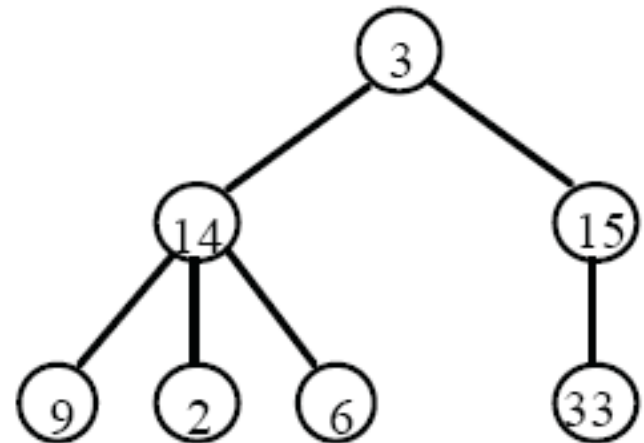


# Tipos de Datos Arborescentes

## Tema III



# Bibliografía

- Capítulo 4 y 6 del libro “Weiss, M.A.: *Data Structures and Algorithm Analysis in C++, Fourth Edition*, Pearson/Addison Wesley, 2014.”
- Capítulos 5, 6, 10 y 13 del libro “Clifford A. Shaffer: *Data Structures and Algorithm Analysis, Edition 3.2 (C++ Version)* March 2013.”
- Capítulos 5 a 9 del libro “Z. J. Hernández, J. C. Rodríguez, etc .: *Fundamentos de Estructuras de Datos. Soluciones en Ada, Java y C++*. Thomson Paraninfo, 2005“
- Libro: “Joyanes, L., Zahonero, I., Fernández, M. y Sánchez, L.: *Estructuras de datos. Libro de problemas*, McGraw Hill, 1999.”, capítulos 8, 9 y 10
- Capítulos 6 a 8 del libro “Martí Oliet, N., Ortega Mallén, Y., Verdejo López, J.A.: *Estructuras de datos y métodos algorítmicos*. 2ª edición: 213 ejercicios resueltos. Garceta, 2013”,
- Capítulo 5 del libro “X. Franch: *Estructuras de datos. Especificación, diseño e implementación*, 3ª edición, Ediciones UPC, 2001”
- Tema III (lecciones 11 a 18) del libro de apuntes “Campos Laclaustra, J.: *Apuntes de Estructuras de Datos y Algoritmos*, segunda edición, 2018 (versión 4, 2022).
- *Y muchos otros....*



# Tipos y estructuras de datos arborescentes

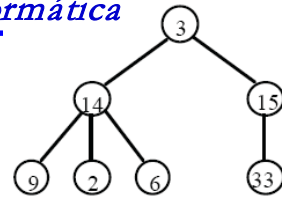
---

*Lección 11*

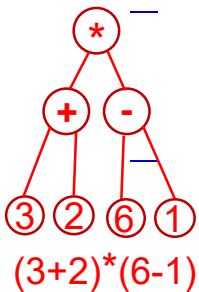
# Esquema

- Introducción
- Aplicación y usos
- Definición y terminología básica
- Recorridos

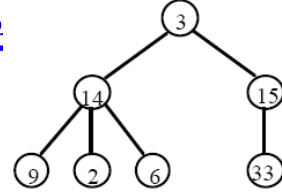
# Introducción



- Los **árboles** son estructuras no lineales que organizan sus elementos, denominados *nodos*, formando jerarquías
- Utilizaremos árboles para representar, almacenar y gestionar **colecciones de datos entre los que existen relaciones no lineales**, normalmente jerárquicas, **y también** como **herramientas** para diseñar e implementar soluciones sencillas y eficientes a problemas.
- Uso muy frecuente en computación:



- Representación de relaciones jerárquicas de todo tipo (sistemas jerárquicos de directorios, árboles genealógicos, árboles de clasificación, organigramas, relaciones de inclusión, dependencias....)
- Diseño y construcción de compiladores, interpretes, procesamiento de textos, etc.
- Representación, almacenamiento y evaluación de expresiones (aritméticas, etc.)
- Representación de gramáticas, y programas, en compilación o interpretación de programas
- Como concepto y herramienta de diseño de estructuras y algoritmos, solución a problemas complejos
  - Árboles de decisión en IA, algoritmos de búsqueda, bases de datos jerárquicas, etc.



# Aplicación y usos de los árboles

- Situaciones habituales al resolver problemas (en EDA) en las que usar árboles:
  - TADs para almacenar y manipular colecciones de datos entre los que existe alguna **relación jerárquica**:
    - Para la definición de TADs arborescentes:
      - TADs que representan árboles y ofrecen operaciones para manipular el árbol
      - Ej.: árbol genealógico, árbol de directorios en un SO, clasificaciones, organigramas, etc.
    - Como estructuras de datos para la representación interna y soporte para implementaciones de otros módulos o TADs
      - Estructuras que reflejen las relaciones existentes entre los datos (jerarquías, dependencias, inclusiones,...)
      - Implementaciones sencillas y eficientes
      - Uso del árbol más o menos transparente según el TAD (Ej: operaciones para manipular la colección o las relaciones, pero no el árbol)
  - Árboles como alternativas eficientes para implementar la representación interna de TADs **contenedores**:
    - Almacenar y manipular colecciones de **datos sin relaciones jerárquicas** entre ellos
    - **mejorar la eficiencia en tiempo** de las operaciones de manipulación con respecto a la eficiencia conseguida con representaciones lineales (uso de árbol oculto en la implementación)
- Estudiaremos diferentes tipos de árboles, con diferentes características, alternativas de implementación y costes en tiempo para sus operaciones...

