Estructura de Datos II

David Concha Gómez

Asignatura obligatoria

Segundo cuatrimestre

Créditos: 6

Moodle de la asignatura

Guía docente

Índice

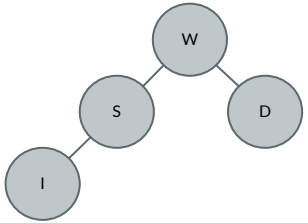
Índice

Introducción a los árboles

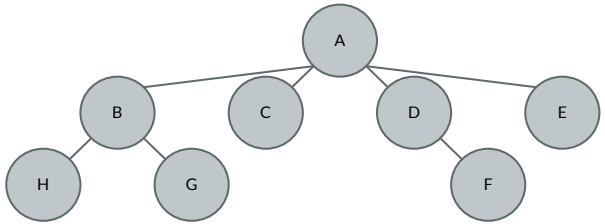
- Modelo abstracto de una estructura jerárquica.

- Un árbol está formado por nodos con una relación padre-hijo

- Aplicaciones
  - Flujos en organizaciones
  - Sistemas de ficheros
  - Entornos de programación
  - etc.

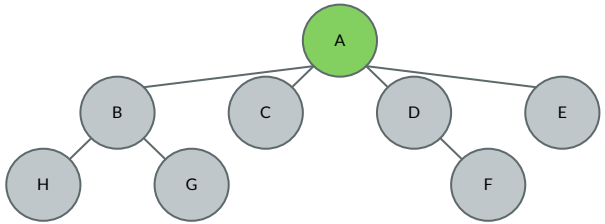


Introducción a los árboles



Raíz: ...

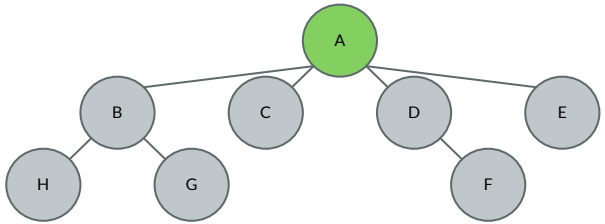
Introducción a los árboles



Raíz

Raíz: Nodo sin padre.

Introducción a los árboles

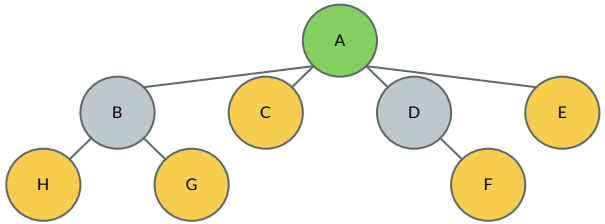


Raíz

Raíz: Nodo sin padre.

Hoja: ...

Introducción a los árboles



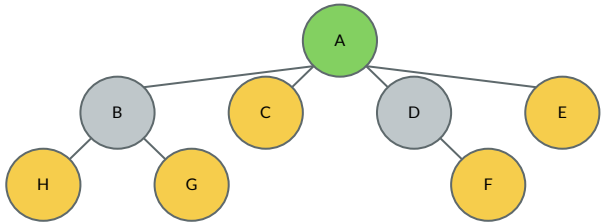
Raíz

Hoja

Raíz: Nodo sin padre.

Hoja: Nodo sin hijos.

Introducción a los árboles



Raíz

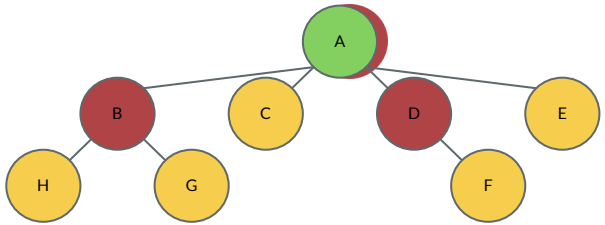
Hoja

Raíz: Nodo sin padre.

Hoja: Nodo sin hijos.

Interno: ...

Introducción a los árboles



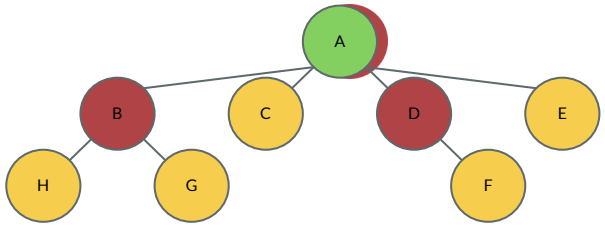
Raíz
Hoja
Interno

Raíz: Nodo sin padre.

Hoja: Nodo sin hijos.

Interno: Nodo con, al menos, un hijo.

Introducción a los árboles



Raíz
Hoja
Interno

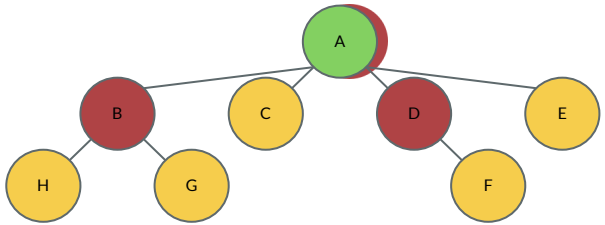
Raíz: Nodo sin padre.

Hoja: Nodo sin hijos.

Interno: Nodo con, al menos, un hijo.

Ancestros: ...

