LLiçó 3. ARBRES

1. Definicions. Propietats i exemples.
2. Arbres amb arrel o arrelats.
3. Algoritmes de recerca de primera profunditat.

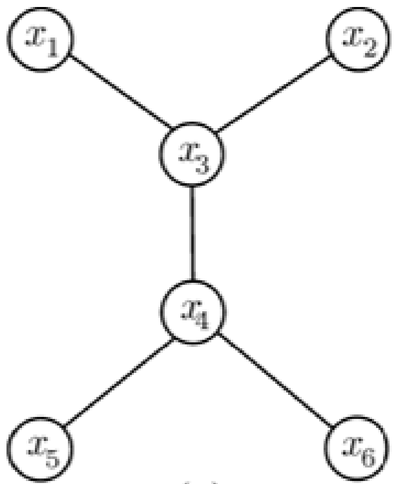
1. DEFINICIONS. PROPIETATS I EXEMPLES

Siga *G* un graf no dirigit.

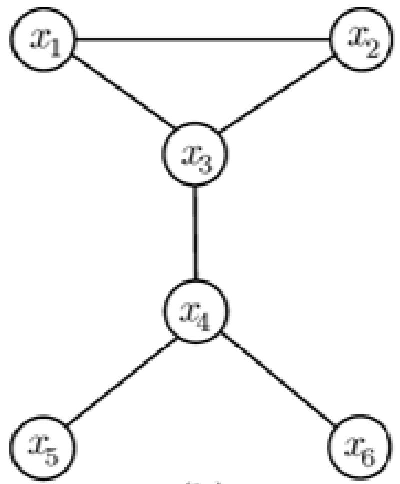
DEFINICIONS:

1. Direm que *G* és un **arbre** si *G* és connex i acíclic.

EXEMPLE:



(a)



(b)

És un arbre perquè és connex i no té cicles.

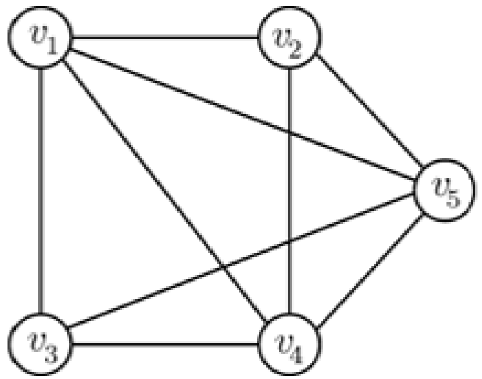
Conté un cicle $\boldsymbol{x_1 x_2 x_3 x_1}$ i per tant no és un arbre.

1. DEFINICIONS. PROPIETATS I EXEMPLES

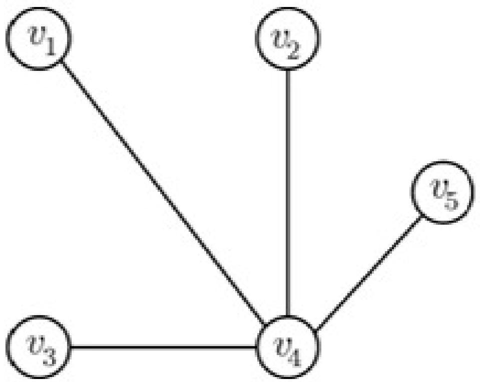
2. Direm que T és un **arbre generador** d'un graf G si T és arbre i subgraf generador de G.

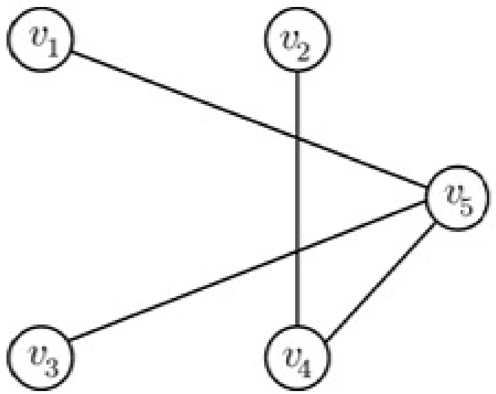
EXEMPLE:

Considerem el graf G:



Els següents arbres, són arbres generadors de G:





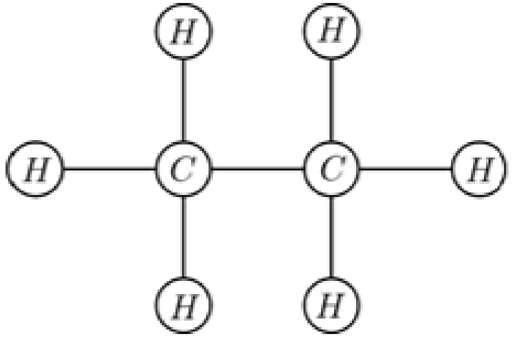
1. DEFINICIONS. PROPIETATS I EXEMPLES

TEOREMA

1. En un arbre dos vèrtexs qualssevol estan units per un únic camí.
2. Un graf G és connex si i només si té un arbre generador.
3. Si G és un arbre, aleshores el nombre d'arestes és igual al nombre de vèrtexs menys 1.
4. Tot arbre T no trivial (més d'1 vèrtex) té almenys dos vèrtexs de grau 1.