# Definición y terminología básica

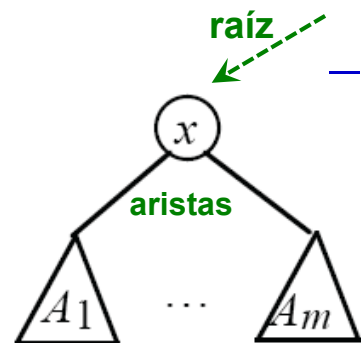
- **Árbol:**

- Definición no recursiva:

- Conjunto de elementos de un mismo tipo, denominados nodos o vértices, que pueden representarse en un grafo no orientado, conexo y acíclico, en el que existe un vértice destacado denominado raíz

- Definición recursiva:

- Un árbol es un conjunto no vacío de elementos del mismo tipo, tal que:
  - Existe un elemento destacado llamado raíz del árbol
  - el resto de los elementos se distribuyen en  $m$  subconjuntos disjuntos ( $0 \leq m$ ), llamados subárboles del árbol original, cada uno de los cuales es a su vez un árbol  $(A_1, \dots, A_m)$



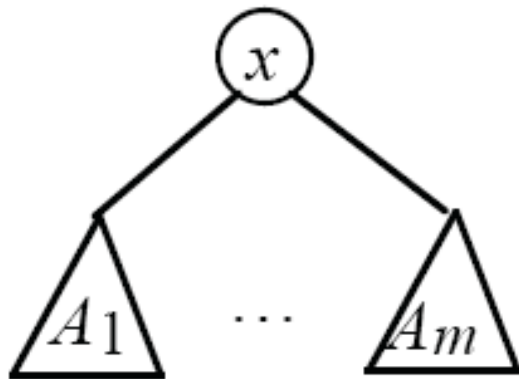
(Los subárboles se *conectan* a la raíz mediante aristas.)

Tanto los subárboles  $A_i$ , como sus nodos raíz, se denominan hijos de la raíz  $x$ )

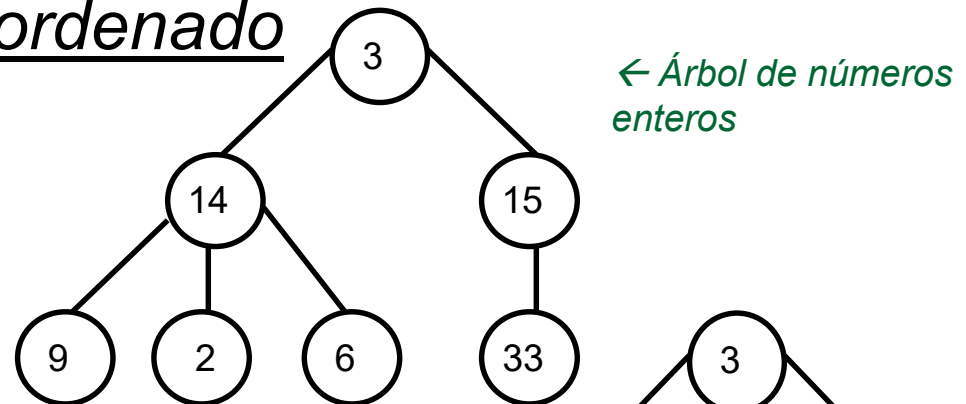
# Definición y terminología básica

- **Árbol ordenado:**

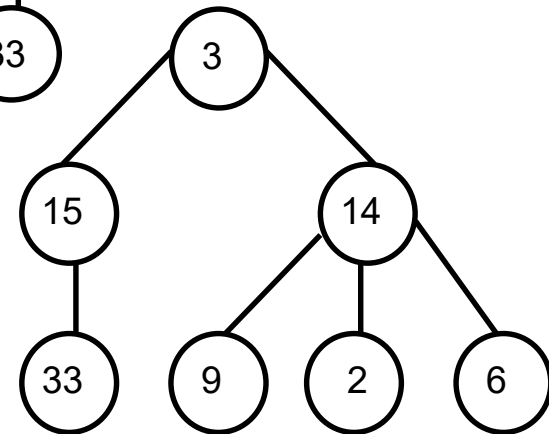
- Si en el conjunto de subárboles de un árbol se supone definida una relación de orden total, el árbol se denomina ordenado



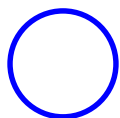
Árbol ordenado con raíz  $X$  y subárboles  $A_1 \dots A_m$



