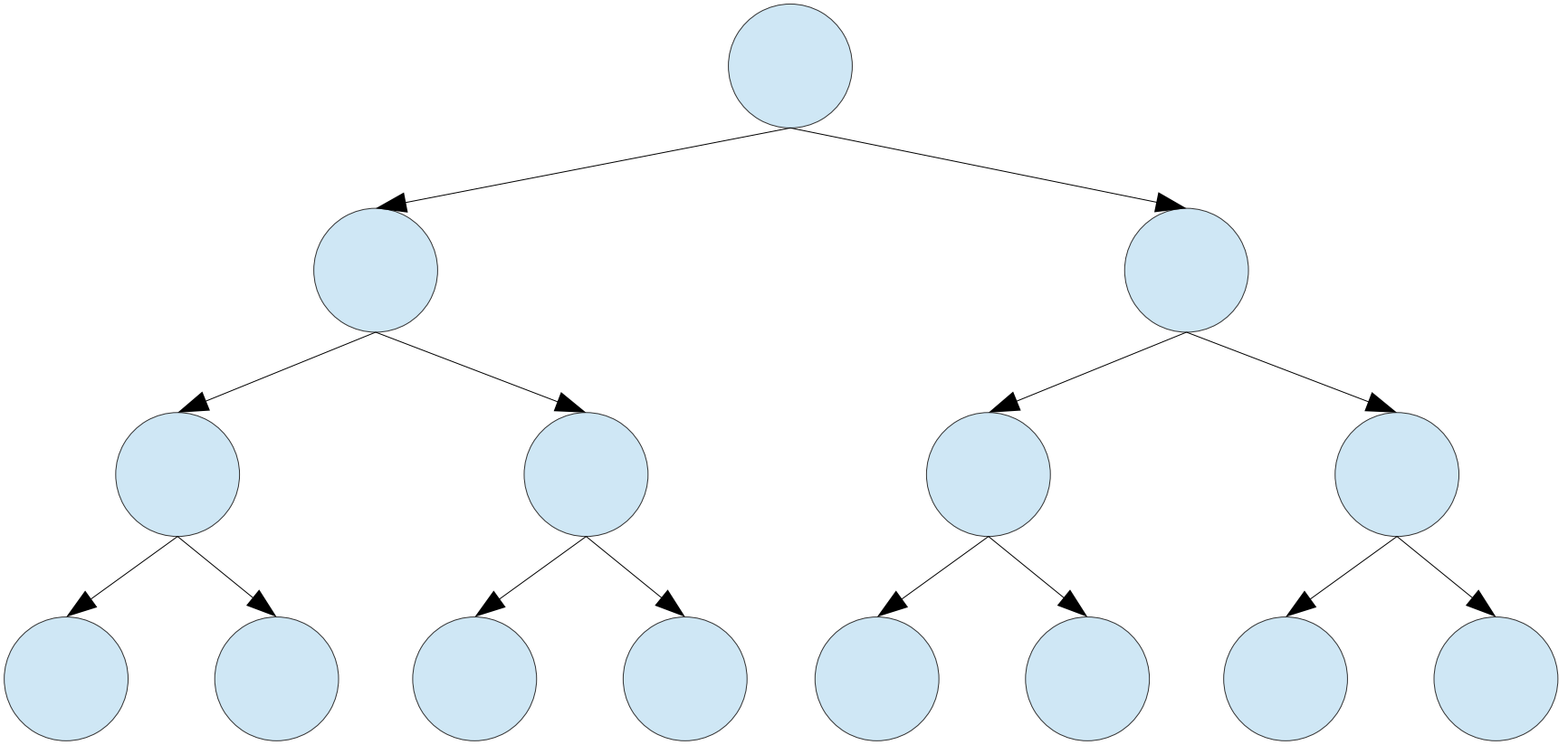


$h = 3$

$4 \leq 9 \leq 15$

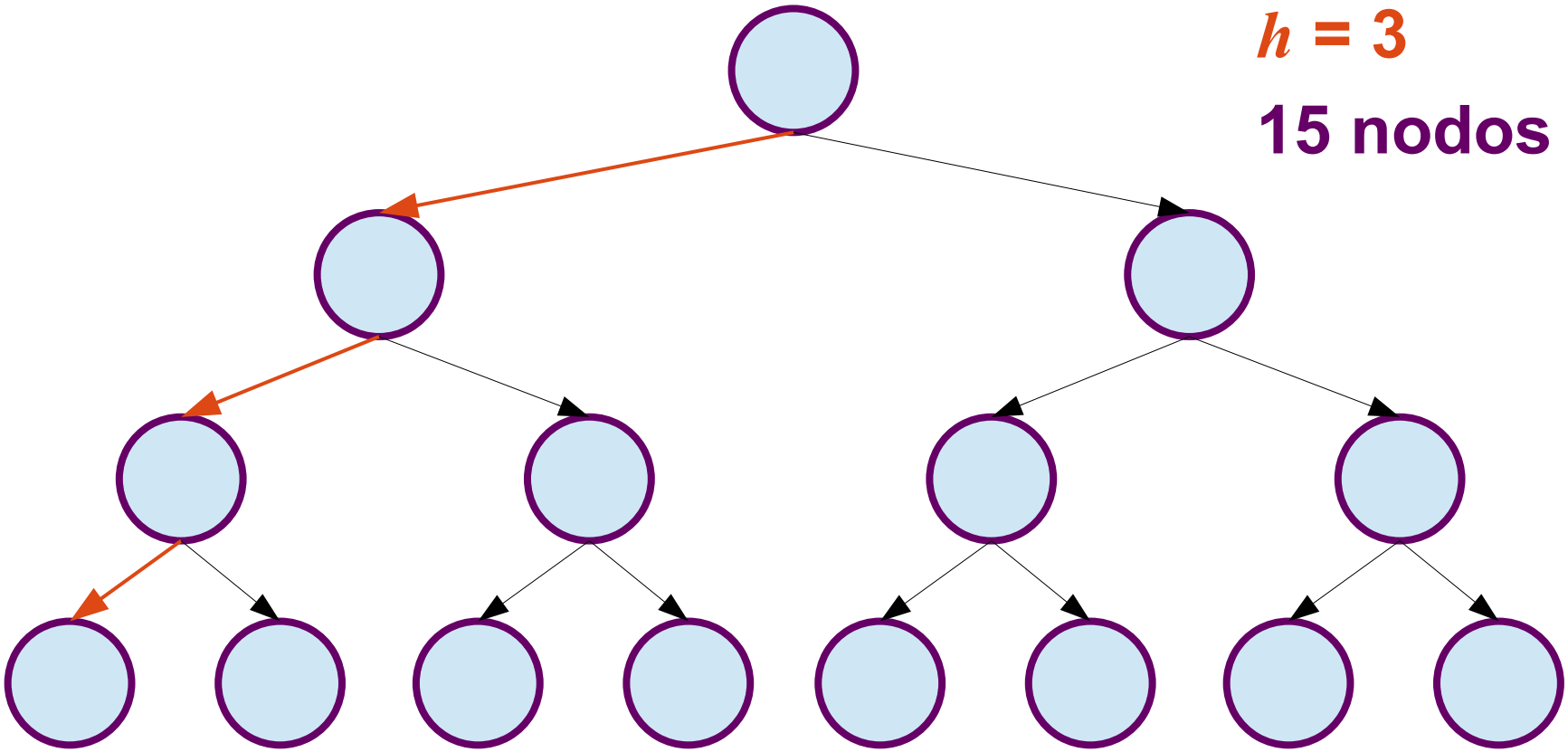
Árbol Binario

- Propiedades
  - Un árbol binario completo, de altura h, tiene exactamente $2^{h+1}-1$ nodos

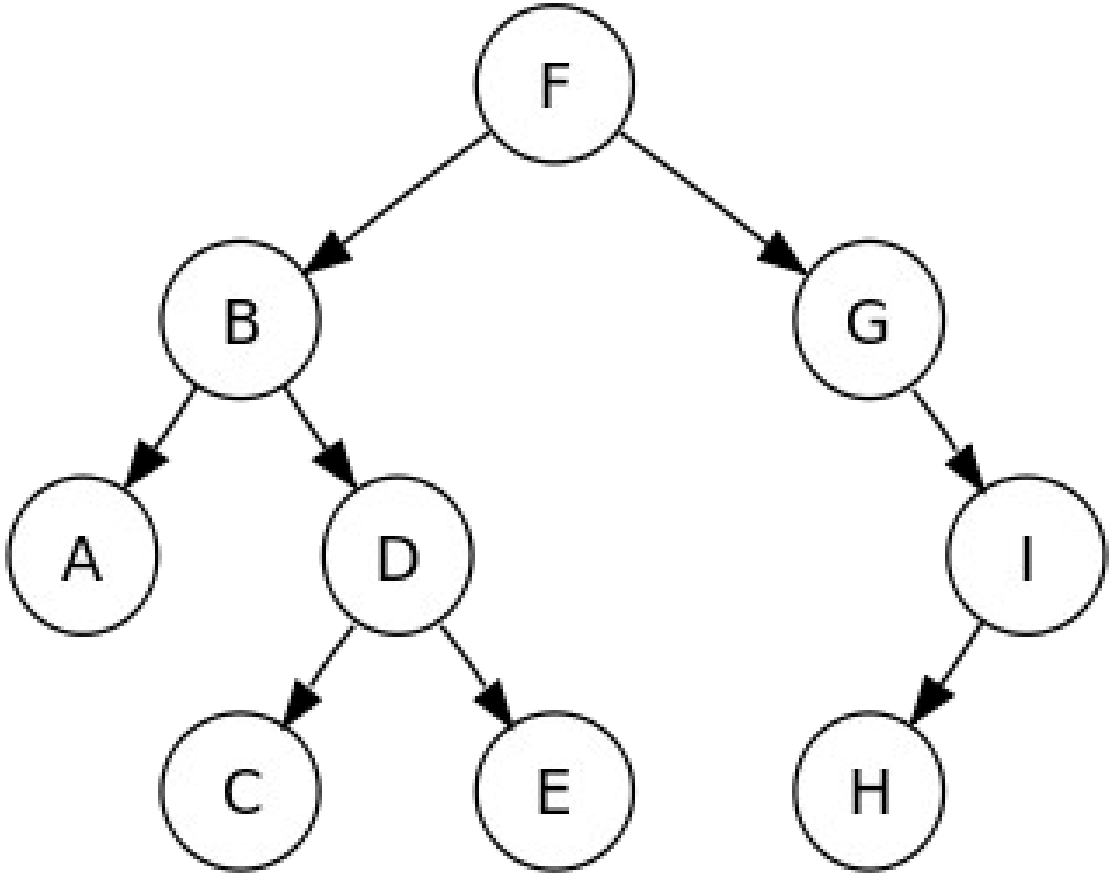


Árbol Binario

- Propiedades
  - Un árbol binario completo, de altura h, tiene exactamente $2^{h+1}-1$ nodos



