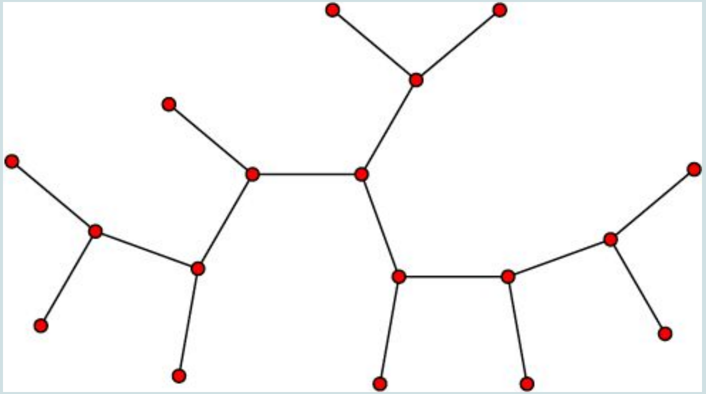
Árboles

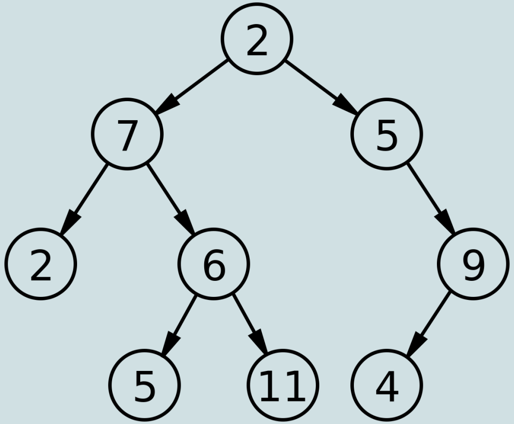
Árboles en teoría de gráficas

Una gráfica es un conjunto de nodos o vértices conectados por medio de aristas. Un **árbol** es una gráfica conexa acíclica.

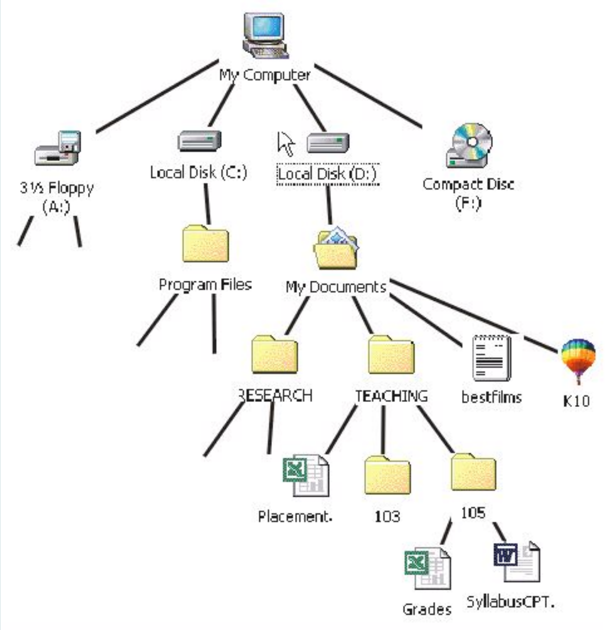


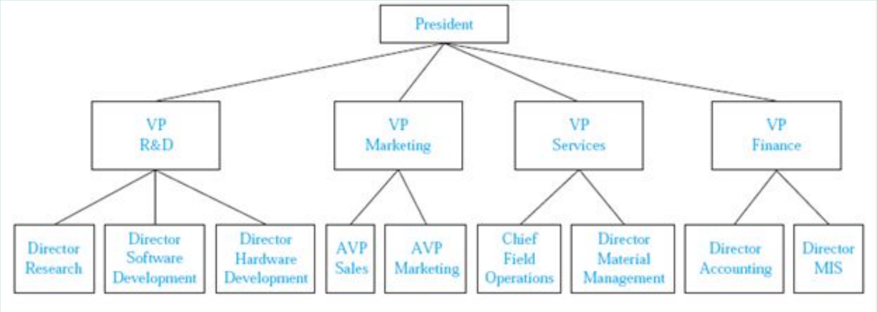
Árboles en computación

En computación, los árboles son estructuras no lineales donde se distingue a un nodo llamándolo **raíz.** Los árboles suelen representarse de arriba a abajo comenzando por la raíz.