Otro árbol distinto, → de números enteros



Leyenda: **Nodo**



**Árbol**

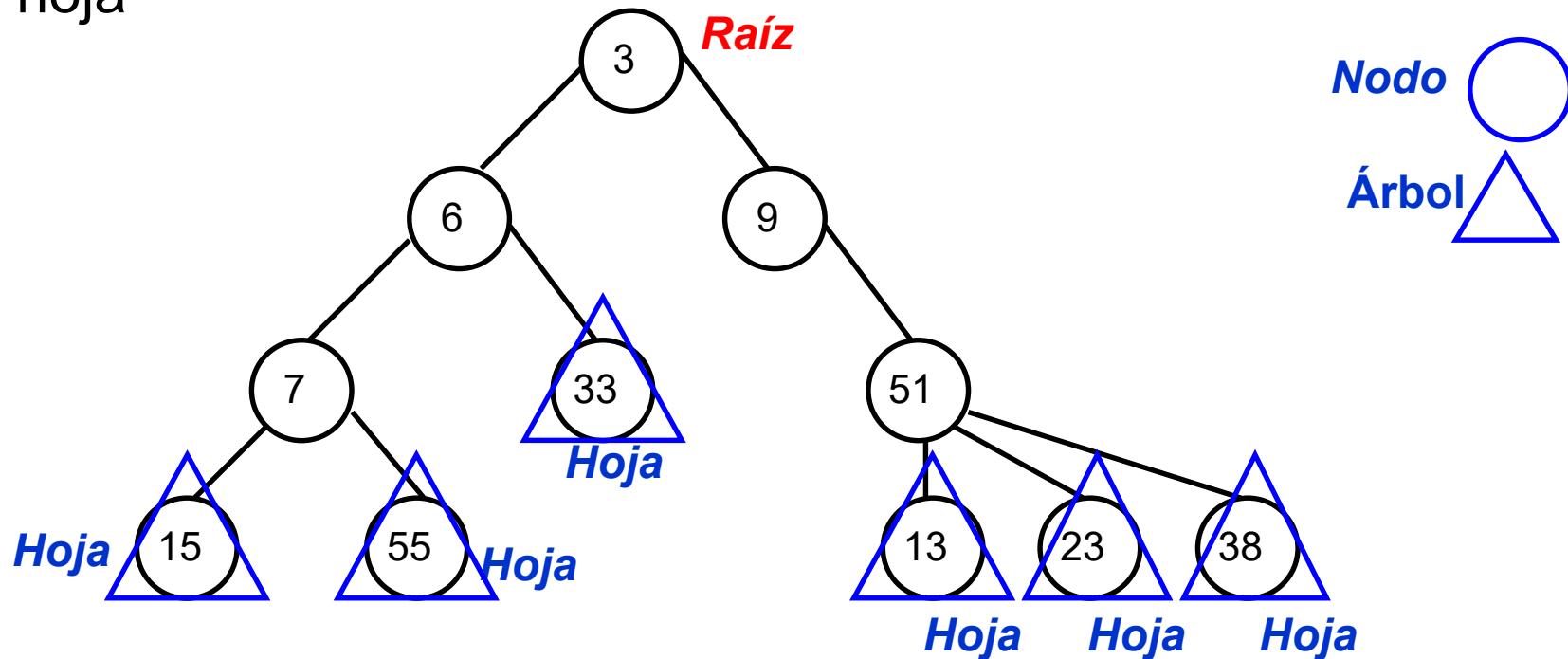




# Definición y terminología básica

- **Hoja:**

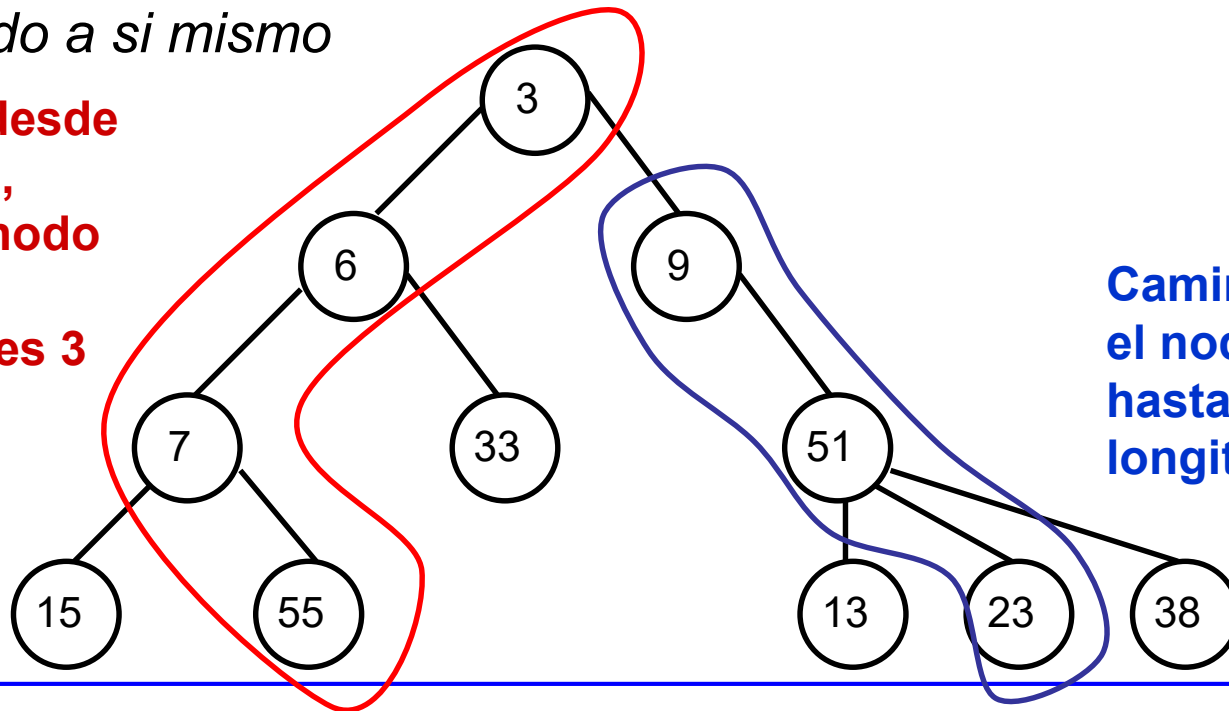
- Un árbol compuesto por un solo elemento se denomina hoja



# Definición y terminología básica

- **Camino:** un camino es una secuencia de nodos  $n_1, \dots, n_s$ ,  $s \geq 1$ , tal que  $n_{i+1}$  es *hijo* de  $n_i$ , para todo  $1 \leq i \leq s-1$
- **Longitud de camino:** número de nodos en la secuencia del camino, menos 1 (es decir, longitud = N° de aristas en el camino)
- Por convenio: *diremos que existe un camino de longitud 0, de todo nodo a si mismo*