Árbol Binario: recorridos



en.wikipedia.org/wiki/Tree_traversal

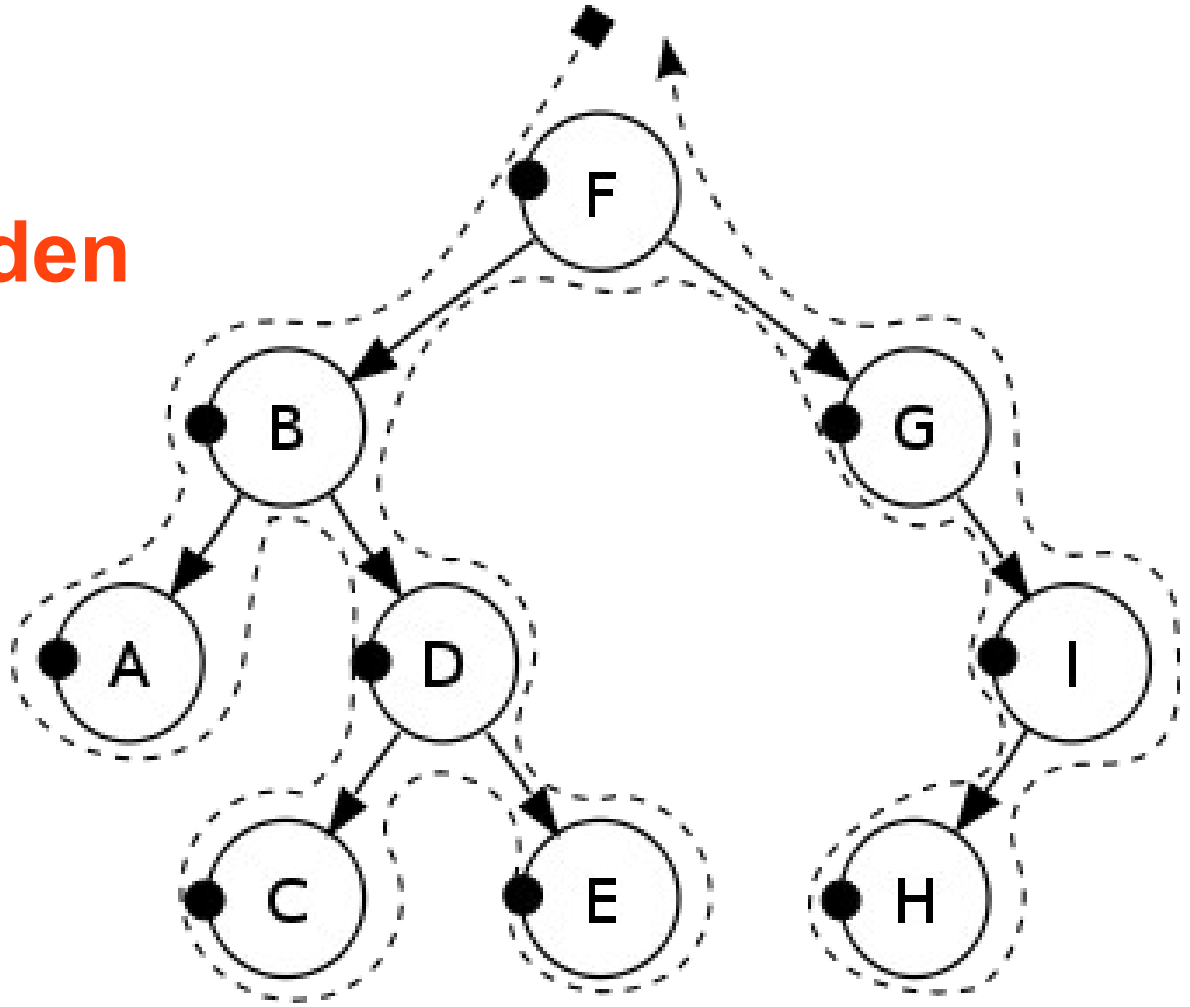
Árbol Binario: recorridos

Pre-orden

1. **visitar la raíz**
2. **recorrer el hijo (subárbol) izquierdo**
   -> hacer *preorden* sobre el hijo izquierdo
3. **recorrer el hijo (subárbol) derecho**
   -> hacer *preorden* sobre el hijo derecho

Árbol Binario: recorridos

Pre-orden



F – B – A – D – C – E – G – I – H

en.wikipedia.org/wiki/Tree_traversal

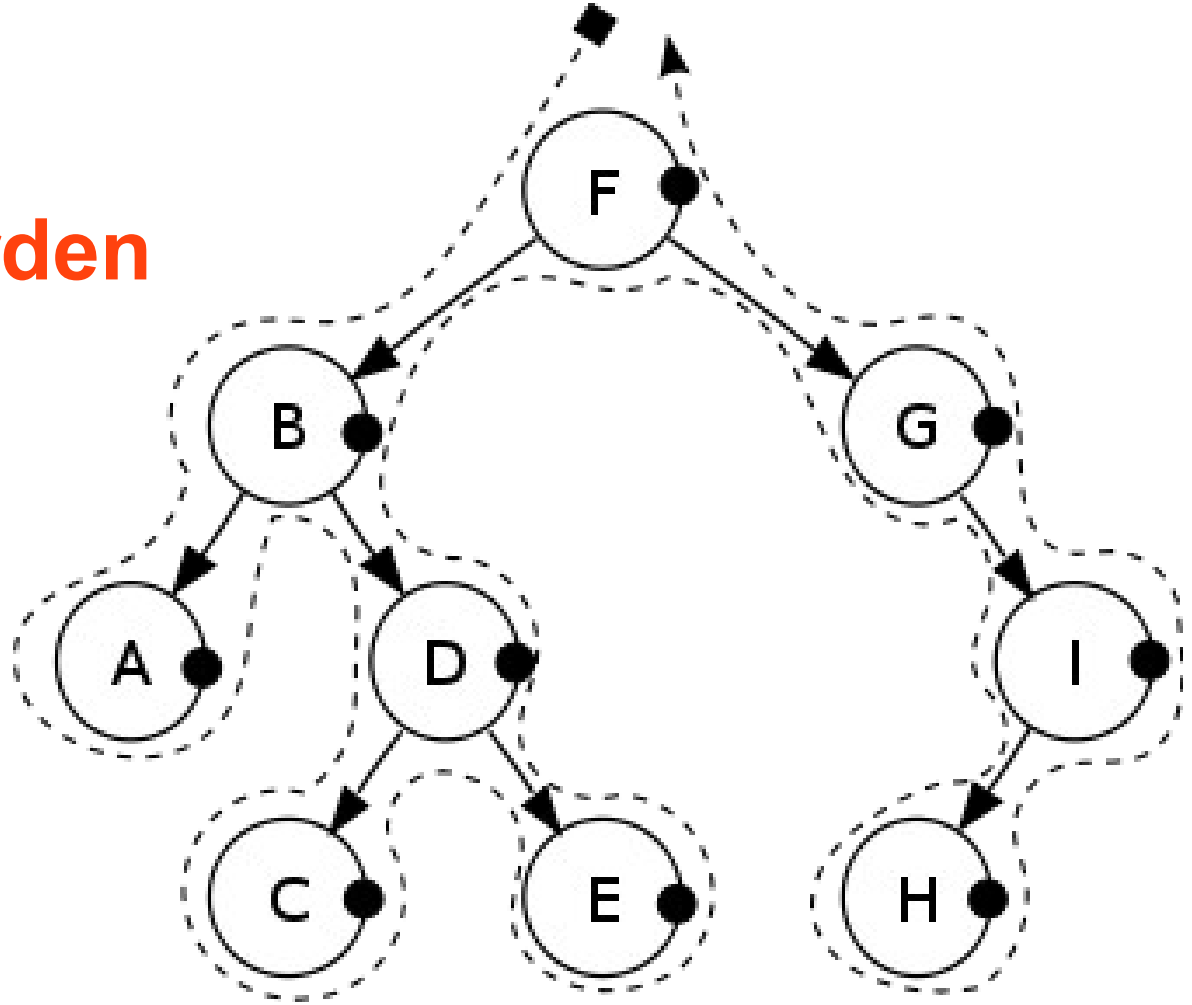
Árbol Binario: recorridos

Pos-orden

1. **recorrer el hijo (subárbol) izquierdo**
   -> hacer *posorden* sobre el hijo izquierdo
2. **recorrer el hijo (subárbol) derecho**
   -> hacer *posorden* sobre el hijo derecho
3. **visitar la raíz**

