# GUÍA DE APRENDIZAJE PARA CONCURSANTES ICPC Y IOI: ÁRBOLES Y SUS PROPIEDADES BÁSICAS

# 1. Introducción

En ciencias de la computación y en informática, un árbol es un tipo abstracto de datos (TAD) ampliamente usado que imita la estructura jerárquica de un árbol, con un valor en la raíz y subárboles con un nodo padre, representado como un conjunto de nodos enlazados. A dicho TAD y sus propiedades básicas estará enfocada la presente guía.

# 2. Conocimientos previos

## 2.1. Registro o estructura

Las estructuras son colecciones de variables relacionadas bajo un nombre. Las estructuras pueden contener variables de muchos tipos diferentes de datos a diferencia de los arreglos que contienen unicamente elementos de un mismo tipo de datos.

## 2.2. Puntero

Un puntero es una variable que almacena la dirección de memoria de un objeto. Los punteros se usan ampliamente en C y C++ para tres propósitos principales:

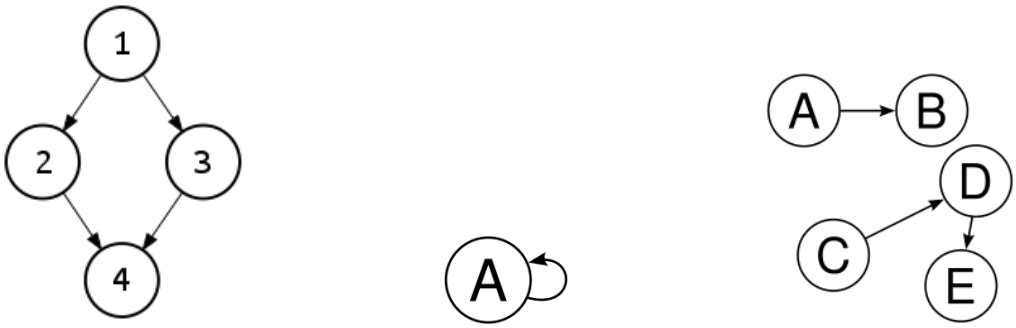
1. para asignar nuevos objetos en el montón.
2. para pasar funciones a otras funciones
3. para iterar sobre elementos en matrices u otras estructuras de datos.

# 3. Desarrollo

Un árbol es una estructura (posiblemente no lineal) de datos compuesta de nodos, vértices y aristas que es acíclica. Un árbol que no tiene ningún nodo se llama árbol vacío o nulo. Un árbol que no está vacío consta de un nodo raíz y potencialmente muchos niveles de nodos adicionales que forman una jerarquía.

Una estructura de datos de árbol se puede definir de forma recursiva (localmente) como una colección de nodos (a partir de un nodo raíz), donde cada nodo es una estructura de datos con un valor, junto con una lista de referencias a los nodos (los hijos), con la condición de que ninguna referencia esté duplicada ni que ningún nodo apunte a la raíz.

Alternativamente, un árbol se puede definir de manera abstracta en su conjunto como un árbol ordenado, con un valor asignado a cada nodo. Ambas perspectivas son útiles: mientras que un árbol puede ser analizado matemáticamente, realmente es representado como una estructura de datos en la que se trabaja con cada nodo por separado (en lugar de como una lista de nodos y una lista de adyacencia entre nodos, como un grafo). Mirando a un árbol como conjunto, se puede hablar de el nodo padre de un nodo dado, pero en general se habla de una estructura de datos de un nodo dado que sólo contiene la lista de sus hijos sin referencia a su padre (si lo hay).




