Arboles:

Un árbol es una colección de elementos, llamados nodos, uno de los cuales se distingue con el nombre de raíz. Los nodos mantienen entre ellos una relación que define una estructura jerárquica (de "paternidad") entre ellos.

Recursivamente: un **árbol** puede verse como una estructura formada por la raíz de dicho árbol y una lista de árboles (los hijos).
Este nodo raíz es el padre de las raíces de los árboles que componen la lista, a partir del cual, se establece la relación de paternidad entre ellos.
El conjunto vacío de nodos es un árbol, llamado nulo o vacío.

Si n es un nodo y A 1 , A 2 , . . . , A k son árboles con raíces n 1 ,n 2 , . . . , n k , respectivamente, se puede construir un nuevo árbol haciendo n el padre de los nodos n 1 , n 2 , . . . , n k .
En este árbol n es la raíz y T 1 , T 2 , . . . , T k son los subárboles de la raíz. Los nodos n 1 , n 2 , . . . , n k se conocen como los hijos del nodo n.

Definiciones:

camino de un nodo *n* a un nodo m: es una secuencia de nodos *n* 1 , *n* 2 , ... , m de tal manera que *n i* es padre de *n i+1* para *i* = 1, 2, . . . , k-1.

longitud de un camino: es el número de nodos en el camino menos uno. El camino de un nodo a si mismo mide 0.

Si existe camino de un nodo *a* a un nodo b, entonces se dice que *a* es un **ancestro** de *b* y que *b* es un **descendiente** de a.
Un ancestro o descendiente de un nodo diferente de sí mismo se dice que es un **ancestro** o **descendiente propi**o, respectivamente.En un árbol la raíz es el único nodo que no tiene ancestros propios.

hoja: nodo sin descendientes propios.

subárbol de un árbol: es un nododel árbol junto con todos sus descendientes.

profundidad de un nodo: longitud del camino único de la raíz al nodo.

profundidad de un árbol: profundidad de la hoja más profunda.

grado de un nodo (Grado de salida de un nodo): número de hijos que tiene. El grado de un nodo hoja es cero.

altura de un nodo en un árbol: la longitud del mayor de los caminos del nodo a cada hoja. La altura de un árbol es la altura de la raíz.

profundidad de un nodo: longitud del único camino de la raíz a ese nodo.

niveles de un árbol: dado un árbol de altura *h* se definen los niveles *0...h* de modo que el nivel *i* está compuesto por todos los nodos de profundidad *i*.

orden de los nodos: los hijos de un nodo usualmente están ordenados de izquierda a derecha. Si se desea ignorar el orden de los dos hijos, se habla de un árbol no-ordenado.

Al recorrer los nodos de un árbol se consideran ciertas formas útiles para ordenar sistemáticamente los nodos de un árbol.

Los modos de realizar los recorridos en profundidad más importantes son llamados: pre-orden, post-orden y en-orden y se definen recursivamente como sigue:

El recorrido preorden de los nodos de A es la raíz n, seguida por los nodos de A 1 en pre-orden, después los nodos de A 2 en pre-orden, y así, hasta los nodos de A *k* en pre-orden.

El recorrido postorden de los nodos de A es los nodos de A 1 en post-orden, seguidos de los nodos de A 2 en post-orden, y así hasta los nodos de A *k* en post-orden, todos ellos seguidos de n.

El recorrido inorden de los nodos de A es los nodos de A 1 en-orden, seguidos por n, seguidos por los nodos de A 2 , . . . , A *k* , cada grupo de nodos en-orden.

A estos recorridos se les agrega el recorrido 'en amplitud' que corresponde a un recorrido 'por niveles ' del árbol.

EL TDA ARBOL

Usaremos un valor especial ARBOL_VACIO para el caso en que el árbol no contenga nodos, y NODO_NULO para expresar que un nodo no existe.

Primitivas del TDA Arbol

CREAR_Arbol: crea un árbol.
Precondición : no tiene
Postcondición.: el árbol vacío.

DESTRUIR. libera los recursos que mantienen el árbol .
Precondición: el árbol debe haber sido creado

PADRE (de un nodo): esta función recibe el valor de un nodo y devuelve el padre del nodo en el árbol.
Si el nodo no tiene padre, devuelve NODO_NULO
Precondición: el nodo pasado al método no es nulo .
Postcondición: ninguna.