Árbol Binario: recorridos

Pos-orden



A – C – E – D – B – H – I – G – F

en.wikipedia.org/wiki/Tree_traversal

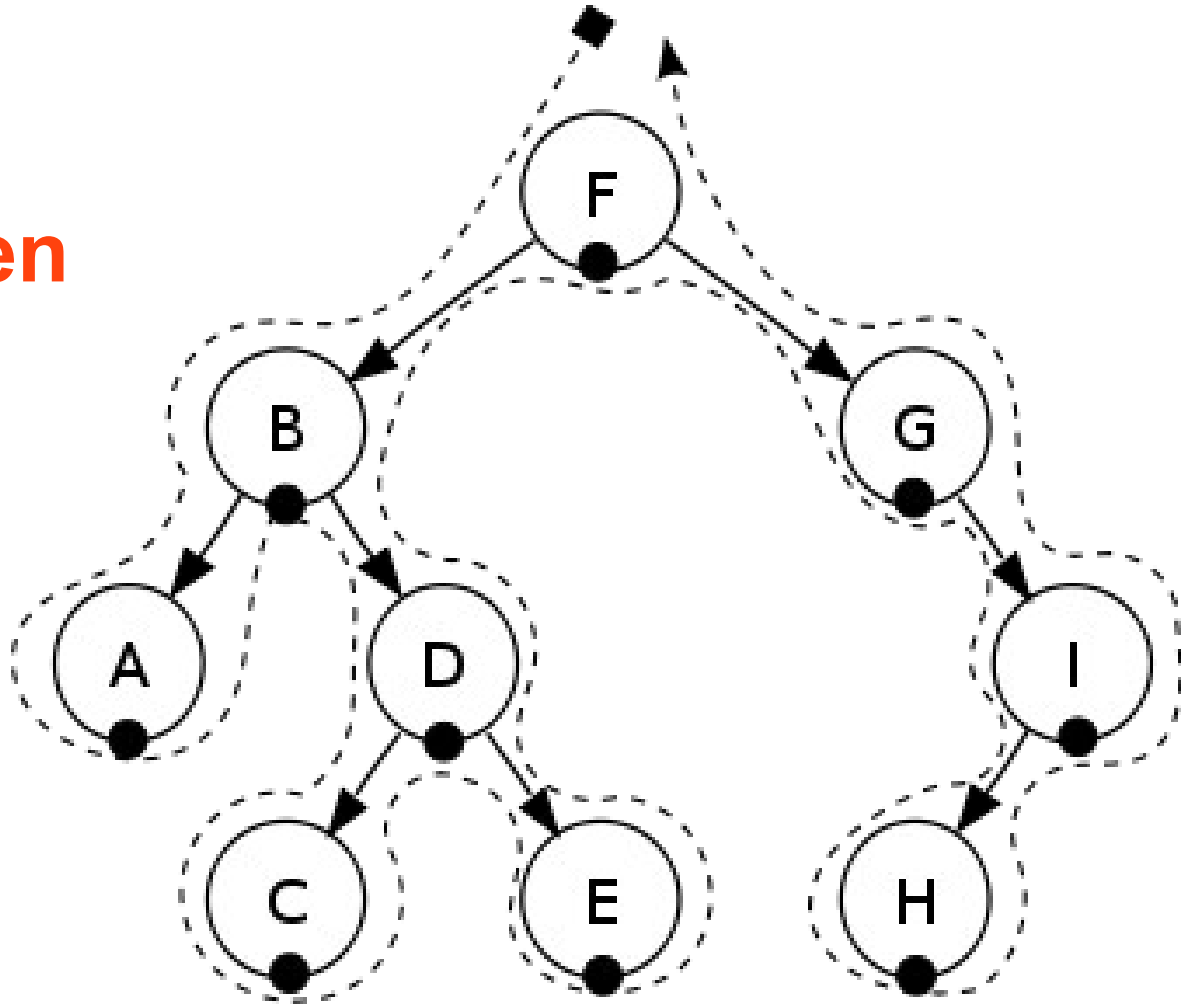
Árbol Binario: recorridos

In-orden

1. **recorrer el hijo (subárbol) izquierdo**
   -> hacer *inorden* sobre el hijo izquierdo
2. **visitar la raíz**
3. **recorrer el hijo (subárbol) derecho**
   -> hacer *inorden* sobre el hijo derecho

Árbol Binario: recorridos

In-orden



A – B – C – D – E – F – G – H – I

en.wikipedia.org/wiki/Tree_traversal

Árbol Binario: recorridos

Por niveles

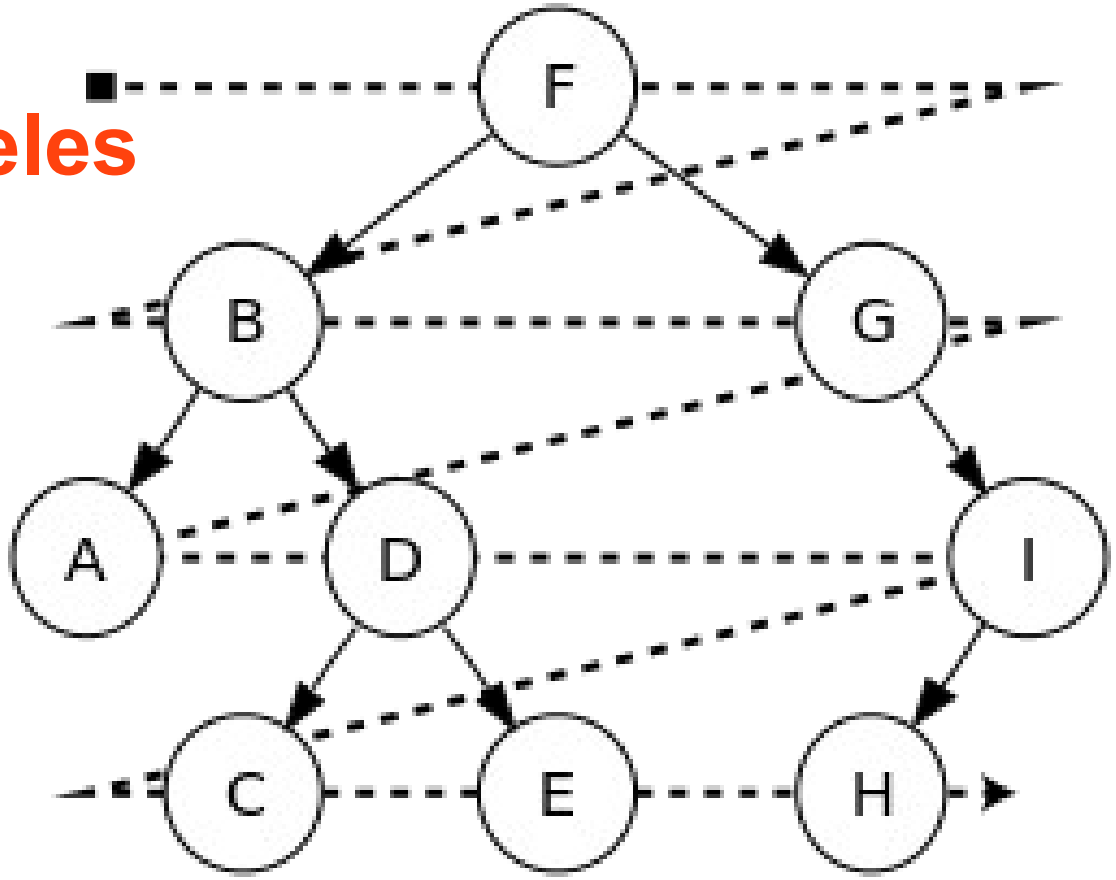
1. **visitar la raíz**
2. **visitar todos los hijos**
3. **visitar todos los nietos (hijos de los hijos)**

...