1. DEFINICIONS. PROPIETATS I EXEMPLES

EXEMPLE: Veurem que si un hidrocarbur té n àtoms de carboni (C), llavors té $2n+2$ d'hidrogen (H). La següent figura mostra un hidrocarbur amb dos àtoms de carboni i sis d'hidrogen:



Siga k el nombre de vèrtexs de grau 1 o àtoms d'hidrogen de l'arbre. Aleshores, tenim un total de $n+k$ vèrtexs i els n àtoms de carboni tenen grau 4. Per tant:

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

Aleshores, $k=2n+2$.

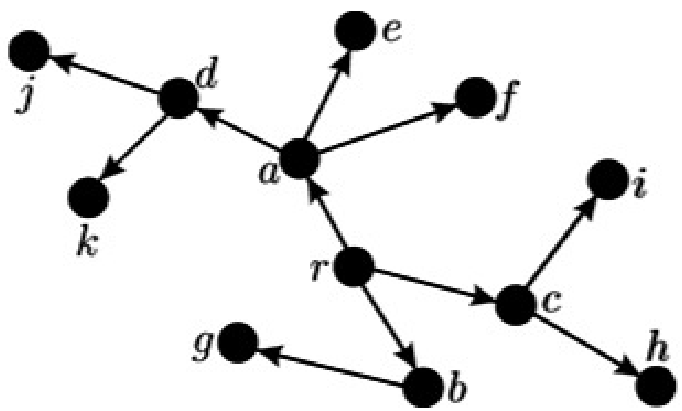
2. ARBRES AMB ARREL O ARRELATS

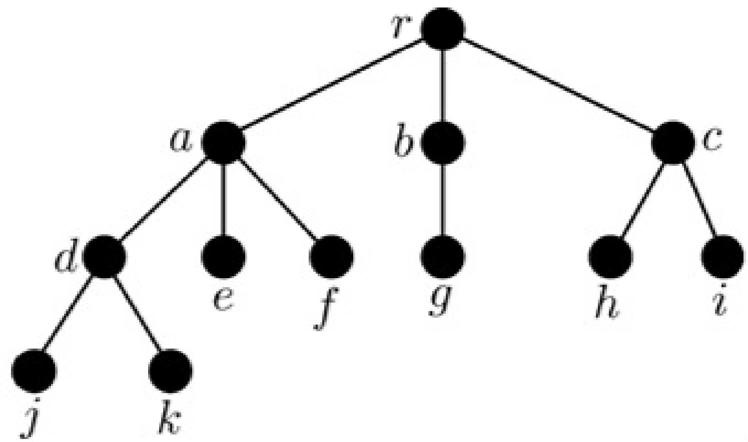
DEFINICIÓ: Siga T un arbre.

Triant un vèrtex $\mathbf{r}_0$ de T que anomenem **arrel**, al ser l'arbre connex, tot altre vèrtex estarà connectat amb $\mathbf{r}_0$. Aleshores, podem definir un graf dirigit T($\mathbf{r}_0$) on tots els arcs siguen extrems finals d'un camí que s'inicia en $\mathbf{r}_0$. A aquest arbre l'anomenarem **arbre arrelat en $\mathbf{r}_0$**.

EXEMPLE: Considerem l'arbre.

Triant el vèrtex **r** com a arrel, obtenim:



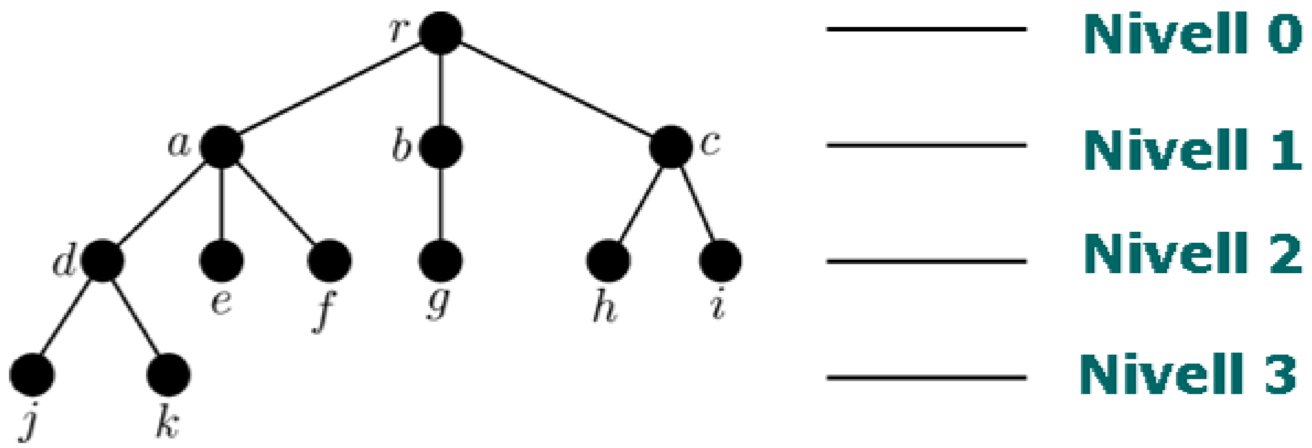


2. ARBRES AMB ARREL O ARRELATS

DEFINICIÓ

Siga T un arbre arrelat i *u* un vèrtex de T. Anomenem **nivell** del vèrtex *u* a la longitud del camí que va de l'arrel a aquest vèrtex. **L'altura** d'un arbre és el valor del nivell màxim.

