Lección 3. ÁRBOLES

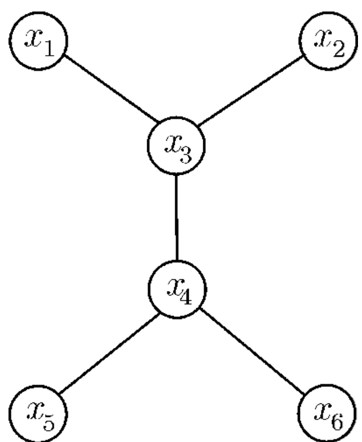
1. DEFINICIONES. PROPIEDADES Y EJEMPLOS

Sea G un grafo no dirigido.

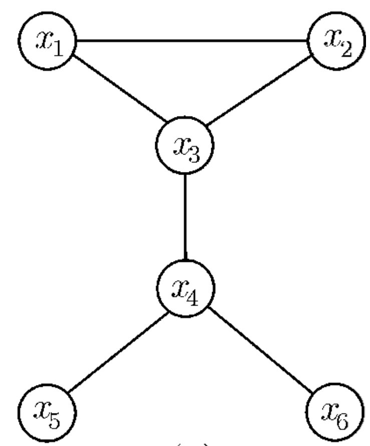
DEFINICIONES:

1. Diremos que G es un **árbol** si G es conexo y acíclico.

EJEMPLO:



(a)



(b)

Es un árbol porque es conexo y no tiene ciclos.

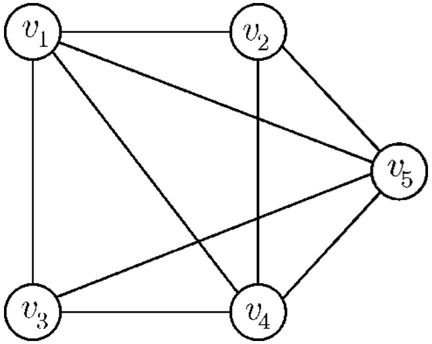
Contiene un ciclo $x_1x_2x_3x_1$ y por tanto no es un árbol.

1. DEFINICIONES. PROPIEDADES Y EJEMPLOS

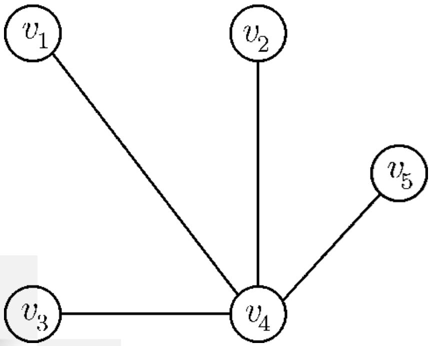
2. Diremos que T es un **árbol generador** de un grafo G si T es árbol y subgrafo generador de G.

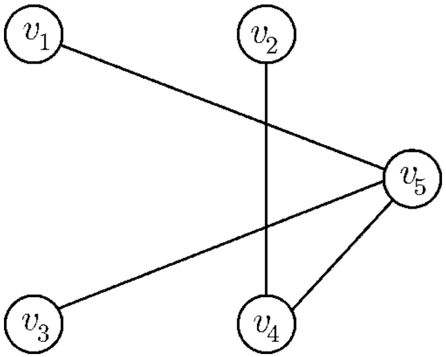
EJEMPLO:

Consideremos el grafo G:



Los siguientes árboles, son árboles generadores de G:





1. DEFINICIONES. PROPIEDADES Y EJEMPLOS

TEOREMA

1. En un árbol dos vértices cualesquiera están unidos por un único camino.

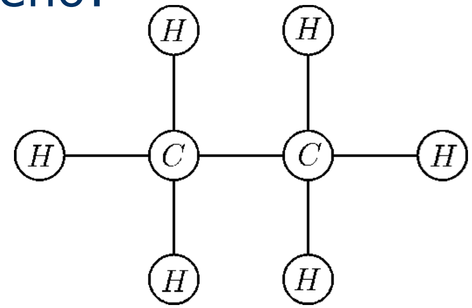
2. Un grafo G es conexo si y sólo si tiene un árbol generador.

3. Si G es un árbol, entonces el número de aristas es igual al número de vértices menos uno.

4. Todo árbol T no trivial (más de 1 vértice) tiene al menos dos vértices de grado 1.

1. DEFINICIONES. PROPIEDADES Y EJEMPLOS

EJEMPLO: Vamos a ver que si un hidrocarburo tiene n átomos de carbono (C), entonces tiene 2n+2 de hidrógeno (H). La siguiente figura muestra un hidrocarburo con dos átomos de carbono y seis de hidrógeno:



Sea k el número de vértices de grado uno o átomos de hidrógeno del árbol. Entonces tenemos un total de n+k vértices y los n átomos de carbono tienen grado 4. Por tanto:

$$4n + k = \sum_{v \in V} d(v) = 2\mathrm{card}(A) = 2(\mathrm{card}(V) - 1) = 2(n + k - 1).$$

Entonces k=2n+2.

