Algoritmos y Estructuras de Datos I

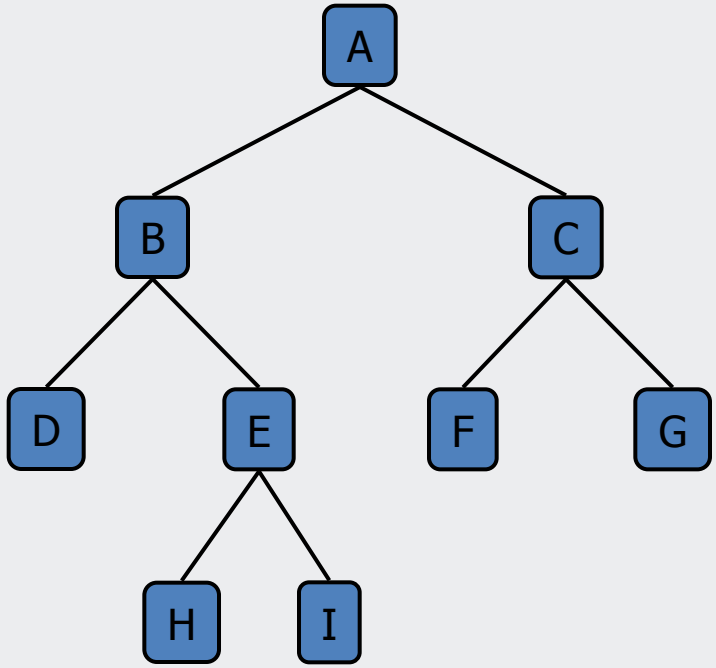
Universidad Católica del Uruguay

Arboles Binarios - I

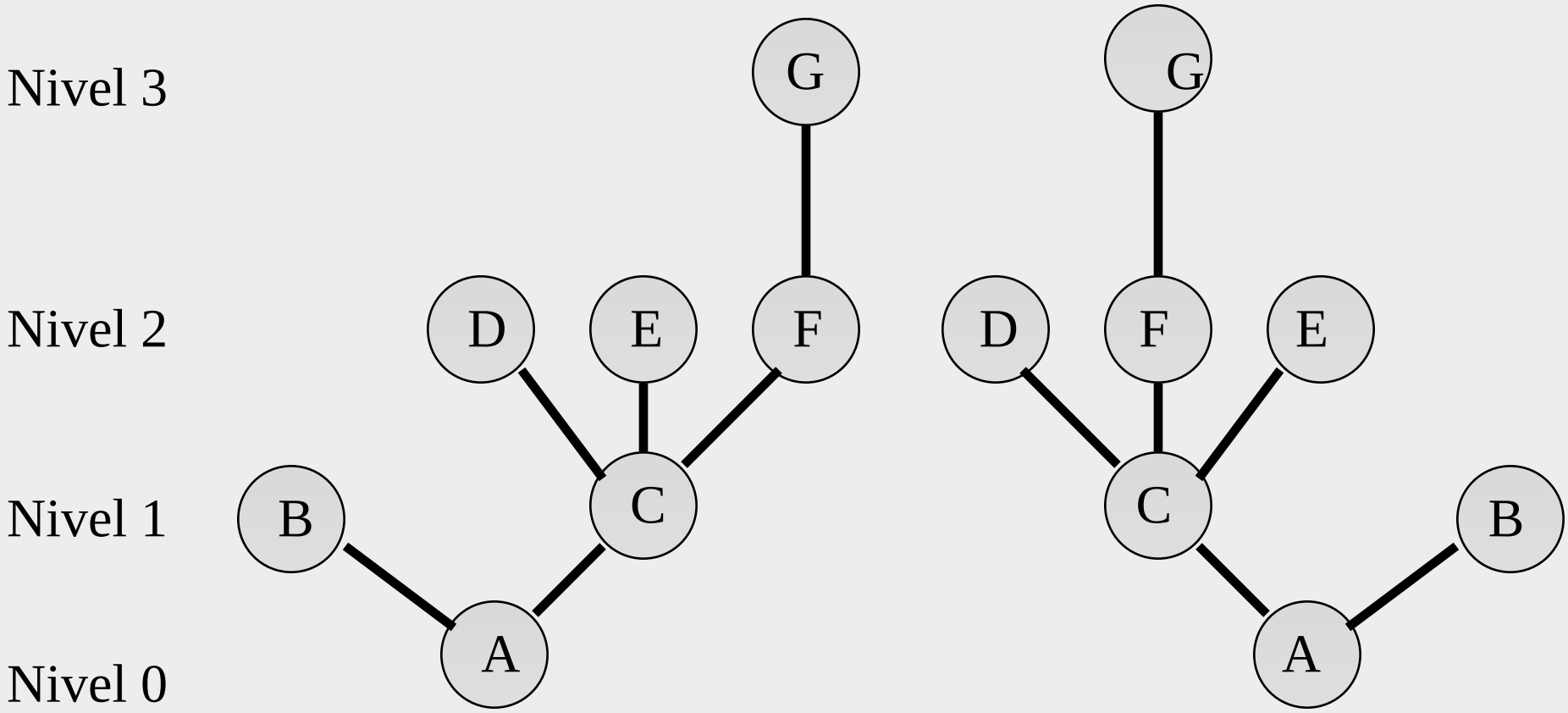
Arboles Binarios

- Un Arbol Binario es un Arbol con las siguientes propiedades:
  - Cada nodo interno tiene como máximo dos hijos (exactamente **dos** para árboles binarios completos)
  - Los hijos de un nodo son un par ordenado
- Los hijos de un nodo interno se denominan Hijo Izquierdo e Hijo Derecho
- Definiciones alternativas:
  - Un árbol con un sólo nodo, o un árbol cuya raíz tiene un par ordenado de hijos, cada uno de los cuales es a su vez un árbol binario
  - Conjunto finito de nodos, que puede estar vacío o consistir de una raíz y dos árboles binarios disjuntos, llamados subárbol izquierdo y derecho de la raíz.

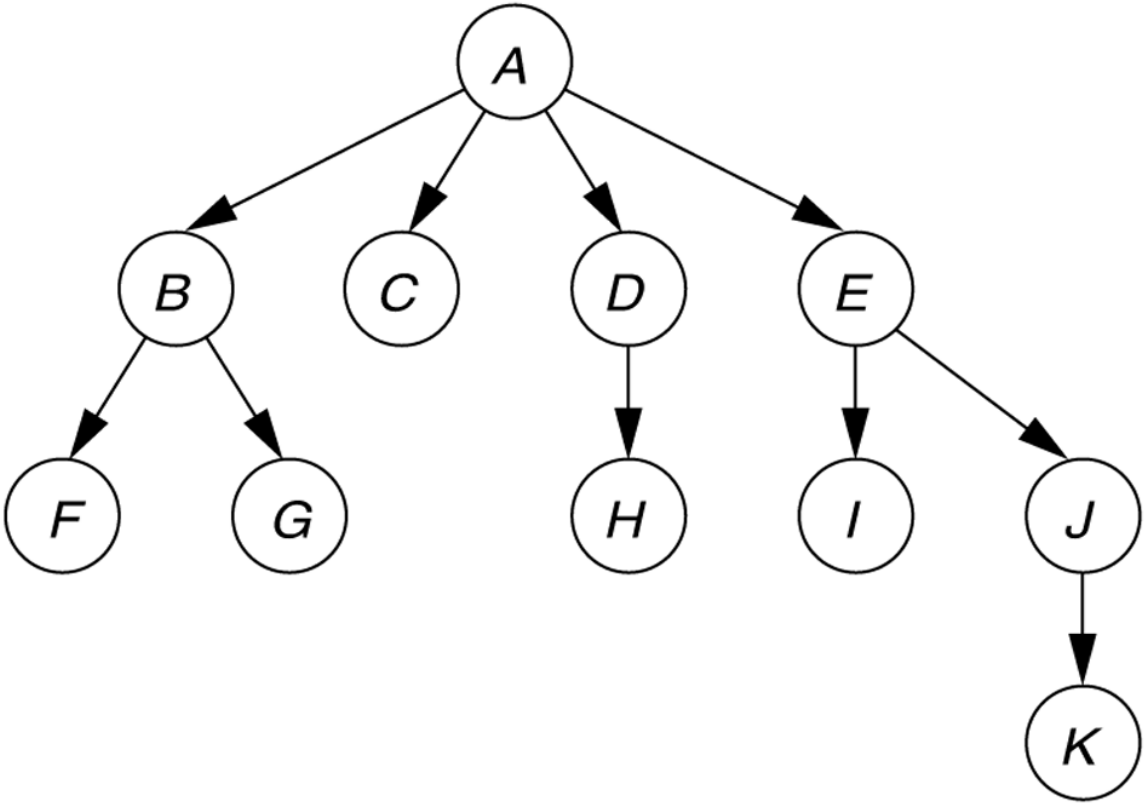
- **Applicaciones:**
  - Expresiones aritméticas
  - Procesos de decisión
  - Búsqueda