EXEMPLE:



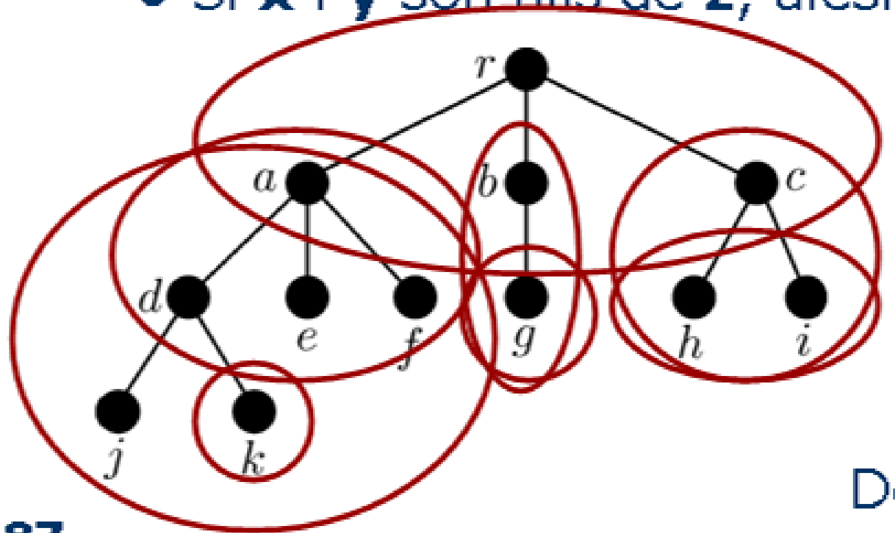
Altura (nivell màxim): 3

2. ARBRES AMB ARREL O ARRELATS

DEFINICIÓ:

Siga T un arbre amb arrel $\mathbf{r}_0$. Suposem que **x, y, z** són vèrtexs de T i que $\mathbf{v}_0\mathbf{v}_1 \ldots \mathbf{v}_{n-1}\mathbf{v}_n$ és un camí en T. Aleshores:

- $\mathbf{v}_{n-1}$ es el **pare** de $\mathbf{v}_n$.
- $\mathbf{v}_0, \ldots, \mathbf{v}_{n-1}$ són els **avantpassats** de $\mathbf{v}_n$.
- $\mathbf{v}_n$ es el **fill** de $\mathbf{v}_{n-1}$.
- Si **x** es un avantpassat de **y**, aleshores **y** és un **descendent** de **x**.
- Si **x** i **y** son fills de **z**, aleshores **x** i **y** son **germans**.



a és el pare de ***d***, ***e***, ***f***

c és el pare de ***h***, ***i***

Els avantpassats de ***k*** són ***d***, ***a***, ***r***

g es fill de ***b***

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

h, ***i*** son germans

g no té germans

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

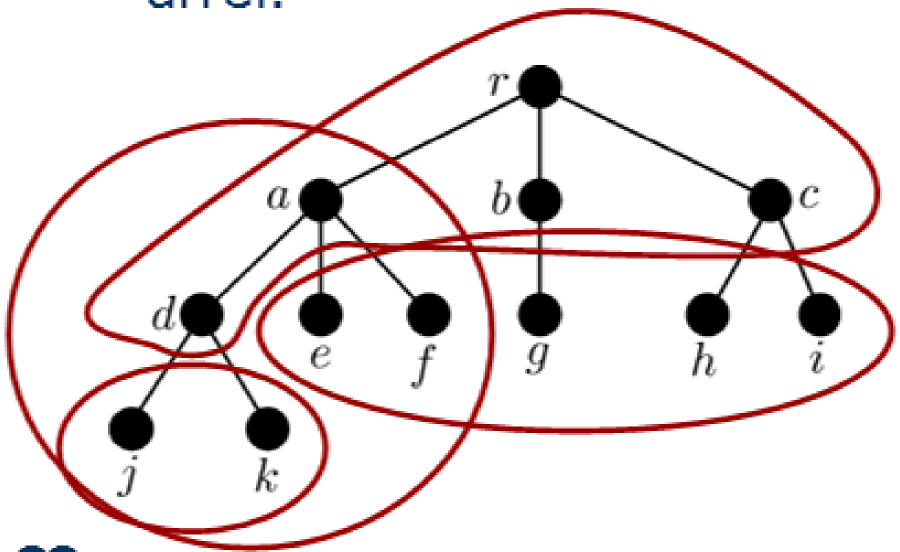
Descendents de ***r***: tots

2. ARBRES AMB ARREL O ARRELATS

DEFINICIÓ:

Siga T un arbre amb arrel $\mathbf{r}_0$. Suposem que **x, y, z** són vèrtexs de T i que $\mathbf{v}_0\mathbf{v}_1 \ldots \mathbf{v}_{n-1}\mathbf{v}_n$ és un camí en T. Aleshores:

- Si **x** no té fills direm que és un vèrtex **terminal**.
- Si **x** no és un vèrtex terminal direm que és **intern**.
- El subgraf de T que consisteix en **x** i tots els seus descendents, amb **x** com a arrel s'anomena **subarbre de T** que té a **x** com a arrel.



