Estructura de Datos II

Árbol

Teoría General de Árboles

Los **árboles** son, sin duda, una de las estructuras de datos no lineales, empleadas en informática, tanto para resolver problemas de hardware como de software. Los árboles de directorios son organizaciones bastante empleadas por cualquier usuario o programador de una computadora. De igual manera cumplen un buen papel en la toma de **decisiones**, válido como árbol de decisiones.

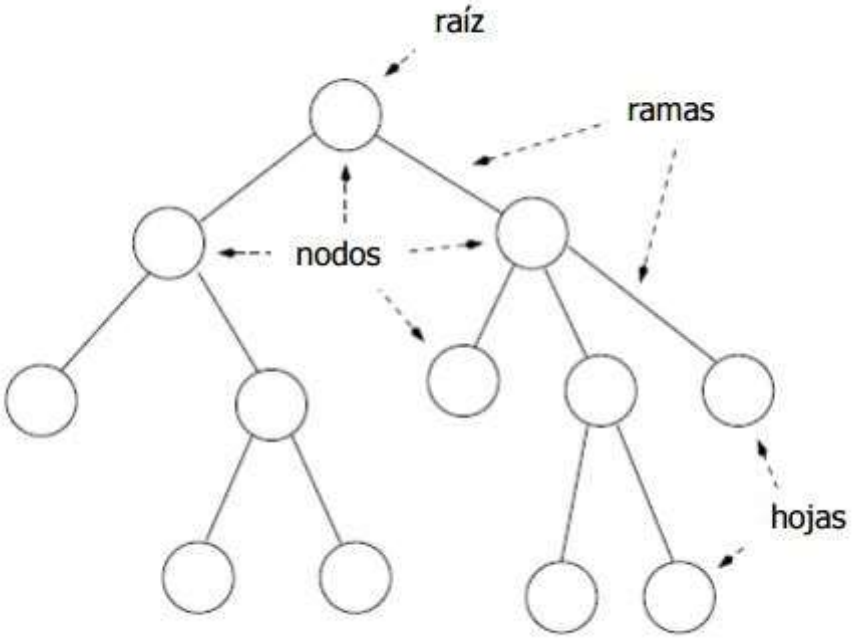
Los árboles **genealógicos** y los **organigramas** son ejemplos comunes. Entre otras aplicaciones, los árboles se emplean para **analizar** circuitos eléctricos y para representar la estructura de fórmulas matemáticas, así como para organizar la información de bases de datos, para representar la **estructura** sintáctica de un programa fuente en compiladores y para la toma de decisiones.

Definición de Árboles

Los **árboles** son estructuras de datos muy similares a las listas doblemente **enlazadas**, en el sentido que tienen punteros que apuntan a otros elementos, pero no tienen una estructura **lógica** de tipo lineal o secuencial como aquellas, sino **ramificada**. Tienen aspecto de árbol, de ahí su nombre.

Su estudio desde el punto de vista matemático pertenece a la teoría de grafos; desde el punto de vista informático son **estructuras de datos,** lo que significa que cada elemento, denominado nodo u hoja, contiene un valor. Su estudio corresponde a la **teoría** de bases de datos, y en esta terminología, los nodos que dependen de otros se denominan hijos. Cada hoja puede tener un máximo de hijos, si no tiene ninguno se dice que es un nodo **terminal**.

Un **árbol** es una estructura de datos no lineal en la que cada nodo puede apuntar a uno o varios nodos. También se suele dar una definición recursiva: un árbol es una estructura compuesta por un dato y varios árboles. Esto son definiciones **simples**. Una representación gráfica de los árboles se puede visualizar en la figura que se presenta a continuación:

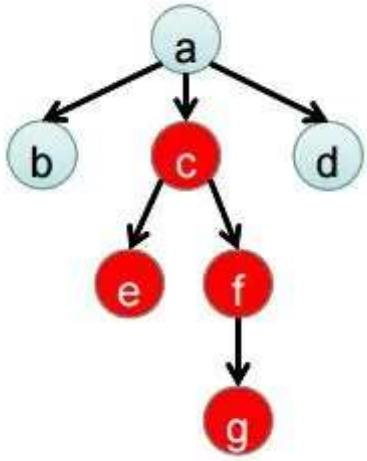


- Un **Árbol** consiste en un nodo (r, denominado nodo raíz) y una lista o conjunto de **subárboles** (A 1, A 2, .. A k).
- Si el orden de los subárboles importa, entonces forman una lista, y se denomina **árbol ordenado** (por defecto un árbol se supone que es ordenado).En caso contrario los subárboles forman un conjunto, y se denomina **árbol no ordenado.**
- Se definen como **nodos hijos** de r a los nodos raíces de los subárboles A 1, A 2, .. A k.
- Si b es un nodo hijo de a entonces a es el **nodo padre** de b.
- Un nodo puede tener cero o más hijos, y uno o ningún padre. El único nodo que no tiene padre es el nodo raíz del árbol.
- Un nodo sin hijos se denomina **nodo hoja o externo**. En caso contrario se denomina **nodo interno**.