Introducción a los árboles



Raíz
Hoja
Interno

Ancestros de F: D y A

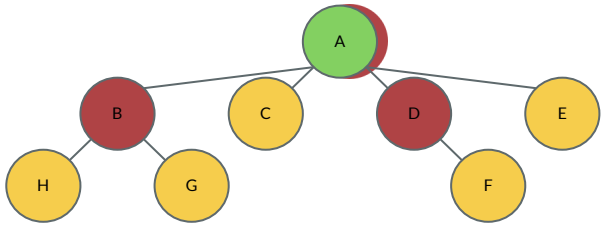
Raíz: Nodo sin padre.

Hoja: Nodo sin hijos.

Interno: Nodo con, al menos, un hijo.

Ancestros: Padres, Abuelos, etc.

Introducción a los árboles



Raíz
Hoja
Interno

Raíz: Nodo sin padre.

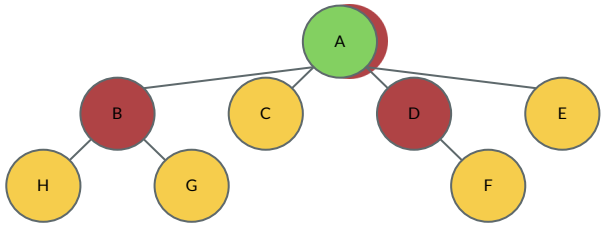
Hoja: Nodo sin hijos.

Interno: Nodo con, al menos, un hijo.

Ancestros: Padres, Abuelos, etc.

Descendiente: ...

Introducción a los árboles



Raíz
Hoja
Interno

Descendiente de B: H y G

Raíz: Nodo sin padre.

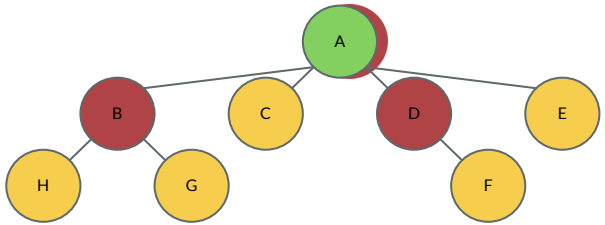
Hoja: Nodo sin hijos.

Interno: Nodo con, al menos, un hijo.

Ancestros: Padres, Abuelos, etc.

Descendiente: hijo, nieto, etc.

Introducción a los árboles



Raíz
Hoja
Interno

Raíz: Nodo sin padre.

Hoja: Nodo sin hijos.

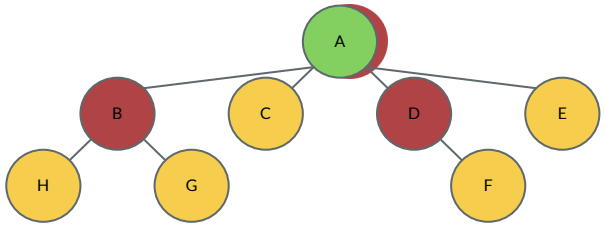
Interno: Nodo con, al menos, un hijo.

Ancestros: Padres, Abuelos, etc.

Descendiente: hijo, nieto, etc.

Profundidad: ...

Introducción a los árboles



- Raíz
- Hoja
- Interno

Profundidad de G: 2

Raíz: Nodo sin padre.

Hoja: Nodo sin hijos.

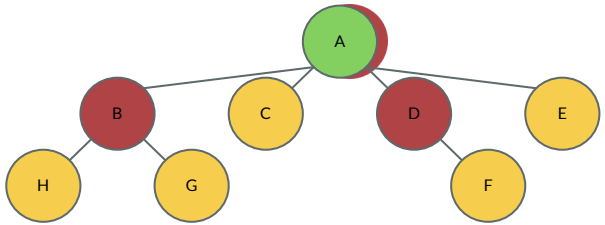
Interno: Nodo con, al menos, un hijo.

Ancestros: Padres, Abuelos, etc.

Descendiente: hijo, nieto, etc.

Profundidad: nº ancestros (nivel - 1).

Introducción a los árboles



Raíz
Hoja
Interno

Raíz: Nodo sin padre.

Hoja: Nodo sin hijos.

Interno: Nodo con, al menos, un hijo.

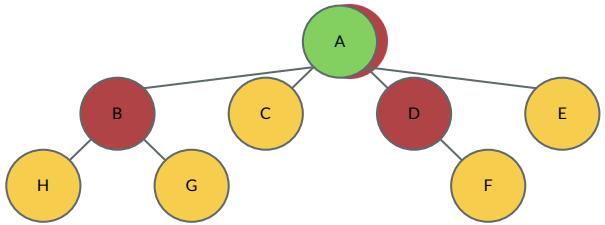
Ancestros: Padres, Abuelos, etc.

Descendiente: hijo, nieto, etc.

Profundidad: nº ancestros (nivel - 1).

Altura: ...

Introducción a los árboles



Raíz
Hoja
Interno

Altura: 3

Raíz: Nodo sin padre.

Hoja: Nodo sin hijos.

Interno: Nodo con, al menos, un hijo.

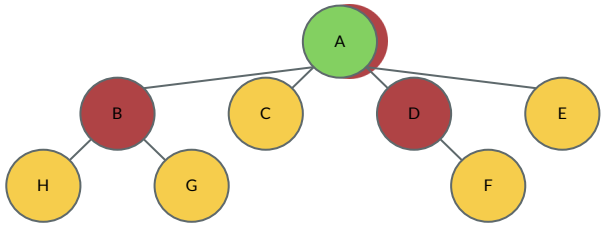
Ancestros: Padres, Abuelos, etc.