Sugerencia: cola de nodos, para cada nodo se encolan sus hijos

Árbol Binario: recorridos

Por niveles

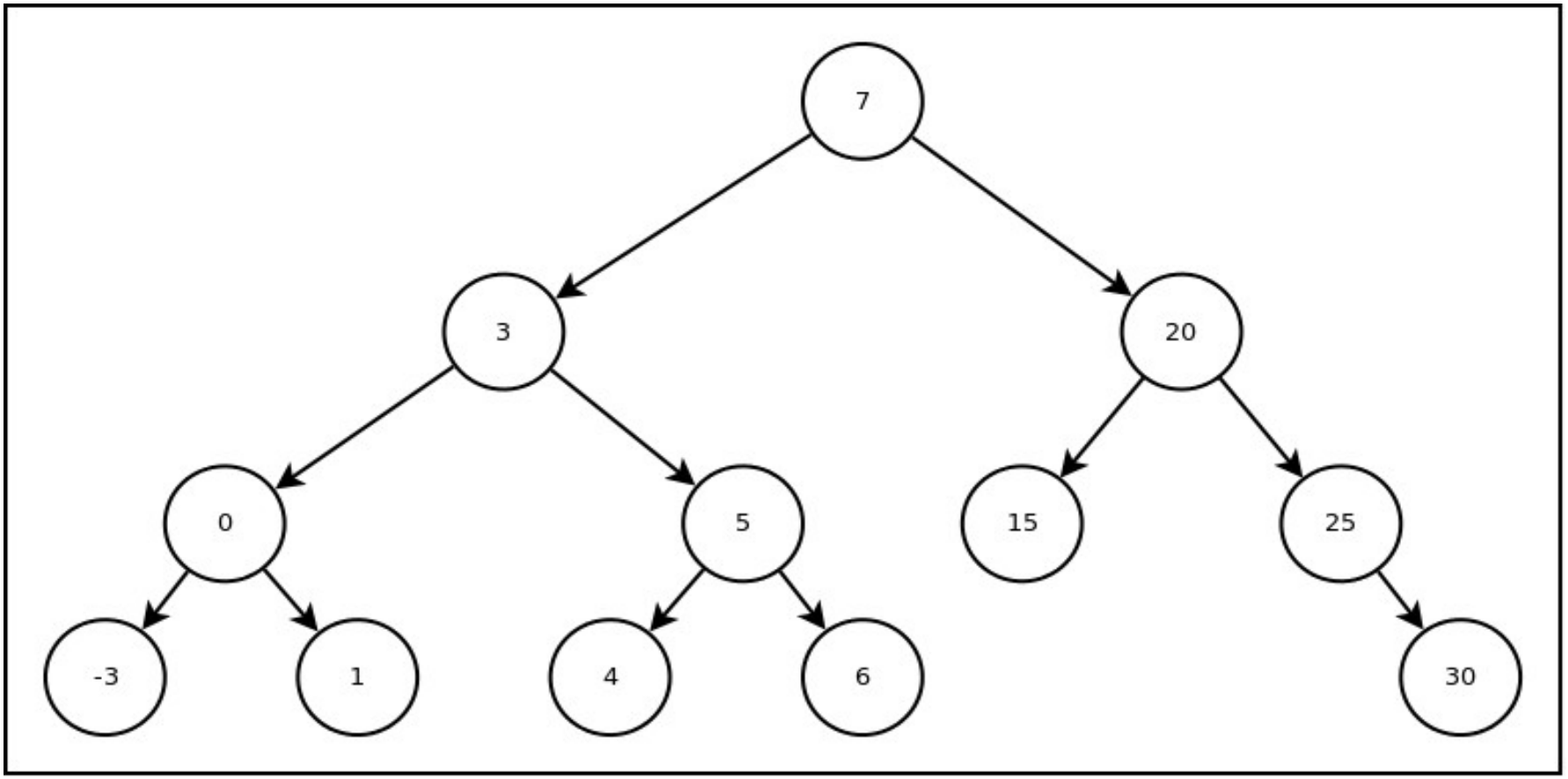


F – B – G – A – D – I – C – E – H

en.wikipedia.org/wiki/Tree_traversal

Árboles Binarios Ordenados

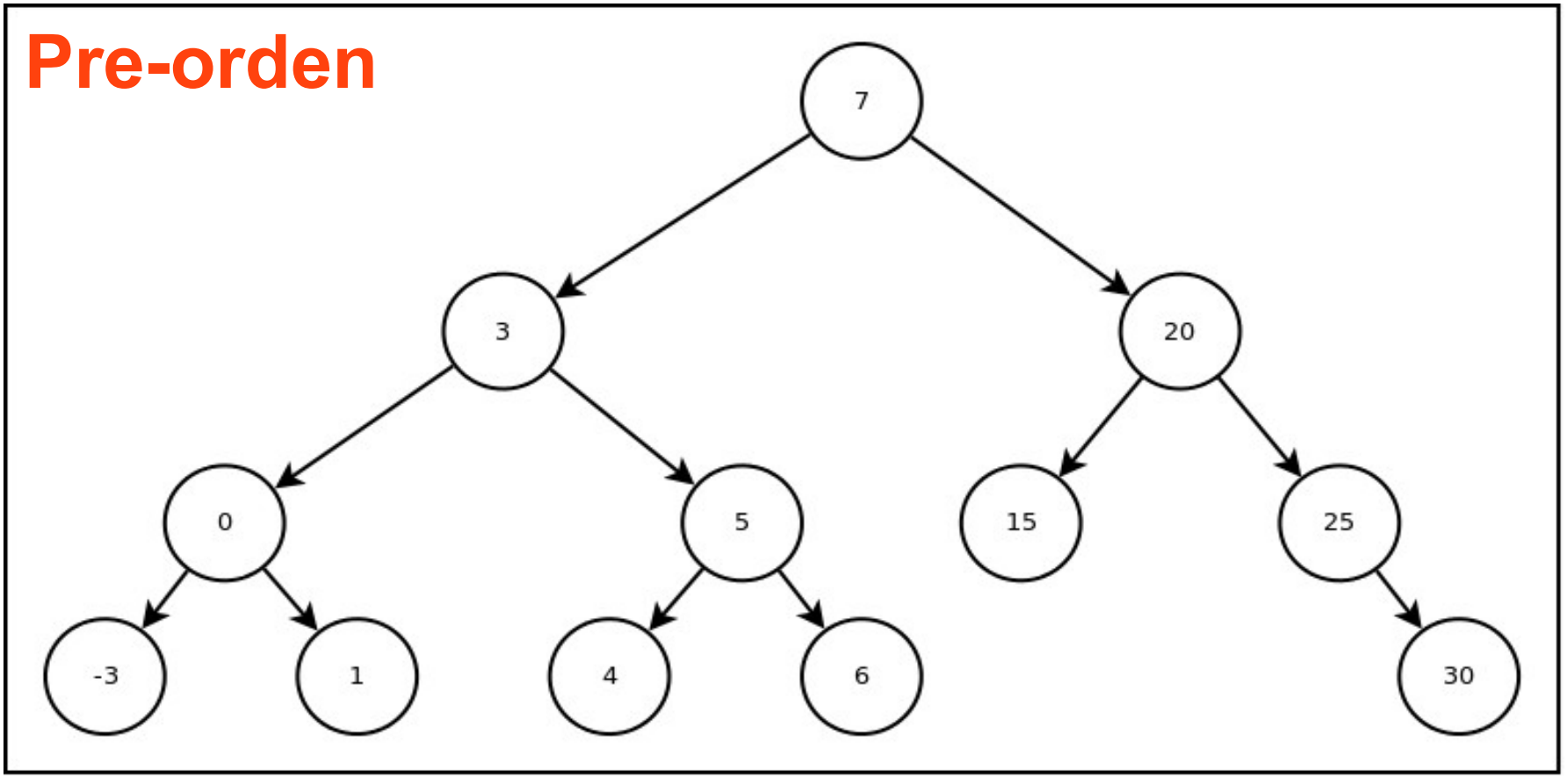
Árbol Binario Ordenado



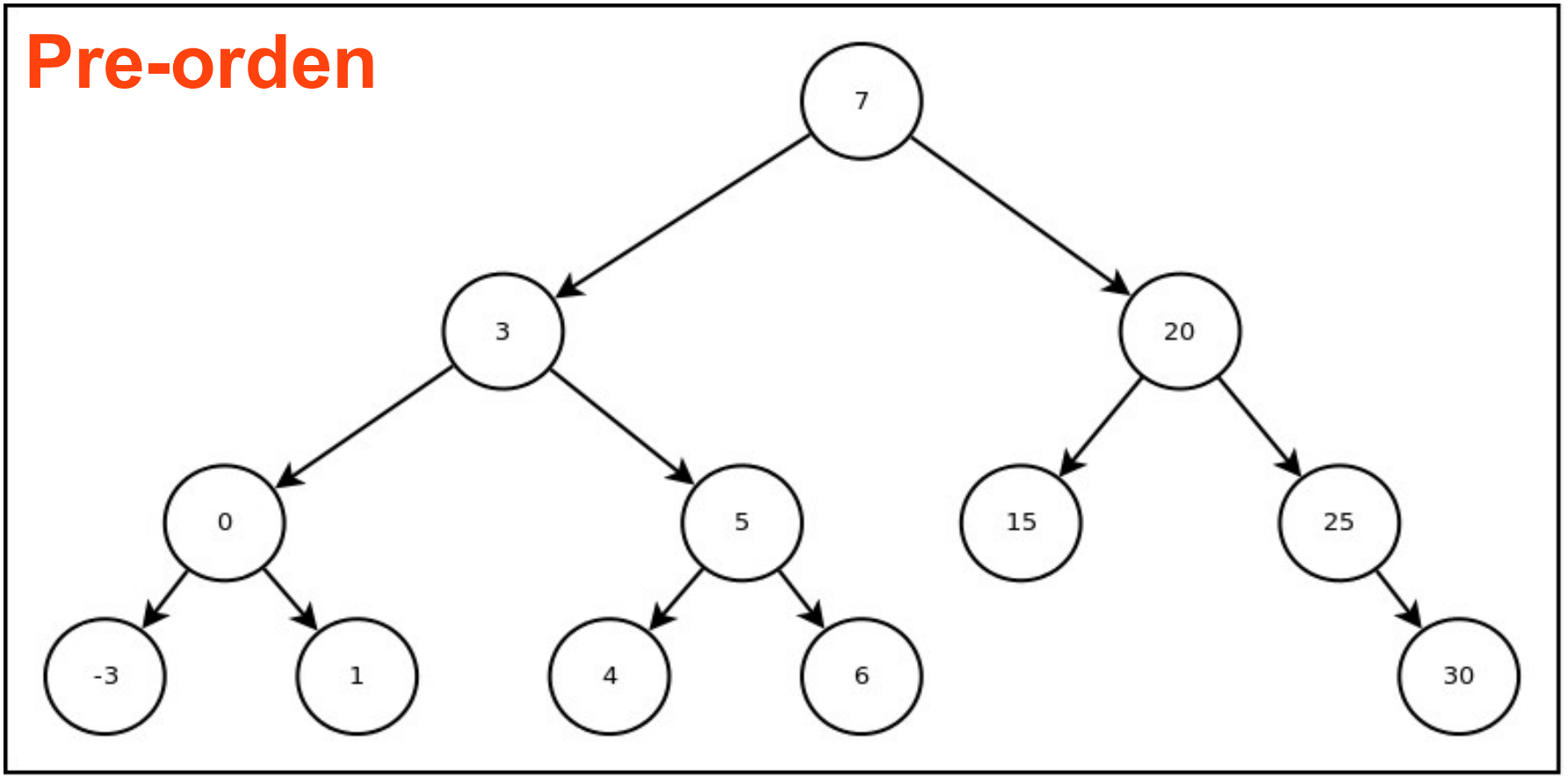
Árbol Binario Ordenado

- Estructura de datos recurrente de orden 2.
  - Motivada por la búsqueda binaria.
- Todos los nodos a la izquierda de la raíz son **menores** que ella.
- Todos los nodos a la derecha de la raíz son **mayores** que ella.
- ¿Se permiten repetidos?