ESTRUCTURAS DE DATOS



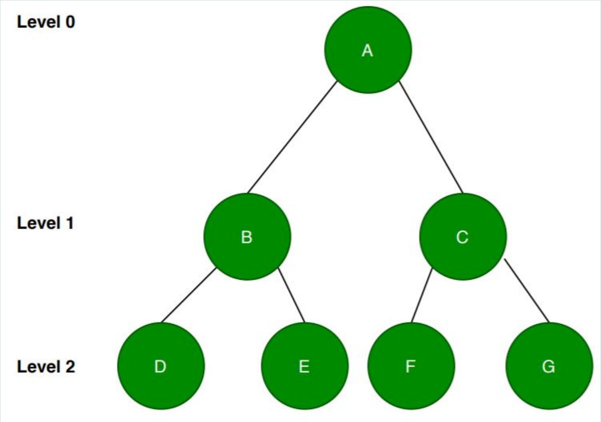


Los árboles suelen usarse para organizar un conjunto de elementos de forma jerárquica.

ESTRUCTURAS DE DATOS

Parentesco entre nodos

Se suele asignar entre nodos el parentesco de una familia donde cada uno de ellos es padre soltero de los nodos conectados a él en un nivel inferior.



Algunos ejemplos de parentesco en este árbol:

- B es padre de E
- F es hijo de C
- A es abuelo de D
- D y E son hermanos
- B es tío de G

ED
ESTRUCTURAS DE DATOS

Árboles binarios

Árbol en el que cada nodo tiene a lo más dos hijos.

Matemáticamente un árbol binario para elementos de un conjunto A se define de la siguiente manera:

1. Un árbol vacío es un árbol binario y de denota por *void.*
2. Si T_1 y T_2 son árboles binarios y **c** ϵ A, entonces *tree(* T_1, c, T_2*)* es un árbol binario, donde T_1 es el subárbol izquierdo, T_2 es el subárbol derecho y **c** la raíz del árbol.
3. Nada más es un árbol binario.

ED
ESTRUCTURAS DE DATOS

Árboles binarios

Para implementarlos se necesita:

- Una referencia al nodo raíz
- Cada nodo debe tener referencias a sus dos hijos (cualquiera de sus hijos puede ser *void* o en Java *null*).

Árbol vacío: Un árbol binario en el que su raíz es *null*.
Hoja: Un nodo en el que sus dos hijos son *null.*

ED
ESTRUCTURAS DE DATOS

Pregunta

¿Cuántas formas distintas puede tener un árbol binario de N nodos?

Altura

Sea v un nodo de un árbol binario, la altura del nodo se denota **h(v)** donde:

- Si v es *null*, entonces **h(v) = -1**
- Sean v_i y v_d los hijos de v, entonces **h(v) = 1 + max(h(v_i), h(v_d))**

La altura de las hojas vale 0

La altura del árbol es la altura de la raíz

Profundidad

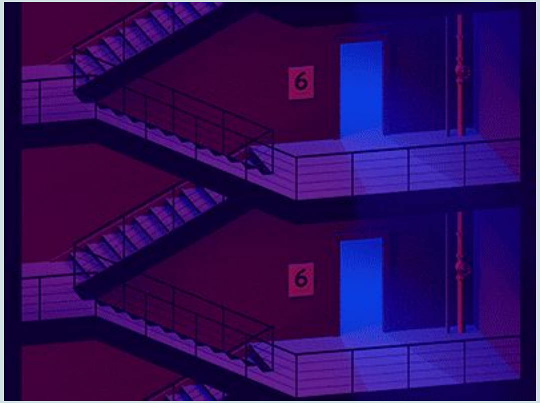
Sea v un nodo de un árbol binario, la profundidad del nodo se denota **d(v)** donde:

- Si v es es la raíz, entonces **d(v) = 0**
- Sea p el padre de v, entonces **d(v) = 1 + d(p)**

Niveles de profundidad

El i-ésimo nivel de profundidad en un árbol binario T, donde $0 \leq i \leq h(\,T\,)$ son todos los nodos tales que **$d(v) = i$.**

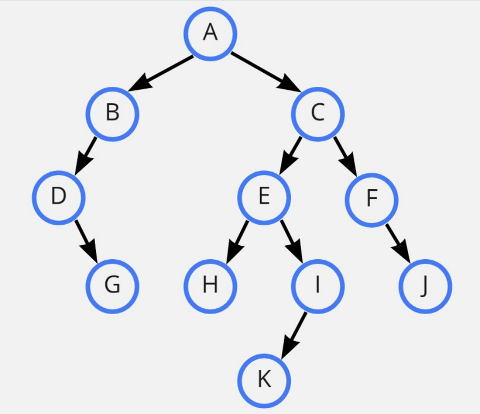
La **profundidad de un árbol** es la cantidad de niveles de profundidad que este posee.



ED
ESTRUCTURAS DE DATOS

Ejercicio

Calcular la altura y la profundidad del siguiente árbol:



Pregunta

¿Cuál es el máximo número de nodos que puede existir en el i-ésimo nivel de un árbol binario?

Árbol binario lleno

El máximo número de nodos que puede haber en un árbol binario de profundidad d es:

$$\sum_{l=0}^{d-1} (\text{máximo número de nodos en el nivel } l)$$

$$= \sum_{l=0}^{d-1} 2^l$$

$$= 2^d - 1$$

Un **árbol binario lleno** es un árbol binario de profundidad d con $2^d - 1$ nodos.

ESTRUCTURAS DE DATOS

Recorridos en árboles

Para explorar los nodos de un árbol binario, al ser una estructura no lineal, hay varias formas de hacerlo. Solo se deben seguir 2 reglas:

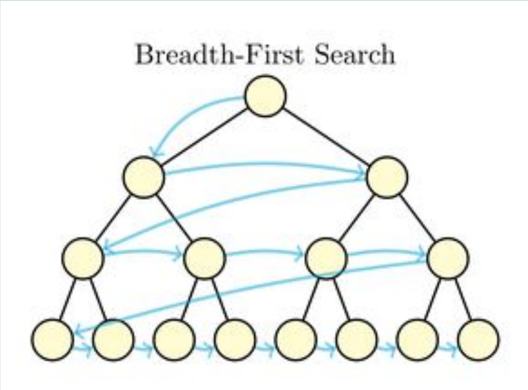
- Todos los nodos deben ser visitados.
- Cada nodo debe ser visitado solo una vez.

ED
ESTRUCTURAS DE DATOS

Recorrido por Amplitud (BFS)

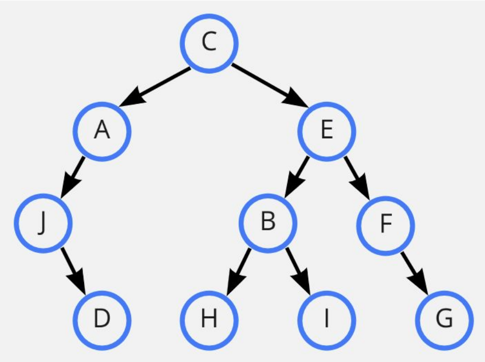
Un recorrido por amplitud o *Breadth-First Search (BFS)* se hace de la siguiente manera:

- En cada nivel los nodos se visitan de izquierda a derecha.
- Los nodos del nivel i deben visitarse antes que los del nivel i+1.