Descendiente: hijo, nieto, etc.

Profundidad: nº ancestros (nivel - 1).

Altura: nivel máximo del árbol.

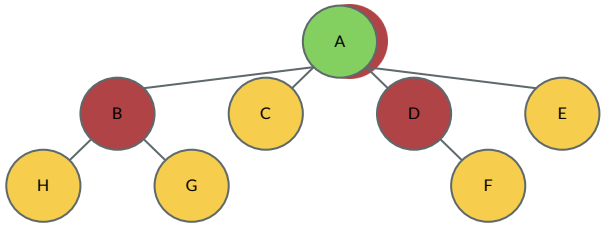
Introducción a los árboles



- Raíz
- Hoja
- Interno

Hermanos: ...

Introducción a los árboles

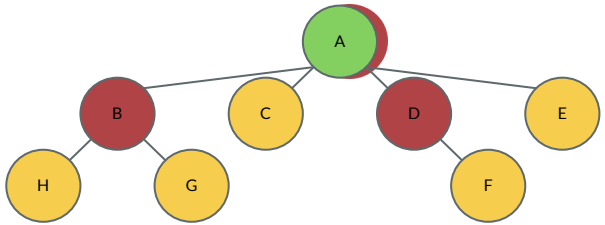


- Raíz
- Hoja
- Interno

H y G son Hermanos

Hermanos: Nodos con el mismo padre.

Introducción a los árboles

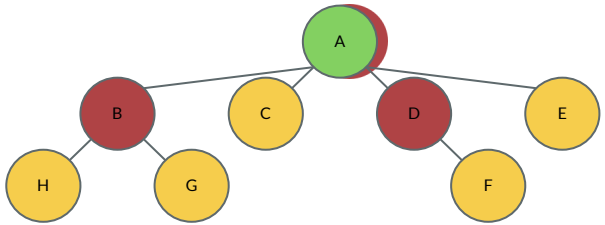


- Raíz
- Hoja
- Interno

Hermanos: Nodos con el mismo padre.

Grado de un Nodo: ...

Introducción a los árboles



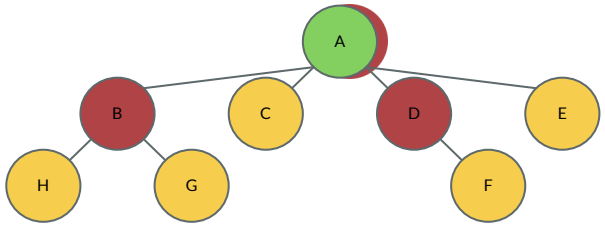
- Raíz
- Hoja
- Interno

El grado de B es 2

Hermanos: Nodos con el mismo padre.

Grado de un Nodo: número de hijos.

Introducción a los árboles



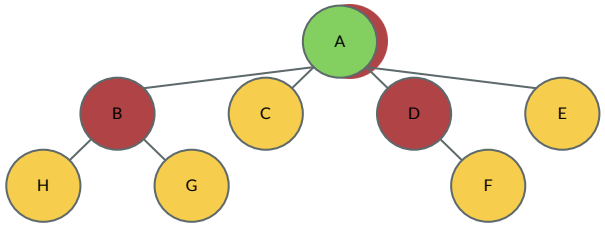
- Raíz
- Hoja
- Interno

Hermanos: Nodos con el mismo padre.

Grado de un Nodo: número de hijos.

Grado de un Árbol: ...

Introducción a los árboles



- Raíz
- Hoja
- Interno

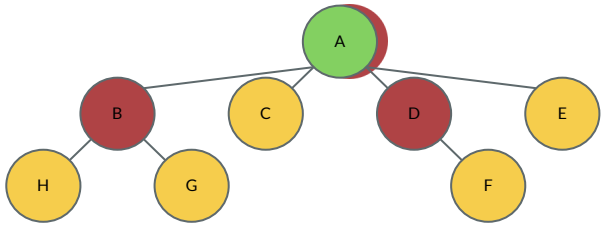
El grado del árbol es 4

Hermanos: Nodos con el mismo padre.

Grado de un Nodo: número de hijos.

Grado de un Árbol: mayor grado.

Introducción a los árboles



- Raíz
- Hoja
- Interno

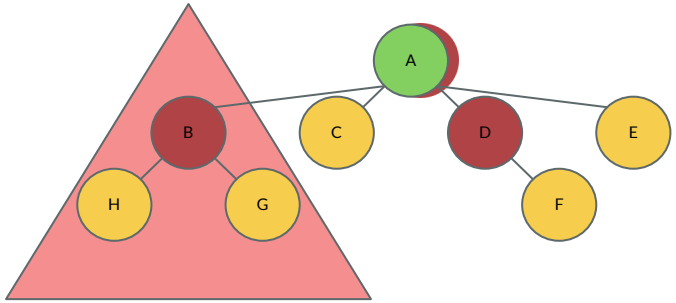
Hermanos: Nodos con el mismo padre.

Grado de un Nodo: número de hijos.

Grado de un Árbol: mayor grado.

Sub-árbol: ...

Introducción a los árboles



- Raíz
- Hoja
- Interno

Hermanos: Nodos con el mismo padre.

Grado de un Nodo: número de hijos.

Grado de un Árbol: mayor grado.

Sub-árbol: un nodo y sus descendientes.