Árbol Binario Ordenado

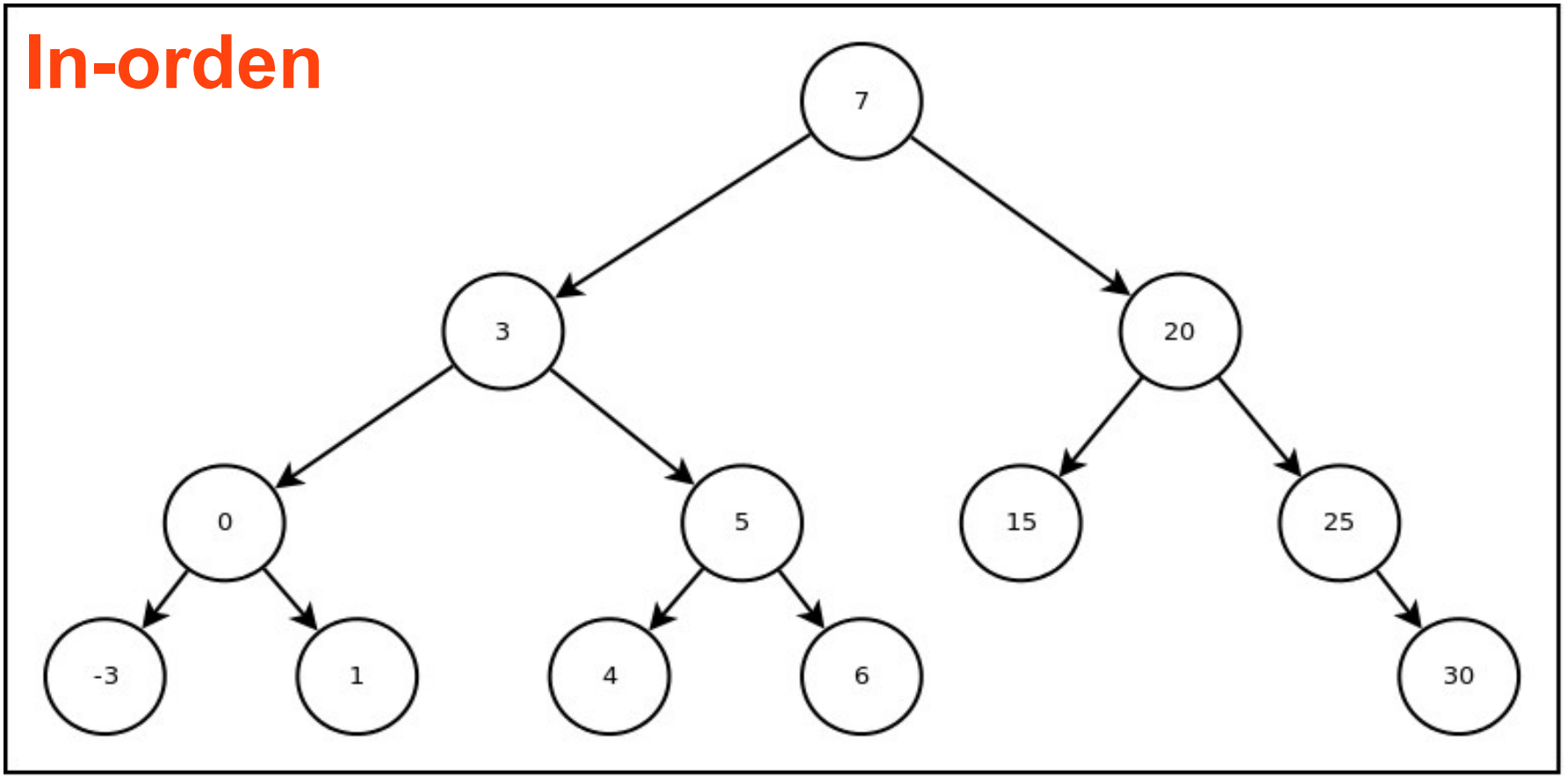


Árbol Binario Ordenado

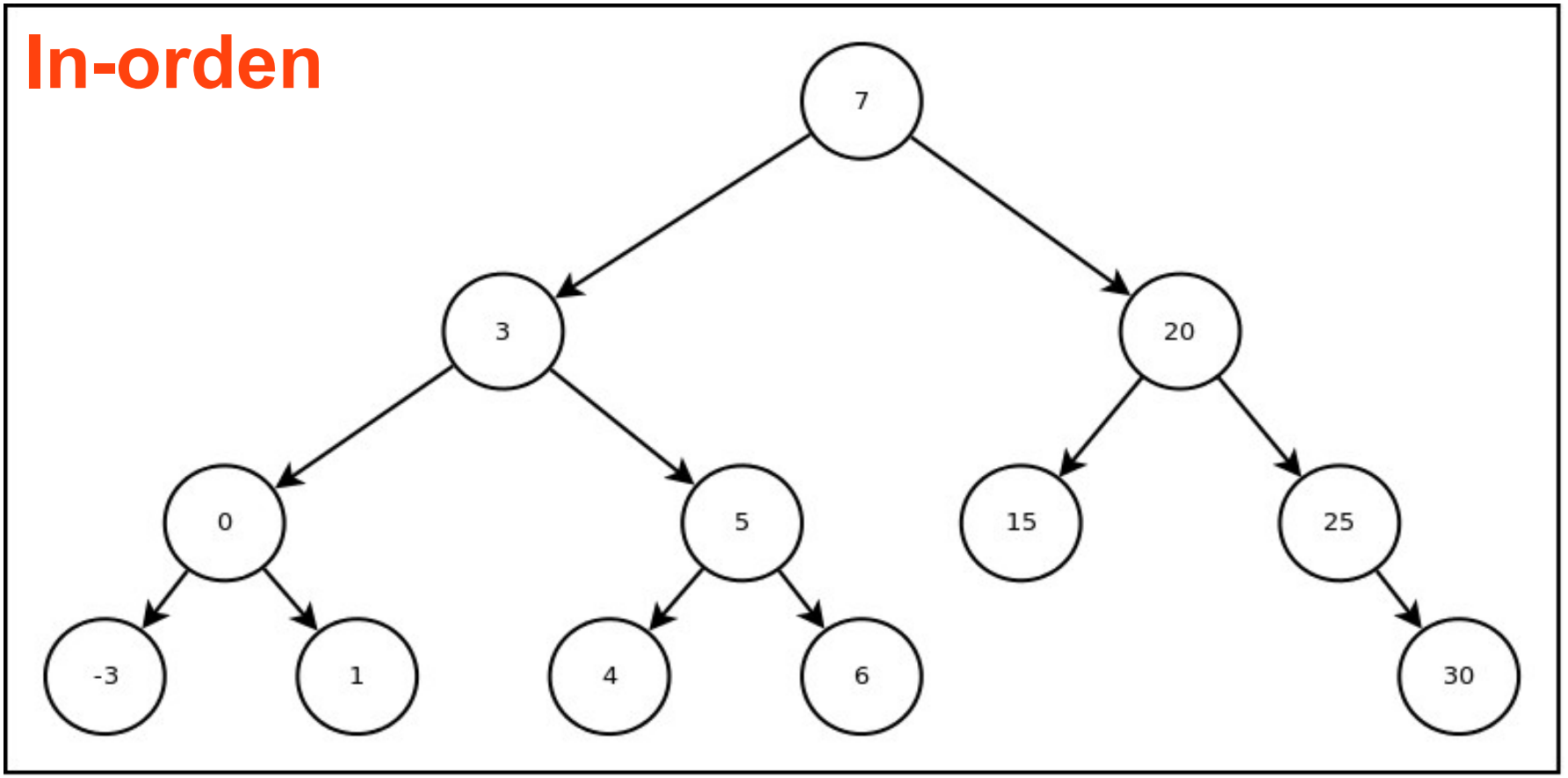


7, 3, 0, -3, 1, 5, 4, 6, 20, 15, 25, 30

Árbol Binario Ordenado

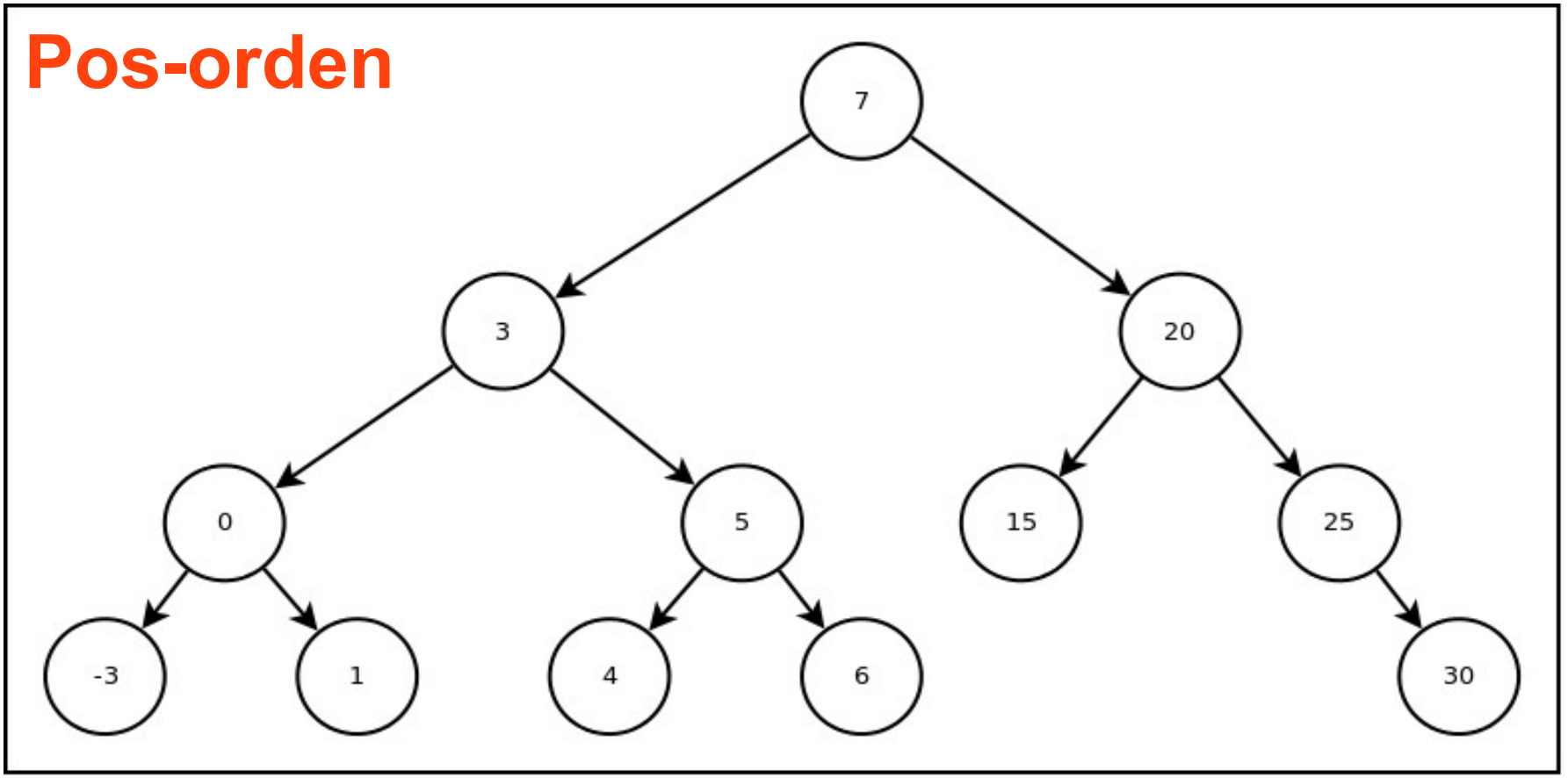


Árbol Binario Ordenado

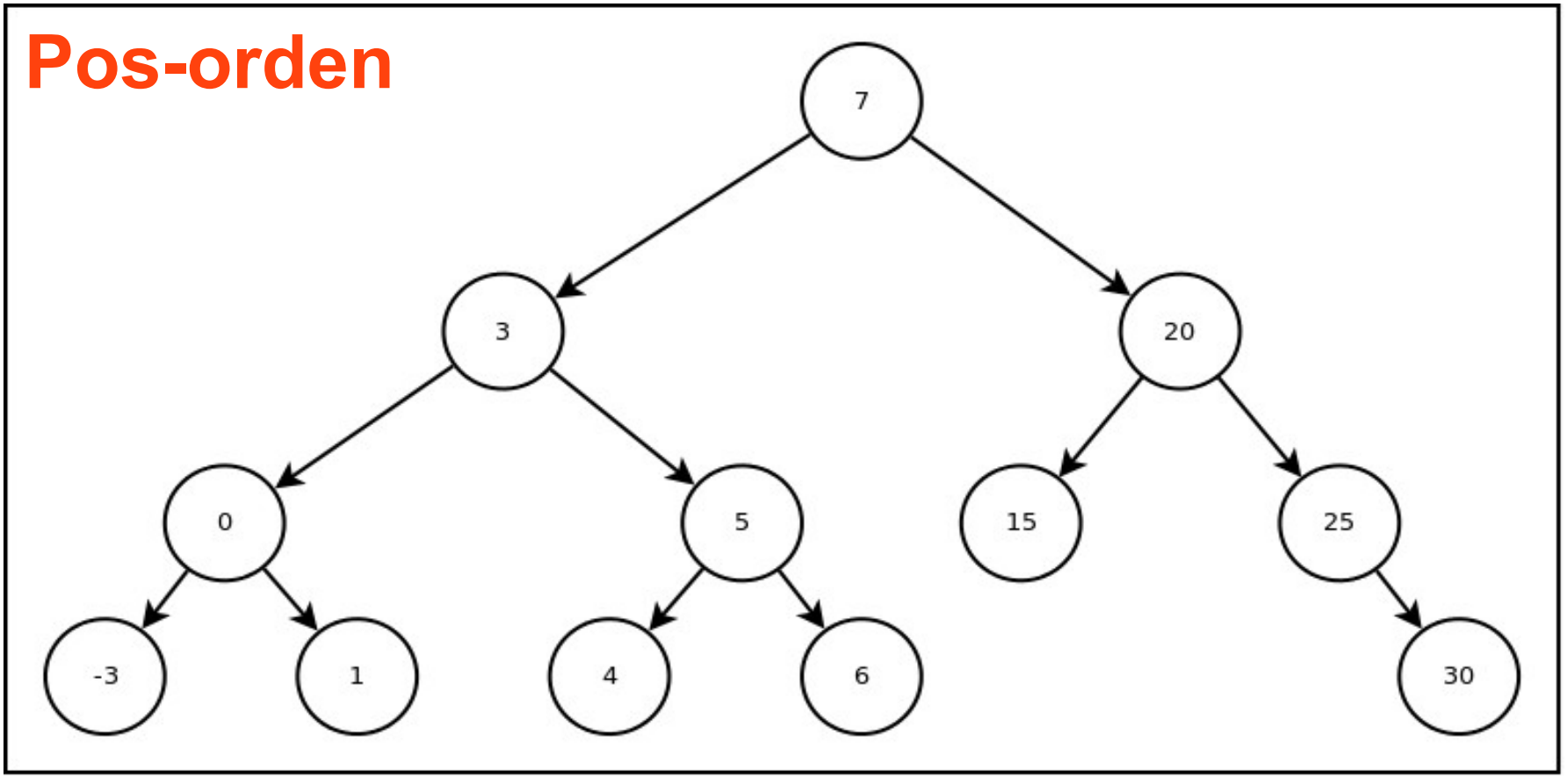


-3, 0, 1, 3, 4, 5, 6, 7, 15, 20, 25, 30

Árbol Binario Ordenado

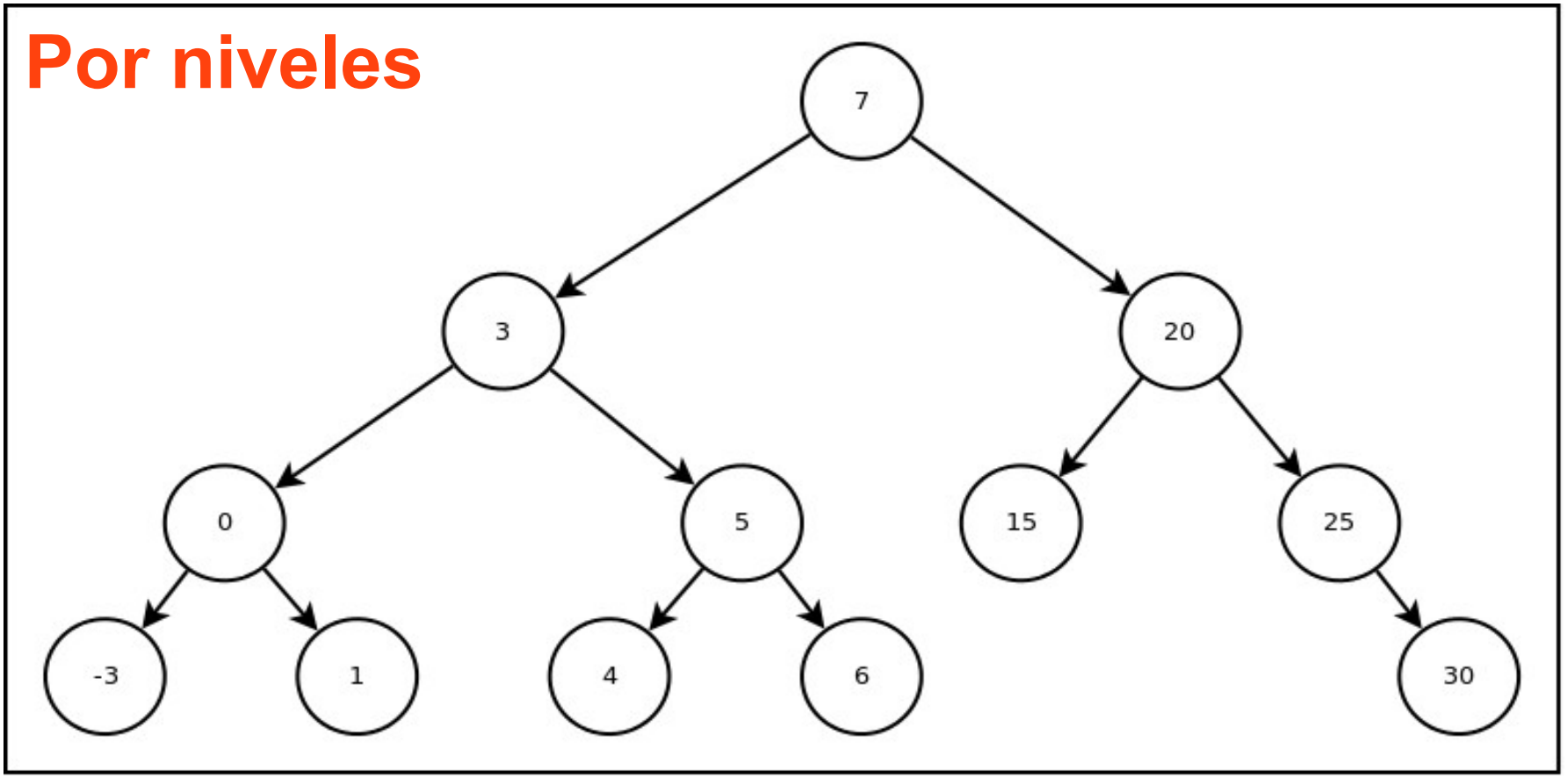


Árbol Binario Ordenado

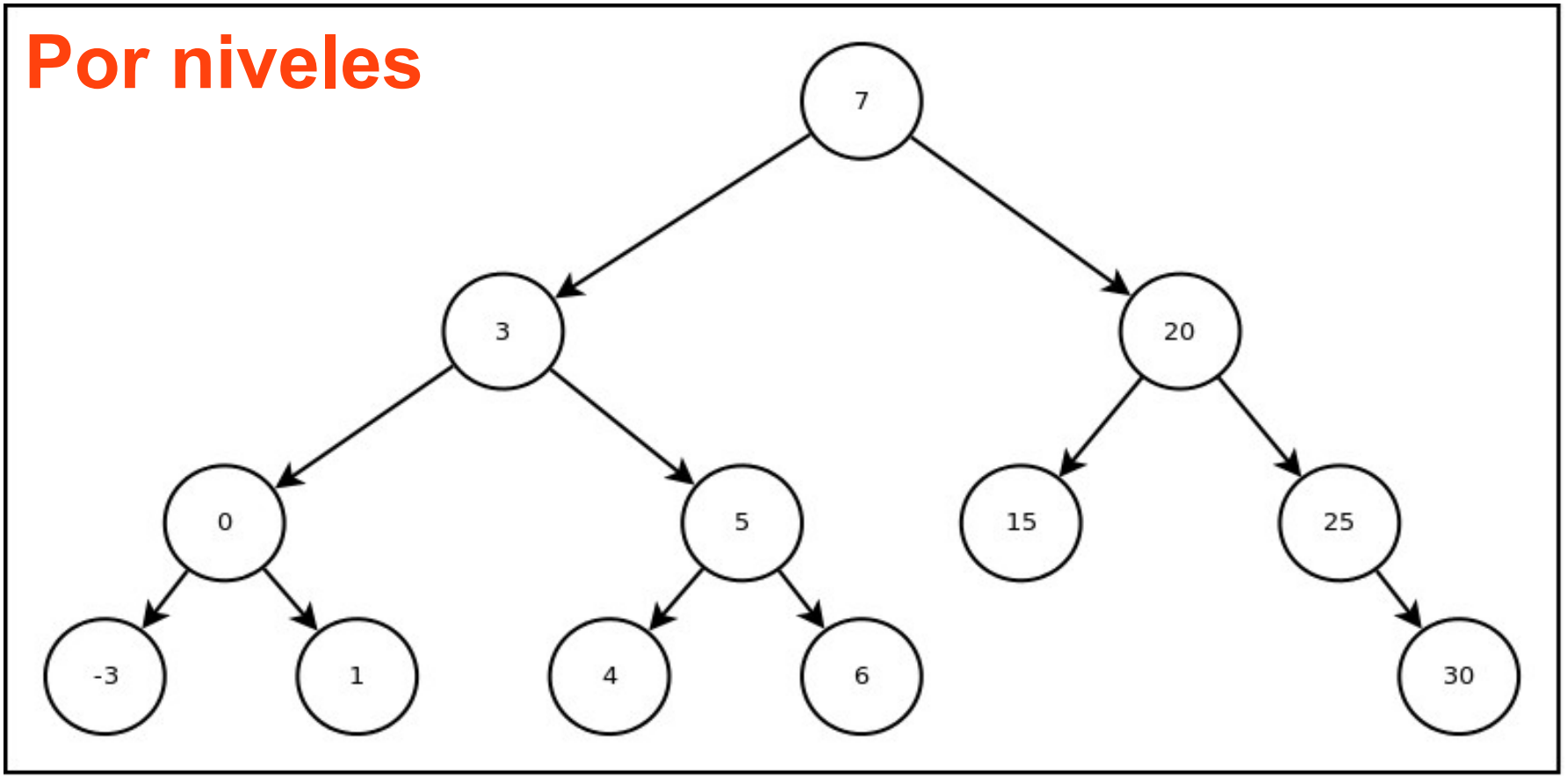


-3, 1, 0, 4, 6, 5, 3, 15, 30, 25, 20, 7

Árbol Binario Ordenado



Árbol Binario Ordenado



7, 3, 20, 0, 5, 15, 25, -3, 1, 4, 6, 30

Árbol Binario Ordenado

- ¿Mínimo del árbol?

- ¿Máximo del árbol?

Árbol Binario Ordenado

- ¿Mínimo del árbol?

  Seguir los hijos izquierdos desde la raíz hasta que el hijo izquierdo sea nulo → el mínimo.

  Primer elemento del inorden y el posorden.

- ¿Máximo del árbol?

  Seguir los hijos derechos desde la raíz hasta que el hijo derecho sea nulo → el máximo.