Ejemplo:

Obtener el recorrido BFS del siguiente árbol:



El recorrido BFS es: C A E J B F D H I G

Implementación de un recorrido BFS

Para este algoritmo necesitaremos una cola.

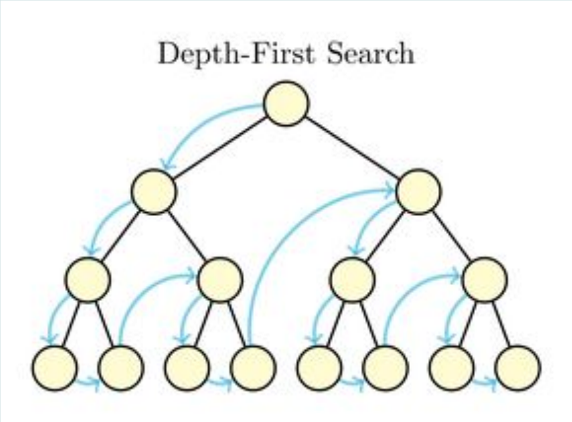
El algoritmo va como sigue:

```
Meter la raíz del árbol a la cola.
Mientras la cola no esté vacía:
    Sacar al nodo v de la cola.
    Procesar al nodo v.
    Si tiene hijo izquierdo, se mete a la cola.
    Si tiene hijo derecho, se mete a la cola.
```

Este algoritmo funciona también para árboles no binarios.

Recorrido por Profundidad (DFS)

En un recorrido por profundidad o *Depth-First Search (DFS)* se hace una exploración que va descendiendo por los niveles del árbol sin necesariamente haber visitado la totalidad de los nodos de niveles superiores.



Hacer **backtracking** es seguir un recorrido DFS en un árbol de decisiones.

Recorridos DFS

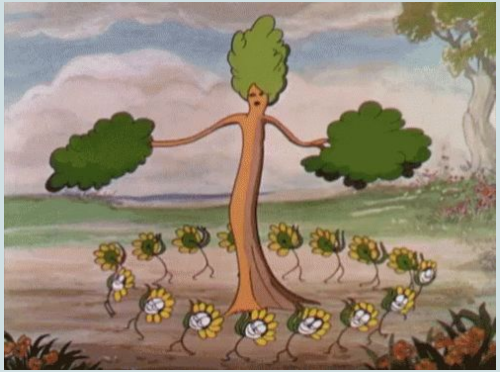
Los 3 recorridos DFS más comunes son: preorden, inorden y postorden.

Preorden	Inorden	Postorden
Preorden(nodo): 1 Procesar nodo 2 Preorden(nodo.izq) 3 Preorden(nodo.der)	Inorden(nodo): 1 Inorden(nodo.izq) 2 Procesar nodo 3 Inorden(nodo.der)	Postorden(nodo): 1 Postorden(nodo.izq) 2 Postorden(nodo.der) 3 Procesar nodo

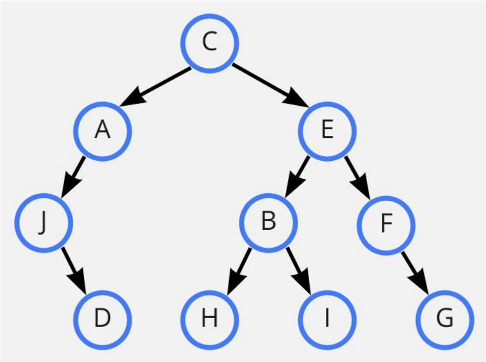
No hay que olvidar verificar que no procesemos nodos vacíos.