ARBOLES



Altura y profundidad o nivel



Nodo	Altura	Nivel
A	3	0
B	1	1
C	0	1
D	1	1
E	2	1
F	0	2
G	0	2
H	0	2
I	0	2
J	1	2
K	0	3

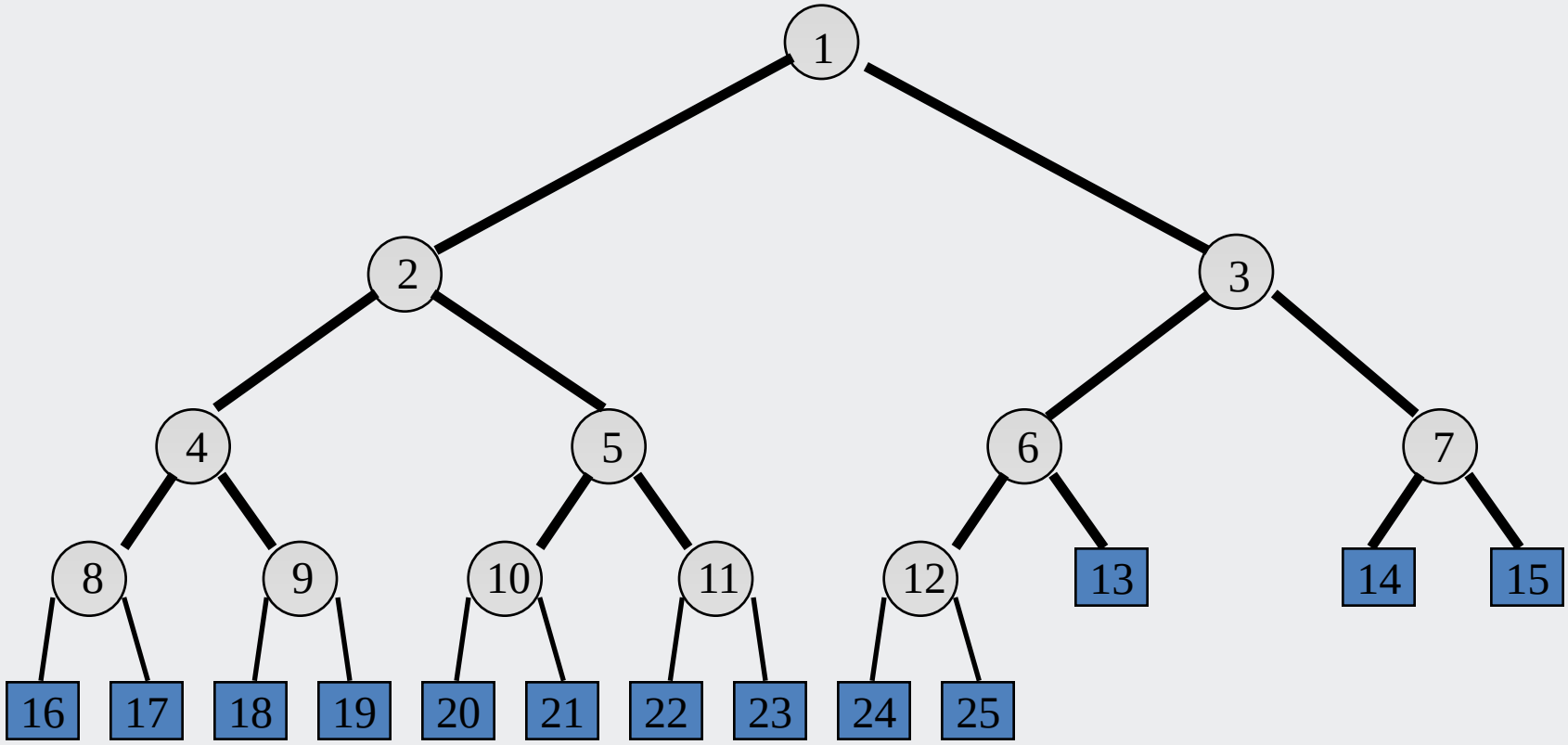
ARBOLES: Representación

- La raíz arriba, las hojas abajo.
- Se dice que cada raíz es el “padre” de las raíces de sus subárboles, los cuales son llamados “hermanos”.
- Los subárboles son llamados “hijos” de su “padre”.
- La raíz del árbol total no tiene padre.
- Ancestros y descendientes.

Lleno y completo

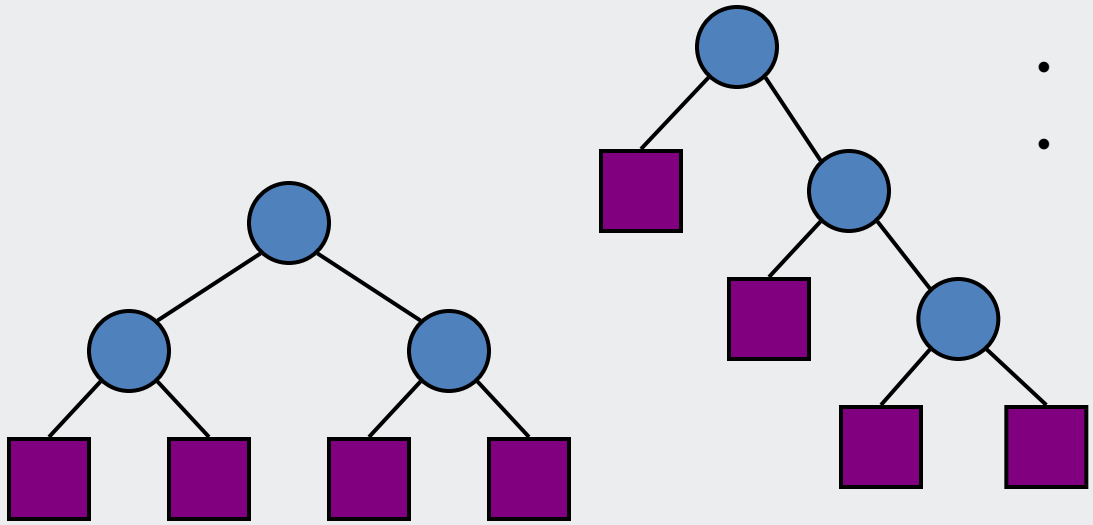
- Arbol binario Lleno
- Arbol binario completo

ARBOL BINARIO COMPLETO



Propiedades de los Arboles Binarios Completos

- **Notación**
  - n número de nodos
  - e número de nodos externos
  - i número de nodos internos
  - h altura



- **Propiedades:**
  - $e = i + 1$
  - $n = 2e - 1$
  - $h \leq i$
  - $h \leq (n - 1)/2$
  - $e \leq 2^h$
  - $h \geq \log_2 e$
  - $h \geq \log_2 (n + 1) - 1$