  Último elemento del inorden y el preorden.

Árbol Binario Ordenado

- Búsqueda: ¿Se facilita la búsqueda con el orden?

¿Complejidad de la búsqueda?

Árbol Binario Ordenado

- Búsqueda: ¿Se facilita la búsqueda con el orden?

  Definida de manera recursiva.

  - Comparar valor buscado con dato en nodo:
    - si es menor, buscar en el subárbol izquierdo.
    - si es mayor, buscar en el subárbol derecho.
    - si es igual, lo encontramos.

¿Complejidad de la búsqueda?

Proporcional a la altura.

¿Caso promedio? O(log n).

¿Peor caso? Árbol como lista → O(n).

Árbol Binario Ordenado

- Inserción:

  Buscar el “buen lugar”:

  - ¿A qué lado debe ir?
  - ¿Hay espacio?

  ¿Complejidad de la inserción?

Árbol Binario Ordenado

- Inserción:
  - Comparar valor a insertar con dato en nodo:
    - avanzar por subárbol izquierdo o derecho.
    - si es igual, ya existe en el árbol (duplicado).
  - Si no está duplicado:
    - crear el nuevo nodo con el dato a insertar.
    - asignarlo como hijo izquierdo o derecho del nodo actual.

¿Complejidad de la inserción?

Proporcional a la altura.

¿Caso promedio? O(log n).

¿Peor caso? Árbol como lista → O(n).

Árbol Binario Ordenado

- Eliminación:

  Seguimiento desde el padre:

  - Mantenimiento de apuntadores.

  Casos a borrar:

  - Sin hijos.
  - Un sólo hijo.
  - Dos hijos.

  ¿Complejidad de la eliminación?

Árbol Binario Ordenado

- Eliminación:
  - Comparar valor a eliminar con dato en nodo:
    - avanzar por subárbol izquierdo o derecho.
    - si es igual, lo encontramos.
  - Si lo encontramos:
    - si no tiene hijos, borrar el nodo.
    - si tiene un hijo, usarlo como nodo de reemplazo.
    - en caso de dos hijos, buscar el nodo mayor del subárbol izquierdo y usarlo como nodo de reemplazo.

¿Complejidad de la eliminación?

¿Caso promedio? O(log n). ¿Peor caso? O(n).

Árbol Binario Ordenado

- Ejercicio:

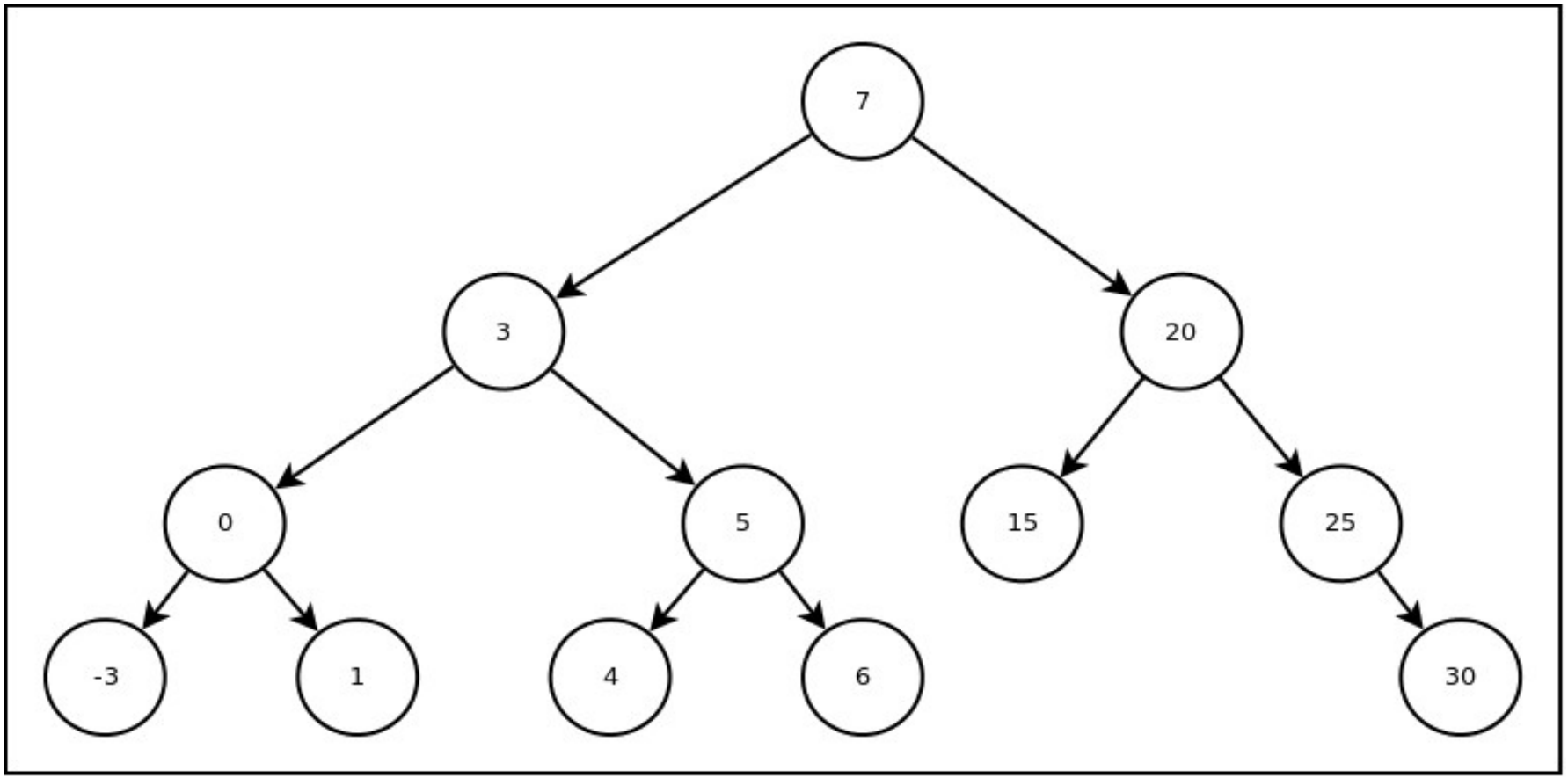
  ¿Cómo queda un árbol binario de búsqueda luego de insertar las siguientes secuencias de elementos?

