



Árboles

- Presentar la estructura no lineal más importante en computación
- Mostrar la especificación e implementación de varios tipos de árboles
- Algoritmos de manipulación de árboles

Contenido

- 1. Terminología fundamental**
 - 1.1. Recorridos de un árbol**
- 2. Árboles binarios**
 - 2.1. Definición
 - 2.2. Especificación
 - 2.3. Implementación
- 3. Árboles binarios de búsqueda**
 - 3.1. Definición
 - 3.2. Especificación
- 4. Árboles binarios equilibrados**
 - 4.1. Árboles AVL
- 5. Árboles generales**
 - 5.1. Especificación
 - 5.2. Implementación

Árboles

■ Bibliografía

- Weiss, Mark Allen; *Estructuras de datos en Java*, Pearson. 2013. Págs. 515-557, 641-704, 797-805
- Lewis J. y Chase J.; *Estructuras de datos con Java. Diseño de estructuras y algoritmos*. Segunda Edición. Pearson Addison Wesley 2006. Pags. 404-411, 416-420
- Rowe, Glen; *An Introduction to Data Structures and Algorithms with Java*, Prentice Hall, 1998. Págs. 282-331

Árbol

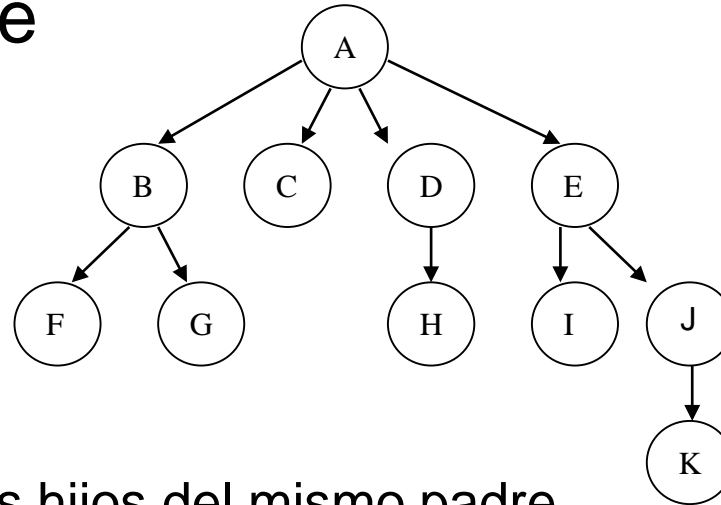
- **Árbol:** estructura **no lineal** y **dinámica** más importante en computación
- **Árbol:** estructura *jerárquica* de una *colección* de objetos
 - Colección de elementos llamados nodos, uno de los cuales se distingue como **raíz**,
 - Una relación (de "paternidad") que impone una estructura jerárquica sobre los nodos

Árbol

- Formalmente, un árbol se puede definir de manera **recursiva** como:
 - Un árbol sin nodos es un árbol, llamado *árbol vacío* o *nulo*.
 - Un solo nodo es, por sí mismo, un árbol. Ese nodo es también la raíz de dicho árbol
 - Supóngase que n es un nodo y que A_1, A_2, \dots, A_K son árboles con raíces n_1, n_2, \dots, n_K , respectivamente. Se puede construir un nuevo árbol haciendo que n se constituya en el padre de los nodos n_1, n_2, \dots, n_K
 - n es la raíz del nuevo árbol y A_1, A_2, \dots, A_K son los **subárboles** (o árboles hijos) de la raíz
 - los nodos n_1, n_2, \dots, n_K reciben el nombre de **hijos** del nodo n y el nodo n recibe el nombre de **padre** de dichos nodos

Árbol

■ Gráficamente



- **Hermanos:** nodos hijos del mismo padre
Ej: B,C,D y E son hermanos
- **Camino** del nodo n_1 al nodo n_K : sucesión de nodos de un árbol n_1, n_2, \dots, n_K , tal que n_i es el padre de n_{i+1} , $i = 1, 2, \dots, K-1$
Ej: A, E, J, K es un camino
 - **Longitud de un camino:** número de nodos del camino menos 1.
Caminos de longitud cero: aquellos que van de cualquier nodo a él mismo
Ej: el camino A, E, J, K tiene longitud 3

Árbol

- Si existe un camino de un nodo **n** a otro **m**, entonces **n** es un **antecesor** de **m**, y **m** es un **descendiente** de **n**
 - En un árbol, la raíz es el único nodo que no tiene antecesores
 - **Hoja** o **nodo terminal**: nodo sin descendientes
- **Subárbol** de un árbol: nodo junto con todos sus descendientes
- **Grado** de un nodo: número de subárboles que tiene. Los *nodos terminales* u *hojas* tienen grado 0

Ej: el grado de A es 4.