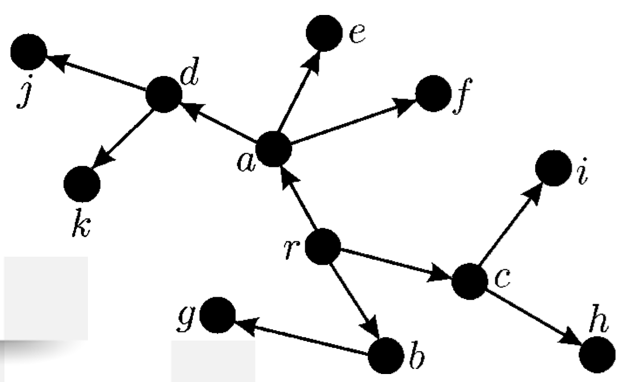
2. ÁRBOLES CON RAIZ O ENRAIZADOS

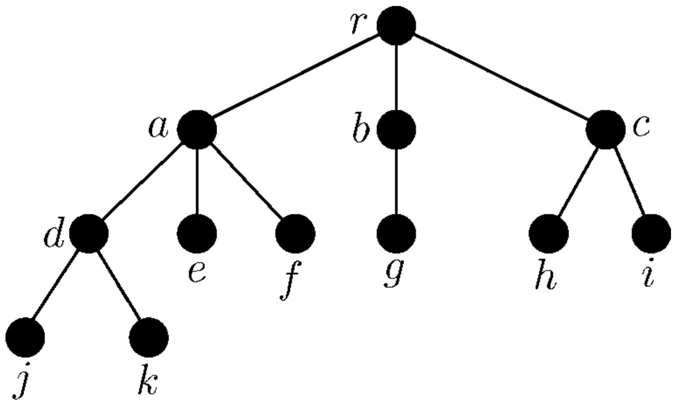
DEFINICIÓN: Sea T un árbol.

Eligiendo un vértice $\mathbf{r_0}$ de T que llamamos raíz, al ser el árbol conexo, todo otro vértice estará conectado con $\mathbf{r_0}$. Podemos entonces definir un grafo dirigido T($\mathbf{r_0}$) donde todos los arcos sean extremos finales de un camino que se inicia en $\mathbf{r_0}$. A este árbol lo llamaremos **árbol enraizado en $\mathbf{r_0}$**.

EJEMPLO: Consideremos el árbol.

Eligiendo el vértice **r** como raíz, obtenemos:



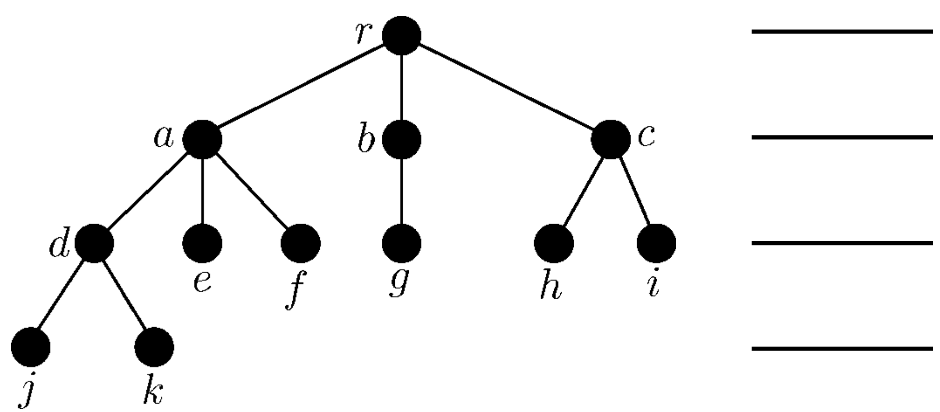


2. ÁRBOLES CON RAIZ O ENRAIZADOS

DEFINICIÓN

Sea T un árbol enraizado y u un vértice de T. Llamamos **nivel** del vértice u a la longitud del camino que va de la raíz a dicho vértice. La **altura** de un árbol es el valor del nivel máximo.