**Camino desde  
el nodo 3,  
hasta el nodo  
55, y su  
longitud es 3**



**Camino desde  
el nodo 9  
hasta el 23, de  
longitud 2**

**NO EXISTE ningún camino desde el nodo 7 hasta el nodo 51, ni viceversa**

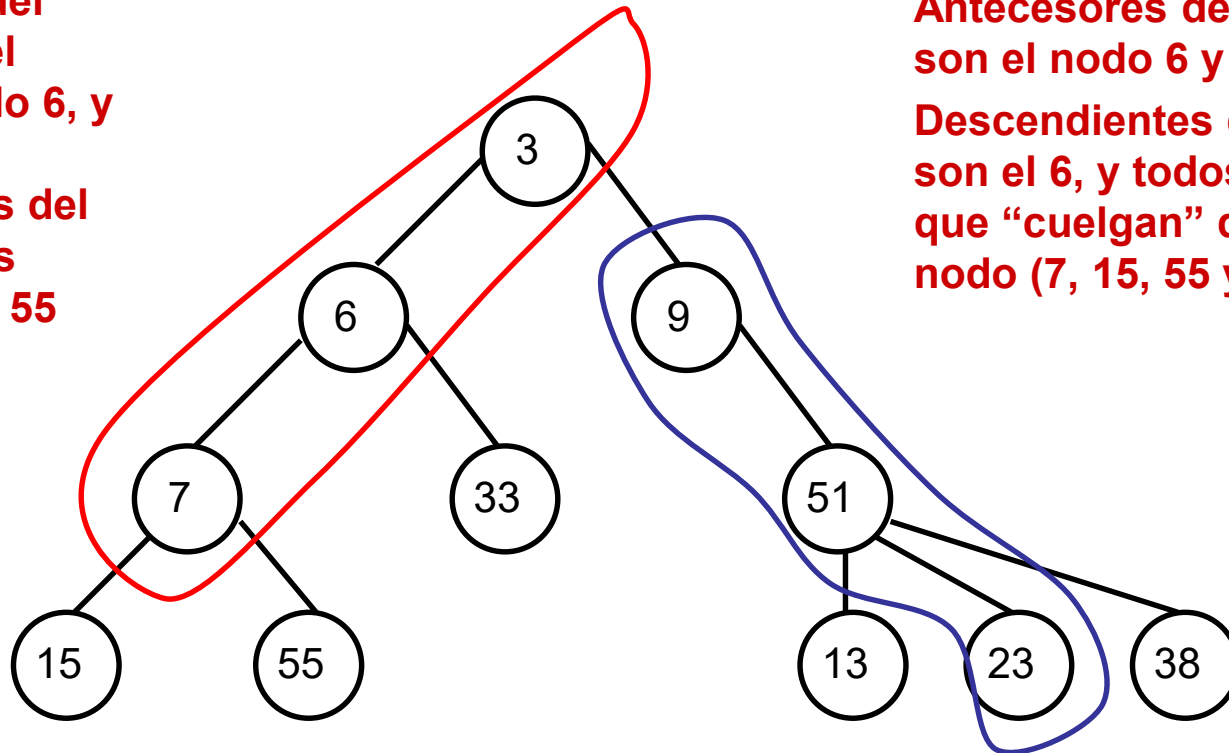
# Definición y terminología básica

- Si en un árbol A existe un camino desde el nodo  $n_1$  hasta el nodo  $n_2$ , se dice que  $n_1$  es **antecesor** de  $n_2$  y que  $n_2$  es **descendiente** de  $n_1$

→ Por tanto todo nodo es antecesor y descendiente de si mismo

→ Todos los nodos del árbol son descendientes de su raíz

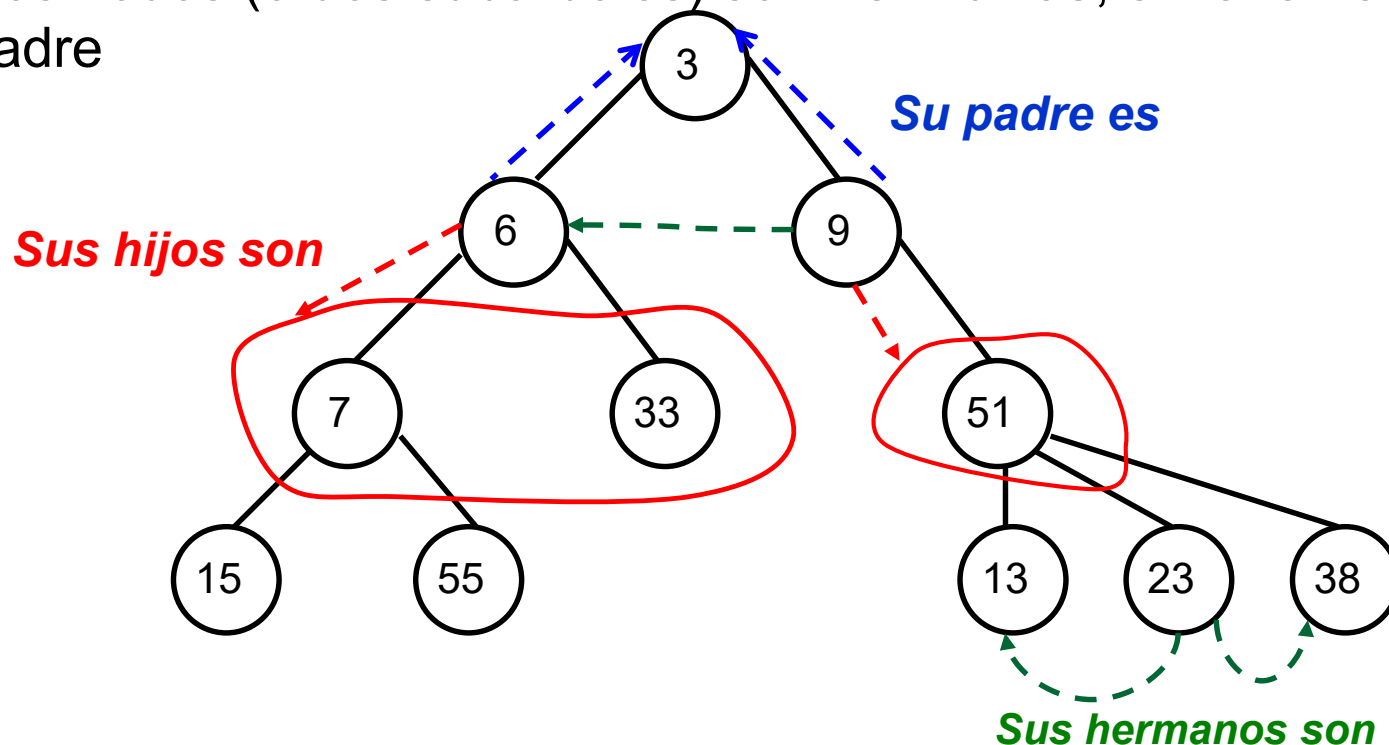
**Antecесores del nodo 7, son: el nodo 7, el nodo 6, y el nodo 3.**  
**Descendientes del nodo 7 son los nodos 7, 15 y 55**