Vèrtexs terminals: ***j*, *k*, *e*, *f*, *g*, *h*, *i***

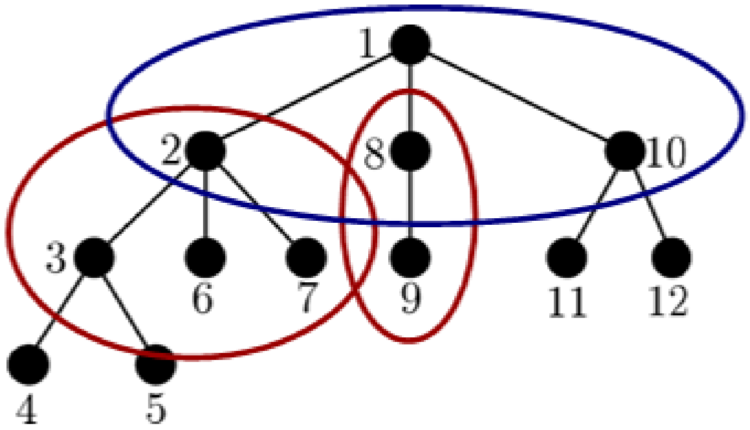
Vèrtexs interns: ***d*, *a*, *b*, *c*, *r***

Subarbre de T que té al vèrtex **a** com a arrel

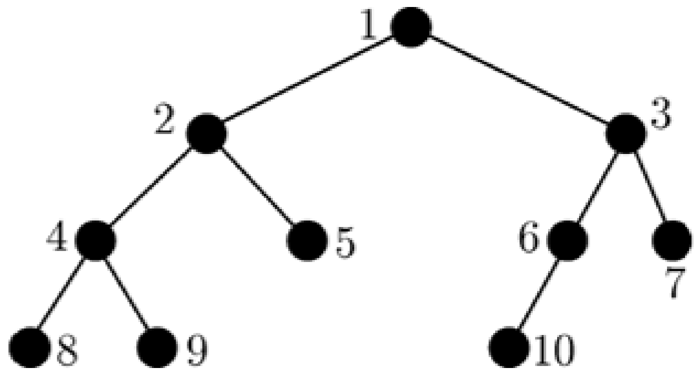
2. ARBRES AMB ARREL O ARRELATS

DEFINICIONS:

1. Un **arbre binari** és un arbre arrelat en el qual cada vèrtex té un fill a la dreta, o un fill a l'esquerra, o un fill a la dreta i un fill a l'esquerra, o bé cap fill.
2. Un **arbre binari complet** és un arbre binari en el qual cada vèrtex té un fill a la dreta i altre a l'esquerra o bé cap fill.



Arbre arrelat NO binari

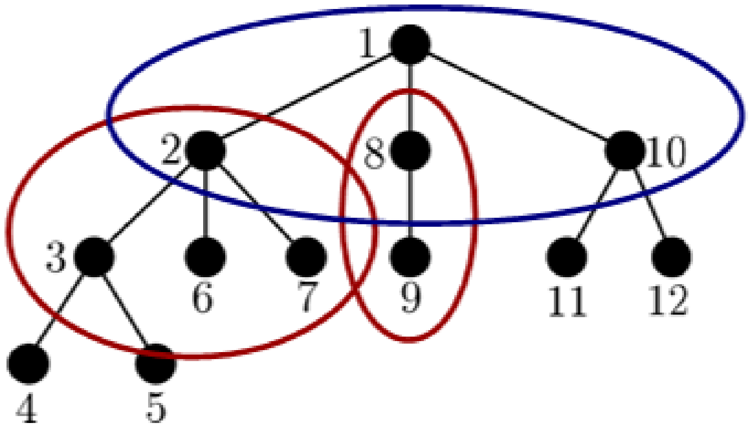


Arbre binari NO complet

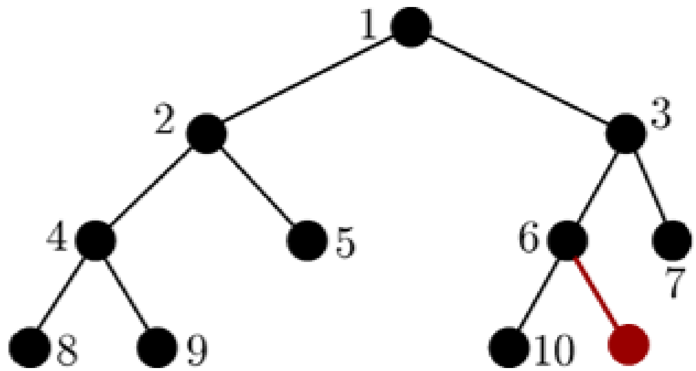
2. ARBRES AMB ARREL O ARRELATS

DEFINICIONS:

1. Un **arbre binari** és un arbre arrelat en el qual cada vèrtex té un fill a la dreta, o un fill a l'esquerra, o un fill a la dreta i un fill a l'esquerra, o bé cap fill.
2. Un **arbre binari complet** és un arbre binari en el qual cada vèrtex té un fill a la dreta i altre a l'esquerra o bé cap fill.