EJEMPLO:



Nivel 0

Nivel 1

Nivel 2

Nivel 3

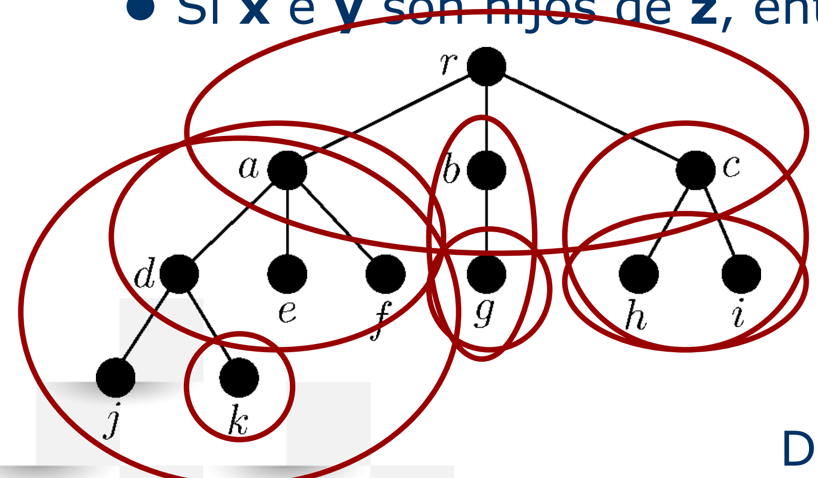
Altura (nivel máximo): 3

2. ÁRBOLES CON RAIZ O ENRAIZADOS

DEFINICIÓN:

Sea T un árbol con raíz $\mathbf{r_0}$. Supongamos que **x, y, z** son vértices de T y que $\mathbf{v_0 v_1 \ldots v_{n-1} v_n}$ es un camino en T. Entonces:

- $\mathbf{v_{n-1}}$ es el **padre** de $\mathbf{v_n}$.
- $\mathbf{v_0, \ldots, v_{n-1}}$ son los **antepasados** de $\mathbf{v_n}$.
- $\mathbf{v_n}$ es el **hijo** de $\mathbf{v_{n-1}}$.
- Si **x** es un antepasado de **y**, entonces **y** es un **descendiente** de **x**.
- Si **x** e **y** son hijos de **z**, entonces **x** e **y** son **hermanos**.



a es el padre de ***d***, ***e***, ***f***

c es el padre de ***h***, ***i***

Los antepasados de ***k*** son ***d***, ***a***, ***r***

g es hijo de ***b***

a, ***b***, ***c*** son hijos de ***r***

h, ***i*** son hermanos

g no tiene hermanos

Descendientes de ***a***: ***d***, ***e***, ***f***, ***j***, ***k***

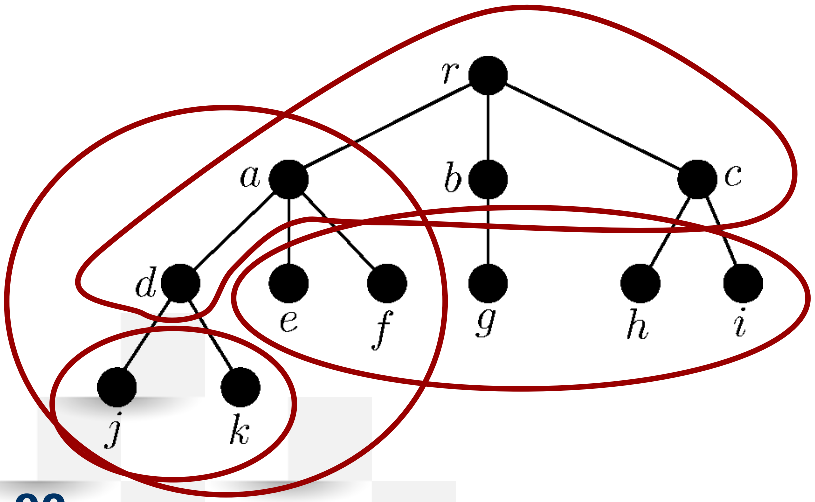
Descendientes de ***r***: todos

2. ÁRBOLES CON RAIZ O ENRAIZADOS

DEFINICIÓN:

Sea T un árbol con raíz $\mathbf{r_0}$. Supongamos que **x, y, z** son vértices de T y que $\mathbf{v_0 v_1 \ldots v_{n-1} v_n}$ es un camino en T. Entonces:

- Si **x** no tiene hijos diremos que es un vértice **terminal**.
- Si **x** no es un vértice terminal diremos que es **interno**.
- El subgrafo de T que consiste en **x** y todos sus descendientes, con **x** como raíz se llama **subárbol de T** que tiene a **x** como raíz.