Arboles y recursividad

Operaciones en arboles binarios

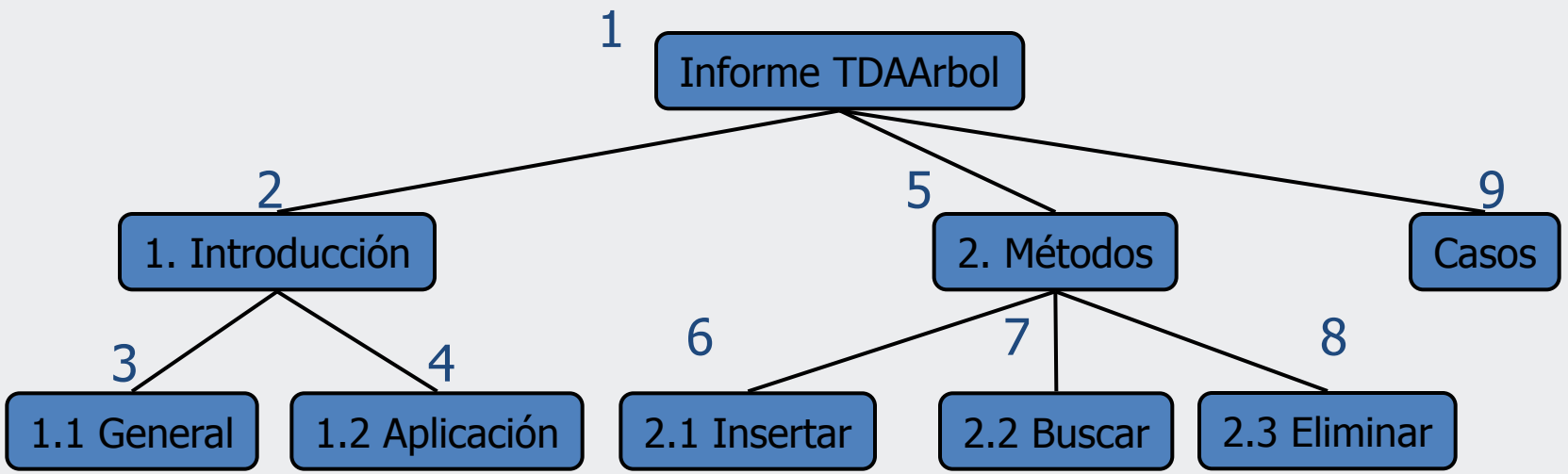
Universidad Católica del Uruguay

- Operaciones básicas.
  - Inserción, búsqueda, eliminación
- Recorridos
  - Preorden
  - Postorden
  - inorden

Recorrida en Preorden

- En un recorrido en preorden, un nodo es visitado antes que sus descendientes
- Aplicación: imprimir un documento estructurado

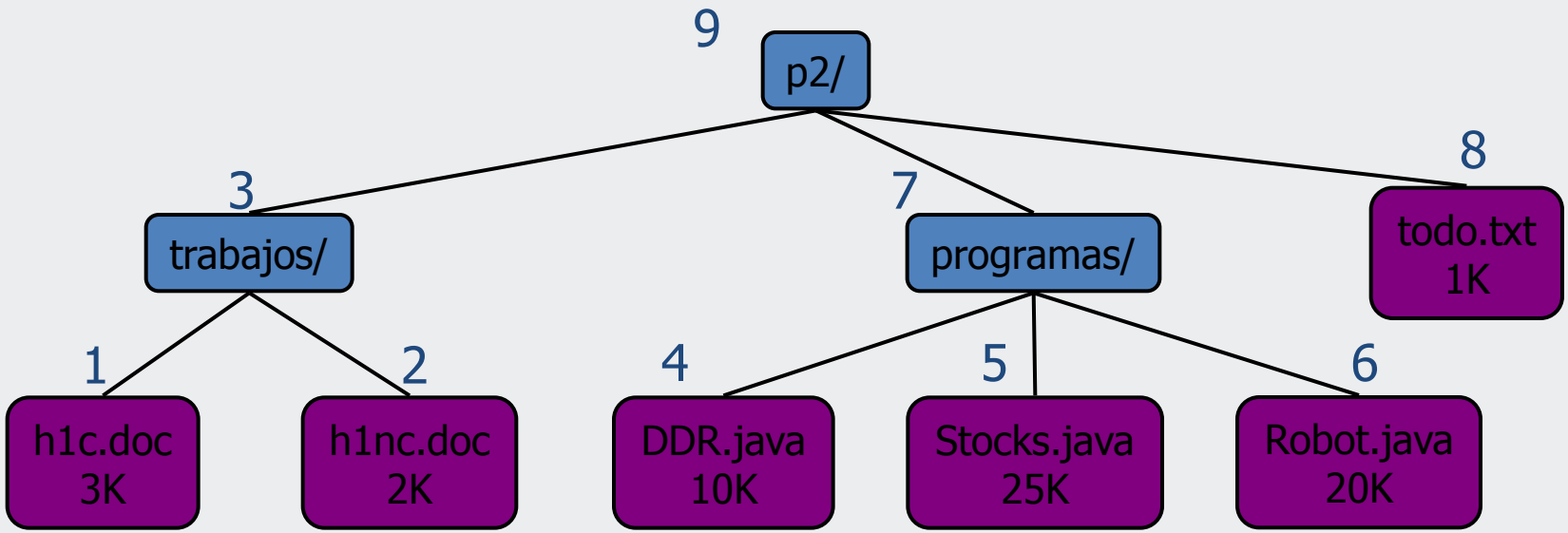
Algoritmo ***preOrden*****(*****v*****)**
  visitar**(*****v*****)**
  Para cada hijo ***w*** de ***v***
    preorden **(*****w*****)**



Recorrida en Postorden

- En un recorrido en postorden, un nodo es visitado después de sus descendientes
- Aplicación: calcular el espacio usado por archivos en un directorio y sus subdirectorios

Algoritmo *postOrden*(*v*)
  Para cada hijo *w* de *v*
    postOrden (*w*)
  visitar(*v*)



Recorrido en Inorden de un árbol binario

- En un recorrido en inorden el nodo es visitado después que su subárbol izquierdo y antes que su subárbol derecho

