TEMA 3
El tipus arbre

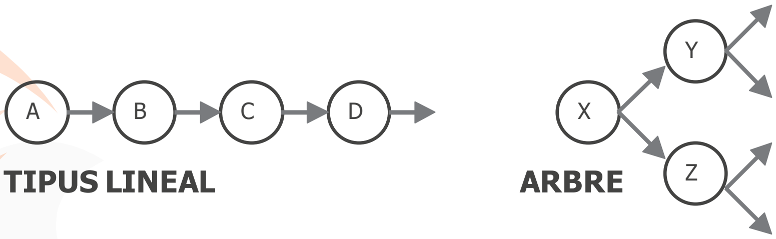
PROGRAMACIÓ I ESTRUCTURES DE DADES

Tipus arbre

- 1. Definicions generals
- 2. Arbres binaris
- 3. Arbres de cerca
  - 3.1. Arbres binaris de cerca
  - 3.2. Arbres AVL
  - 3.3. Arbres 2-3
  - 3.4. Arbres 2-3-4

1. Definicions generals (I)

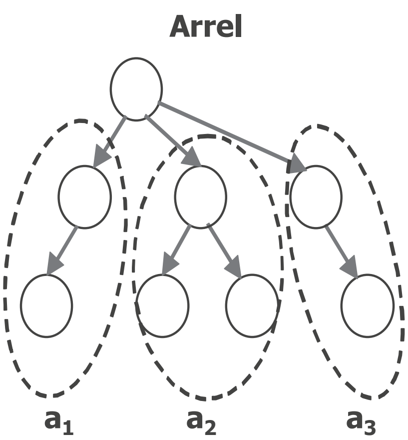
- L'estructura de dades arbre apareix perquè els elements que el constitueixen mantenen una estructura jeràrquica, obtinguda a partir d'estructures lineals, quan eliminem el requisit de que cada element té com a màxim un successor:



- Els elements dels arbres es diuen nodes

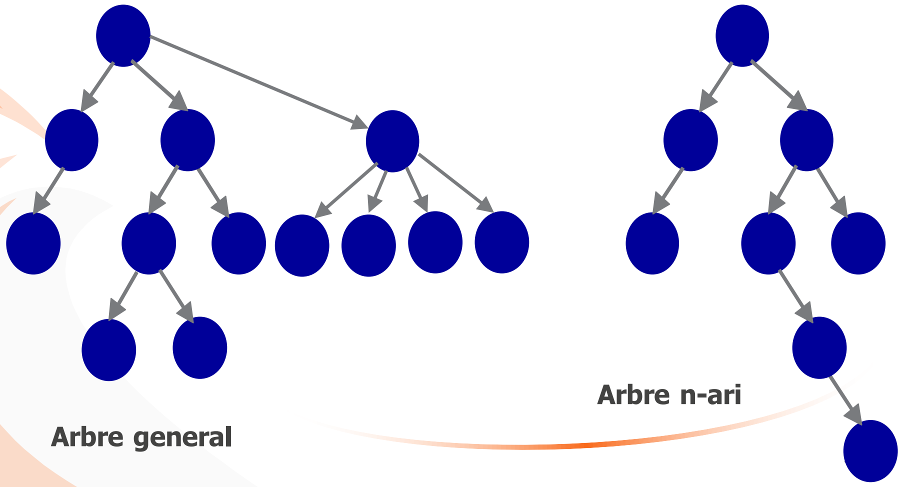
1. Definicions generals (II)

- Definició inductiva d'arbre:
  - un únic node és un arbre (**arrel**)
  - donats *n* arbres $a_1, \ldots, a_n$ es pot construir uno nou com resultat d'arrelar un nou node amb els *n arbres*. Els **arbres** a_i passen a ser **subarbres** del nou arbre i el nou node es converteix en arrel del nou arbre
- Arbre buit o nul $\Rightarrow$ 0 nodes