Arbre arrelat NO binari



Arbre binari complet

Índex

2. ARBRES AMB ARREL O ARRELATS

TEOREMA

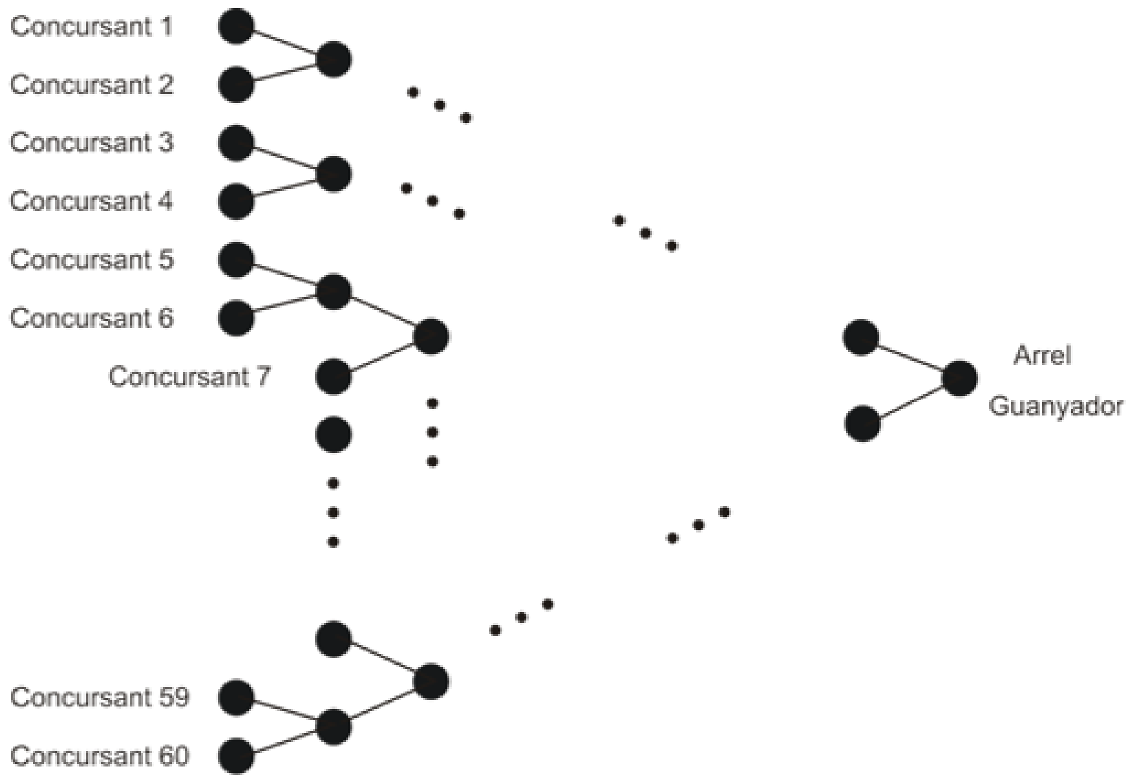
Si T és un arbre binari complet amb *i* vèrtexs interns, aleshores T té *i+1* vèrtexs terminals i *2i+1* vèrtexs en total.

EXEMPLE:

En un torneig d'eliminació simple amb 60 concursants,

quants partits s'han de jugar?

El graf que representa els partits del torneig és un arbre binari de la forma:



2. ARBRES AMB ARREL O ARRELATS

TEOREMA

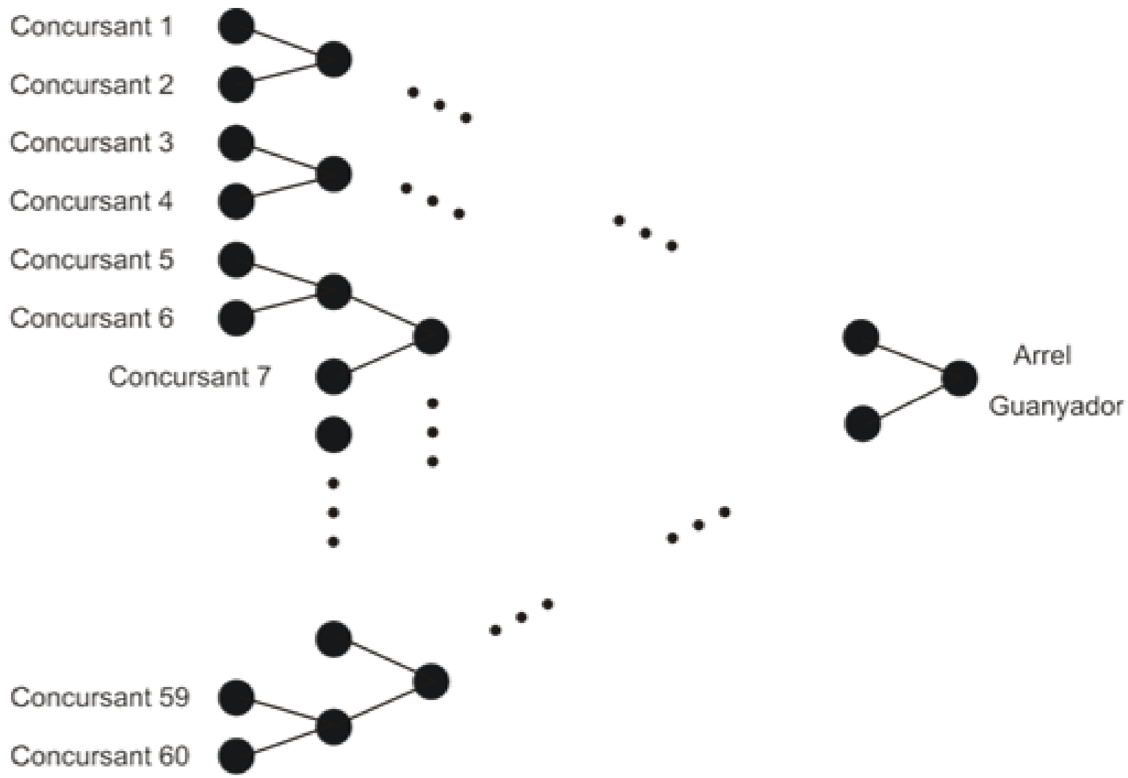
Si T és un arbre binari complet amb i vèrtexs interns, aleshores T té *i+1* vèrtexs terminals i *2i+1* vèrtexs en total.