Definiciones (II)

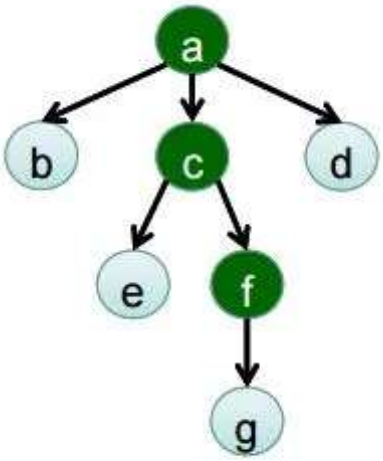
Se define un **camino** en un árbol como cualquier secuencia de nodos del árbol, n 1 … n p, que cumpla que cada nodo es padre del siguiente en la secuencia (es decir, que ni es el padre de ni+1). La longitud del camino se define como el número de nodos de la secuencia menos uno (p-1).

- Los **descendientes** de un nodo (c en el diagrama) son aquellos nodos accesibles por un camino que comience en el nodo.



Descendientes

- Los **ascendientes** de un nodo (f en el diagrama) son los nodos del camino que va desde la raíz a él.



Ascendientes

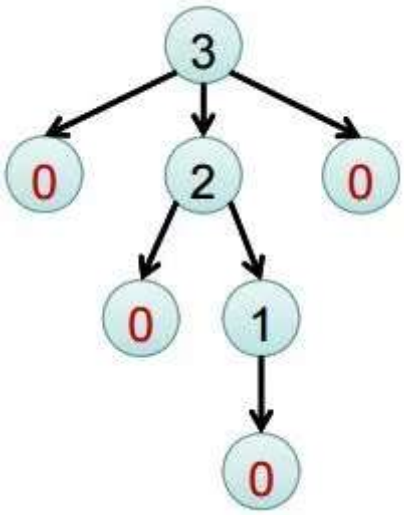
Altura

- Se define la **altura de un nodo** en un árbol como la longitud del camino más largo que comienza en el nodo y termina en una hoja.La altura de un nodo hoja es cero (0).

La altura de un nodo es igual a la mayor altura de sus hijos + 1.

La altura de un árbol se define como la altura de la raíz.

- La **altura de un árbol** determina la eficiencia de la mayoría de operaciones definidas sobre árboles.



Profundidad

- Se define la **profundidad de un nodo** en un árbol como la longitud del camino (único) que comienza en la raíz y termina en el nodo. También se denomina nivel.

- La profundidad de la raíz es 0
- La profundidad de un nodo es igual a la profundidad de su padre + 1

Recorrido de Árboles

1. **Preorden**: Se pasa por la raíz y luego se recorre en preorden cada uno de los subárboles. Recursivo.

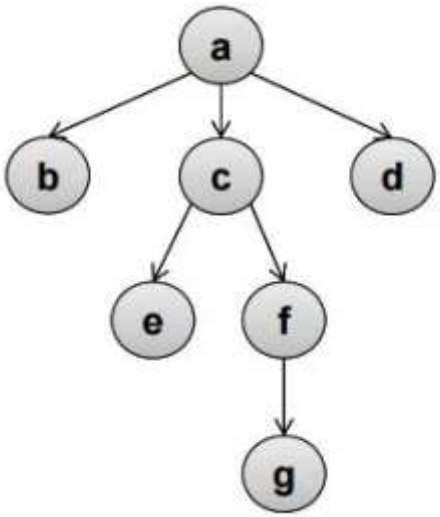
2. **Postorden**: Se recorre en postorden cada uno de los subárboles y luego se pasa por la raíz. Recursivo.
3. **Inorden**: Se recorre en inorden el primer subárbol (si existe). Se pasa por la raíz y por último se recorre en inorden cada uno de los subárboles restantes. Tiene sentido fundamentalmente en árboles binarios. Recursivo.
4. **Por Niveles:** Se etiquetan los nodos según su profundidad (nivel). Se recorren ordenados de menor a mayor nivel, a igualdad de nivel se recorren de izquierda a derecha.**No recursivo:** Se introduce el raíz en una cola y se entra en un bucle en el que se extrae de la cola un nodo, se recorre su elemento y se insertan sus hijos en la cola.