Árbol

- **Altura** de un nodo: longitud del camino más largo de ese nodo a una hoja. La ***altura del árbol*** es la altura de la raíz
Ej: B tiene altura 1, E altura 2 y H altura 0.
La altura del árbol es 3
- **Nivel o profundidad** de un nodo: longitud del único camino desde la raíz a ese nodo. Por definición, la raíz tiene *nivel 0*. La ***profundidad de un árbol*** se define como el máximo de los niveles de los nodos del árbol
Ej: B tiene nivel 1, H nivel 2.
La profundidad del árbol es 3

Árbol

- Estrategias para recorrer un árbol y procesar sus nodos:

- ☐ Recorridos en profundidad
- ☐ Recorridos en anchura

- **Recorridos en profundidad**

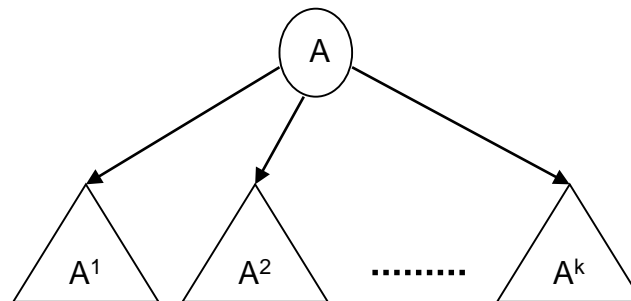
Tres formas de recorrer un árbol en profundidad:

- ☐ **Preorden** u orden previo
- ☐ **Inorden** u orden simétrico
- ☐ **Postorden** u orden posterior

Árbol

Preorden:

- Si el árbol A es nulo \Rightarrow la lista vacía es el listado en preorden
- Si el árbol A tiene un solo nodo \Rightarrow ese nodo constituye el listado del árbol A en preorden
- Si el árbol A tiene más de un nodo, es decir, tiene como raíz el nodo n y los subárboles A_1, A_2, \dots, A_K :



El listado en preorden de A es:

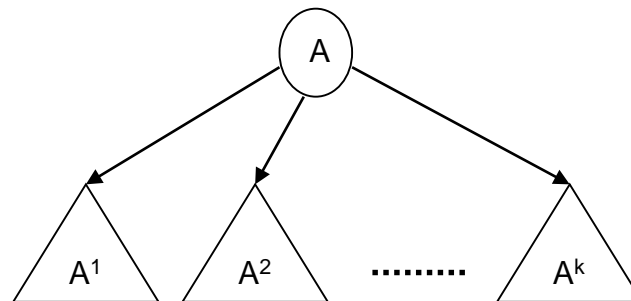
raíz del árbol A , preorden (A_1), preorden (A_2)... preorden (A_k)

Ej: A, B, F, G, C, D, H, E, I, J, K

Árbol

Inorden:

- Si el árbol A es nulo \Rightarrow la lista vacía es el listado en inorden
- Si el árbol A tiene un solo nodo \Rightarrow ese nodo constituye el listado del árbol A en inorden
- Si el árbol A tiene más de un nodo, es decir, tiene como raíz el nodo n y los subárboles A_1, A_2, \dots, A_K :



El listado en inorden de A es:

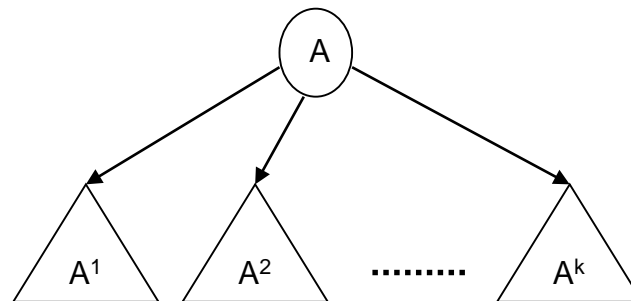
inorden (A_1), raíz del árbol A , inorden (A_2), ... inorden (A_k)

Ej: F, B, G, A, C, H, D, I, E, K, J

Árbol

Postorden:

- Si el árbol A es nulo \Rightarrow la lista vacía es el listado en postorden
- Si el árbol A tiene un solo nodo \Rightarrow ese nodo constituye el listado del árbol A en postorden
- Si el árbol A tiene más de un nodo, es decir, tiene como raíz el nodo n y los subárboles A_1, A_2, \dots, A_K :



El listado en postorden de A es:

postorden (A_1), postorden (A_2)... postorden (A_k) raíz del árbol A

Ej: F, G, B, C, H, D, I, K, J, E, A

■ Recorrido en anchura

Exploración del árbol por niveles, empezando por el nivel 0, luego el nivel 1, ... y dentro de cada nivel, listando los nodos de izquierda a derecha.

Ej: A, B, C, D, E, F, G, H, I, J, K