Vértices terminales: ***j***, ***k***, ***e***, ***f***, ***g***, ***h***, ***i***

Vértices internos: ***d***, ***a***, ***b***, ***c***, ***r***

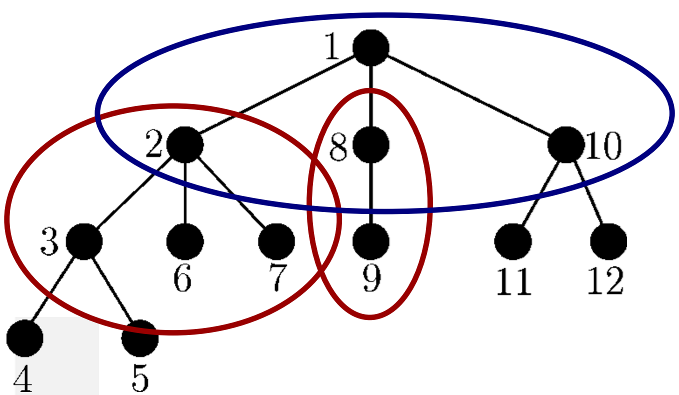
Subárbol de T que tiene al vértice ***a*** como raíz

2. ÁRBOLES CON RAIZ O ENRAIZADOS

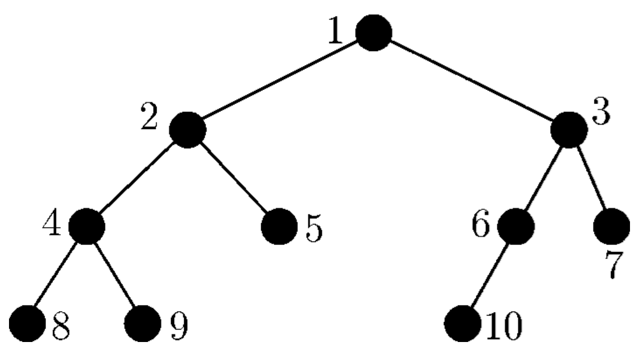
DEFINICIONES:

1. Un **árbol binario** es un árbol enraizado en el cual cada vértice tiene un hijo a la derecha, o un hijo a la izquierda, o un hijo a la derecha y un hijo a la izquierda, o bien ningún hijo.

2. Un **árbol binario completo** es un árbol binario en el que cada vértice tiene un hijo a la derecha y otro a la izquierda o bien ningún hijo.



Árbol enraizado NO binario



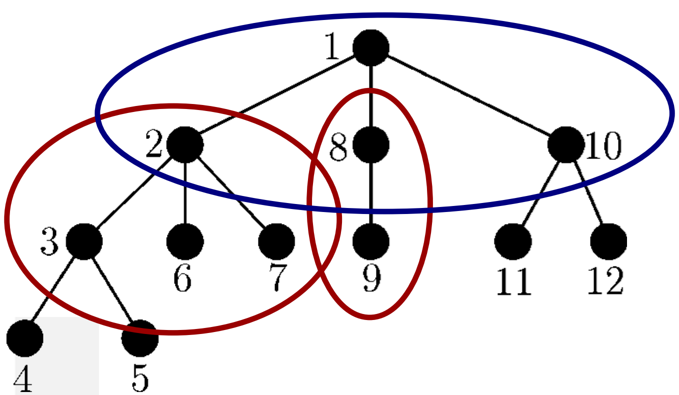
Árbol binario NO completo

2. ÁRBOLES CON RAIZ O ENRAIZADOS

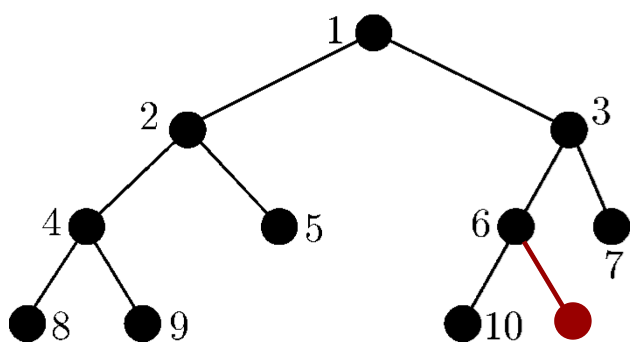
DEFINICIONES:

1. Un **árbol binario** es un árbol enraizado en el cual cada vértice tiene un hijo a la derecha, o un hijo a la izquierda, o un hijo a la derecha y un hijo a la izquierda, o bien ningún hijo.

2. Un **árbol binario completo** es un árbol binario en el que cada vértice tiene un hijo a la derecha y otro a la izquierda o bien ningún hijo.