En la imagen anterior tenemos tres representaciones que no son árboles por diferentes cuestiones que analizaremos a continuación:

- En la primera representación (izquierda) nos es un árbol debido a que el nodo $4$ tiene más de un padre lo cual en un árbol no ocurre ya que cada nodo tiene un solo padre exceptuando uno que no tendrá ningún padre el cual será la raiz del árbol.
- En la segunda representación (centro) no podemos decir que es un árbol ya que a pesar que presenta un solo nodo o vértice existe un ciclo, además el nodo A tiene un solo padre pero no existe un nodo sin padre que pueda ser la raíz del árbol
- En la tercera representación (derecha) no existen ciclos y pero existen dos nodos sin padre cuando debe existir uno solo. Además todos los nodos están conectados entre sí. Para este caso podemos decir que estamos en presencia de un bosque con dos arboles. El primer árbol conformado por los nodos $B$ y $A$ siendo este último la raiz del mismo. El segundo árbol lo componen los nodos $E$,$D$ y $C$ siendo este la raíz.

### 3.1. Tipo de datos frente a la estructura de datos

Hay una distinción entre un árbol como un tipo de datos abstracto y como una estructura concreta de datos, de forma análoga a la distinción entre una lista y lista enlazada. Como tipo de dato, un árbol tiene un valor e hijos, y los hijos son a su vez subárboles; el valor y los hijos de un árbol se interpreta como el valor del nodo raíz y los subárboles de los hijos del nodo raíz. Para permitir que los árboles puedan ser finitos, hay que, o bien permitir que la lista de los hijos pueda estar vacía, o permitir que los árboles puedan estar vacíos, en cuyo caso la lista de los hijos pueden ser de tamaño fijo (factor de ramificación, especialmente 2 o binario), si se desea.

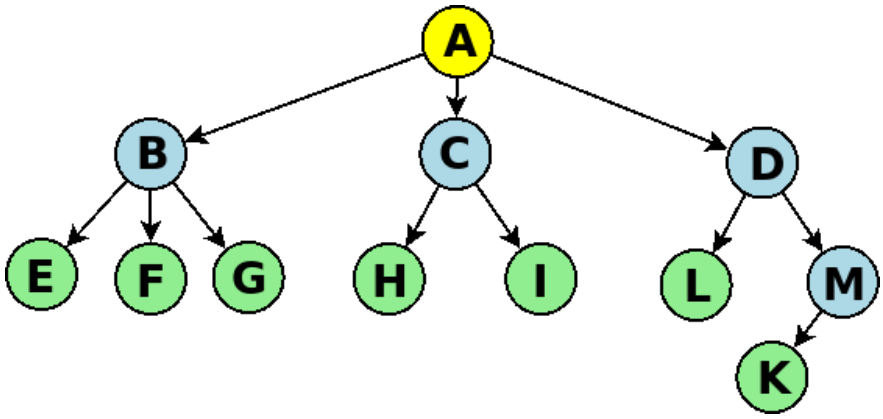
Como una estructura de datos, un árbol vinculado es un grupo de nodos, donde cada nodo tiene un valor y una lista de referencias a otros nodos (sus hijos). Esta estructura de datos realmente define a un grafo dirigido, porque puede tener bucles o varias referencias al mismo nodo, del mismo modo que una lista enlazada. Luego también existe el requisito de que no hay dos referencias que apuntan al mismo nodo (que cada nodo tiene como máximo un solo padre, y de hecho todos lo tienen, a excepción de la raíz), y un árbol que viola esto es árbol corrupto.

Debido al uso de referencias a los árboles en la estructura de datos de un árbol enlazado, a menudo se habla de árboles en los que implícitamente están siendo representados por referencias

al nodo raíz, ya que esto es, normalmente, la forma en la que se aplican realmente. Por ejemplo, en lugar de un árbol vacío, uno puede tener una referencia nula: un árbol nunca está vacío, sino que una referencia a un árbol puede ser nula.

### 3.2. Terminología utilizada en árboles

Para explicar los diferentes conceptos que se manejan en un árbol vamos a utilizar la siguiente representación que se muestra a continuación.