Algoritmo ***TNodoAB.InOrden***

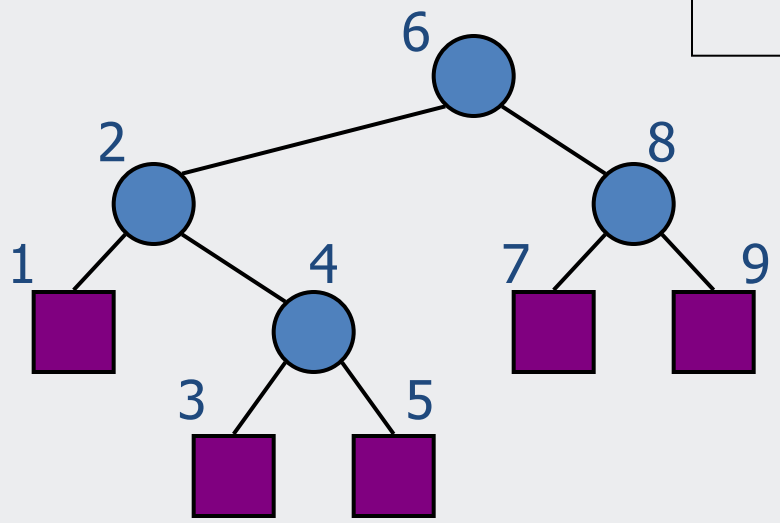
Si ***tiene HijoIzquierdo***

HijoIzquierdo.inOrden

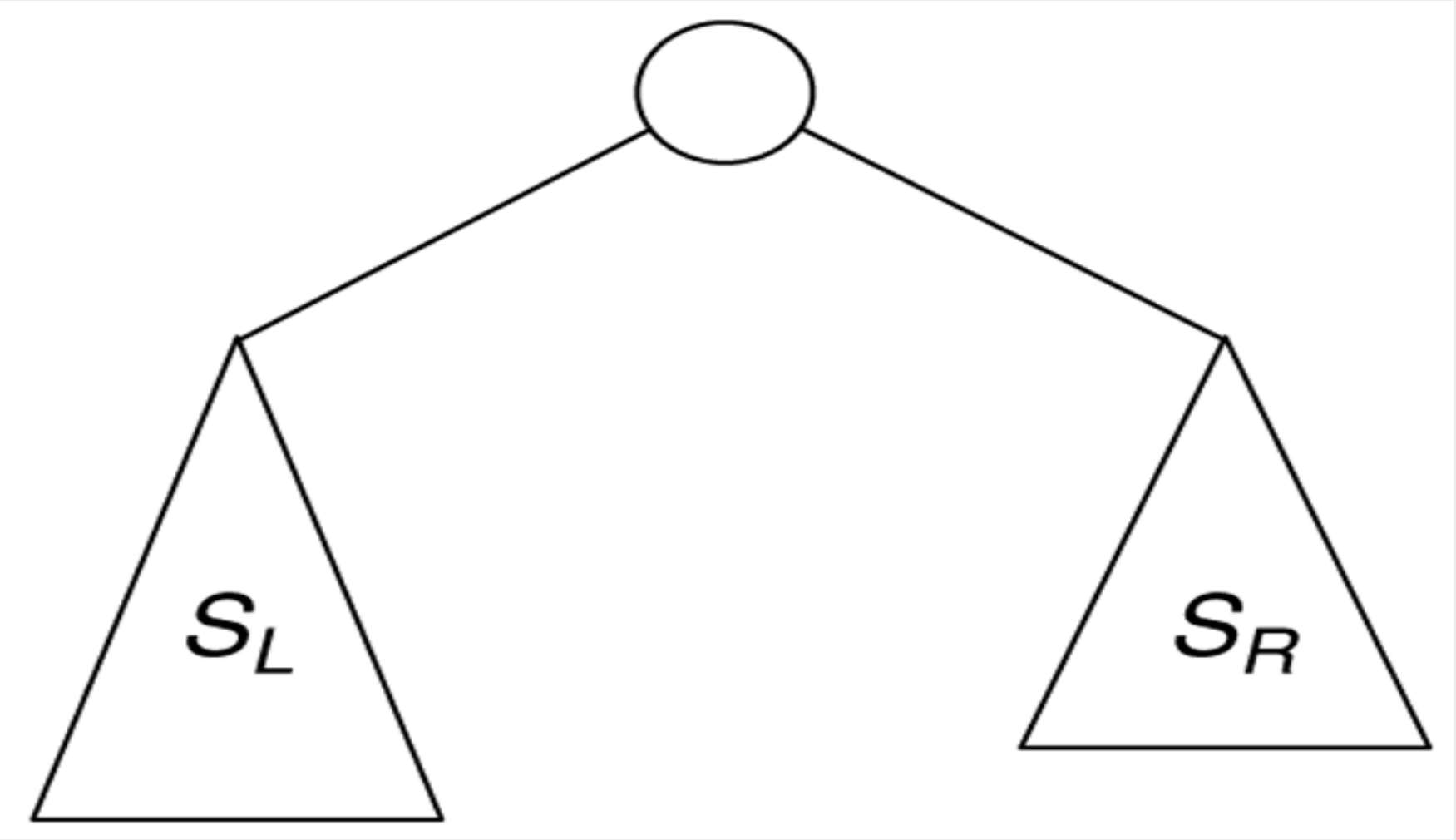
visitar(v)

Si ***tiene HijoDerecho***

HijoDerecho. inOrden

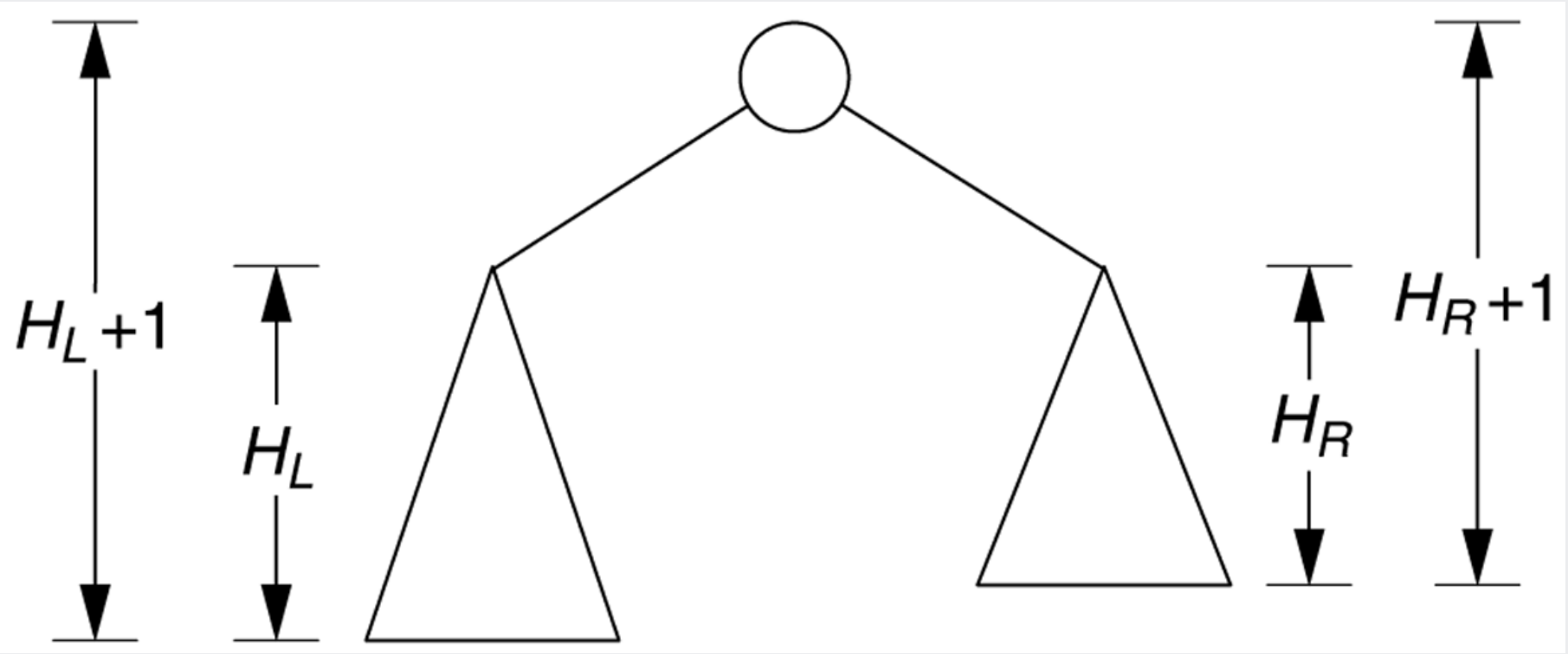


Vista recursiva usada para calcular el tamaño de un árbol $ST = SL + SR + 1$



Vista recursiva usada para calcular la altura de un nodo:

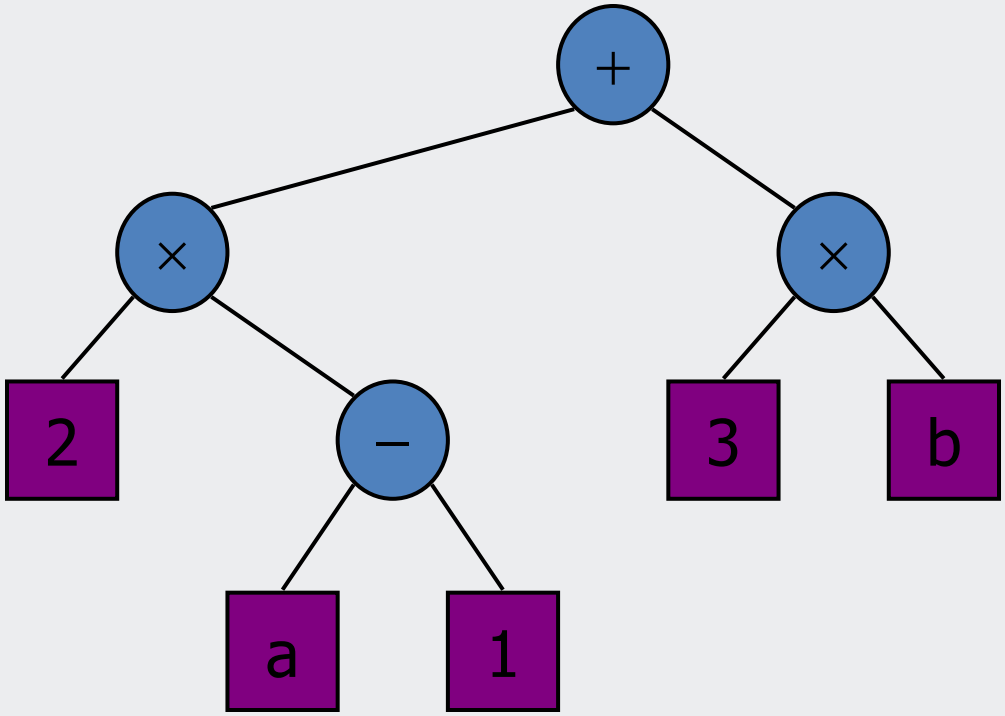
HT = Max (HL + 1, HR + 1)



Arbol de Expresión Aritmética

- Arbol binario asociado con una expresión aritmética:
  - Nodos internos: operadores
  - Nodos externos: operandos