Índice

Operaciones sobre árboles

- Básicas
  - isEmpty: devuelve si es vacío.
- Construcción
  - Add: diferenciar entre raíz y nodo.
  - Cut: devuelve un sub-árbol.
  - Attach: añade un sub-árbol.
- Acceso
- Consulta

```
void isEmpty(Tree t);

Iterator add(T element, parent=nullptr);
// insertar raiz
// r = add(3);
// insertar nodo no raiz
// h1 = add(4, *r);

Tree cut(Node node);

Iterator attach(Node node, Tree, t);
```

Operaciones sobre árboles

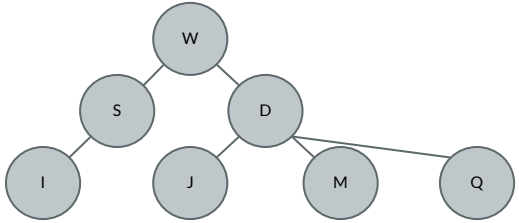
- Básicas
- Construcción
- Acceso
  - Root: devuelve la raíz.
  - Padre: devuelve el nodo padre.
  - Children: devuelve un iterador a los hijos.
- Consulta
  - isRoot: devuelve si es la raíz.
  - isLeaf: devuelve si es hoja.
  - isInternal: devuelve si no es hoja.

```
Iterator Root();

Iterator Parent(Node node);

Iterator Children(Node node);

bool isRoot(Node node);

bool isLeaf(Node node);

bool isInternal(Node node);
```

Índice

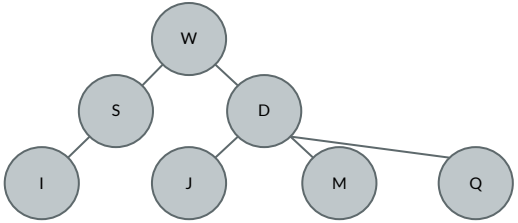
Árboles N-arios

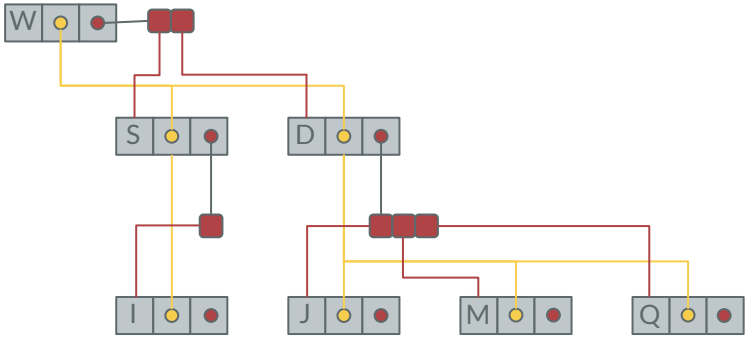
- Árboles con N hijos.



LinkedTree

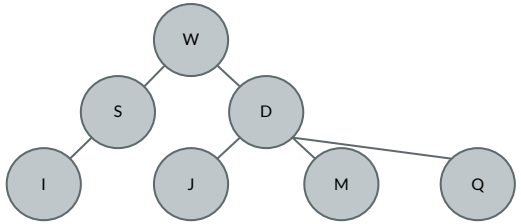
- Cada nodo contiene:
  - El elemento
  - Referencia al padre
  - Referencia a colección de hijos

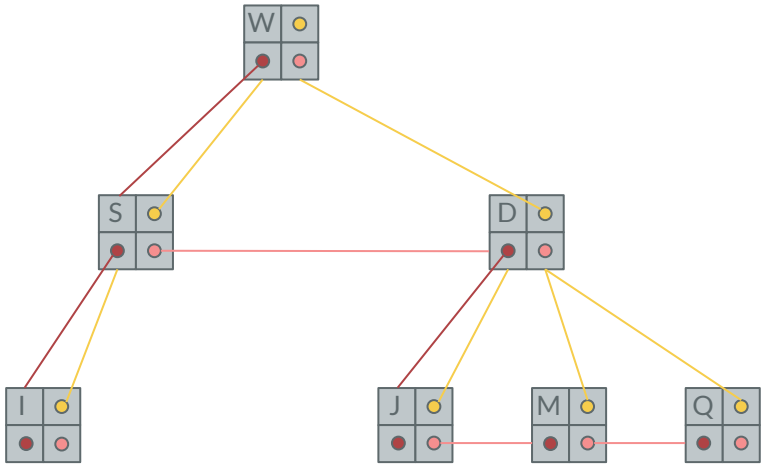




LCRSTree

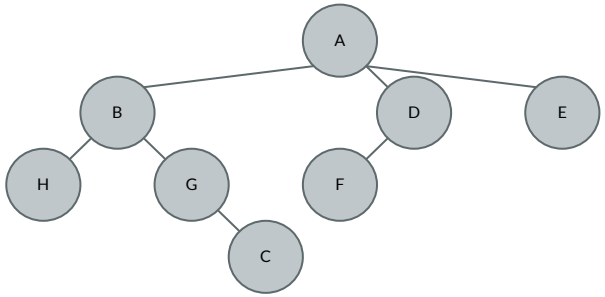
- Cada nodo contiene:
  - El elemento
  - Referencia al padre
  - Referencia al primer hijo
  - Referencia al siguiente hermano