**Antecесores del nodo 6, son el nodo 6 y el nodo 3**  
**Descendientes del nodo 6 son el 6, y todos los nodos que “cuelgan” de dicho nodo (7, 15, 55 y 33).**

# Definición y terminología básica

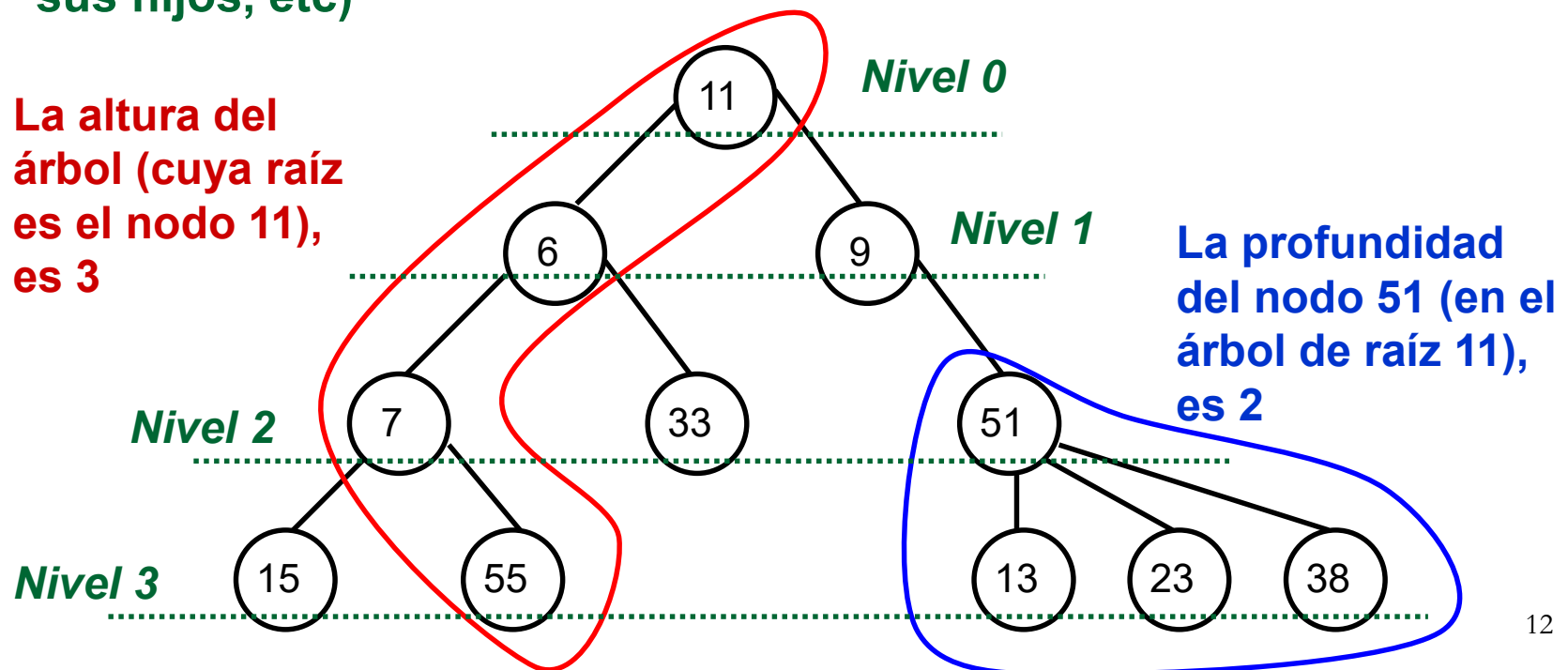
- Los **antecesoros** o **descendientes** de un nodo, distintos del mismo nodo, se denominan **propios**
- El **padre** de un nodo, es su primer antecesor propio, si existe
- Los **hijos** de un nodo, son sus primeros descendientes propios, si existen (son los nodos raíz de sus subárboles)
- Dos nodos (o dos subárboles) son **hermanos**, si tienen el mismo padre



# Definición y terminología básica

- La **altura** de un árbol es la longitud del camino más largo que puede encontrarse en el árbol
- La **profundidad de un nodo en un árbol**, es la longitud del único camino existente desde la raíz del árbol hasta el nodo
- Un **nivel**, es el conjunto de nodos de un árbol con igual profundidad (**en el nivel 0 solo está la raíz del árbol, en el nivel 1 sus hijos, etc**)

