Programación y Estructura de Datos

Temario de Teoría y resumen del Seminario de C++

Universitat d'Alacant
Universidad de Alicante

Resumen

Explicación del documento

Eduardo Espuch

Curso 2019-2020

Índice

prueba

1. Introducción a los TADs, los tipos lineas

prueba

1.1. Introducción a los TADs

prueba

1.2. Vectores

prueba

1.3. Listas

prueba

1.4. Pilas

prueba

1.5. Colas

prueba

2. La eficiencia de los algoritmos

prueba

2.1. Noción de complejidad

prueba

2.1.1. Complejidad temporal, tamaño del problema y paso

prueba

2.2. Cotas de complejidad

prueba

2.2.1. Cota superior, inferior y promedio

prueba

2.3. Notación asintótica

prueba

2.4. Obtención de cotas de complejidad

prueba

3. El tipo árbol

Vamos a comenzar recordando brevemente que es un árbol, dado previamente en la asignatura de Matemáticas Discreta. Podéis verlo con más detalle en los apuntes de esa asignatura.

Los arboles son grafos conexos y aciclicos y diremos que es un árbol generador si es árbol y subgrafo generador a la vez.

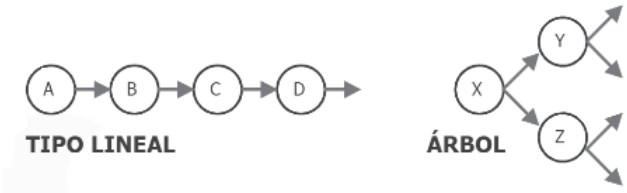
1. Dos vértices cualesquiera están conectados por un único camino.
2. Un grafo sera conexo si y solo si tiene un árbol generador.
3. El numero de relaciones en un árbol difiere en uno al numero de vértices de este
4. Todo árbol no trivial tiene al menos 2 vértices de grado 1.

Dado un vértice r_0, el cual actúa como extremo inicial, únicamente y todo vértice está conectado con éste, podemos definir un árbol enraizado a r_0 y distinguiremos los distintos niveles o alturas n a los vértices cuyos caminos sean de longitud n.

Cabe decir que, aunque es recomendable conocer que es un árbol en un contexto matemático, no es necesario.

3.1. Definiciones generales

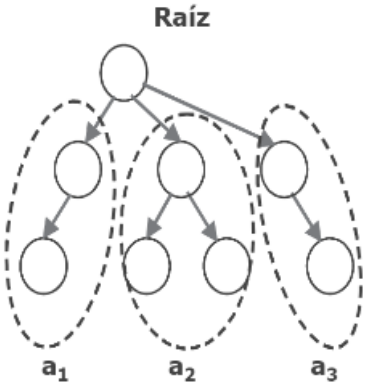
Diremos que la estructura de datos del tipo árbol aparece a la hora de establecer una jerarquía entre los elementos que lo constituyen, obtenido a partir de eliminar el requisito de que cada elemento en una estructura de datos del tipo lineal solo podría tener como máximo un sucesor. Llamaremos a los elementos de estructura como **nodos**.



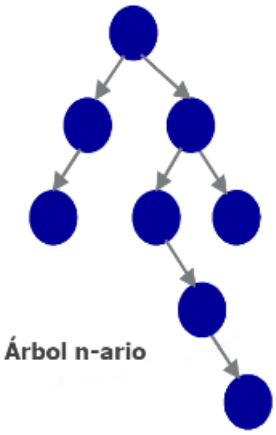
Procedamos a enlistar una serie de propiedades y definiciones respecto al concepto de la estructura del tipo árbol:

- Un único nodo puede constituir un árbol, ademas de ser la raíz de éste.
- Diremos que un **árbol es vació** o **nulo** si este tiene 0 nodos, es decir, no tiene una raíz.
- La información almacenada dentro del nodo se denomina **etiqueta**.

- Dados n arboles $a_1, \ldots, a_n$ se puede construir uno nuevo al enraizar todos arboles a un nuevo nodo, pasando estos a ser **subárboles** del nuevo árbol, siendo el nuevo nodo la raíz de éste.