ED
ESTRUCTURAS DE DATOS

Ejemplo:



Obtener los recorrido DFS preorden, inorden y postorden del siguiente árbol:



El recorrido DFS-preorden es: C A J D E B H I F G

El recorrido DFS-inorden es: J D A C H B I E F G

El recorrido DFS-postorden es: D J A H I B G F E C

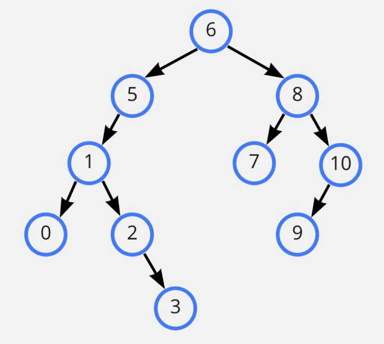
Pregunta

¿A partir de cuántos de los recorridos que hemos visto se puede reconstruir su árbol correspondiente?

Árbol Binario de Búsqueda (BST)

Un Árbol Binario de Búsqueda o *Binary Search Tree (BST)* es un árbol binario T que cumple que:

Para cualquier nodo v ϵ T, siendo $T_i(v)$ y $T_d(v)$ sus subárboles izquierdo y derecho respectivamente, todo nodo **u** ϵ $T_i(v)$ cumple que **u** ≤ v y todo nodo **w** ϵ $T_d(v)$ cumple que v ≤ w.



Simulador de Árboles Binarios de Búsqueda

Pedro Ulises Cervantes González - Facultad de Ciencias

Algoritmo para agregar en un BST

Definamos una función que inserta elementos en un BST como sigue:

```
agregar( arbol, u ):
  if ( arbol.raiz == null ):
     arbol.raiz = u
     return
  agregarAux( arbol.raiz, u )
```

```
agregarAux( nodo, u ):
  if ( u < nodo ):
     if ( nodo.izq == null ):
        nodo.izq = u
     else
        agregarAux( nodo.izq, u )
  if ( u > nodo):
     if ( nodo.der == null ):
        nodo.der = u
     else
        agregarAux( nodo.der, u )
```

Agregar elementos de esta forma ignora a los elementos repetidos.

Ejercicio

Agregar a un BST los elementos 56, 24, 45, 67, 3, 79 y 60 en ese orden.

Algoritmo para buscar en un BST

Definamos una función que busca elementos en un BST como sigue:

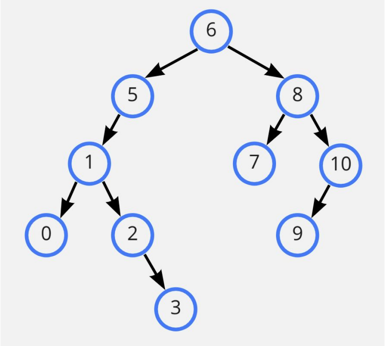
```
busca( nodo, u ):
  if ( nodo == null ):
      Fracaso
  if ( nodo == u ):
      Éxito
  if ( u < nodo ):
      busca( nodo.izq, u )
  if ( u > nodo ):
      busca( nodo.der, u )
```

Pregunta

¿Dónde está el mínimo y el máximo elemento en un BST?

Ejercicio:

Obtener el recorrido DFS inorden:

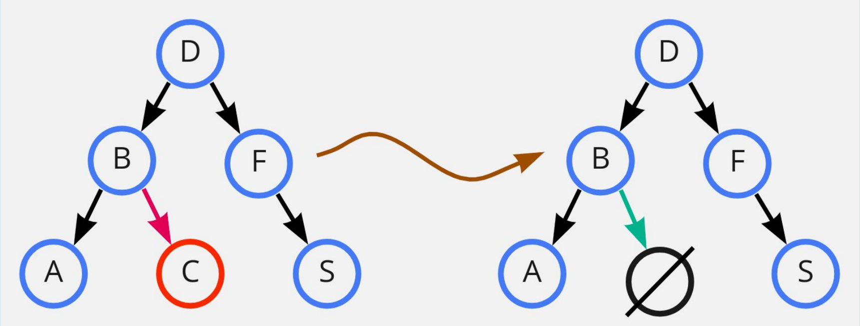


El recorrido DFS-inorden es: 0 1 2 3 5 6 7 8 9 10

Un recorrido DFS-inorden de un BST devuelve a los elementos en el orden correcto.