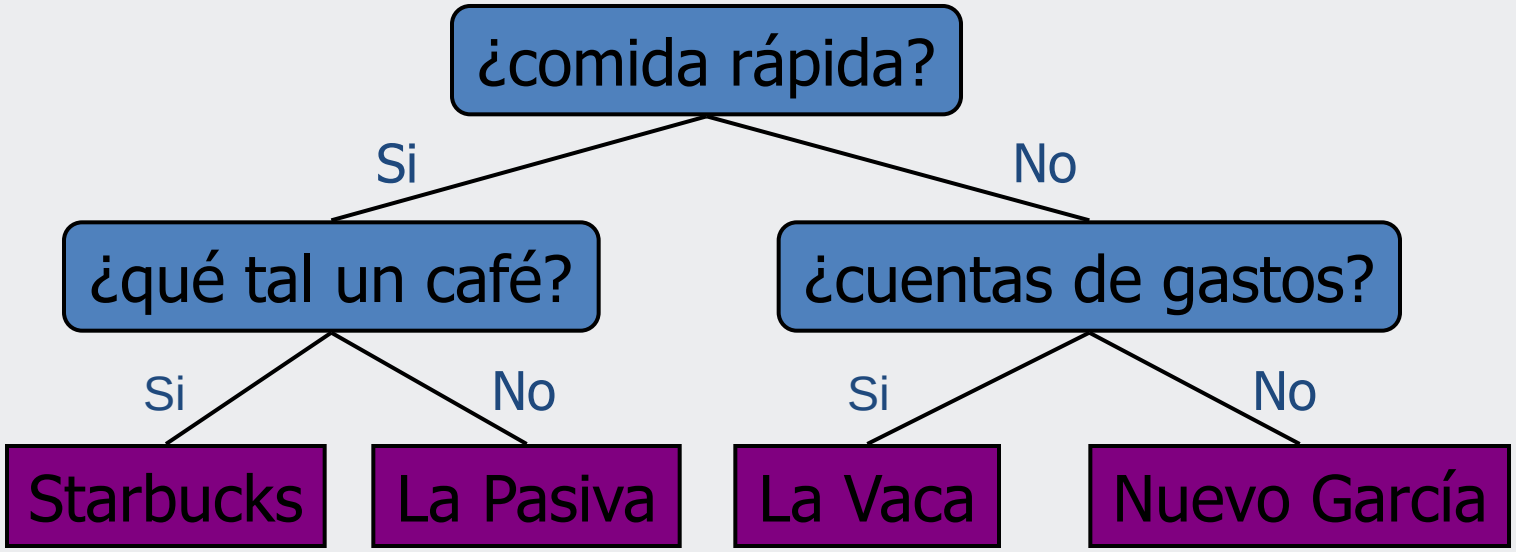
- Ejemplo: árbol de expresión aritmética para la expresión $(2 \times (a - 1) + (3 \times b))$



Aplicación de Arbol Binario: Arbol de Decisión

- Arbol binario asociado con un proceso de decisión
  - Nodos internos : preguntas con respuestas si / no
  - Nodos externos : decisiones

- Ejemplo: decisión de cena



Impresión de expresiones aritméticas

- Especialización del recorrido en inorden
  - Imprimir el operando u operador al visitar el nodo
  - Imprimir "(" antes de recorrer el subárbol izquierdo
  - Imprimir ")" después de recorrer el subárbol derecho

Algoritmo *TNodoAB.printExpression*

Si *tieneHijoIzquierdo*

imprimir("(")

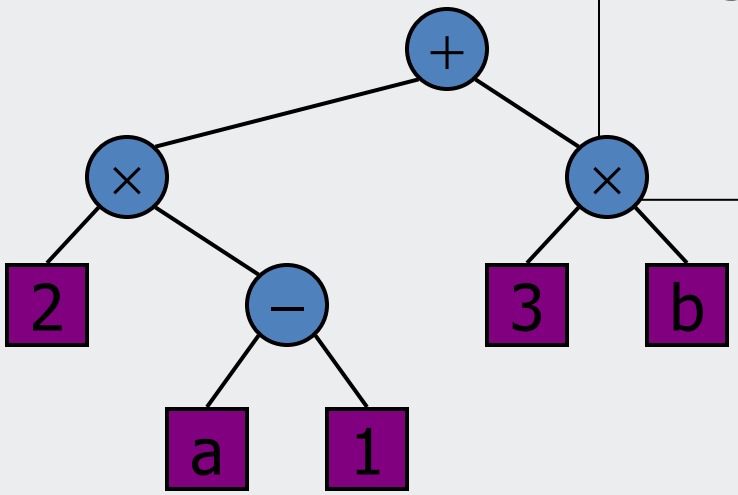
HijoIzquierdo.printExpression

imprimir

Si *tieneHijoDerecho*

HijoDerecho .printExpression

imprimir(")")



$$((2 \times (a - 1)) + (3 \times b))$$

Evaluar expresiones aritméticas

- Especialización del recorrido en postorden
  - Método recursivo que retorna el valor de un subárbol
  - Al visitar un nodo interno, combina los valores de los subarboles

Algoritmo *TNodoAB.evalExpr*

Si *esHoja*

Devolver *elemento*

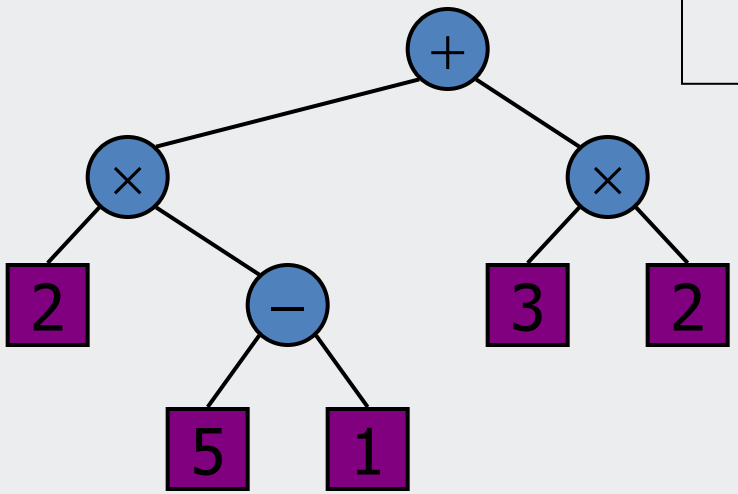
sino

x ← *HijoIzquierdo .evalExpr*

y ← *HijoDerecho.evalExpr*

◊ ← operador contenido

devolver *x* ◊ *y*



Ejercicios Arboles Binarios

- Defina árbol binario. Dé un ejemplo de organización usando esta estructura.
- Represente la siguiente expresión aritmética utilizando un árbol binario:

  a - (b * (c + d / (f + g)) + h * (i - j * (k + l)))

  y dé un algoritmo para, utilizando este árbol, evaluar la expresión cuando las variables toman valores.

- El recorrido en preorden de un cierto árbol binario produce

  ADFGHKLPQRWZ

  y el recorrido en inorden produce

  GFHKDLAWRQPZ

  Dibuje el árbol binario correspondiente.

Ejercicios Arboles Binarios

Universidad Católica del Uruguay

Sean p(x), s(x) e i(x) las posiciones del nodo x en preorden, postorden e inorden respectivamente.

•Marque en el cuadro siguiente las posiciones que pueden ser ciertas simultáneamente.

	i(n) < i(m)	*s(n) < s(m)*	*p(n) < p(m)*
n* es ancestro de *m	**10**	**100**	**1000**
n* es descendiente de *m	**10**	**100**	**1000**
n* está a la izquierda de *m	**10**	**100**	**1000**
n* está a la derecha de *m	**10**	**100**	**1000**

Ejercicios Arboles Binarios

- Escriba un algoritmo para contar las hojas de un árbol binario

- Dado un árbol binario de elementos de tipo entero, escriba un algoritmo que calcule la suma de todos los elementos.

- Escribir un algoritmo que determine la cantidad de nodos que se encuentran en un árbol binario en un nivel n

ARBOL BINARIO DE BUSQUEDA

- Se dice que un árbol binario es de búsqueda si está organizado de forma que:
  - para cada nodo t_i todas las claves del subárbol izquierdo de t_i son menores que la clave de t_i y todas las claves del árbol derecho son mayores.
- Si el árbol tiene n nodos y está balanceado, su altura será de log(n)
- Una búsqueda en este caso puede tomar log(n) comparaciones o menos.

Árbol binario de búsqueda

Universidad Católica del Uruguay

TDA ArbolBinario

Raiz : ElementoArbolBinario

Primitivas

Buscar (UnaEtiquetaqueta) : ElementoArbolBinario...

....