Índice

Recorridos

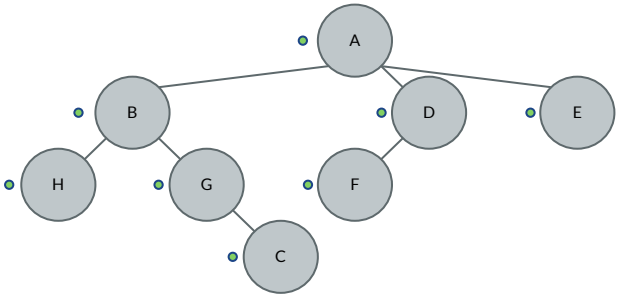


- Preorden
- Postorden

void preorder(Tree t)

void postorder(Tree t)

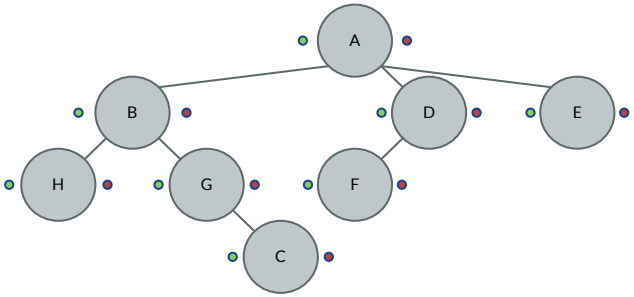
Recorridos



- Preorden
- Postorden

```
void preorder(Tree t)

void postorder(Tree t)
```

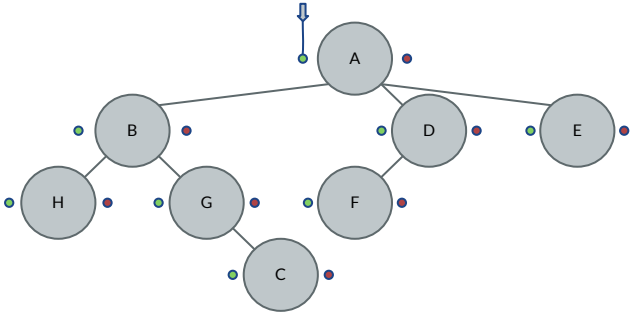
Recorridos



- Preorden
- Postorden

```
void preorder(Tree t)

void postorder(Tree t)
```

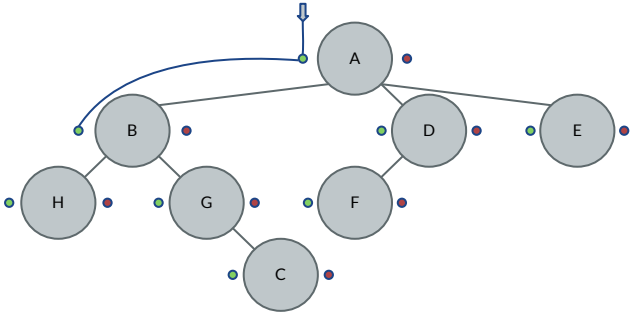
Recorridos



- Preorden {A, }
- Postorden { }

```
void preorder(Tree t)

void postorder(Tree t)
```

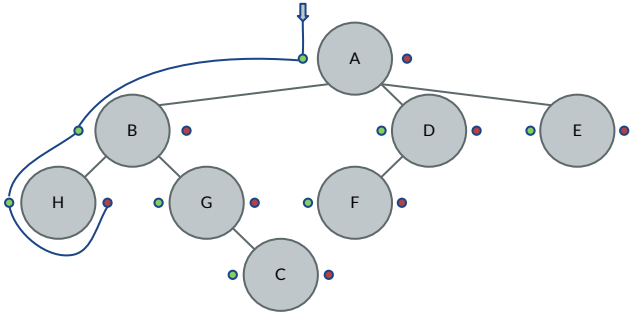
Recorridos



- Preorden {A, B, }
- Postorden { }

```
void preorder(Tree t)

void postorder(Tree t)
```

Recorridos

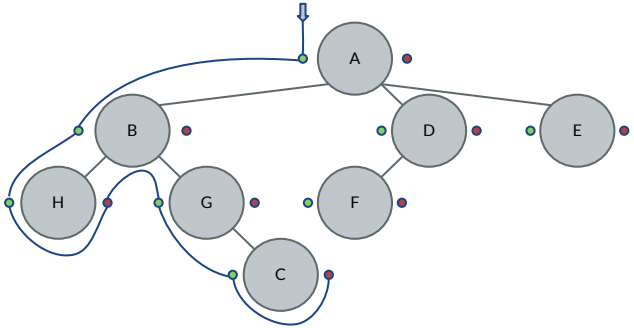


- Preorden {A, B, H, }
- Postorden {H, }

void preorder(Tree t)

void postorder(Tree t)

Recorridos

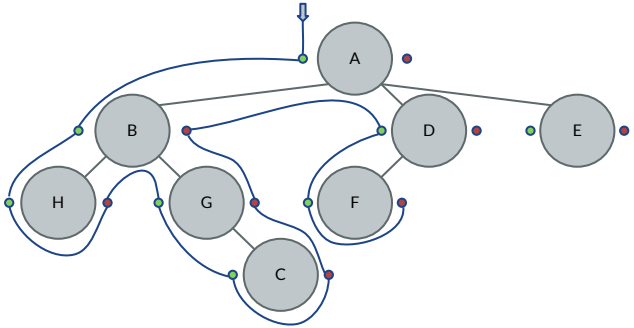


- Preorden {A, B, H, G, C, }
- Postorden {H, C, }

void preorder(Tree t)

void postorder(Tree t)