Árbol enraizado NO binario



Árbol binario completo

2. ÁRBOLES CON RAIZ O ENRAIZADOS

TEOREMA

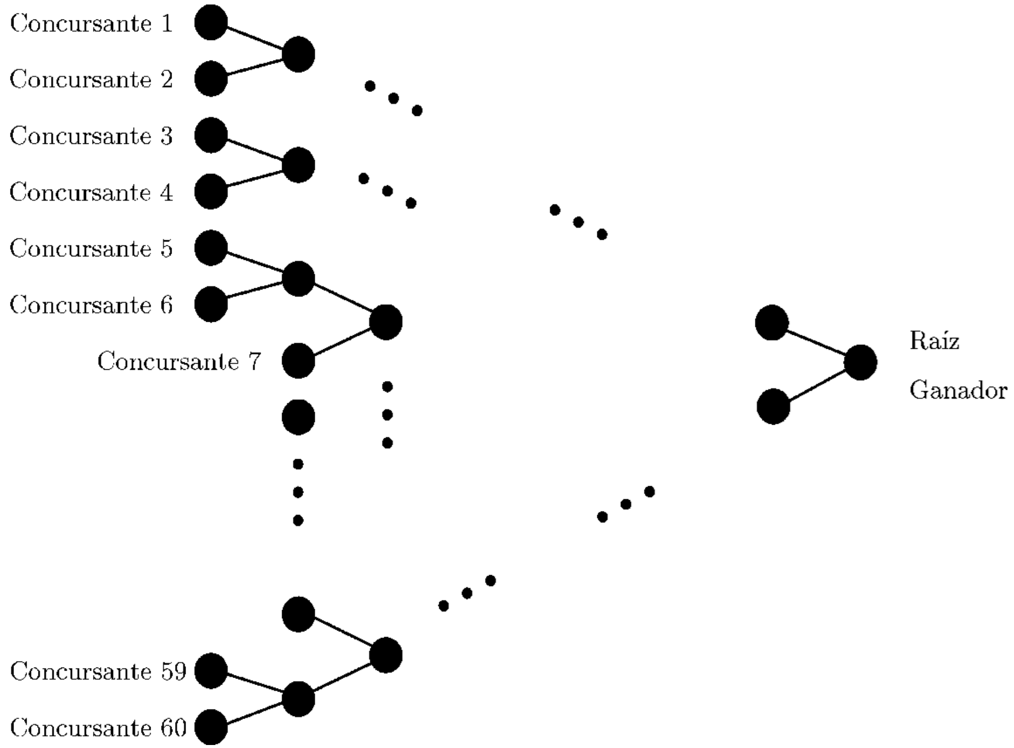
Si T es un árbol binario completo con i vértices internos, entonces T tiene i+1 vértices terminales y 2i+1 vértices en total.

EJEMPLO:

En un torneo de eliminación simple con 60 concursantes

¿cuántos partidos se tienen que jugar?

El grafo que representa los partidos del torneo es un árbol binario de la forma:



2. ÁRBOLES CON RAIZ O ENRAIZADOS

TEOREMA

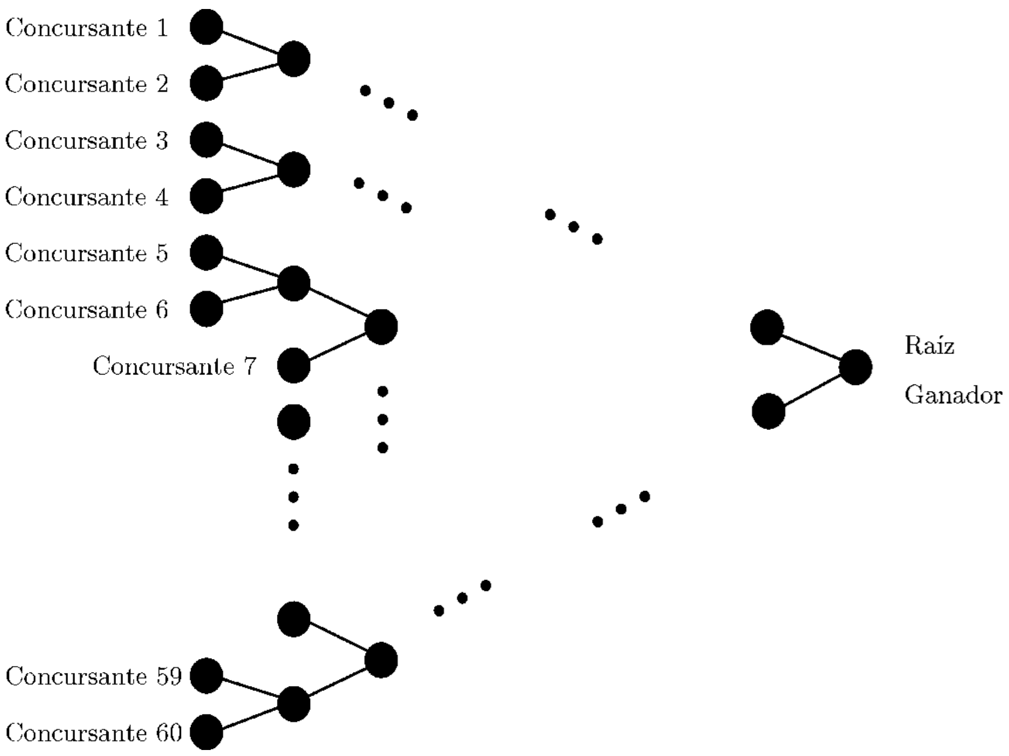
Si T es un árbol binario completo con i vértices internos, entonces T tiene i+1 vértices terminales y 2i+1 vértices en total.