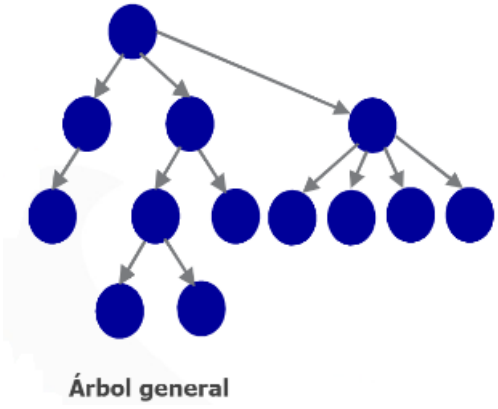


- **Árbol n-ario** es un arraigado árbol en el que cada nodo no tiene mas que n sucesores. Se podría decir que todo árbol solo podrá tener como máximo n árboles. Lo llamaremos **árbol binario** cuando $n = 2$.

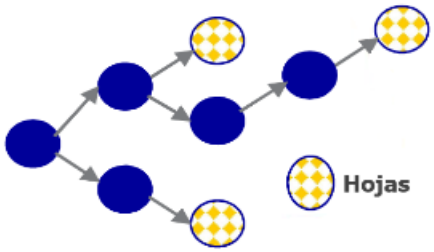


  Es mas, llamaremos grado de un árbol al número máximo de sucesores que podrán tener los subárboles, siendo en el árbol n-ario un grado de n.

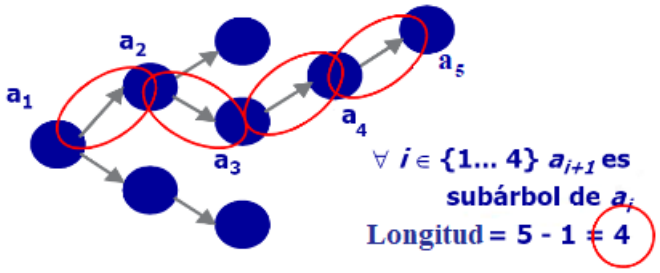
- **Árbol general** es un árbol sin una restricción en el número de sucesores por nodos, es decir, que podrá tener una cantidad de subárboles indeterminada.



- Llamaremos **hojas** a los arboles compuestos por un único nodo, o dicho de otra manera, cuando un nodo no tiene sucesores.

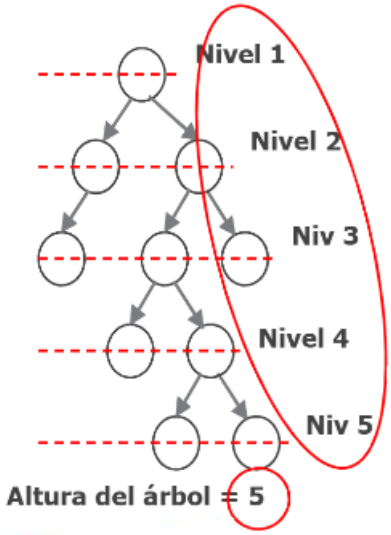


- **Camino:**
  - una secuencia $a_1, \ldots, a_s$ de árboles tal que $\forall i \in \{1, \ldots, s-1\}$, a_{i+1} es subárbol de a_i. Pueden darse casos de caminos de un árbol a si mismo, siendo la sucesión el propio árbol.
  - La **longitud** del camino corresponde a la cantidad árboles menos uno ya. Se considera que la longitud de un árbol a si mismo es de 0. Esto se debe a que los caminos consisten en enlistar los nodos que son sucesores pero la longitud es el numero de sucesiones necesarias para alcanzar desde la raíz el ultimo elemento del camino.



- Terminologías para sucesores:
  - Diremos que x es **ascendiente** de y si existe un camino con x como la raíz del árbol e y uno de los sucesores directos o indirectos de x. También podemos decir que y es **descendiente** de x.

    Debido a la definición de camino, todo árbol es ascendiente(descendiente) de sí mismo.
  - Diremos que son **ascendiente(descendientes) propios** todos los ascendiente(descendientes) excluidos del propio árbol.
  - **Padre** es el primer ascendiente propio de un árbol. No todos los arboles tienen padre.
  - **Hijos** son los primeros descendientes propios, si existen, de un árbol.
  - **Hermanos** son los subárboles con el mismo padre.
  - **Profundidad** de un subárbol es la longitud del único camino desde la raíz a dicho subárbol. Recordad que un árbol/subárbol consiste en el nodo y sus sucesores.