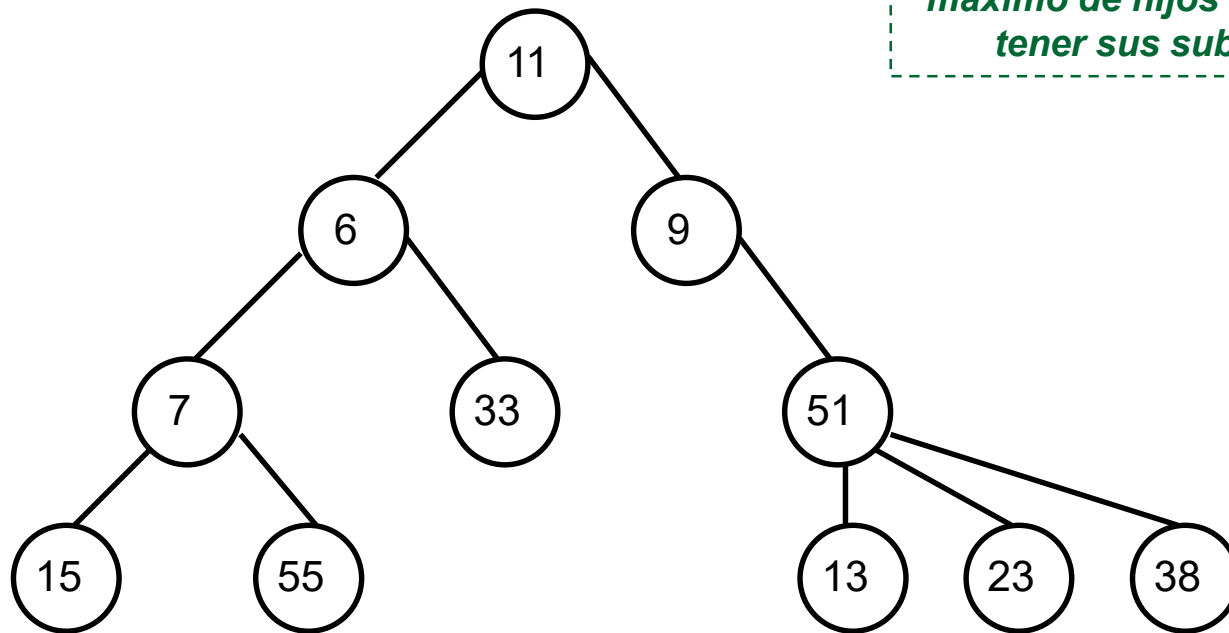
La altura del árbol (cuya raíz es el nodo 11), es 3



# Definición y terminología básica

- El **grado** de un árbol es el número máximo de hijos que pueden tener sus subárboles
  - Árbol n-ario → árbol de grado n
  - Árbol binario ~~→~~ árbol de grado 2
  - Etc...

*Árbol general es aquel en el que no se considera ninguna limitación en el número máximo de hijos que pueden tener sus subárboles*

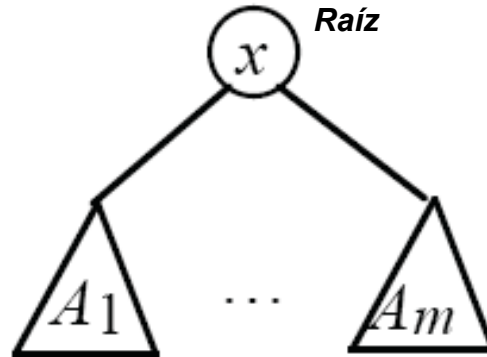


**Árbol de grado 3**

# Definición y terminología básica

- **Árbol N-ario:**

- Un árbol n-ario (con  $n \geq 1$ ) es un conjunto no vacío de elementos del mismo tipo tal que:
  - Existe un elemento destacado llamado raíz del árbol
  - el resto de los elementos se distribuyen en  $m$  subconjuntos disjuntos ( $0 \leq m \leq n$ ), llamados subárboles del árbol original, cada uno de los cuales es a su vez un árbol n-ario



*Un árbol n-ario no puede ser vacío*

# Definición y terminología básica

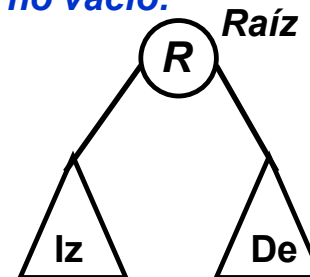
- **Árbol binario:**

- Conjunto de elementos o nodos del mismo tipo, tal que:
  - o bien es el conjunto vacío, y entonces se llama **árbol vacío**
  - o bien es no vacío, en cuyo caso existe un elemento destacado llamado **raíz**, y el resto de los elementos se distribuyen en **dos** subconjuntos disjuntos, llamados **subárbol izquierdo** y **subárbol derecho**, cada uno de los cuales es un árbol binario

Árbol binario vacío:



Árbol binario no vacío:





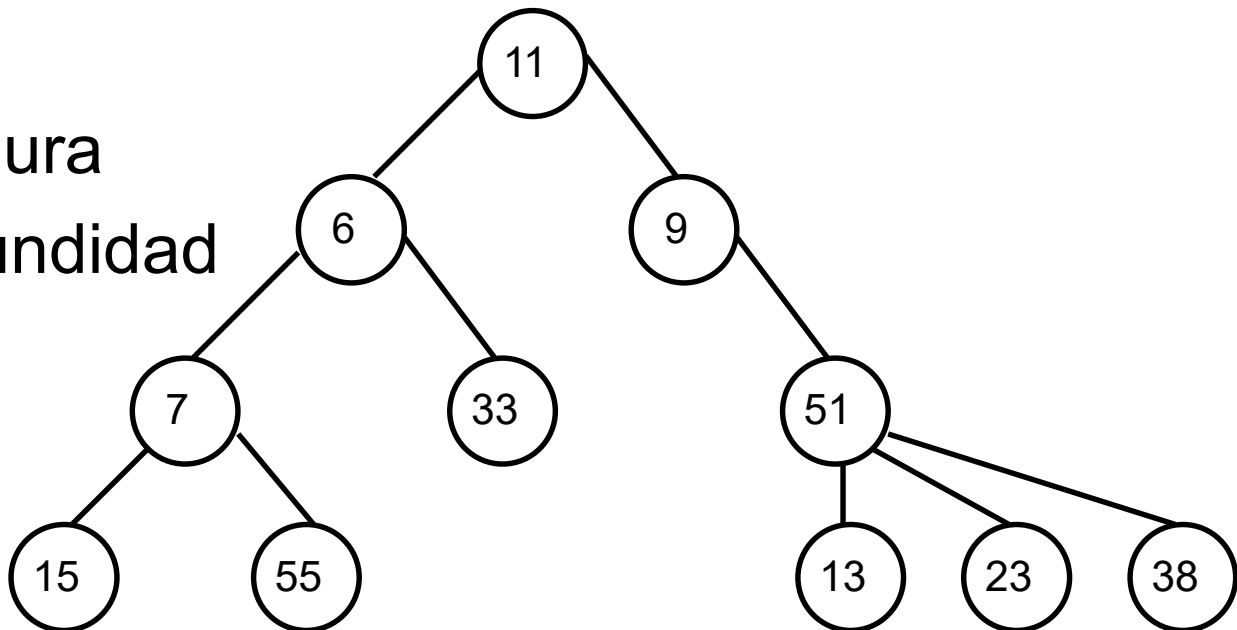
# Recorridos en árboles

*Un recorrido de un árbol consiste en visitar todos los elementos del árbol una sola vez*

Recorridos:

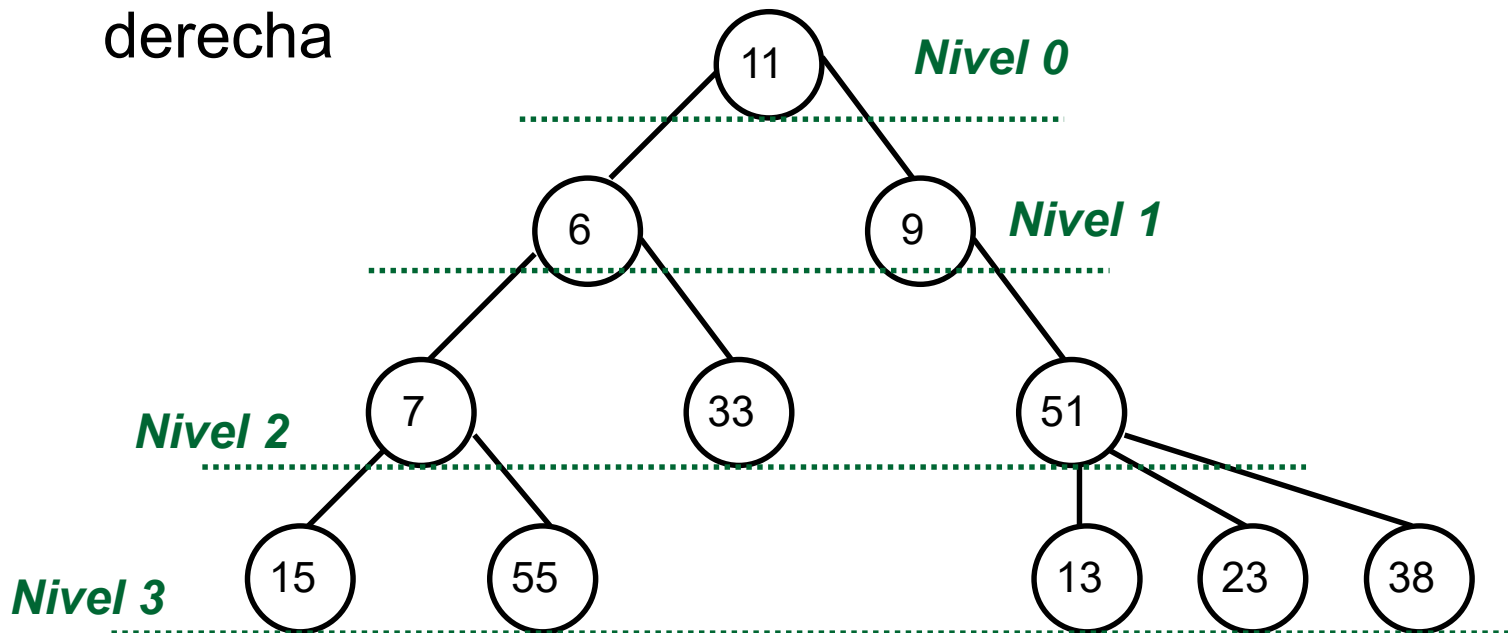
→ en anchura

→ en profundidad



# Recorrido en anchura

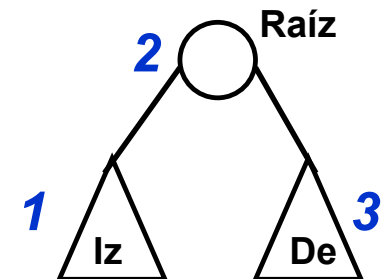
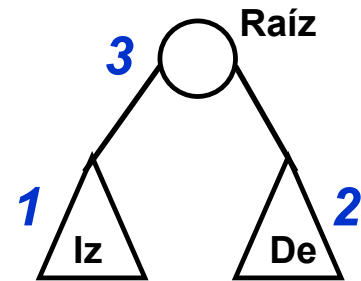
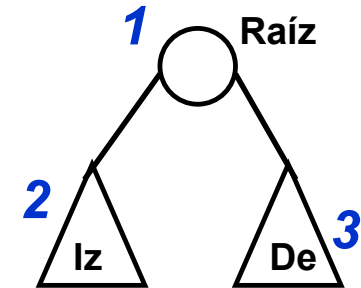
- El **recorrido en anchura** de un árbol consiste en visitar todos los elementos del árbol una sola vez, de la forma:
  - primero se visitan los elementos del nivel 0, luego los del nivel 1, y así sucesivamente,
  - En cada nivel, se visitan los elementos de izquierda a derecha



# Recorridos en árboles binarios

## Recorridos en profundidad de **árboles binarios**:

- Recorrido en **pre-orden**:
  1. se visita la raíz
  2. se recorre en *pre-orden* el hijo izquierdo
  3. se recorre en *pre-orden* el hijo derecho
- Recorrido en **post-orden**:
  1. se recorre en *post-orden* el hijo izquierdo
  2. se recorre en *post-orden* el hijo derecho
  3. se visita la raíz
- Recorrido en **in-orden**:
  1. se recorre en *in-orden* el hijo izquierdo
  2. se visita la raíz
  3. se recorre en *in-orden* el hijo derecho