EJEMPLO:

nº de participantes = nº de vértices terminales.

nº de partidos = nº de vértices internos.

El nº de vértices terminales es i+1=60, de modo que i=59 es el número de vértices internos. Es decir, se han de jugar 59 partidos.



2. ÁRBOLES CON RAIZ O ENRAIZADOS

TEOREMA

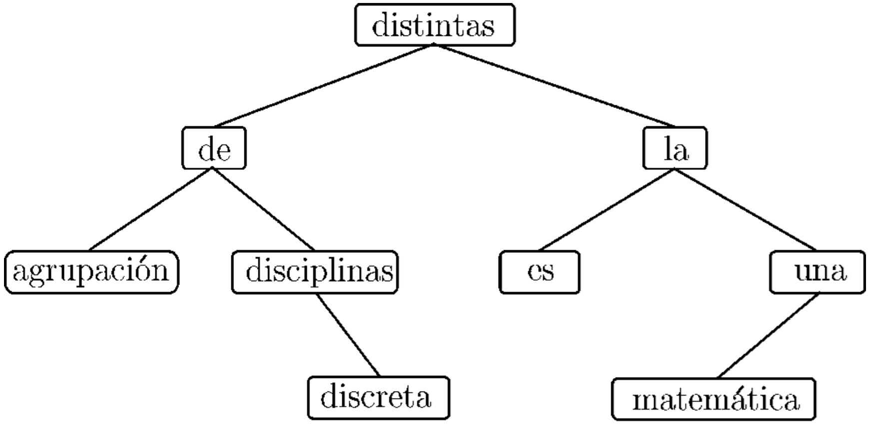
Sea T un árbol binario de altura h y con t vértices terminales, entonces $t \leq 2^h$.

DEFINICIÓN:

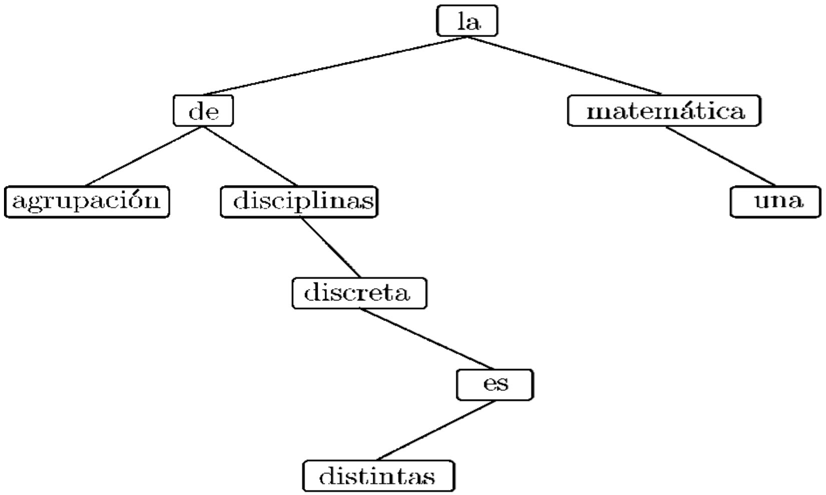
Un **árbol binario de búsqueda** es un árbol binario T en donde se han asociado datos a los vértices. Los datos se disponen de manera que para cualquier vértice **v** en T, cada dato en el subárbol a la izquierda (derecha, respectivamente) de **v** es menor que (mayor que, respectivamente) el dato correspondiente a **v**.

2. ÁRBOLES CON RAIZ O ENRAIZADOS

EJEMPLO: Las palabras de la frase "*La matemática discreta es una agrupación de distintas disciplinas*", se pueden colocar en un árbol binario de búsqueda de múltiples formas:



Árbol binario de búsqueda T_1



Árbol binario de búsqueda T_2

2. ÁRBOLES CON RAIZ O ENRAIZADOS

ALGORITMO DE BÚSQUEDA - NOTACIÓN

Sea T un árbol binario de búsqueda con raíz **RAIZ**.
Si **v** es un vértice, se define:

- IZQUIERDA(**v**) es el hijo a la izquierda de **v**.
- DERECHA(**v**) es el hijo a la derecha de **v**.
- Si **v** no tiene hijos a la izquierda haremos IZQUIERDA(**v**) = λ.
- Si **v** no tiene hijos a la derecha haremos DERECHA(**v**) = λ.
- VALOR(**v**) proporciona el dato asociado al vértice **v**.

