DEPARTAMENTO DE
INGENIERÍA
INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

# Análisis de Algoritmos y Estructura de Datos

## TDA árbol, TDA AB, TDA ABB
## Refuerzo y ejercicios

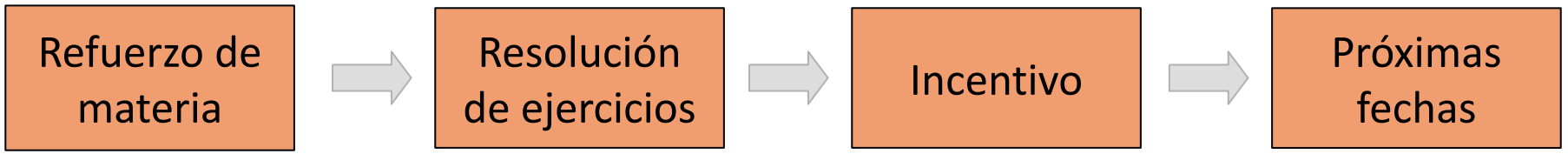
Prof. Violeta Chang C

Semestre 2 – 2023

# TDA árbol

- **Contenidos**:
  - Especificación de TDA árbol
  - Operaciones con árboles
  - Árboles binarios y de búsqueda
- **Objetivos**:
  - Explicar estructura de datos de TDA árbol
  - Comprender funcionamiento de operaciones con árboles y determinar su complejidad
  - Entender algoritmos de recorrido de árboles
  - Conocer tipos de árboles especiales: binarios y de búsqueda

# TDA árbol



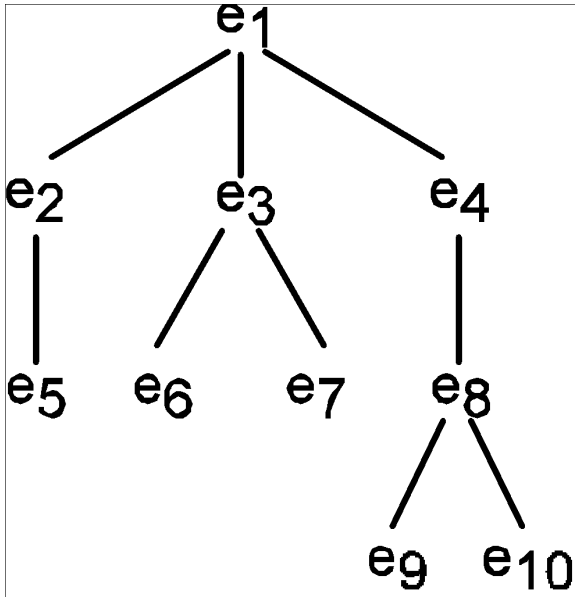

Ruta de la sesión

# Refuerzo de materia

# Especificación de TDA Árbol

- **Estructura de datos**
  - Un árbol (o tree) es una colección no lineal con componentes homogéneos y capacidad ilimitada
  - Un árbol se considera como un grafo acíclico
  - A un árbol con 0 nodos se le conoce como árbol vacío
  - Un único nodo es un árbol cuya raíz es el único nodo existente
  - Puntero externo: apunta a la raíz del árbol
  - Tipos de nodos:
    - Nodo raíz: nodo inicial del árbol
    - Nodo interno: nodo que posee antecesor y sucesor
    - Nodo hoja: nodo que no posee sucesores (hijos)

# Especificación de TDA árbol




**raíz**: $e_1$
**nodos internos**: $e_2$, $e_3$, $e_4$, $e_8$
**hojas**: $e_5$, $e_6$, $e_7$, $e_9$, $e_{10}$

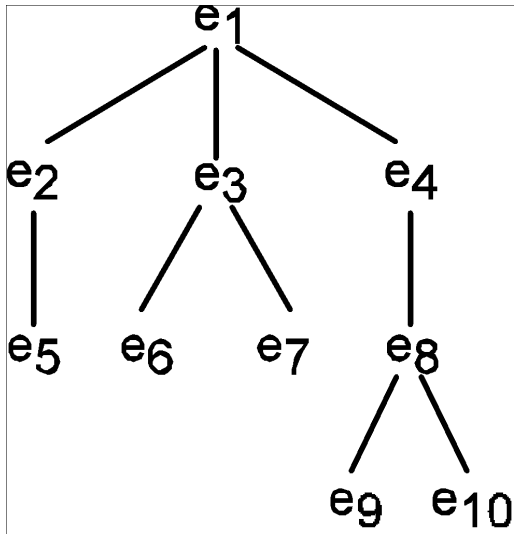
$e_1$ es el **padre** de $e_2$, $e_3$, $e_4$, entonces $e_2$, $e_3$, $e_4$ son **hermanos** y son los **hijos** de $e_1$.