•**Nivel** de un nodo:

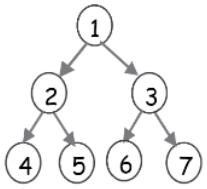
-el nivel de un árbol vacío es 0
-el nivel de la raíz es 1
-si un nodo está en el nivel i,
sus hijos están en el nivel $i + 1$

•**Altura(profundidad)** de un árbol:

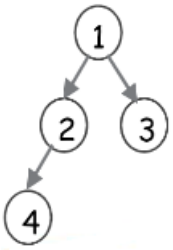
-es el máximo nivel de los nodos de un árbol

Hay casos en los que la gente inicializa la raíz al nivel 0 o el nivel mas bajo es donde esta la hoja con el camino de mayor longitud a la raíz (de igual manera, pudiendo ser el nivel inicial 1 o 0). Y la altura, según la definición dada, se aplicaría sin ningún problema a la forma que se decida usar. Para esta asignatura se considerara lo mostrado previamente, pero considerar esto como posible ya que la gente puede hacer lo que quiera siempre que se explique al principio.

•**Árbol lleno** es árbol en el que todos sus subárboles tienen n hijos(con n siendo el grado del árbol) y todas sus hojas tienen la misma profundidad.



•**Árbol completo** es un árbol cuyos nodos corresponden a los nodos numerados (la numeración se realiza desde la raíz hacia las hojas y, en cada nivel, de izquierda a derecha) de 1 a n en el árbol lleno del mismo grado, siendo todo lleno uno completo.



En algunos casos, se dice que un árbol completo n-ario cuando todos los nodos tienen n sucesores o son hoja, parecido a lo que seria un árbol lleno, pero aquí se dice que es un árbol que se va llenando de izquierda a derecha por nivel, nunca se podrá tener un una hoja a profundidad x y otra a un nivel inferior a $x - 1$.

3.2. Árboles Binarios

prueba

3.3. Árboles de búsqueda

prueba

3.3.1. Árboles binarios de búsqueda

prueba

3.3.2. Árboles AVL

prueba

3.3.3. Árboles 2-3

prueba

3.3.4. Árboles 2-3-4

prueba

4. Conjuntos

prueba

4.1. Definiciones generales

prueba

4.2. Diccionario

prueba

4.2.1. Tabla de dispersión

prueba

4.3. Cola de prioridad

prueba

4.3.1. Montículo

prueba

4.3.2. Cola de prioridad doble

prueba

5. Grafos

prueba

5.1. Concepto de grafo y terminología

prueba

5.2. Especificación algebraica

prueba

5.3. Representación de grafos

prueba

5.4. Grafos dirigidos

prueba

5.4.1. Recorrido en profundidad o DFS

prueba

5.4.2. Recorrido en anchura o BFS

prueba

5.4.3. Grafos acíclicos dirigidos o GAD

prueba

5.4.4. Componentes fuertemente conexos

prueba

5.5. Grafos no dirigidos

prueba

5.5.1. Algoritmos de recorrido

prueba

6. Seminario de C++

prueba

6.1. Clases y objetos

prueba

6.2. Composición y herencia

prueba

6.3. Constructor y destructor

prueba

6.4. Funciones y clases amigas, reserva de memoria

prueba

6.5. Sobrecarga de operadores

prueba
Ejemplo de referencias:

Referencias

[1] *X86 Assembly/Floating Point*

[2] *x86 and amd64 instruction reference*

[3] *x86/x64 SIMD Instruction List (SSE to AVX512)*

[4] *Biblioteca de instrucciones para MMX y SSE*

[5] *Pagina de wikipedia de SPEC CPU*

[6] *Página donde se explica que es GCC y expone todo lo que lleva, estilo g++, .c, etc*

[7] *Información respecto al paquete de pruebas .mcf, asociado a las tarjetas de red*

[8] *Información respecto a OODBMS y como funciona la transcripción de datos a dispositivos de almacenamiento*