Recorrido de Árboles (II)

- **Preorden:** a,b,c,e,f,g,d

- **Postorden:** b,e,g,f,c,d,a

- **Inorden:** b,a,e,c,g,f,d
- **Por Niveles:** a,b,c,d,e,f,g

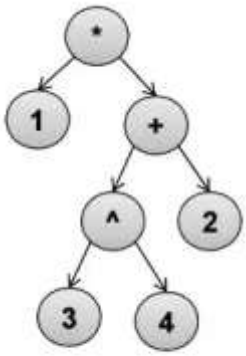


Parentizado sobre subárboles

- **Preorden:** a (b) (c (e) (f (g))) (d) g
- **Postorden:** (b) ((e) ((g) f) c) (d) a
- **Inorden:** (b) a ((e) c ((g) f)) (d)
- **Por Niveles:** (a) (b c d) (e f) (g)

Expresiones Matemáticas

- **Preorden** ⇒ Notación prefija: * 1 + ^ 3 4 2
- **Postorden** ⇒ Notación postfija: 1 3 4 ^ 2 + *
- **Inorden** ⇒ Notación habitual: 1 * ((3 ^ 4) + 2)



Evaluación de expresiones

Se recorre el arbol en **postorden**:
Si es un operando, se inserta en **pila**
Si es un operador:
- Se extraen dos operandos
- Se aplica el operador
- Se inserta en pila el resultado

Al final, la pila debe contener un único valor, el resultado.

Estructura para la creación de un árbol de orden dos

El **nodo** típico de un árbol difiere de los nodos para el manejo de las **listas**, aunque sólo en el número de nodos. A continuación se presenta un ejemplo de nodo para **crear** árboles de orden dos:

```
struct Arbol {
    int dato;
    struct Arbol *rama1;
```

```
struct Arbol *rama2;
};
```

Generalizando más se puede **declarar** un a constantes llamada **orden** que se le asigna el valor de **5**:

```
#define ORDEN 5
struct Arbol {
  int dato;
  struct Arbol *rama[ORDEN];
  };
```

El **movimiento** a través de árboles, salvo que se implementen punteros al nodo **padre**, será siempre partiendo del nodo raíz hacia un **nodo hoja**. Cada vez que se llegue a un nuevo nodo se podrá optar por cualquiera de los nodos a los que apunta para avanzar al siguiente nodo.

Un **ejemplo** de estructura en árbol es el sistema de **directorios y ficheros** de un sistema operativo. Aunque en este caso se trata de árboles con nodos de dos tipos, nodos directorio y nodos fichero, se podría considerar que los nodos hoja son **ficheros** y los nodos rama son **directorios**.

Operaciones Básicas con Árboles

Salvo que se trabaje con algún tipo de **árboles especiales**, las **inserciones** serán siempre en punteros de nodos hoja o en punteros libres de nodos rama. Ya que con estas estructuras no es tan fácil **generalizar**, existen muchas variedades de árboles.

Nuevamente se tiene casi el mismo concepto de operaciones de las que se disponía con las listas enlazadas:

- **Añadir o insertar elementos a un árbol.**
- **Buscar o localizar elementos dentro del árbol.**
- **Borrar elementos creados en el árbol.**
- **Moverse a través del árbol por cada uno de sus ramas.**
- **Recorrer el árbol completo.**

Los algoritmos de **inserción y borrado** dependen en gran medida del tipo de árbol que se esté implementando, de modo que por ahora se dejarán a un lado y se centrará la atención en el modo de recorrer los árboles.

CUESTIONARIO

1. **¿Que es un Árbol?**
2. **¿Que es un Árbol Descendiente?**
3. **¿Que es la altura de un Árbol?**
4. **¿Defina el concepto de profundidad en un Árbol?**
5. **Mencione los 4 recorridos de Arboles.**

Fuente: http://www.infor.uva.es/~cvaca/asigs/doceda/tema4.pdf (http://www.infor.uva.es/~cvaca/asigs/doceda/tema4.pdf)