EXEMPLE:

nombre de participants = nombre de vèrtex terminals.

nombre de partits = nombre de vèrtex interns

El nombre de vèrtex terminals és i+1=60, de manera que i=59 és el nombre de vèrtex interns. És a dir, s'han de jugar 59 partits.



2. ARBRES AMB ARREL O ARRELATS

TEOREMA

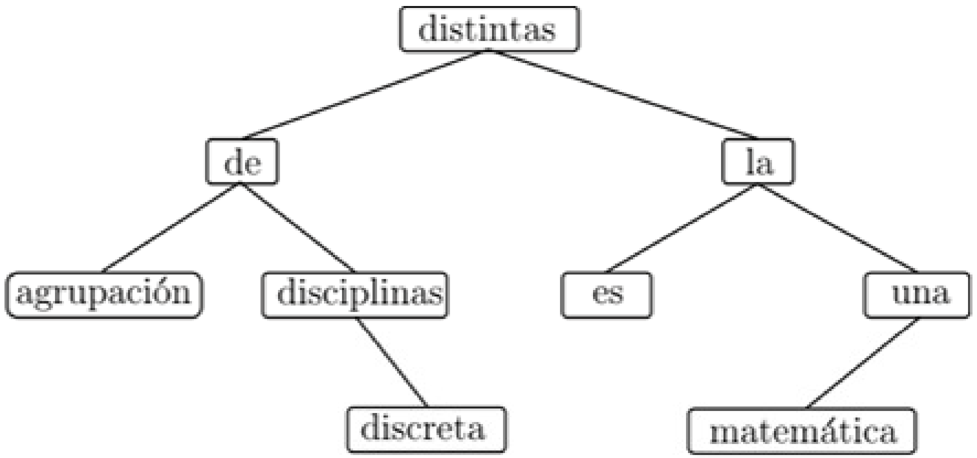
Siga T un arbre binari d'altura h i amb t vèrtex terminals, aleshores $t \leq 2^h$.

DEFINICIÓ:

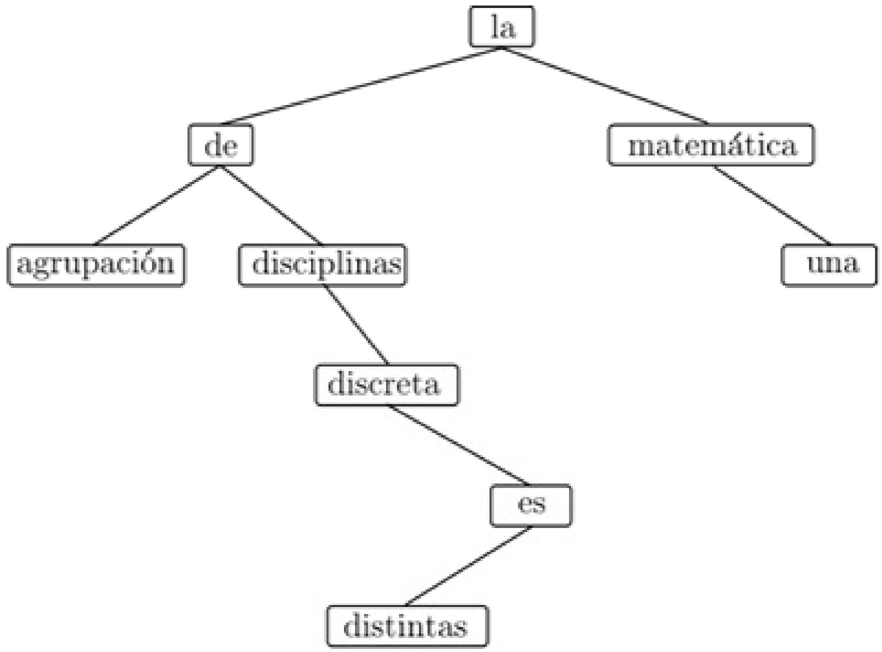
Un **arbre binari de recerca** és un arbre binari T on s'han associat dades als vèrtexs. Les dades es disposen de manera que per a qualsevol vèrtex **v** en T, cada dada en el subarbre a l'esquerra (dreta, respectivament) de **v** és menor que (major que, respectivament) la dada corresponent a **v**.

2. ARBRES AMB ARREL O ARRELATS

EXEMPLE: Les paraules de la frase "La matemàtica discreta és una agrupació de distintes disciplines", es poden col·locar en un arbre binari de recerca de múltiples formes:



Arbre binari de recerca T_1



Arbre binari de recerca T_2

2. ARBRES AMB ARREL O ARRELATS

ALGORITME DE RECERCA - NOTACIÓ

Siga T'un arbre binari de recerca amb arrel **ARREL**.
Si **v** és un vèrtex, es defineix:

- ESQUERRA(**v**) és el fill a l'esquerra de **v**.
- DRETA(**v**) es el fill a la dreta de **v**.
- Si **v** no té fills a l'esquerra farem ESQUERRA(**v**) = λ.
- Si **v** no té fills a la dreta farem DRETA(**v**) = λ.
- VALOR(**v**) proporciona la dada associada al vèrtex **v**.