ArbolBinario.Buscar(UnaEtiquetaqueta)

COM

```
resultado = nulo
si Raiz <> nulo ENTONCES
    resultado = Raiz.Buscar(UnaEtiquetaqueta)
Devolver resultado
```

FIN

Arbol binario de búsqueda

TDA ElementoArbolBinario

Etiqueta : TipoEtiqueta

HijoIzquierdo,HijoDerecho : ElementoArbolBinario

Operaciones

Insertar(unElementoArbolBinario)

Buscar(UnaEtiquetaqueta)

Eliminar(UnaEtiquetaqueta)

...

TArbolBB

```
public class TArbolBB {

TElementoAB raiz;
public TArbolBB () {…} // a Implementar

public boolean esVacio() {…} // a Implementar

public boolean insertar(TElementoAB unElemento) {…}

public TElementoAB buscar (Comparable UnaEtiquetaqueta) {…}

public void preOrden() {…} // a Implementar

public void inOrden() {…}

public void postOrden() {…} // a Implementar
}
```

TElementoAB

```
public class TElementoAB {
   Comparable etiqueta;
   TElementoAB hijoIzq;
   TElementoAB hijoDer;
   Object datos;

// a implementar
public TElementoAB(Comparable UnaEtiquetaqueta, Object unosDatos) {…}

public boolean insertar(TElementoAB unElemento) {…}

public TElementoAB buscar(Comparable UnaEtiquetaqueta) {…}

public void preOrden() {…}

public void inOrden() {…}

public void postOrden() {...}
}
```

Arbol binario de busqueda

ElementoArbolBinario.Buscar(UnaEtiquetaqueta) : ElementoArbolBinario

```
COM
 RESULTADO = nulo
  SI UnaEtiquetaqueta = Etiqueta ENTONCES
       RESULTADO = THIS
   SINO
     SI UnaEtiquetaqueta < Etiqueta ENTONCES
             SI HijoIzquierdo <> nulo ENTONCES
                                RESULTADO = HijoIzquierdo.Buscar(UnaEtiquetaqueta)
             FINSI
      SINO
             SI HijoDerecho <> nulo ENTONCES
                                RESULTADO = HijoDerecho.Buscar(UnaEtiquetaqueta)
       FINSI
       FINSI
  FINSI
  devolver RESULTADO
FIN
```

TArbolBB -Buscar una Etiqueta

```
// de TArbolBB
public TElementoAB buscar (Comparable
UnaEtiquetaqueta) {
   if (esVacio()) {
        return null;
   } else {
        return raiz.buscar(UnaEtiquetaqueta);
    }
}
```

TElementoAB-Buscar Etiqueta

```
// de TElementoAB

public TElementoAB buscar(Comparable UnaEtiquetaqueta) {
   if (UnaEtiquetaqueta.compareTo(etiqueta) == 0) {
        return this;
   } else {
        if (UnaEtiquetaqueta.compareTo(etiqueta) < 0) {
            if (hijoIzq != null) {
                return hijoIzq.buscar(UnaEtiquetaqueta);
            } else {
                return null;
            }
        } else {
            if (UnaEtiquetaqueta.compareTo(etiqueta) > 0) {
                if (hijoDer != null) {
                    return hijoDer.buscar(UnaEtiquetaqueta);
                } else {
                    return null;
                }
            } else {
                return null;
            }
  }
}
}
```

Inserción en árboles binarios

```
ArbolBinario.Insertar(unElementoArbolBinario)
COM
  SI Raiz = nulo ENTONCES
        Raiz = unElementoArbolBinario
 SINO
    Raiz.Insertar(unElementoArbolBinario)
FIN
```

Inserción en árboles binarios

```
ElementoArbolBinario.Insertar(UnElementoArbolBinario);
COM
  SI Etiqueta = unElementoArbolBinario.Etiqueta ENTONCES
              SALIR // ya está en el árbol
  FINSI
  SI unElementoArbolBinario.Etiqueta < Etiqueta ENTONCES
      SI HijoIzquierdo = nulo ENTONCES
              HijoIzquierdo ←UnNodoArbolBinario
      SINO HijoIzquierdo.Insertar(UnNodoArbolBinario)
      FINSI
  SINO
      SI HijoDerecho = nulo ENTONCES
              HijoDerecho ←unElementoArbolBinario
      SINO HijoDerecho.Insertar(unElementoArbolBinario)
      FINSI
   FINSI
FIN
```

Implementación de Recorridos ABB

Universidad Católica del Uruguay

- Preorden
- Inorden
- Postorden

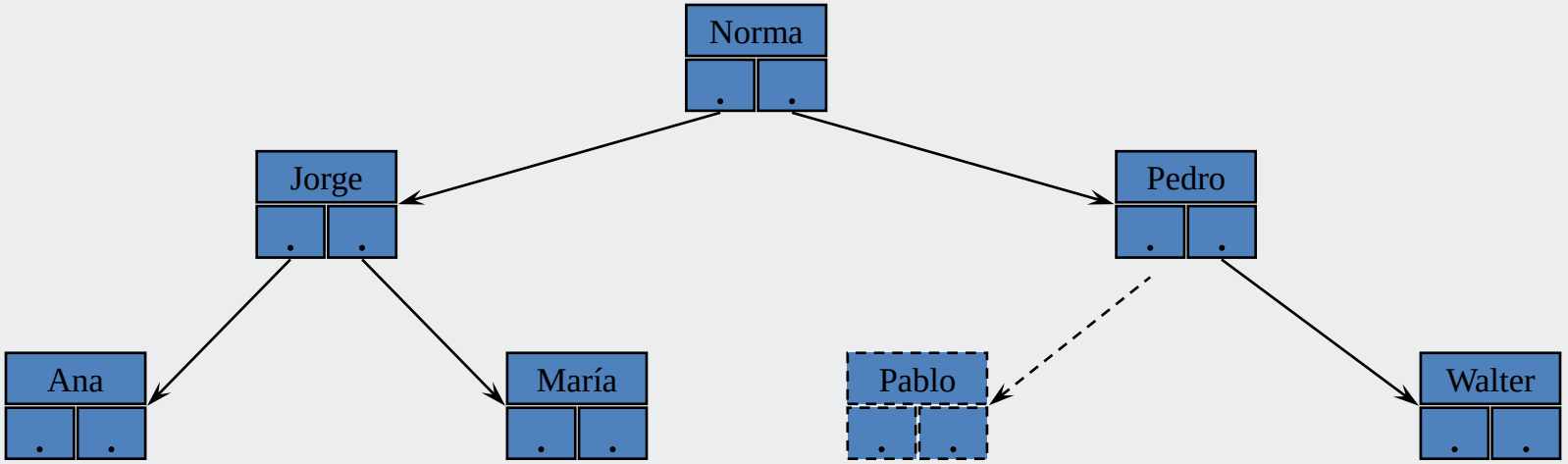
TArbolBB - Inorden

```
public String inOrden(){
    if (raiz == null) return "arbol vacio";
        else return raiz.inOrden();
}
```

TElementoAB - Inorden

```
public String inOrden() {
    String tempStr = "";
    if (hijoIzq != null) {
        tempStr = hijoIzq.inOrden();
    }
    tempStr = tempStr + imprimir();
    if (hijoDer != null) {
        tempStr = tempStr + hijoDer.inOrden();
    }
    return tempStr;
}
```

Ejemplo: Búsqueda e Inserción en árboles binarios



TDA
TPalabra
clave **string**
cuenta **integer**
izquierdo, derecho: TPalabra

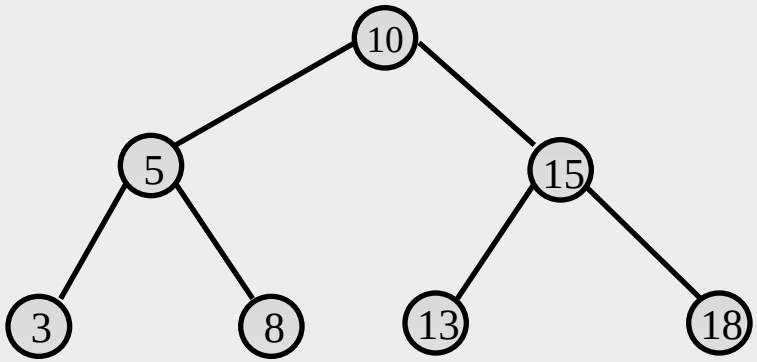
CrearArbol
AbrirArchivo
MIENTRAS NO FIN ARCHIVO HACER
LEER(archivo, palabra);
Arbol.Insertar(palabra);
end;