- **Nodos:** Se le llama nodo a cada elemento que contiene un árbol.
- **Nodo Raíz:** Se refiere al primer nodo de un árbol, solo un nodo del árbol puede ser la raíz (nodo de fondo amarillo).
- **Nodo Padre:** Se utiliza este termino para llamar a todos aquellos nodos que tiene al menos un hijo (nodos de fondos amarillo y azules).
- **Nodo Hijo:** Los hijos son todos aquellos nodos que tiene un padre (nodos de fondo azul y verde).
- **Nodo Hermano:** Los nodos hermanos son aquellos nodos que comparte a un mismo padre en común dentro de la estructura. Los nodos $B$,$C$ y $D$ son hermanos, los nodos $E$,$F$ y $G$ también son hermanos.
- **Nodo Hoja:** Son todos aquellos nodos que no tienen hijos, los cuales siempre se encuentran en los extremos de la estructura (nodos de fondo verde).
- **Nodo Rama:** Estos son todos aquellos nodos que no son la raíz y que además tiene al menos un hijo (nodos de fondos verdes).

Los árboles además de los nodos tiene otras propiedades importantes que son utilizadas en diferente ámbitos los cuales son:

- **Nivel:** Nos referimos como nivel a cada generación dentro del árbol. Por ejemplo, cuando a un nodo hoja le agregamos un hijo, el nodo hoja pasa a ser un nodo rama pero además el

árbol crece una generación por lo que el árbol tiene un nivel mas. Cada generación tiene un número de nivel distinto que las demas generaciones.

- Un árbol vacío tiene 0 niveles
- El nivel de la raíz es 1
- El nivel de cada nodo se calculado contando cuantos nodos existen sobre el, hasta llegar a la raíz + 1, y de forma inversa también se podría, contar cuantos nodos existes desde la raíz hasta el nodo buscado + 1.

El nodo $A$ tiene nivel 1. Con nivel 2 están los nodos $B$,$C$ y $D$ mientras el nodo $K$ tiene nivel 4 y el resto de los nodos sin mencionar nivel 3.

- **Altura:** Le llamamos altura al número máximo de niveles de un árbol. La altura es calculado mediante recursividad tomando el nivel mas grande de los dos sub-árboles de forma recursiva de la siguiente manera:

$$altura = max(altura(hijo1), altura(hijo2), altura(hijoN)) + 1$$

En el caso del árbol de ejemplo su altura es cuatro.

- **Peso:** Conocemos como peso a el número de nodos que tiene un Árbol. Este factor es importante por que nos da una idea del tamaño del árbol y el tamaño en memoria que nos puede ocupar en tiempo de ejecución(Complejidad Espacial en análisis de algoritmos.)

  El peso se puede calcular mediante cualquier tipo de recorrido el cual vaya contando los nodo a medida que avanza sobre la estructura. El peso es un árbol es igual a la suma del peso de los sub-árboles hijos + 1.

$$peso = peso(hijo1) + peso(hijo2) + peso(hijoN) + 1$$

- **Orden:** El orden de un árbol es el número máximo de hijos que puede tener un nodo. Notemos que un árbol con Orden = 1 no tendría sentido ya que seria una estructura lineal. Ya que cada nodo solo podría tener un hijo. Este valor no lo calculamos, si no que ya lo debemos conocer cuando diseñamos nuestra estructura, ya que si queremos calcular esto lo que obtendremos es el grado
- **Grado:** El grado se refiere al número mayor de hijos que tiene alguno de los nodos del árbol y esta limitado por el orden, ya que este indica el número máximo de hijos que puede tener un nodo.

  El grado se calcula contando de forma recursiva el número de hijos de cada sub-árbol hijo y el numero de hijos del nodo actual para tomar el mayor, esta operación se hace de forma recursiva para recorrer todo el árbol.

$$grado = max(contarHijos(hijo1), contarHijos(hijo2), contarHijos(hijoN), contarHijos(nodo))$$

- **Sub-árbol:** Conocemos como sub-árbol a todo árbol generado a partir de una sección determinada del árbol, Por lo que podemos decir que un árbol es un nodo raíz con $N$ sub-árboles.

  Existen escenarios donde podemos sacar un Sub-Árboles del Árbol para procesarlo de forma separada, de esta forma el Sub-Árboles pasa a ser un Árbol independiente, También podemos eliminar Sub-Árboles completos, Agregarlos,entre otras operaciones. En árbol un posible subárbol sería el que conforman los nodos $B$, $E$, $F$ y $G$ donde el nodo B es la raiz.