El **grado** de $e_1$ es 3. El grado de $e_3$ es 2 y el grado de $e_9$ es 0.

La **aridad** del árbol es 3 → $e_1$ tiene 3 hijos y no hay otro nodo en el árbol que tenga más hijos.

La **altura** del árbol es 4 → el camino más largo desde la raíz a una hoja tiene 4 nodos.

# Especificación de TDA árbol




**Ruta**

| | |
|---|---|
| $e_1 - e_3$ | numNodos: 2 |
| $e_1 - e_3 - e_7$ | numNodos: 3 |
| $e_1 - e_4 - e_8 - e_{10}$ | numNodos: 4 |

**Altura de un nodo**

| | |
|---|---|
| $e_1$ | altura: 4 |
| $e_3$ | altura: 2 |
| $e_{10}$ | altura: 1 |

**Profundidad de un nodo (NIVEL)**

| | |
|---|---|
| $e_1$ | profundidad: 1 |
| $e_2$, $e_3$, $e_4$ | profundidad: 2 |
| $e_5$, $e_6$, $e_7$, $e_8$ | profundidad: 3 |
| $e_9$, $e_{10}$ | profundidad: 4 |

**Ancestros de un nodo**

$e_{10}$: $e_8$, $e_4$, $e_1$

**Descendientes de un nodo**

$e_4$: $e_8$, $e_9$, $e_{10}$

- **Definición recursiva para ancestro**

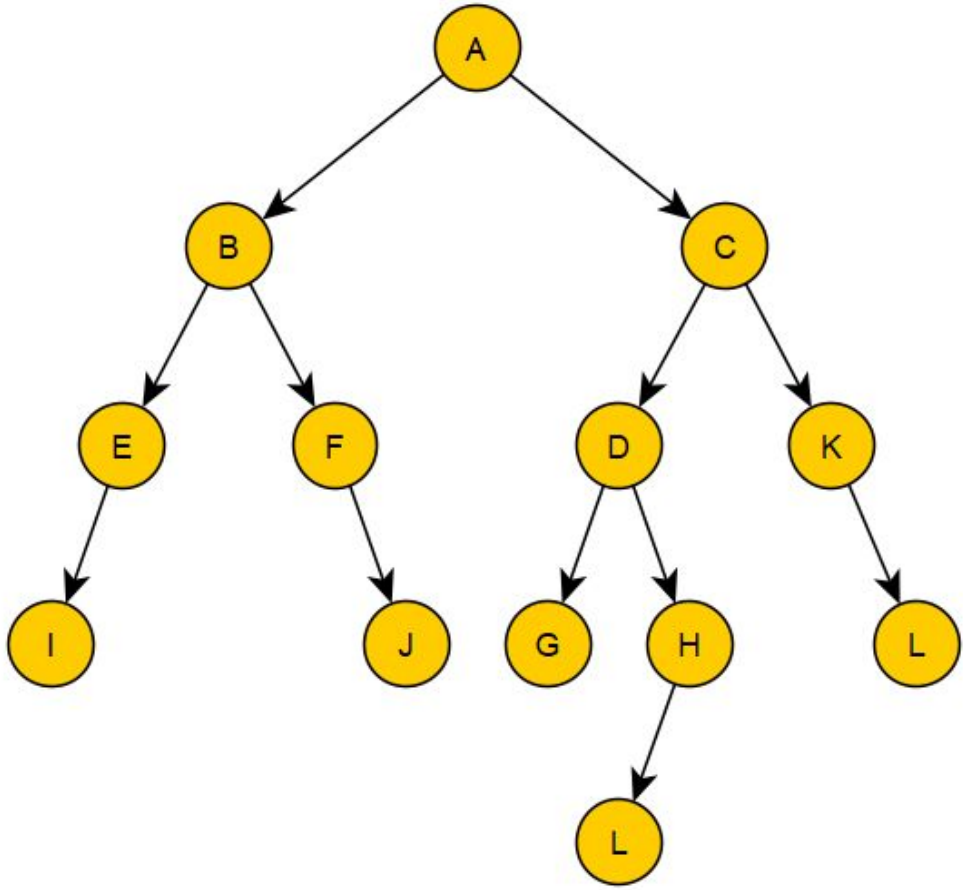
ancestros(*n*) =
- vacío, si *n* es la raíz del árbol — *caso base*
- padre de *n* + ancestros(padre de *n*), si *n* no es la raíz — *caso recursivo*