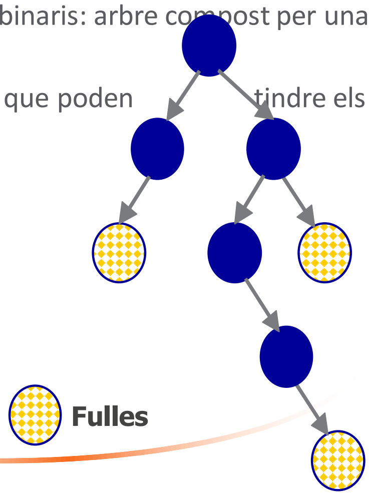
1. Definicions generals (III)

- El proces d'arrelar pot involucrar:
  - un nombre indeterminat de subarbres (**arbres generals**)
  - o bé, un nombre màxim *n* de subarbres (**arbres n-aris**)



1. Definicions generals (IV)

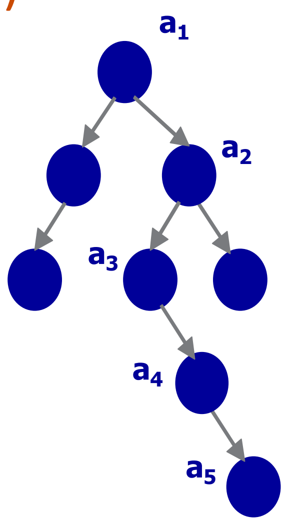
- Un arbre n-ari amb *n* = 2 es denomina **arbre binari**
- La informació emmagatzemada en els nodes de l'arbre es denomina **etiqueta**
- Les **fulles** són arbres amb un sol node (arbres binaris: arbre compost per una arrel i 2 subarbres buits)
- El **grau** d'un arbre és el nombre màxim de fills que poden tindre els seus subarbres (si l'arbre es *n*-ari, el grau és *n*)



1. Definicions generals (V)

- Camí:
  - és una seqüència $a_1, ..., a_s$ d'arbres tal que $\forall\ i \in \{1... s - 1\}$, a_{i+1} és subarbre de a_i
  - el nombre de subarbres de la seqüència menys un, es denomina **longitud del camí**

    (Considerarem que hi ha un camí de longitud 0 de tot subarbre a si mateix)



$\forall\ i \in \{1... 4\}$ a_{i+1} és subarbre de a_i

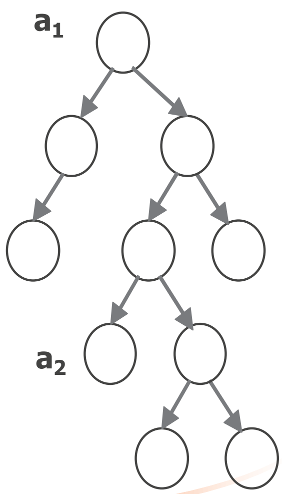
Longitud = 5 - 1 = 4

1. Definicions generals (VI)

- a_1 es ascendent de a_2 (i a_2 és descendent de a_1) si hi ha un camí $a_1, ..., a_2$

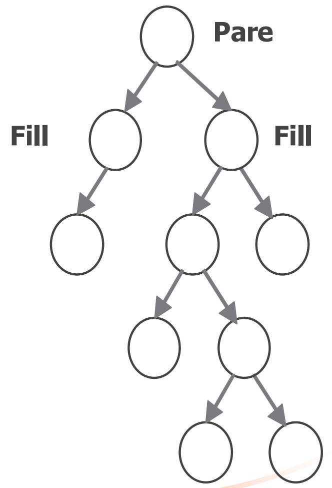
  (Segons la definició de camí, tot subarbre és ascendent/descendent de si mateix)

- Els ascendents (descendents) d'un arbre, exclòs l'arbre mateix, es denominen ascendents (descendents) propis



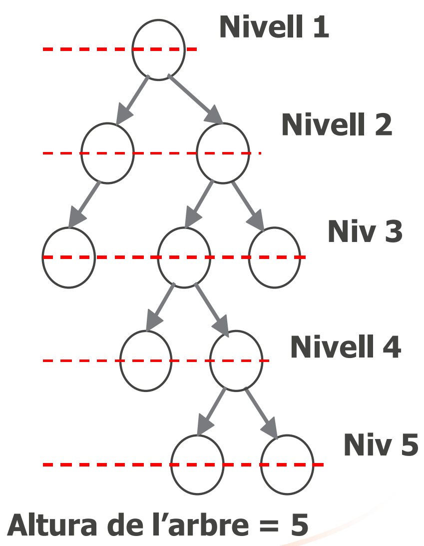
1. Definicions generals (VII)