Eliminar hojas de un BST

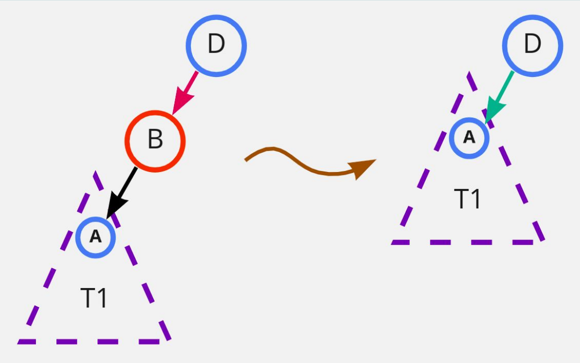
Para eliminar una hoja de un árbol basta con actualizar la referencia que tiene su padre a *null*.



Hay que hacer una verificación extra para saber si la hoja a eliminar es la raíz del árbol.

Eliminar nodos con un solo hijo en un BST

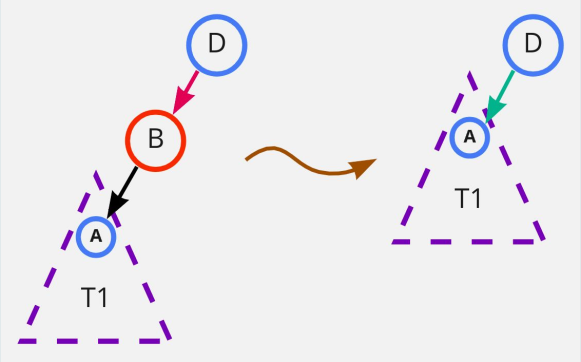
Para hacer esto se hace un **ascenso**, que consiste en actualizar referencias para que el padre del nodo eliminado ahora tenga como hijo a su nieto.

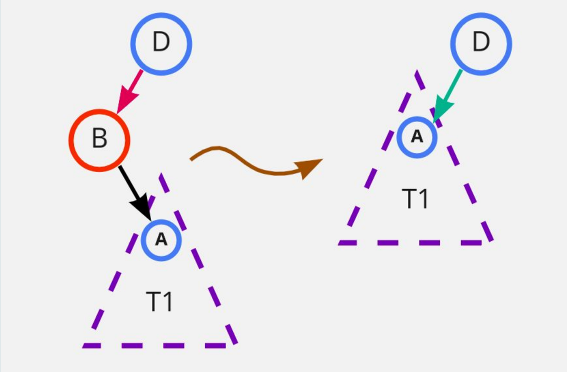


Hay que hacer una verificación extra para saber si el nodo a eliminar es la raíz del árbol.

ED
ESTRUCTURAS DE DATOS

Ascensos





Si para eliminar, el nodo que ocupará su lugar será su hijo izquierdo, entonces es un **ascenso izquierdo**.

Si es el derecho, entonces es un **ascenso derecho.**

ED
ESTRUCTURAS DE DATOS

Algoritmo para eliminar en un BST

Hay dos alternativas para eliminar un elemento en un BST:

1 Buscar al nodo v que contiene el elemento a eliminar.
2 a) Si es hoja, se remueve del árbol.
  b) Si tiene solo un hijo, se reemplaza v con su hijo.
  c) Si tiene dos hijos:

Sea u el máximo en $T_i(v)$

Sea u el mínimo en $T_d(v)$

Intercambiar el valor de v por el de u.
Eliminar u.

Pedro Ulises Cervantes González - Facultad de Ciencias

Pregunta

¿Cuál es la complejidad de agregar, buscar y eliminar en un BST?