Búsqueda y eliminación en árboles binario de búsqueda

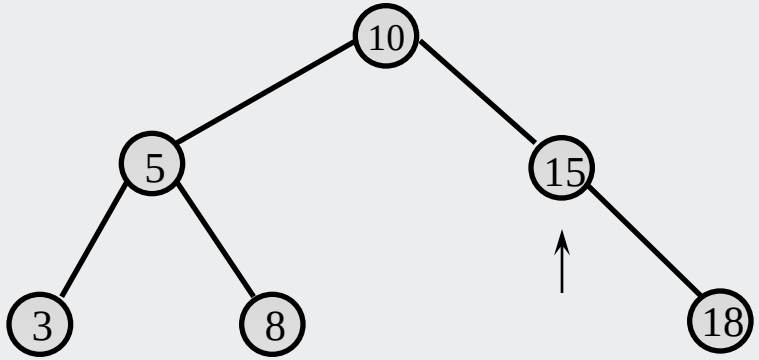
Universidad Católica del Uruguay

- Se busca el nodo con la etiqueta a eliminar.
- Si no existe un nodo con esa etiqueta, la eliminación es infructuosa.
- Si existe y es una hoja, se elimina el nodo directamente.
- Si no es una hoja, pero le falta alguno de sus dos sub árboles, la operación es sencilla, ya que basta con una reasignación de referencias.
- Si el nodo a eliminar es un nodo interno completo, deben sustituirse sus datos y etiqueta por el los del nodo de mayor etiqueta de su sub árbol izquierdo (su “inmediato anterior” en el orden lexicográfico dado).
- Luego se elimina ese elemento, que será una hoja, o le faltará el sub árbol derecho.

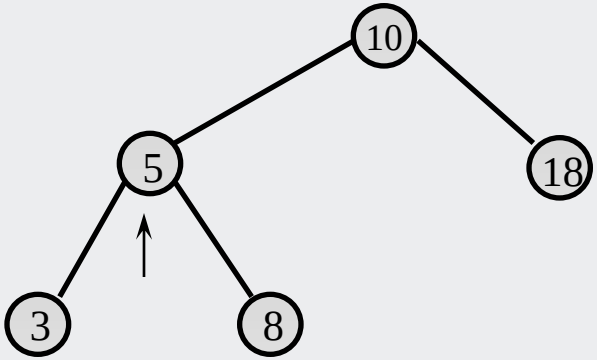
ELIMINACION EN ARBOL BINARIO DE BUSQUEDA



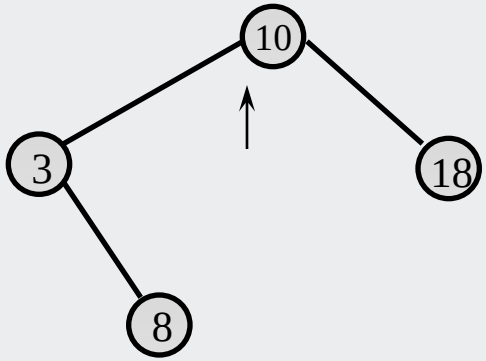
(a)



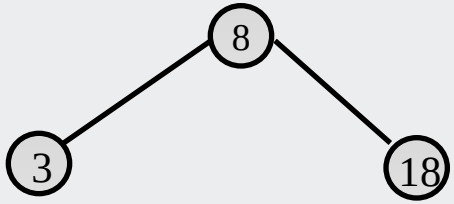
(b)



(c)

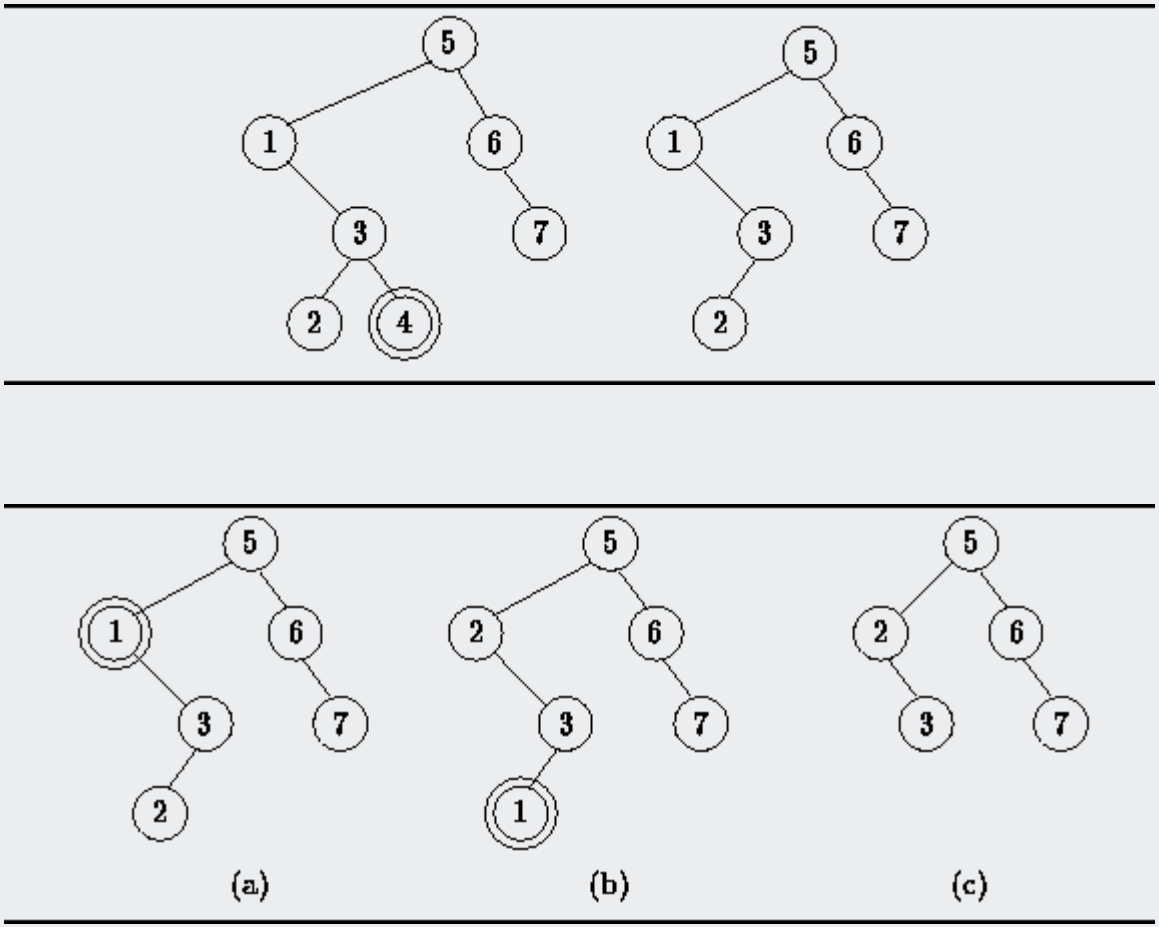


(d)