HIJO_IZQUIERDA(de un nodo): devuelve el descendente más a la izquierda en el siguiente nivel del nodo recibido en el árbol. Devuelve NODO_NULO si el nodo recibido no tiene hijo a la izquierda.
Precondición: el nodo recibido no es nulo.
Postcondición: ninguna.

HERMANO_DERECHO(de un nodo): devuelve el descendiente (a la derecha del nodo recibido) del padre del nodo, en el árbol
Precondición : el nodo recibido no es nulo.
Postcondición: no tiene .

RAIZ(T). Devuelve el nodo que está en la raíz del árbol o NODO_NULO si el árbol está vacío.
Precondición: el árbol debe haber sido creado.
Postcondición: no tiene.

INSERTAR_HIJO_IZQUIERDA (de un nodo determinado): inserta un árbol recibido como hijo a la izquierda de un nodo determinado que pertenece al árbol .
Precondición : el árbol debe haber sido creado, y no estar vacío y el nodo recibido no es nulo.
Postcondición: el árbol alterado con el nuevo nodo agregado

INSERTAR_HERMANO_DERECHA(de un nodo determinado): inserta el árbol recibido como hermano a la derecha de un nodo que pertenece al árbol .
Precondición : el árbol debe haber sido creado, y no estar vacío y el nodo recibido no es nulo.
Precondición : el árbol ha sido creado y no está vacío y el nodo recibido no es nulo.

PODAR_HIJO_IZQUIERDA(de un nodo determinado): devuelve el subárbol con raíz hijo a la izquierda del nodo recibido. Al nodo recibido se le elimina el hijo izquierdo.del árbol
Precondición : el árbol ha sido creado y el nodo recibido no es nulo.
Postcondición: árbol alterado por la eliminación del subárbol.

PODAR_HERMANO_DERECHA: idem operación anterior, pero a derecha

ARBOLES BINARIOS

Un **árbol binario** un árbol cuyos nodos tienen a lo sumo dos hijos o bien es un árbol nulo.

Los hijos de un árbol binario se indican como **hijo izquierdo e hijo derecho.**

Un árbol binario puede definirse como un árbol que en cada nodo puede tener como máximo grado 2 (máximo dos hijos).

TDA Arbol Binario:

Para construir el TDA Arbol Binario bastaría con utilizar las primitivas de los árboles generales pero dado la gran importancia y peculiaridades que tienen este tipo de árboles, construiremos una serie de operaciones específicas. Consideraremos las siguientes:

CREAR: crea un árbol vacío.
Postcondición : árbol creado y vacío.

DESTRUIR: libera los recursos que mantienen el árbol.
Precondición: el árbol debe haber sido creado.

PADRE(de un nodo determinado) :devuelve el padre del nodo recibido. En caso de no existir el padre, devuelve NODO_NULO.
Precondición : el nodo recibido no es NODO_NULO.
Postcondición : ninguna

HIJO_IZQUIERDO(de un nodo determinado): devuelve el hijo a la izquierda del nodo recibido o NODO_NULO si no existe .
Precondición: el nodo recibido no es NODO_NULO.
Postcondición : ninguna.

HIJO_DERECHO(de un nodo determinado): devuelve el hijo a la derecha del nodo recibido o NODO_NULO si no existe .
Precondición: el nodo recibido no es NODO_NULO.
Postcondición: ninguna.

RAIZ: devuelve el nodo que está en la raíz del árbol o NODO_NULO si el árbol es nulo.
Precondición: el árbol debe haber sido creado.
Postcondición: ninguna

INSERTAR_HIJO_IZQUIERDA(recibe un árbol y un nodo determinado): inserta el árbol como hijo a la izquierda del nodo.
Si ya existe el hijo a izquierda, la primitiva se encarga de que sea destruído junto con sus descendientes.
Precondiciones: el árbol recibido no es ARBOL_VACIO y el nodo no es NODO_NULO.
Postcondición: árbol modificado por la inserción del nuevo nodo

INSERTAR_HIJO_DERECHA(recibe un árbol y un nodo determinado): inserta el árbol como hijo a la derecha del nodo.
Si ya existe el hijo a derecha, la primitiva se encarga de que sea destruído junto con sus descendientes.
Precondiciones: el árbol recibido no es ARBOL_VACIO y el nodo no es NODO_NULO.
Postcondición: árbol modificado por la inserción del nuevo nodo