Recorridos

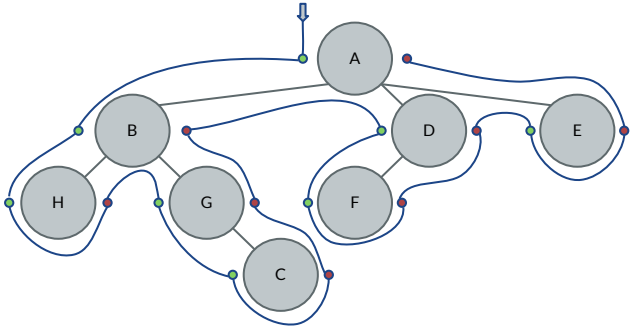


- Preorden {A, B, H, G, C, D, F, }
- Postorden {H, C, G, B, F, }

void preorder(Tree t)

void postorder(Tree t)

Recorridos

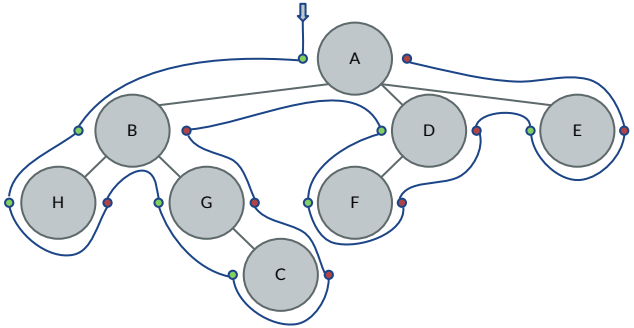


- Preorden {A, B, H, G, C, D, F, E}
- Postorden {H, C, G, B, F, D, E, A}

void preorder(Tree t)

void postorder(Tree t)

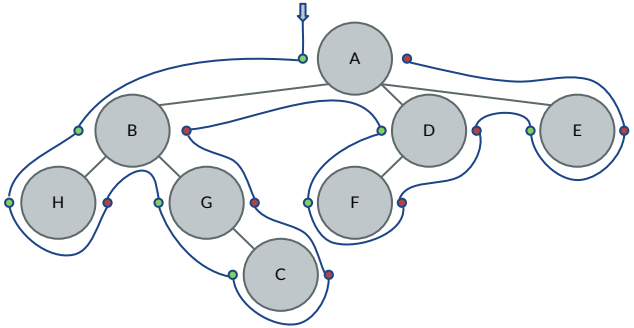
Recorridos



- Preorden {A, B, H, G, C, D, F, E}
- Postorden {H, C, G, B, F, D, E, A}

```
void preorder(Tree t) {
        auto root = t.getRoot();
        visit(root);
        for(auto& h: t.children(root)) {
                preorder(h);
        }
}

void postorder(Tree t)
```

Recorridos



- Preorden {A, B, H, G, C, D, F, E}
- Postorden {H, C, G, B, F, D, E, A}

```
void preorder(Tree t) {
        auto root = t.getRoot();
        visit(root);
        for(auto& h: t.children(root)) {
                preorder(h);
        }
}

void postorder(Tree t) {
        auto root = t.getRoot();
        for(auto& h: t.children(root)) {
                postorder(h);
        }
        visit(root);
}
```

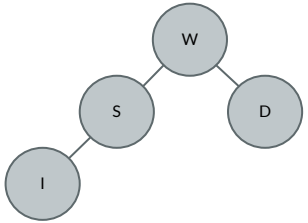
Índice

Árbol binario

- Un árbol n-ario en el que cada nodo tiene como máximo 2 hijos.

- Árbol binario no solo es un árbol con dos nodos como máximo, hay un hijo izquierdo y un hijo derecho.

- Es más fácil pensar como una estructura recursiva.

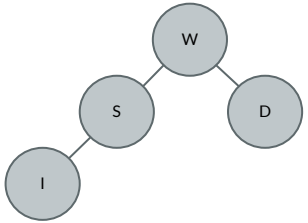


Árbol binario

- Ejemplos de uso son los árboles de expresión o los de decisión.

- Idealmente el número de nodos crece exponencialmente con la altura.

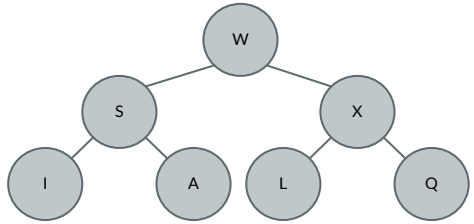
- Árbol binario completo: Todos los nodos internos tienen 2 hijos.

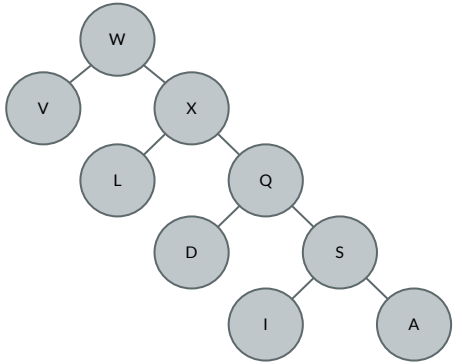


Árbol binario

- Árbol binario completo.
  - n: número de nodos
  - e: número de hojas
  - i: número de nodos internos
  - h: altura del árbol

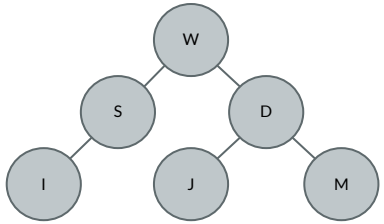
- Se cumple:
  - e = i + 1
  - n = i + e = 2e - 1
  - h < n
  - h ≤ e