# Especificación de TDA árbol

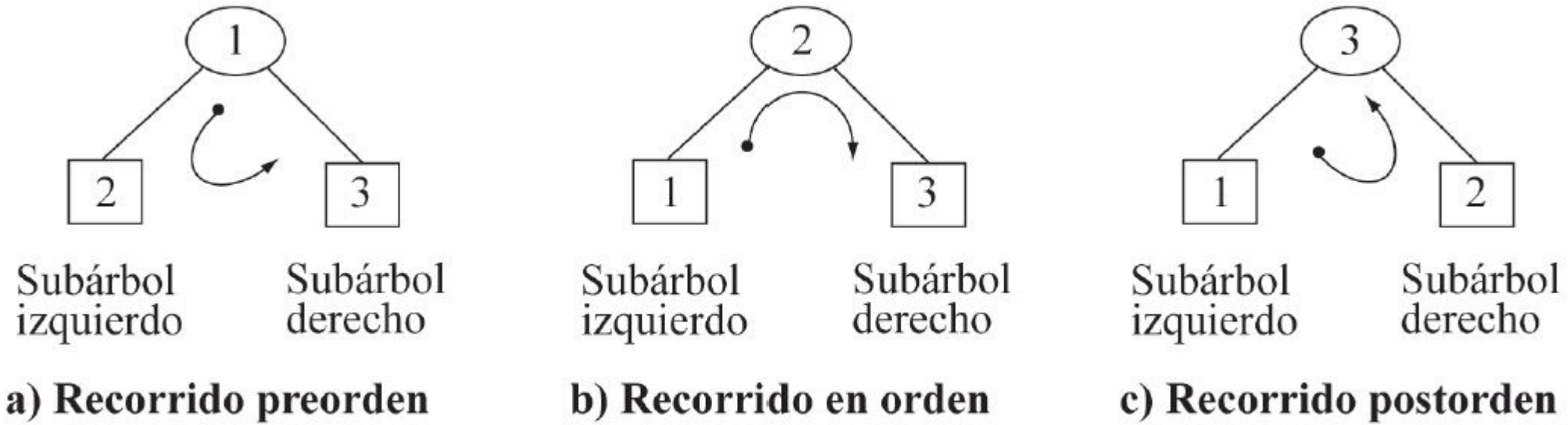
- **Operaciones:**
  - **esArbolVacío(T)**: determina si árbol T está vacío o no
  - **raíz(T)**: retorna la raíz del árbol T
  - **padre(T,nodo)**: retorna el padre de nodo. Si T es la raíz retorna nulo
  - **esHoja(T,nodo)**: indica si nodo es o no una hoja
  - **insertarNodo(T,nodo)**: inserta nodo en árbol T
  - **eliminarNodo(T,nodo)**: elimina nodo de árbol T
  - **buscarDato(T,dato)**: busca nodo con dato en árbol T
  - **recorrerArbol(T)**: muestra contenido de cada nodo de árbol T

# TAD Árbol Binario

- **Estructura de datos**
  - Similar a TAD árbol, con la restricción que cada nodo puede tener a lo más 2 hijos
- **Operaciones**
  - TAD árbol
  - **insertar(T,Padre ,tipo,x)**: O(n)
  - **eliminar(T,x)**: O(n)
  - **buscar(T,d)**: O(n)



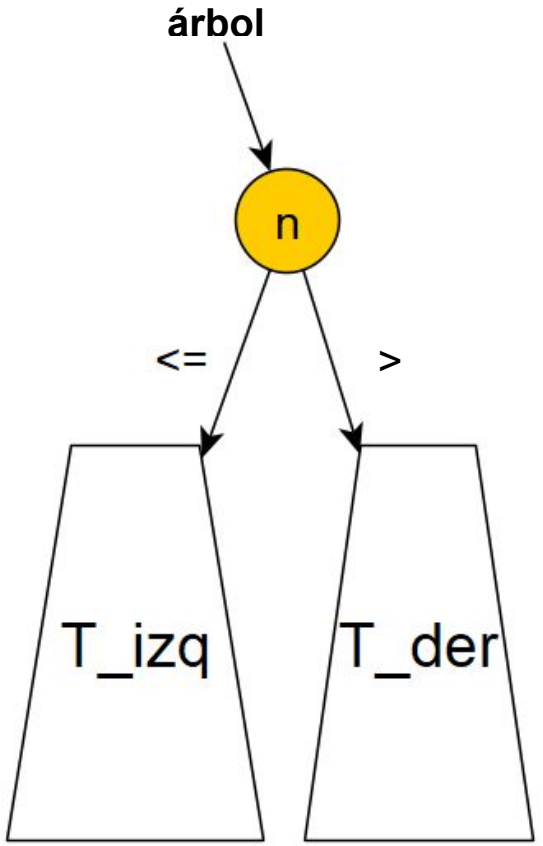
# Recorrido de árbol binario




**¿Cuándo se visita la raíz?**

# TDA árbol binario de búsqueda

- **Estructura de datos**
  - **Puntero externo**: apunta a la raíz del árbol
  - Cada nodo contiene:
    - dato
    - puntero a hijo izquierdo
    - puntero a hijo derecho
  - Un árbol es ABB ssi:
    - $n_i \leq n, \forall n_i \in T_{izq}\ y\ T_{izq}\ es\ \ ABB$
    - $n < n_i, \forall n_i \in T_{der}\ y\ T_{der}\ es\ \ ABB$