2. ARBRES AMB ARREL O ARRELATS

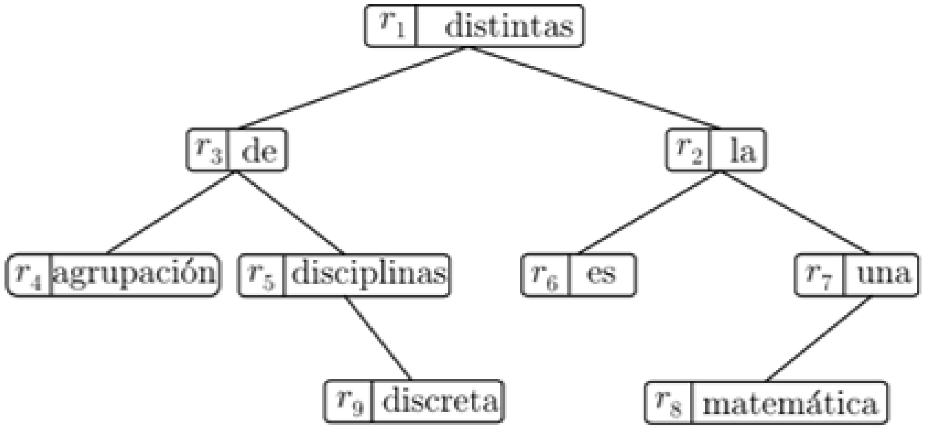
ALGORITME DE RECERCA

Per a una dada W, aquest algoritme proporciona el vèrtex que conté a W o λ si la dada no està en l'arbre.

- **Pas 1.** P := ARREL
- **Pas 2.** Si P = λ, STOP.
  En qualsevol altre cas si VALOR(P) = W, STOP
  (P és el vèrtex que conté la dada W)
- **Pas 3.** Si W > VALOR(P), prenga's P :=DRETA(P), i anar a 2.
  En qualsevol altre cas , es pren P :=ESQUERRA(P), i anar a 2.

2. ARBRES AMB ARREL O ARRELATS

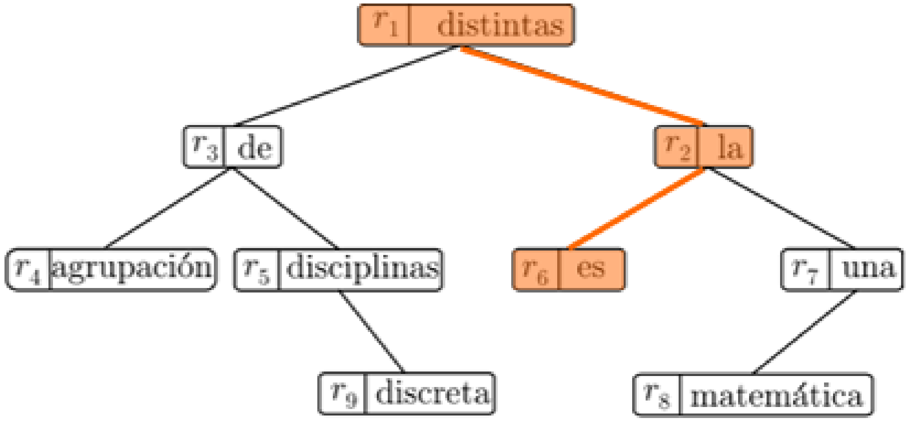
EXEMPLE: Presentem el següent arbre binari de recerca en què s'indiquen, per a cada vèrtex, el seu nom i la dada que conté:



Arbre binari de recerca T_1

Suposem que volem buscar la dada W="es" en l'arbre. L'algoritme realitzaria els passos següents:

2. ARBRES AMB ARREL O ARRELATS



Arbre binari de recerca T_1

- $P=r_1$.
- Com es > VALOR(r_1)=distintas, prenem P=DRETA(r_1)=r_2.
- Com es < VALOR(r_2)=la, prenem P=ESQUERRA(r_2)=r_6.
- Com VALOR(r_6)=es, aleshores PARAR

$P=r_6$ és el vèrtex que conté la dada “es”.

3. ALGORITMES DE BUSCA DE PRIMERA PROFUNDITAT

DEFINICIÓ:

Un **arbre arrelat ordenat** és un arbre arrelat tal que el conjunt de fills de cada pare està ordenat linealment d'esquerra a dreta.

EXEMPLE:

Tot arbre binari és un arbre arrelat ordenat.

Un algoritme de recorregut d'un arbre és un algoritme per a llistar, visitar o buscar tots els vèrtexs d'un arbre arrelat ordenat finit. Els tres algoritmes més usuals són els que donen els recorreguts preorde, postordre i inordre (aquest últim únicament per a arbres binaris).

3. ALGORITMES DE BUSCA DE PRIMERA PROFUNDITAT

ALGORITME PREORDE(v)

Pas 1. Llistar els subarbres amb els fills de **v** com a arrel [Utilitzar PREORDRE(**w**) per a llistar T per a cada fill **w** de **v**].