2. ÁRBOLES CON RAIZ O ENRAIZADOS

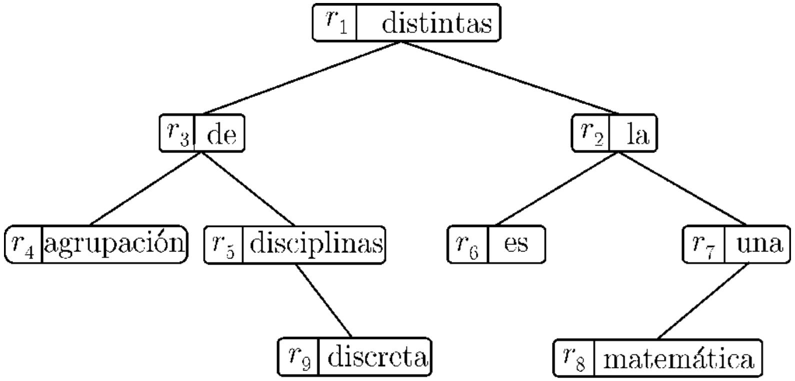
ALGORITMO DE BÚSQUEDA

Para un dato W, este algoritmo proporciona el vértice que contiene a W o λ si el dato no está en el árbol.

- **Paso 1.** P := RAIZ
- **Paso 2.** Si P = λ, STOP.
  En otro caso si VALOR(P) = W, STOP
  (P es el vértice que contiene el dato W.)
- **Paso 3.** Si W > VALOR(P), tómese P :=DERECHA(P), e ir a 2.
  En otro caso, tómese P :=IZQUIERDA(P), e ir a 2.

2. ÁRBOLES CON RAIZ O ENRAIZADOS

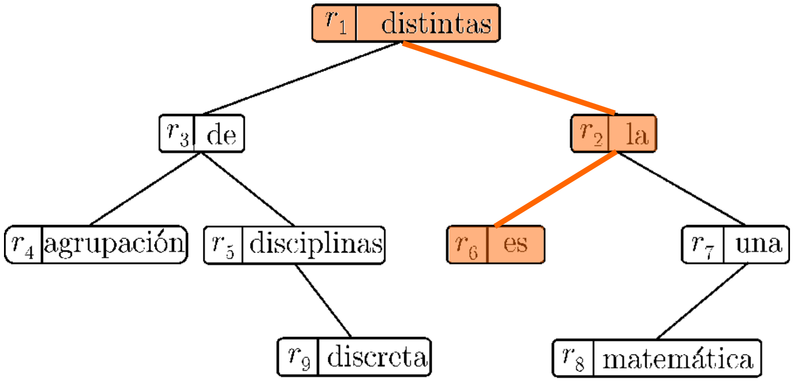
EJEMPLO: Presentamos el siguiente árbol binario de búsqueda en el que se indican, para cada vértice, su nombre y el dato que contiene:



Árbol binario de búsqueda T_1

Supongamos que queremos buscar el dato W="es" en dicho árbol. El algoritmo realizaría los siguientes pasos:

2. ÁRBOLES CON RAIZ O ENRAIZADOS



Árbol binario de búsqueda T_1

- P=r_1.
- Como es > VALOR(r_1)=distintas, tomamos P=DERECHA(r_1)=r_2.
- Como es < VALOR(r_2)=la, tomamos P=IZQUIERDA(r_2)=r_6.
- Como VALOR(r_6)=es, entonces PARAR

P=r_6 es el vértice que contiene el dato "es".

3. ALGORITMOS DE BÚSQUEDA DE PRIMERA PROFUNDIDAD

DEFINICIÓN:

Un **árbol enraizado ordenado** es un árbol enraizado tal que el conjunto de hijos de cada padre está ordenado linealmente de izquierda a derecha.

EJEMPLO:

Todo árbol binario es un árbol enraizado ordenado.

Un algoritmo de recorrido de un árbol es un algoritmo para listar, visitar o buscar todos los vértices de un árbol enraizado ordenado finito. Los tres algoritmos más usuales son los que dan los recorridos preorden, postorden e inorden (este último únicamente para árboles binarios).

3. ALGORITMOS DE BÚSQUEDA DE PRIMERA PROFUNDIDAD

ALGORITMO PREORDEN(v)

Paso1. Listar los subárboles con los hijos de **v** como raíz [Utilizar PREORDEN(**w**) para listar T para cada hijo **w** de **v**].

Paso2. Listar T_v poniendo en sucesión **v** seguido por las listas del paso 1 en el orden de izquierda a derecha. Si **v** no tiene hijos, la lista de T_v es solamente **v**.