- **Bosque:** Un bosque es un conjunto de árboles $n \geq 0$ disjuntos.
- **Descendiente:** Un nodo accesible por descenso repetido de padre a hijo. Por ejemplo el nodo $K$ es descendiente del nodo $D$ mientras el nodo $F$ no es descendiente del nodo $D$.
- **Ancestro:** Un nodo accesible por ascenso repetido de hijo a padre. Por ejemplo el nodo $D$ es ancestro del nodo $K$ mientras el nodo $D$ no es ancestro del nodo $F$.
- **Brazo:** La conexión entre un nodo y otro.
- **Camino:** Una secuencia de nodos y brazos conectados con un nodo descendiente. Un posible camimo es $D$,$M$ y $K$ no es un camino $B$,$A$ y $C$.
- **Longitud del camino:** Cantidad de nodos que se deben recorrer para llegar desde la raíz a un nodo determinado
- **Altura de un nodo:** La altura de un nodo es el número de brazos en el camino más largo entre ese nodo y una hoja.
- **Profundidad:** La profundidad de un nodo es el número de brazos desde la raíz del árbol hasta un nodo.
- **Rama:** Una ruta del nodo raíz a cualquier otro nodo.

### 3.3. Tipos de árboles

Los arboles pueden clasificarse tomando en cuenta su estructura y funcionamiento. A continuación, se presentan los tipos de árboles más utilizados:

- Árboles Binarios
- Árbol de búsqueda binario auto-balanceable
- Árboles AVL
- Árboles Rojo-Negro
- Árbol AA
- Árbol de segmento
- Árbol de rango
- Árboles Multicamino
- Árboles B (Árboles de búsqueda multicamino autobalanceados)

- Árbol-B+
- Árbol-B*

## 4. Implementación

Hay muchas maneras diferentes para representar e implementar árboles computacionalmente; las representaciones más comunes representan a los nodos dinámicamente como registros (estructuras) con punteros a sus hijos, a sus padres, o a ambos, o como elementos de un vector, con relaciones entre ellos determinadas por sus posiciones en este (por ejemplo, un montículo binario). Basicamente van existir dos variantes de implementación: implementación estáticas y implementación dinámica enlazada

En general un nodo en un árbol no tendrá punteros a sus padres, pero esta información puede ser incluida (ampliando la estructura de datos para incluir también un puntero al vector) o almacenarse por separado. Alternativamente, los enlaces ascendentes pueden ser incluidos en los datos del nodo hijo, como en un árbol binario enlazado. Dentro de las operaciones más comunes que podemos encontrar en un árbol está:

- Enumerar todos los elementos
- Enumerar la sección de un árbol
- Buscar un elemento
- Añadir un nuevo elemento en una determinada posición del árbol
- Borrar un elemento
- Podar: Borrar una sección entera de un árbol
- Insertar: Añadir una sección entera a un árbol
- Buscar la raíz de algún nodo
- Representar el árbol con un vector, arreglo o lista

## 5. Aplicaciones

Dentro de las aplicaciones y uso de los árboles se encuentran la representación un dato jerárquicamente y listas ordenadas de datos, el almacenamiento de un dato de tal modo que su búsqueda sea eficiente (ver búsqueda en árboles binarios y recorrido de árboles), como un flujo de trabajo para la composición de imágenes digitales y en algoritmos de encaminamiento

## 6. Complejidad

La complejidad de un árbol tanto espacial como temporal va a depender en gran medida de la manera que se implemente y de las operaciones que se definan.

## 7. Ejercicios

En la programación competitiva la utilización del árbol se ve sobre todo en ejercicios cuya área de conocimientos son estructuras de datos y teoría de grafos. En la primera la idea conceptual del árbol permite conceptualizar e implementar estructuras basadas en este modelo con ciertas restricciones de acuerdo al problema que permiten manejar de forma optima de acuerdo a las restricciones una coleción de datos. En el segundo caso el árbol es visto como un caso especial del grafo.