Árbol Binario Ordenado

7, 3, 0, -3, 1, 5, 4, 6, 20, 15, 25, 30

Árbol Binario Ordenado

7, 3, 0, -3, 1, 5, 4, 6, 20, 15, 25, 30

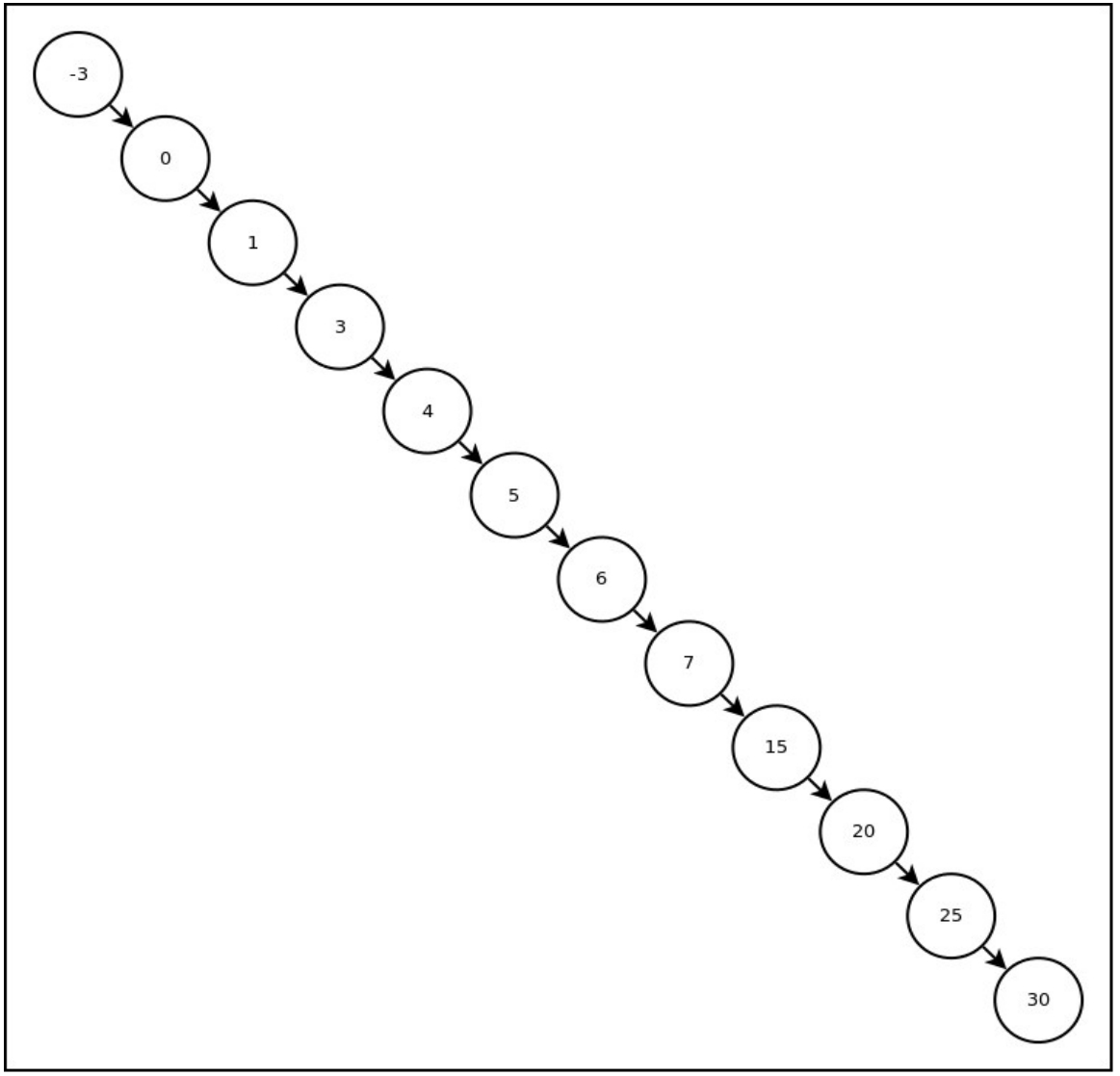


Árbol Binario Ordenado

-3, 0, 1, 3, 4, 5, 6, 7, 15, 20, 25, 30

Árbol Binario Ordenado

-3, 0, 1, 3, 4, 5, 6, 7, 15, 20, 25, 30

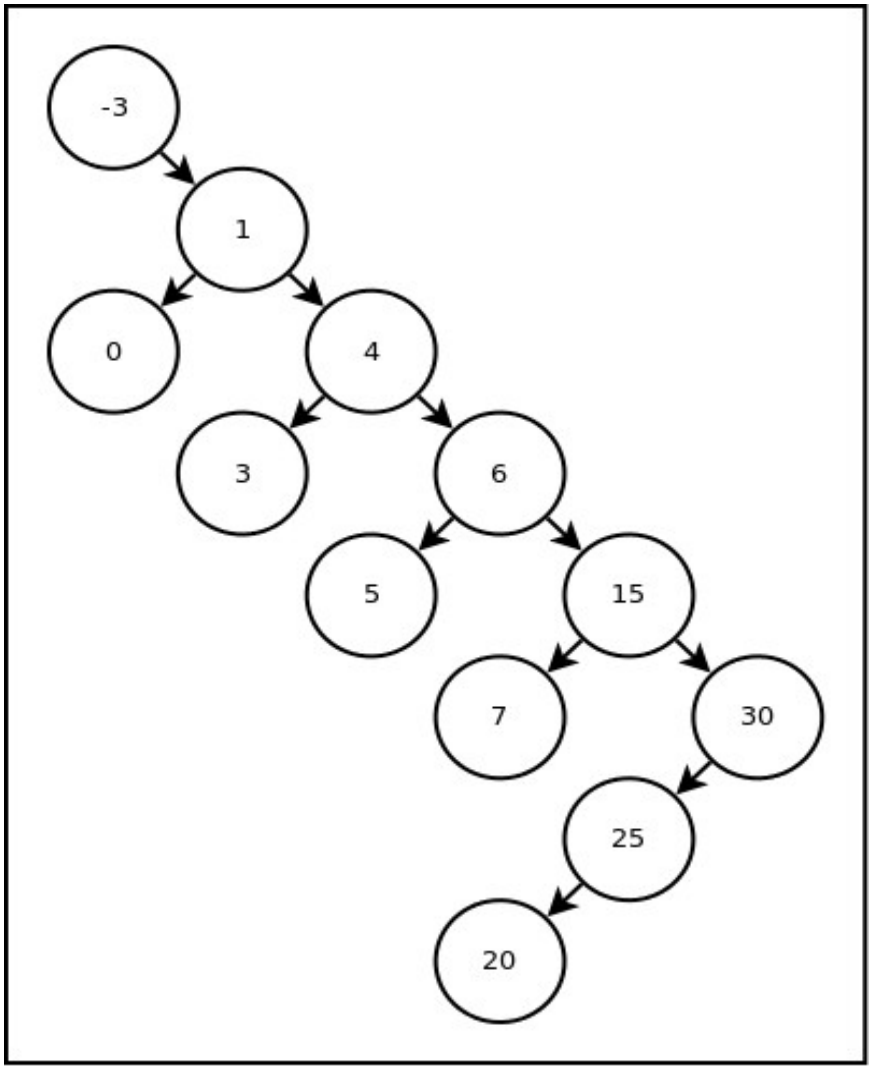


Árbol Binario Ordenado

-3, 1, 0, 4, 6, 5, 3, 15, 30, 25, 20, 7

Árbol Binario Ordenado

-3, 1, 0, 4, 6, 5, 3, 15, 30, 25, 20, 7



¿Cómo garantizar árboles “bonitos”?

- Balanceando:
  - Evitar “listas”.
  - Evitar ramas cortas.
- Garantizar búsqueda / inserción / eliminación en O(log n).
- Árboles AVL y RN.

Representación de árboles

- Árbol binario

  - elemento base de construcción: Nodo Binario

  - Nodo Binario contiene:

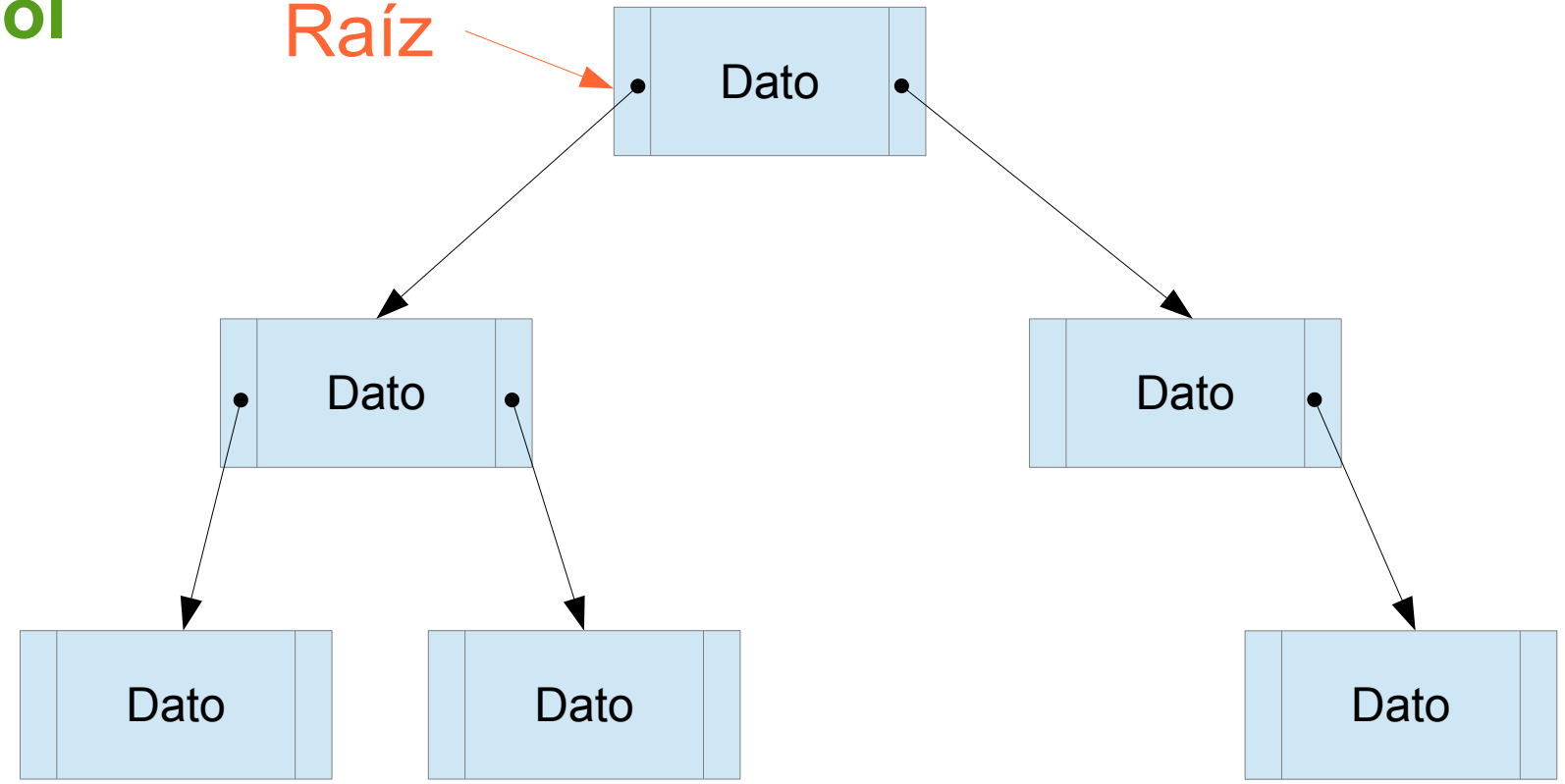
    - Un dato (primario o definido por el usuario)
    - Un vínculo (apuntador) al hijo (subárbol) izquierdo
    - Un vínculo (apuntador) al hijo (subárbol) derecho

  - Un árbol binario se define entonces como un apuntador al Nodo Binario de inicio (raíz)

Representación de árboles

Nodo

Árbol



TAD Nodo Binario

- Auxiliar para la definición de un árbol binario
- datos miembros
  - dato del nodo
  - apuntador a nodo (hijo, subárbol) izquierdo
  - apuntador a nodo (hijo, subárbol) derecho
- operaciones
  - constructores, destructores
  - asignar, extraer el dato del nodo
  - asignar, extraer una referencia a nodo (hijo)

NodoBinario.h

```
template< class T >
class NodoBinario {
  protected:
    T dato;
    NodoBinario<T> *hijoIzq;
    NodoBinario<T> *hijoDer;

  public:
    T& obtenerDato();
    void fijarDato(T& val)
    NodoBinario<T>* obtenerHijoIzq();
    NodoBinario<T>* obtenerHijoDer();
    void fijarHijoIzq(NodoBinario<T> *izq);
    void fijarHijoDer(NodoBinario<T> *der);
};
```

TAD Árbol Binario Ordenado

- Estado: conjunto mínimo de datos
  - Referencia al nodo raíz

TAD Árbol Binario Ordenado

- Interfaz: comportamiento (operaciones)
  - constructores, destructores
  - obtener, fijar el nodo raíz, la referencia izquierda, la referencia derecha
  - verificación de árbol vacío
  - tamaño del árbol (en nodos), altura del árbol
  - inserción, eliminación de un nodo
  - verificación de un nodo en el árbol
  - recorridos: preorden, inorden, posorden, niveles

ArbolBinarioOrd.h

```
template< class T >
class ArbolBinarioOrd {
  protected:
    NodoBinario<T> *raiz;

  public:
    bool esVacio();
    T& datoRaiz();
    int altura();
    int tamano();
    bool insertar(T& val);
```

ArbolBinarioOrd.h

```
    bool eliminar(T& val);
    bool buscar(T& val);
    void preOrden();
    void inOrden();
    void posOrden();
    void nivelOrden();
};
```

Tarea

- Revisar los videos en la siguiente lista de reproducción: https://www.youtube.com/playlist?list=PLXNX3bqAk3JMYibwD-XgAicV2AeyT4gR9
- Realizar (en grupos) la implementación del TAD Árbol Binario Ordenado, siguiendo los lineamientos dados en los videos.

Referencias

- www.cs.umd.edu/~mount/420/Lects/420lects.pdf
- www.cs.nmsu.edu/~epontell/courses/cs272/disp/trees/2004/tree2004.pdf
- www.csd.uwo.ca/~vmazalov/CS1027a/notes/CS1027-Trees_6up.pdf
- people.cis.ksu.edu/~schmidt/300s05/Lectures/Week7b.html
- pages.cs.wisc.edu/~ealexand/cs367/NOTES/AVL-Trees/index.html
- www.dcs.gla.ac.uk/~pat/52233/slides/AVLTrees1x1.pdf