PODAR_HIJO_IZQUIERDA(de un nodo determinado): devuelve el subárbol con raíz hijo a la izquierda del nodo recibido, al cual se le quita el subárbol izquierdo.
Precondición: el nodo recibido no es NODO_NULO.
Postcondición: árbol modificado por el borrado de un nodo

PODAR_HIJO_DERECHO(de un nodo determinado): devuelve el subárbol con raíz hijo a la derecha del nodo recibido, al cual se le quita el subárbol derecho.
Precondición: el nodo recibido no es NODO_NULO.
Postcondición: árbol modificado por el borrado de un nodo

ARBOLES BINARIOS DE BUSQUEDA

Un árbol binario de búsqueda(ABB) es un árbol binario con la propiedad de que todos los elementos almacenados en el subárbol izquierdo de cualquier nodo *x* son menores que el elemento almacenado en *x*, y todos los elementos almacenados en el subárbol derecho de *x* son mayores que el elemento almacenado en *x*.

Si se listan los nodos del ABB inorden se obtiene la lista de nodos ordenada.
Esta característica de ABB facilita el diseño de un procedimiento para realizar la búsqueda

Se prueba que una búsqueda o una inserción en un ABB requiere $O(\log_2 n)$ operaciones en el caso medio, en un árbol construido a partir de n claves aleatorias, y en el peor caso una búsqueda en un ABB con n claves puede implicar revisar las n claves, o sea, es O(n).

Primitivas del árbol binario de búsqueda

Crear : crea el álbol BB
Precondición : ---
Postcondición: árbol creado

Insertar (un elemento determinado): agrega el elemento al árbol manteniendo el orden del mismo.
Precondición : el árbol debe haber sido creado.
Postcondición: árbol alterado por la inserción del nuevo elemento.

Borrar(un elemento determinado) : borra el elemento indicado.

Precondiciones: el árbol debe haber sido creado y el elemento debe estar en el mismo.
Postcondiciones: árbol modificado por el borrado del elemento.
Buscar(un elemento determinado): determina si un elemento está o no en el árbol
Precondiciones: el árbol debe haber sido creado
Postcondiciones. ninguna

ARBOLES EQUILIBRADOS AVL

Un árbol binario está equilibrado (según han definido Addelson-Velskii y Landis) si, para cada uno de sus nodos ocurre que las alturas de sus dos subárboles difieren a lo sumo en 1.

Los árboles que cumplen esta condición son denominados a menudo árboles AVL.

La especificación coincide con la del Arbol binario de búsqueda.
La implementación varía en las operaciones que modifican la altura de un nodo: insertar y borrar, porque para mantener tras cada inserción y borrado la condición de equilibrio, se llevan a cabo algunos reajustes locales, cambiando punteros.

A través de los árboles AVL se llega a un procedimiento de búsqueda análogo al de los ABB pero con la ventaja de garantizar un caso peor de $O(\log_2 n)$,y manteniendo el árbol siempre equilibrado.

Arbol AVL de altura h con mínima cantidad de nodos:

En un árbol de tal característica, debe haber una raíz, un subárbol AVL mínimo de altura h-1 y otro subárbol AVL también mínimo de altura h-2.

El número de nodos $n(T_h)$ está dado por la relación de recurencia

$$\mathbf{n(T_h) = 1 + n(T_{h-1}) + n(T_{h-2})}$$

Esta relación es similar a la que aparece en los números de Fibonacci ($F_n = F_{n-1} + F_{n-2}$) (los valores para $n(T_h)$ están relacionados con los valores de la sucesión. de Fibonacci)

$$\mathbf{n(T_h) = F_{h+2} - 1}$$

Resolviendo se llega a

$$\mathbf{\log_2(n+1) <= h < 1.44 \log_2(n+2)-0.33}$$

La altura para un AVL de n nodos, nunca excede al 44% de la longitud de la altura de un árbol completamente equilibrado con esos n nodos.
En consecuencia, aún en el peor de los casos llevaría un tiempo $O(\log_2 n)$ al encontrar un nodo con una clave dada.