Estructuras de datos jerárquicas

# Arboles

Es una estructura de datos (almacenamiento) combina las ventajas de los arreglos ordenados y las listas enlazadas.

Listas En 🡪 cualquier tamaño, búsqueda lenta, estructura dinamina

Array 🡪 búsqueda rápida por acceso indexado

Nodos (círculos) conectados por aristas (líneas).

Son entidades matemáticas

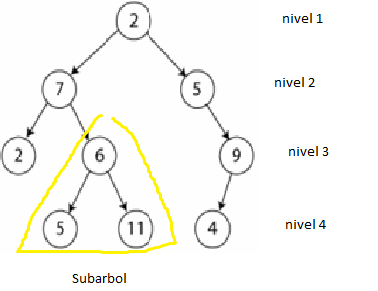
Es una especialización de los grafos.

Pequeños en el tope, grandes al final

(Binarios, heap, AVL, B B+ B\*, Expression tres, N-ary trees)

Maximo 2 hijos

Solo existe una ruta entre dos pares de nodos (cada nodo tiene una ruta única)



# Funciones

Funcionan mucho para hacer sistemas de archivos. Permiten organizar los archivos. Para facilitar la búsqueda de la información

En bases de datos la información se almacena en arboles B.

Algoritmos de compresión

Arboles de expresión para ciertos compiladores

# Terminología

Recorrer, es el procedo de ir de un X a un nodo Y

Ruta, es la lista de nodos en un recorrido

Raíz, esta en la cabeza del árbol, solo puede haber una. (Dentro del árbol en general existen subárboles, y por lo tanto raíces de subárboles).

Cualquier nodo tiene UNA UNICA arista hacia arriba (un nodo solo tiene un padre). Solo no aplica para la raíz.

Cualquier nodo puede tener una o mas líneas hacia abajo (incluso ninguna), estas líneas conectan con los nodos hijos (esto para árboles en general)

Nodo hoja, es un nodo que no tiene hijos.

Cualquier nodo puede ser considerado una raíz de un subárbol, e incluye absolutamente todos los nodos jerárquicamente debajo de este.

Dos nodos con el mismo padre se llaman Nodos hermanos.

Largo de una ruta (desde cualquier parte), es la cantidad de aristas presentes en esa ruta.

Siempre hay una ruta del nodo a el mismo de largo 0.

Profundidad de un nodo (desde raíz), el largo de la ruta única desde el nodo a la raíz. La profundidad de la raíz es cero.

La altura de un nodo es el largo de la ruta mas larga hasta un nodo hoja

Un nodo es visitado cuando se realiza alguna clase de acción sobre la información (si solo paso por él no se puede considerar como visitado).

Se recorre un árbol se VISITAN todos los nodos de un árbol.

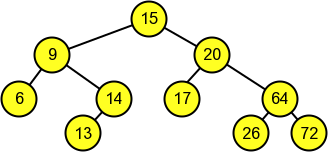
# Arboles binarios

Un árbol donde cada nodo tiene a lo sumo 2 hijos. Es decir (0,1,2). Si esto no se cumple no es binario.

Los hijos se referencian como hijo izquierdo e hijo derecho.

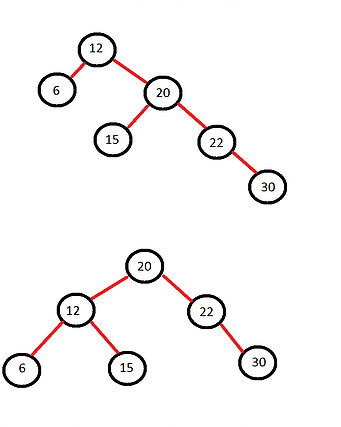
## Árbol binario de búsqueda

Para todo nodo se cumple que : “mi hijo izquierdo es menor y mi hijo derecho es mayor que yo”.



Dependiendo del orden de como entren los datos puede tenerse un árbol desbalanceado

Uno de los dos lados tiene más niveles que otro.



Se puede implementar con arreglos y con estructuras de datos dinámicas.

Pero no es tan conveniente utilizar arreglos.

Clase Nodo <T extends Comparable<? Super T>>

Nodo<T> left;

Nodo<T> right;

Clase arbolBinario <T extends Comparable<? Super T>>

Private Node<T> root;

## Operaciones

Boolean Contiene ()

True si esta el elemento

False si esta vacio o no lo tine

Public booleab contains (element)

Return this.contains(element, this.root);

Private Boolean contains (element, nodo){

If (node == null){

Return false;

Else{

Int compareResult = element.compareTo(node.element);

If (compareResult < 0)

Return contains(element, node.left);

Else

If (compareResult > 0)

Return contains(element, node.right);

Else

Return true;

FindMin()

findMin(nodo)

If this.isEmp

Return findMin(node.left)

Minimo (Full izq, hasta encontrar uno sin izquierda)

Maximo (Full der, hasta encontrar uno sin derecha)

Insert

Recorra el árbol con contiene y asegurese q no existe el dato a insertar. Decida que hacer en caso de que si esté. Si no esta inserte el nodo en la posiscion a donde seguiruia buscando .