- **Operaciones**
  - TAD árbol
  - **insertar(T,x)**: O(*log* n) mejor caso
  - **eliminar(T,x)**: O(*log* n) mejor caso
  - **buscar(T,d)**: O(*log* n) mejor caso

# Ejercicios

# Ejercicio en parejas

- **<u>Ejercicio1</u>**: Para los recorridos que se muestran a continuación (todos del mismo árbol binario), dibujar el árbol correspondiente

  - Preorden: X,M,B,Y,E,P,Q,F,N,A,W,H,V,J,T,Z,D,G,R,I,C
  - Inorden: B,M,E,Y,P,X,F,Q,N,V,H,J,W,T,A,D,Z,G,I,R,C

# Ejercicio en parejas

- **Ejercicio2**: Escribir un algoritmo en seucodódigo para que dadas dos listas enlazadas, cada una con el recorrido preorden e inorden, respectivamente, devuelva un árbol binario de búsqueda (en caso de poder generarse un ABB) reconstruido a partir de ambos recorridos. En caso de que el árbol resultante no fuese un ABB, el algoritmo debería devolver un puntero nulo. Calcular y justificar la complejidad del algoritmo propuesto.

# Ejercicio asíncrono

- **Ejercicio3**: Escribir un algoritmo en seudocódigo que, dado un árbol T y un valor x, verifique si es o no un árbol binario de búsqueda (ABB). En caso de ser ABB, el algoritmo debe devolver la cantidad de hojas del árbol cuyo valor sea mayor que x. Cuál es la complejidad del algoritmo propuesto?

# Ejercicio asíncrono

- **Ejercicio4**: La operación de concatenar toma dos conjuntos S1 y S2 organizados como árboles, donde cada elemento de S1 es menor que cualquier elemento de S2, y los combina. Escribir un algoritmo de complejidad O(h) para concatenar dos árboles binarios de búsqueda, tal que h es la altura máxima de ambos árboles. Justificar cómo el algoritmo propuesto cumple con la complejidad solicitada.

# Ejercicio asíncrono

- **Ejercicio5**: La profundidad máxima de un árbol binario de búsqueda es el número de nodos en el camino desde la raíz hasta la hoja más lejana. Escribir un algoritmo de complejidad O(n) para devolver la profundidad máxima de un árbol binario de búsqueda de n nodos. Justificar cómo el algoritmo propuesto cumple con la complejidad solicitada.

# Incentivo - 15+5+5 minutos

- Dos elementos de un árbol binario de búsqueda han sido intercambiados por error. Escribir un algoritmo de complejidad O(n) para identificar (mostrar) esos dos elementos. Justificar cómo el algoritmo propuesto cumple con la complejidad solicitada.

# Incentivo - 15+5+5 minutos

Puntaje

| | | |
|---|---|---|
| I. | El algoritmo propuesto apunta a resolver el problema planteado. (SI 1 punto / NO 0 punto) | |
| II. | El algoritmo resuelve correctamente el problema planteado. (SI 1 punto / NO 0 punto) | |
| III. | El algoritmo está escrito en seudocódigo ordenado (SI 1 punto / NO 0 punto) | |
| IV. | El algoritmo está escrito en el formato establecido (SI 1 punto / NO 0 punto) | |
| V. | El algoritmo identifica entradas correctamente (SI 1 punto / NO 0 punto) | |
| VI. | El algoritmo identifica y declara salidas de manera correcta. (SI 1 punto / NO 0 punto) | |
| VII. | Calcula correctamente la complejidad (SI 1 punto / NO 0 punto) | |
| VIII. | Justifica la complejidad del algoritmo propuesto. (SI 1 punto / NO 0 punto) | |
| | PUNTOS | |

# Actividad de cierre

código: 7126 7374

# Próximas fechas...

- Resumen de la semana:
  - TDA árbol
  - TDA árbol binario
  - TDA árbol binario de búsqueda

  ~~cátedra~~ – ~~refuerzo~~ – laboratorio

- Próxima semana:
  - TDA árbol AVL

U4 - S11