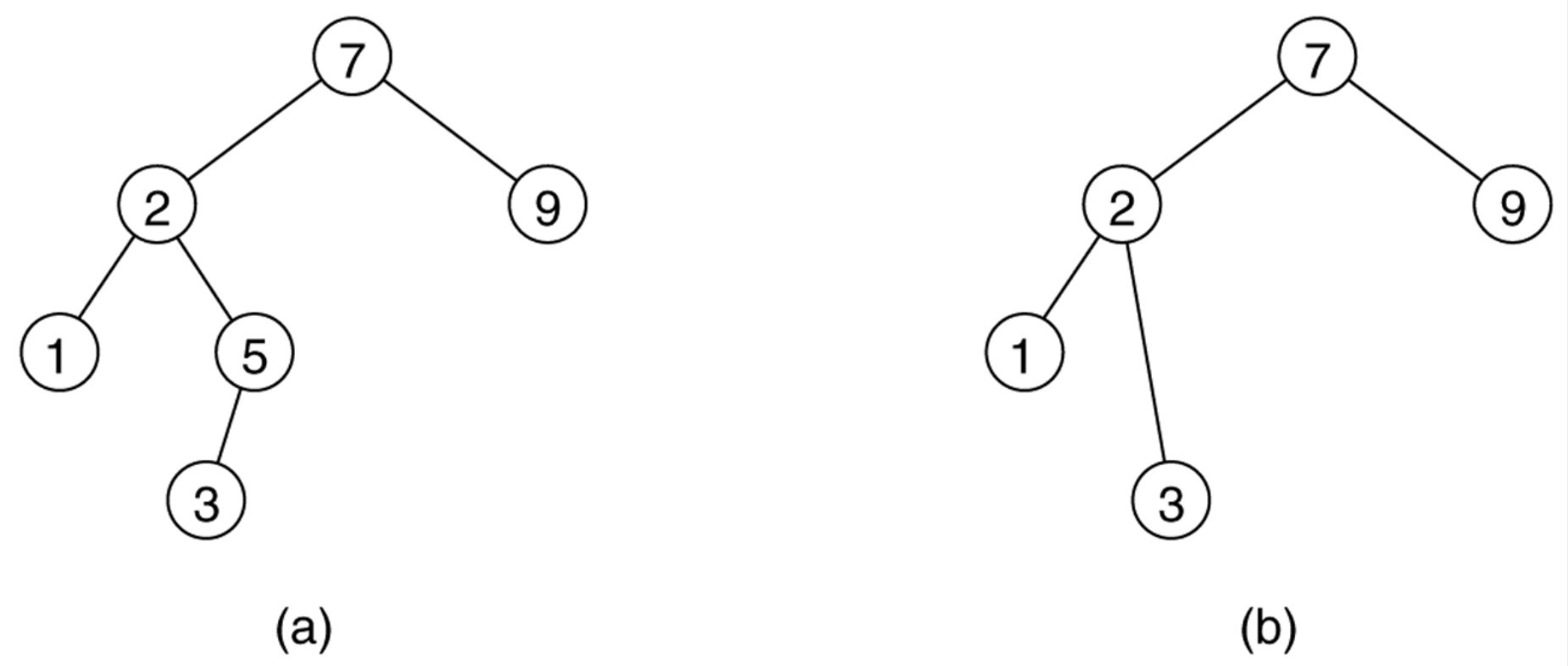


(e)

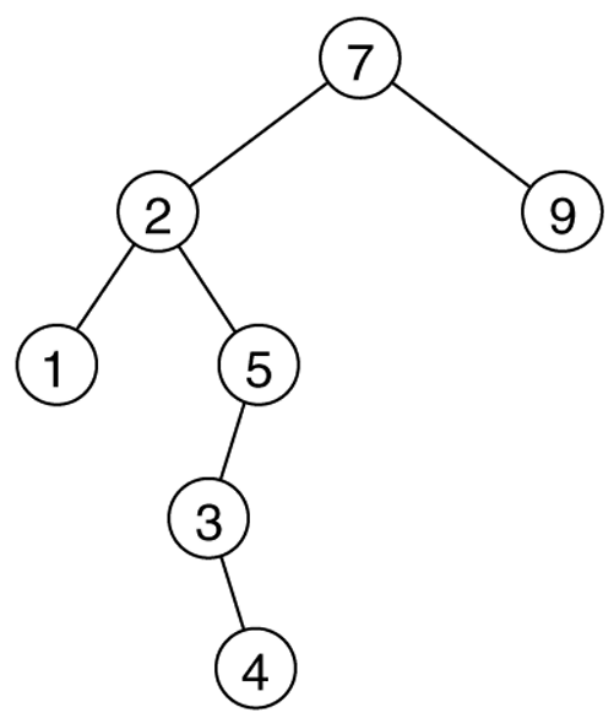
Eliminación



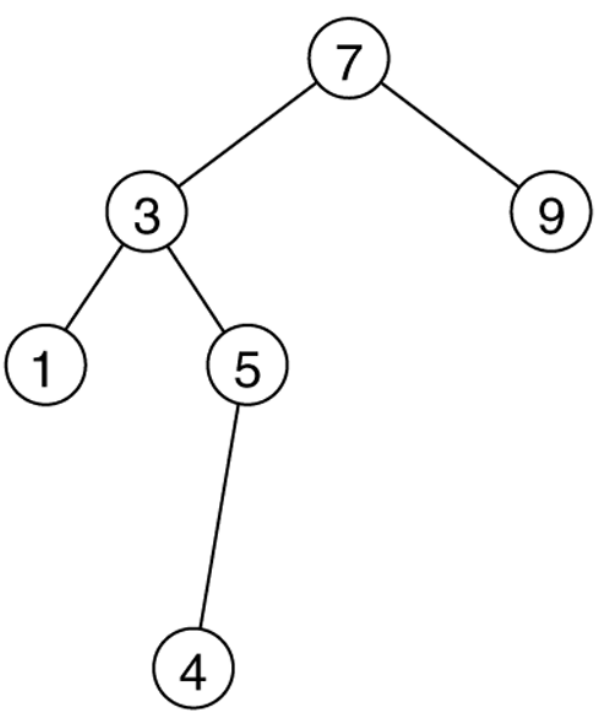
Eliminación del nodo 5 con 1 hijo: (a) antes y (b) después



Eliminación del nodo 2 con dos hijos: (a) antes y (b) después



(a)



(b)

Búsqueda y eliminación en árboles binarios de búsqueda

ArbolBinario.Eliminar(UnaEtiqueta)

```
COM
  SI Raiz <> nulo ENTONCES
        Raiz ← Raiz.Eliminar(UnaEtiqueta)
  SI NO
      mensaje “árbol vacío”
FIN
```

Búsqueda y Eliminación en árboles binarios de búsqueda

ElementoArbolBinario.Eliminar (UnaEtiquetaqueta) : de Tipo ElementoArbolBinario

```
COM
(1) Si UnaEtiqueta < etiqueta entonces                  // si esta, está en el subárbol izquierdo
    Si hijoIzq <> nulo entonces
          hijoIzq ← hijoIzq.eliminar(UnaEtiqueta)       //actualiza el hijo, con el mismo u otro valor
       Finsi
    retornar (this)                                     //al padre le devuelve el mismo hijo
    Finsi

  (2) Si UnaEtiqueta > etiqueta entonces                // si esta, está en el subárbol derecho
       Si hijoDer <> nulo entonces
        hijoDer ← hijoDer.eliminar(UnaEtiqueta)         //actualiza el hijo, con el mismo u otro valor
       Finsi
      retornar (this)                                   // al padre le devuelve el mismo hijo
     Finsi

   (3) retornar quitaElNodo                             // esta, hay que eliminarlo
                                                        // al padre le devuelve el nuevo hijo
Fin

// Cuando encuentra el nodo a eliminar llama, por claridad, al método que hace el trabajo
```

Búsqueda y Eliminación en árboles binarios de búsqueda

Universidad Católica del Uruguay

ElementoArbolBinario.quitaElNodo: de Tipo ElementoArbolBinario;

Comienzo

```
(1)  Si hijoIzq = nulo entonces                  // le falta el hijo izquierdo o es hoja
         retornar hijoDer                        // puede retornar un nulo

(2) Si hijoDer = nulo entonces                   // le falta el hijo derecho
         retornar hijoIzq

(3)  // es un nodo completo
       elHijo ← hijoIzq                          // va al subárbol izquierdo
       elPadre ← this
       mientras elHijo.hijoDer <> nulo hacer
          elPadre ← elHijo
          elHijo ← elHijo.hijoDer
       fin mientras                              // elHijo es el mas a la derecha del subárbol izquierdo

      Si elPadre <> this entonces
        elPadre.hijoDer ← elHijo.hijoIzq
        elHijo.hijoIzq ← hijoIzq
      Finsi

      elHijo.hijoDer ← hijoDer
      retornar elHijo                            // elHijo quedara en lugar de this
```

Fin

Análisis de Búsqueda e Inserción en Arbol Binario de Búsqueda

- Los algoritmos de inserción y búsqueda analizados no controlan el balanceo del árbol.
- Se desconoce a priori la forma en que el árbol ha de crecer.
- Si el árbol estuviera balanceado, para encontrar una clave se precisarían aproximadamente log n comparaciones.
- El peor caso se da para n/2 comparaciones.
- El problema es encontrar la cantidad promedio de comparaciones.
- n claves, n! árboles correspondientes a n! permutaciones de claves.

Ejercicios Arboles Binarios de Búsqueda

- ¿Qué es un árbol binario de búsqueda? Desarrolle el algoritmo de búsqueda correspondiente y evalúe su rendimiento.
- Desarrolle el algoritmo de eliminación en árbol binario de búsqueda y evalúe su rendimiento
- Muestre el resultado de insertar los elementos 3,1,4,6,9,2,5 y 7 en un árbol binario de búsqueda inicialmente vacío. Muestre después el resultado de eliminar la raíz