Pas 2. Llistar T_v posant en successió **v** seguit per les llistes del pas 1 en l'orde d'esquerra a dreta. Si **v** no té fills, la llista de T_v és només **v**.

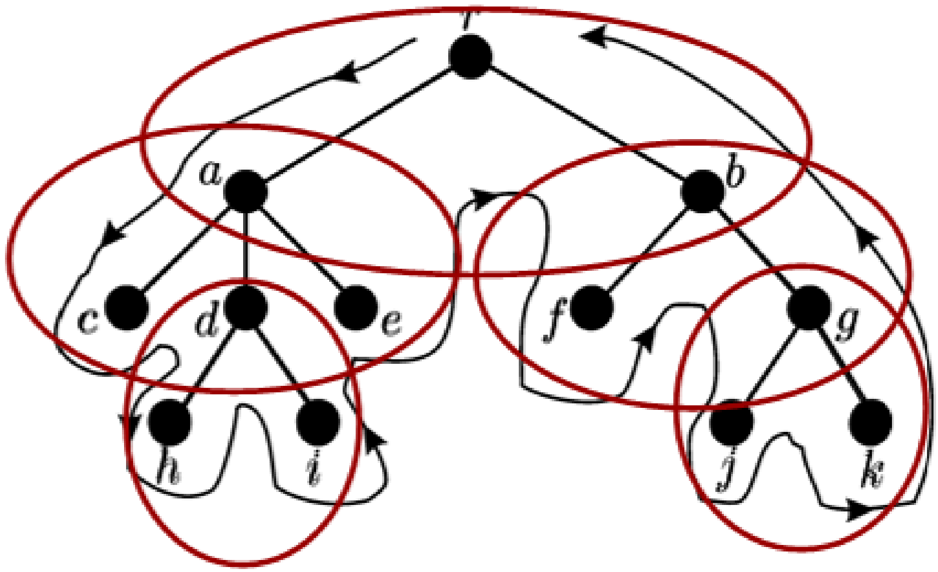
EXEMPLE:

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

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

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

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



3. ALGORITMES DE BUSCA DE PRIMERA PROFUNDITAT

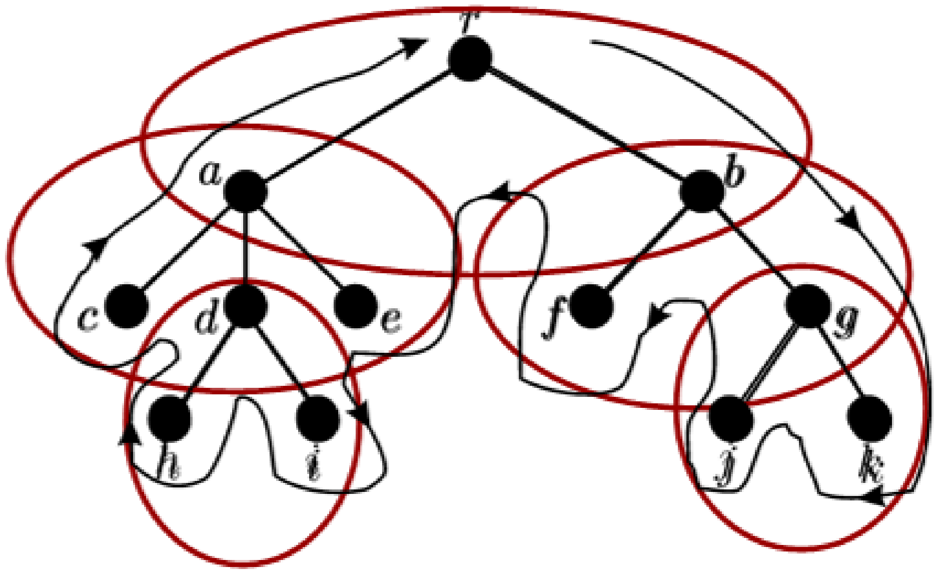
ALGORITME POSTORDRE(v)

Pas 1. Llistar els subarbres amb els fills de **v** com a arrel [Utilitzar POSTORDRE(**w**) per a llistar T per a cada fill **w** de **v**].

Pas 2. Llistar T_v posant en successió les llistes del pas 1 en l'orde d'esquerra a dreta seguides per **v**. Si **v** no té fills, la llista de T_v és només **v**.

EXEMPLE:

$$\begin{aligned} T_r &\equiv T_a\ T_b\ r \\ &\equiv T_c\ T_d\ T_e\ a\ T_f\ T_g\ b\ r \\ &\equiv c\ T_h\ T_i\ d\ e\ a\ f\ T_j\ T_k\ g\ b\ r \\ &\equiv c\ h\ i\ d\ e\ a\ f\ j\ k\ g\ b\ r \end{aligned}$$



3. ALGORITMES DE BUSCA DE PRIMERA PROFUNDITAT

ALGORITME INORDE(v)

Pas 1. Llistar el subarbre de l'esquerra [Utilitzar INORDRE(**w**) per al fill **w** a l'esquerra de **v**].

Pas 2. Llistar el subarbre de la dreta [Utilitzar INORDRE(**w**) per al fill **w** a la dreta de **v**].

Pas 3. Llistar T_v posant en successió les llistes del pas 1, després **v** i després el resultat del pas 2. Si **v** no té fills, la llista de T_v es només **v**.

EXEMPLE:

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

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

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

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