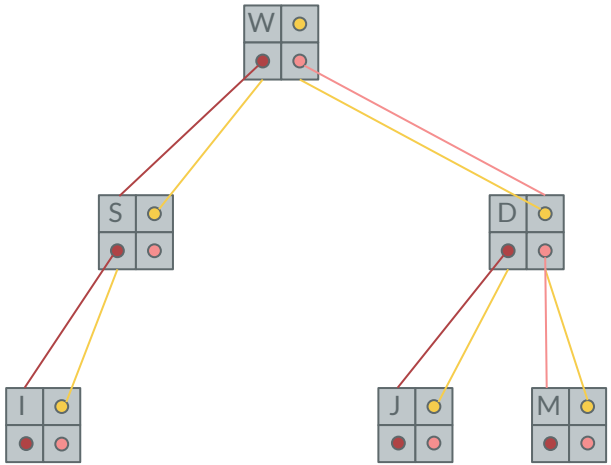




LinkedBinaryTree

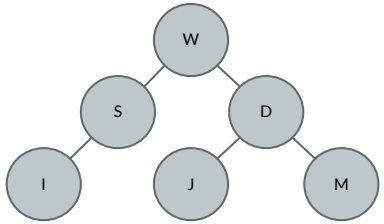
- Cada nodo contiene:
  - El elemento
  - Referencia al padre
  - Referencia al hijo izquierdo
  - Referencia al hijo derecho





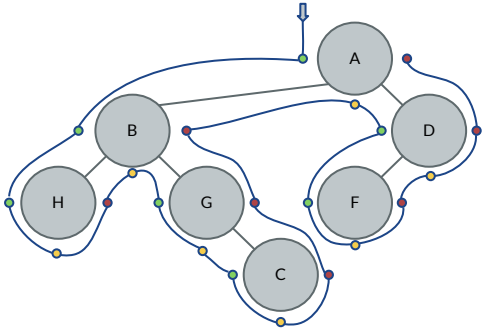
ArrayBinaryTree

- La posición de cada elemento establece la relación:
  - La raíz está en la posición 0.
  - Sea i la posición de un nodo:
    - Su hijo izquierdo está en la posición 2 * i + 1.
    - Su hijo derecho está en la posición 2 * i + 2.



0	1	2	3	4	5	6	7
W	S	D	I	-	J	M	-

Recorridos



- Preorden {A, B, H, G, C, D, F}
- Postorden {H, C, G, B, F, D, A}
- Inorden {H, B, G, C, A, F, D}

```
void preorder(Tree t) {
        auto root = t.getRoot();
        visit(root);
        preorder(hi);
        preorder(hd);
}

void postorder(Tree t) {
        auto root = t.getRoot();
        postorder(hi);
        postorder(hd);
        visit(root);
}

void inorder(Tree t) {
        auto root = t.getRoot();
        inorder(hi);
        visit(root);
        inorder(hd);
}
```

Motículo

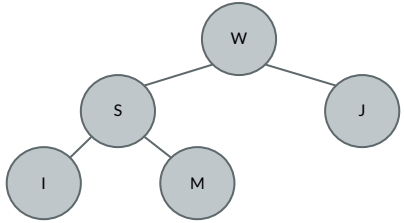
- Árbol binario que se construye en orden y es casi completo:
  - Antes de comenzar un nivel se completa el nivel anterior.

- Todo nodo es mayor que sus hijos.

- Suele implementarse con un ArrayBinaryTree.

Montículo

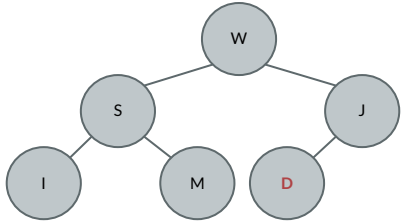
- Inserción:
  - Se inserta en la siguiente posición libre.
  - Si el nodo es mayor que su padre, se intercambian:
    - Se propaga hasta llegar a la raíz.



0	1	2	3	4	5	6
W	S	J	I	M	-	-

Montículo

- Inserción:
  - Se inserta en la siguiente posición libre.
  - Si el nodo es mayor que su padre, se intercambian:
    - Se propaga hasta llegar a la raíz.

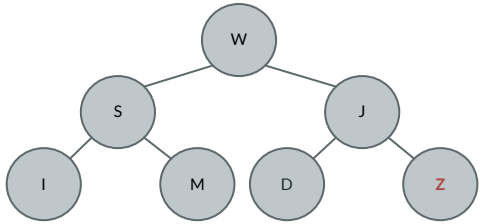


0	1	2	3	4	5	6
W	S	J	I	M	D	-

Insertamos D y como es menor que su padre, terminamos.

Montículo

- Inserción:
  - Se inserta en la siguiente posición libre.
  - Si el nodo es mayor que su padre, se intercambian:
    - Se propaga hasta llegar a la raíz.



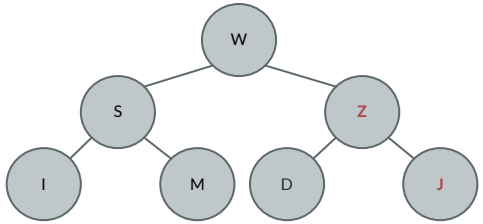
0	1	2	3	4	5	6
W	S	J	I	M	D	Z

Insertamos Z pero es mayor que su padre...