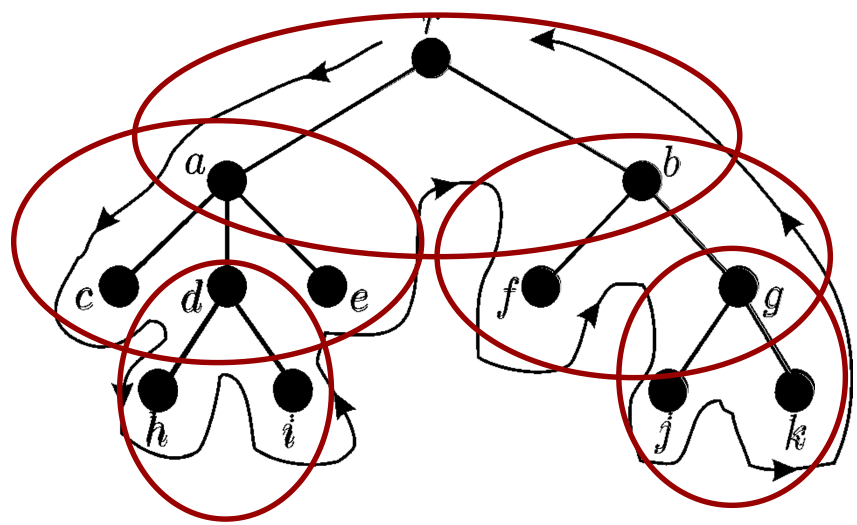
EJEMPLO:

$T_r \equiv \underline{r\ T_a\ T_b}$

$\equiv r\ \underline{a\ T_c\ T_d\ T_e}\ \underline{b\ T_f\ T_g}$

$\equiv r\ a\ c\ \underline{d\ T_h\ T_i}\ e\ b\ f\ \underline{g\ T_j\ T_k}$

$\equiv r\ a\ c\ d\ h\ i\ e\ b\ f\ g\ j\ k$



Índice

3. ALGORITMOS DE BÚSQUEDA DE PRIMERA PROFUNDIDAD

ALGORITMO POSTORDEN(v)

Paso1. Listar los subárboles con los hijos de **v** como raíz

[Utilizar POSTORDEN(**w**) para listar T para cada hijo **w** de **v**].

Paso2. Listar T_v poniendo en sucesión las listas del paso 1 en el orden de izquierda a derecha seguidas por **v**. Si **v** no tiene hijos, la lista de T_v es sólamente **v**.

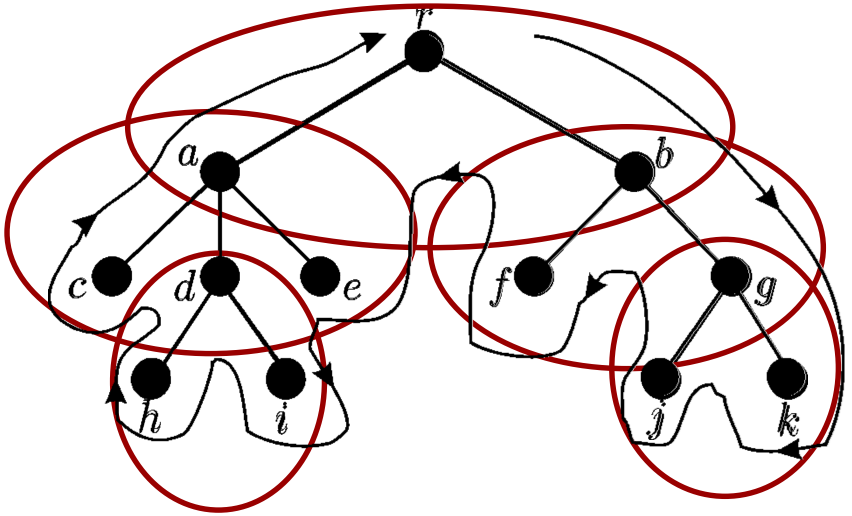
EJEMPLO:

$T_r \equiv \underline{T_a\ T_b}\ r$

$\equiv \underline{T_c\ T_d\ T_e}\ a\ \underline{T_f\ T_g}\ b\ r$

$\equiv c\ \underline{T_h\ T_i}\ d\ e\ a\ f\ \underline{T_j\ T_k}\ g\ b\ r$

$\equiv c\ h\ i\ d\ e\ a\ f\ j\ k\ g\ b\ r$



3. ALGORITMOS DE BÚSQUEDA DE PRIMERA PROFUNDIDAD

ALGORITMO INORDEN(v)

Paso1. Listar el subárbol de la izquierda [Utilizar INORDEN(**w**) para el hijo **w** a la izquierda de **v**].

Paso2. Listar el subárbol de la derecha [Utilizar INORDEN(**w**) para el hijo **w** a la derecha de **v**].

Paso3. Listar T_v poniendo en una sucesión las listas del paso 1, después **v** y luego el resultado del paso 2. Si **v** no tiene hijos, la lista de T_v es solamente **v**.

EJEMPLO:

$T_r \equiv \underline{T_a\ r\ T_b}$

$\equiv \underline{T_c\ a\ T_d}\ r\ \underline{T_e\ b\ T_f}$

$\equiv c\ a\ \underline{T_g\ d\ T_h}\ r\ e\ b\ \underline{T_i\ f\ T_j}$

$\equiv c\ a\ g\ d\ h\ r\ e\ b\ i\ f\ j$