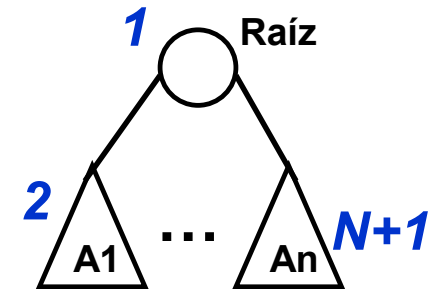


# Recorridos en árboles n-arios

## Recorridos en profundidad de **árboles n-arios**:

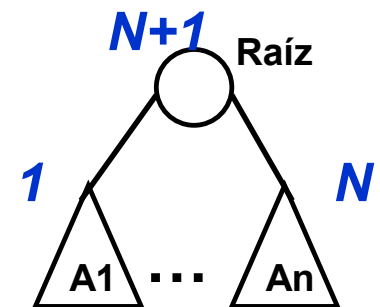
- Recorrido en **pre-orden**:

1. se visita la raíz
2. se recorren en *pre-orden* todos los subárboles, de izquierda a derecha

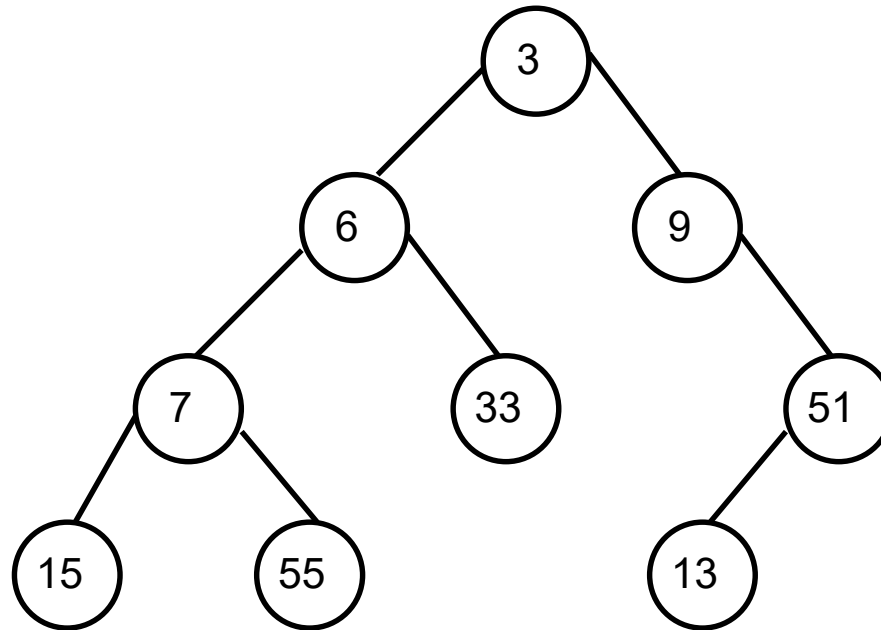


- Recorrido en **post-orden**:

1. se recorren en *post-orden* todos los subárboles, de izquierda a derecha
1. se visita la raíz

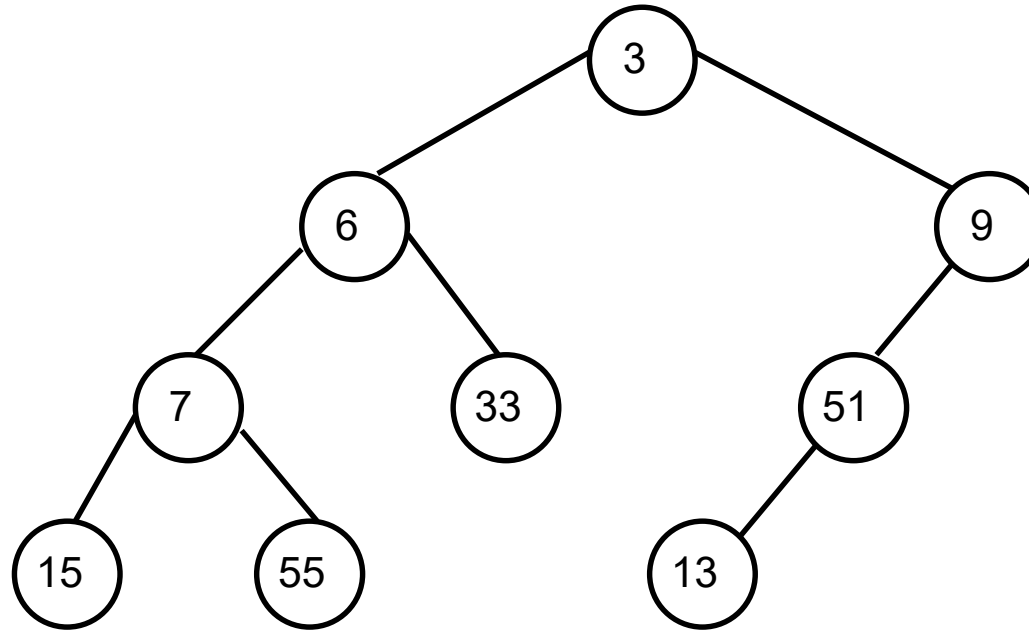


# Recorridos en árboles binarios



- Ejercicio: Cuál será la secuencia de números que se obtendrá al recorrer el árbol utilizando:
  - Recorrido en *pre-orden*
  - Recorrido en *post-orden*
  - Recorrido en *in-orden*

# Recorridos en árboles binarios



- Ejercicio: Cuál será la secuencia de números que se obtendrá al recorrer el árbol utilizando:
  - Recorrido en *pre-orden*
  - Recorrido en *post-orden*
  - Recorrido en *in-orden*