Montículo

- Inserción:
  - Se inserta en la siguiente posición libre.
  - Si el nodo es mayor que su padre, se intercambian:
    - Se propaga hasta llegar a la raíz.

padre = (i -1) / 2;



0	1	2	3	4	5	6
W	S	Z	I	M	D	J

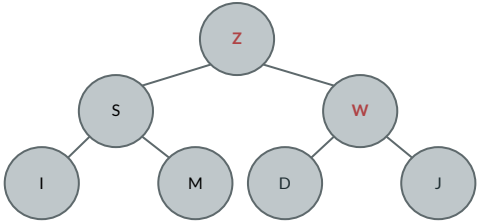
Insertamos Z pero es mayor que su padre, intercambiamos.

Montículo

- Inserción:
  - Se inserta en la siguiente posición libre.
  - Si el nodo es mayor que su padre, se intercambian:
    - Se propaga hasta llegar a la raíz.

padre = (i -1) / 2;

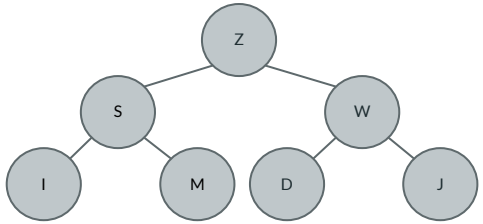
0	1	2	3	4	5	6
Z	S	W	I	M	D	J



Pero sigue siendo mayor que su padre, intercambiamos.

Montículo

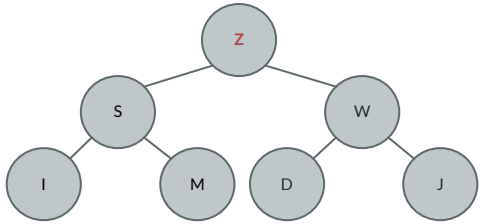
- Inserción:
  - Se inserta en la siguiente posición libre.
  - Si el nodo es mayor que su padre, se intercambian:
    - Se propaga hasta llegar a la raíz.



0	1	2	3	4	5	6
Z	S	W	I	M	D	J

Montículo

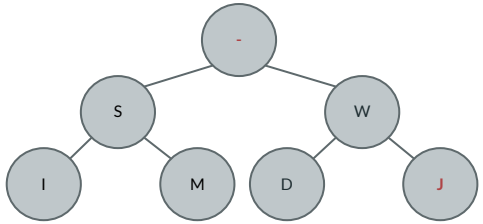
- Borrado:
  - Se borra la raíz.
  - Se sustituye por el último nodo:
    - Se sustituye por su hijo más grande hasta que se cumple la condición de inserción.



0	1	2	3	4	5	6
Z	S	W	I	M	D	J

Montículo

- Borrado:
  - Se borra la raíz.
  - Se sustituye por el último nodo:
    - Se sustituye por su hijo más grande hasta que se cumple la condición de inserción.

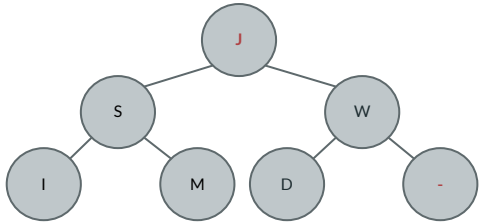


0	1	2	3	4	5	6
-	S	W	I	M	D	J

Se sustituye por el último nodo insertado.

Montículo

- Borrado:
  - Se borra la raíz.
  - Se sustituye por el último nodo:
    - Se sustituye por su hijo más grande hasta que se cumple la condición de inserción.

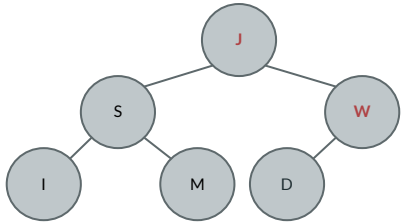


0	1	2	3	4	5	6
J	S	W	I	M	D	-

Se sustituye por el último nodo insertado.

Montículo

- Borrado:
  - Se borra la raíz.
  - Se sustituye por el último nodo:
    - Se sustituye por su hijo más grande hasta que se cumple la condición de inserción.

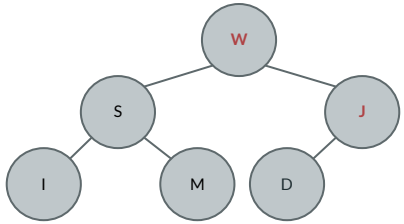


0	1	2	3	4	5	6
J	S	W	I	M	D	-

Se sustituye por el hijo más grande hasta cumplir la condición.

Montículo

- Borrado:
  - Se borra la raíz.
  - Se sustituye por el último nodo:
    - Se sustituye por su hijo más grande hasta que se cumple la condición de inserción.

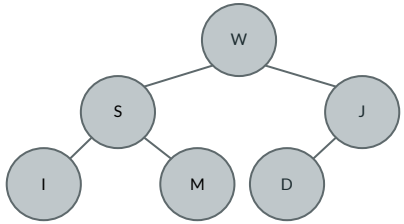


0	1	2	3	4	5	6
W	S	J	I	M	D	-

Si se cumple la condición, se para.

Montículo

- Borrado:
  - Se borra la raíz.
  - Se sustituye por el último nodo:
    - Se sustituye por su hijo más grande hasta que se cumple la condición de inserción.



0	1	2	3	4	5	6
W	S	J	I	M	D	-

Si se cumple la condición, se para.

Estructura de Datos II

David Concha Gómez

Asignatura obligatoria

Segundo cuatrimestre

Créditos: 6

Moodle de la asignatura

Guía docente