- **Pare** és el primer ascendent propi, si existeix, d'un arbre
- **Fills** són els primers descendents propis, si existeixen, d'un arbre
- **Germans** són subarbres amb el mateix pare
- **Profunditat** d'un subarbre és la longitud de l'**únic** camí desde l'arrel a tal subarbre



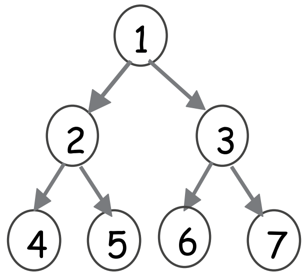
1. Definicions generals (VIII)

- Nivell d'un node:
  - el nivell d'un arbre buit és 0
  - el nivell de l'arrel és 1
  - si un node està en el nivell *i*, els seus fills están en el nivell *i + 1*
- **Altura** (profunditat) d'un arbre:
  - és el màxim nivell dels nodes d'un arbre

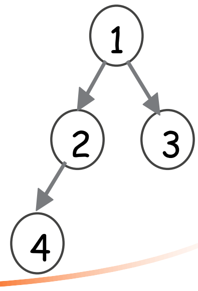


1. Definicions generals (IX)

- **Arbre ple** és un arbre en què tots els subarbres tenen *n* fills (sent *n* el grau de l'arbre) i totes les fulles tenen la mateixa profunditat



- **Arbre complet** és un arbre els nodes del qual corresponen als nodes numerats (la numeració es fa des de l'arrel cap a les fulles i, en cada nivell, d'esquerra a dreta) d'1 a *n* en l'arbre ple del mateix grau. **Tot arbre ple és complet**



2. Arbres binaris

- Definició d'arbre binari i propietats
- Especificació algebraica
- Recorreguts
- Enriquiment de l'especificació
- Representació seqüencial i enllaçada
- Altres operacions interessants
- Exercicis

2. Arbres binaris

DEFINICIÓ

- Un arbre binari és un conjunt d'elements del mateix tipus tal que:
  - o be és el conjunt buit, i en aquest cas es denomina **arbre buit** o **nul**
  - o bé no és buit i, per tant, hi ha un element únic anomenat **arrel**, i la resta dels elements es distribueixen en dos subconjunts disjunts, cada un dels quals és un arbre binari anomenat, respectivament **subarbre esquerre i subarbre dret** de l'arbre original

2. Arbres binaris

PROPIETATS (I)

- Propietats:
  - El màxim nombre de nodes en un nivell *i* d'un arbre binari es

$N(i) = 2^{i-1}$, $i \geq 1$

Demostració

Base inducció
nivell 1 (arrel): $N(1) = 2^{1-1} = 2^0 = 1$ (es compleix)

Pas inductiu

Es vol provar $N(i\text{-}1) \Rightarrow N(i)$, és a dir, a partir de la suposició "temporal" que N és certa per a *i-1,* hem de provar que és certa per a *i*

nivell *i - 1*: $N(i\text{-}1) = 2^{(i-1)-1} = 2^{i-2}$ (suposem cert)

nivell *i* : $\mathbf{N(i)} = N(i\text{-}1) * 2 = 2^{i-2} * 2 = 2^{i-2+1} = \mathbf{2^{i-1}}$

2. Arbres binaris

PROPIETATS (II)

– El máxim nombre de nodes en un arbre binari d'altura k és $N(k) = 2^k - 1,\ k \geq 1$

Demostració

nivell 1: $2^{1-1} = 1$ node

nivell 2: $2^{2-1} = 2$ nodes

nivell 3: $2^{3-1} = 4$ nodes

. . .

nivell k: 2^{k-1} nodes

Altura k = $2^{1-1} + 2^{2-1} + \ldots + 2^{k-1}$ = [S P G. ($r = 2$, $a_1 = 2^0$, $n = k$)]

$= 1\ (2^k - 1) / 2 - 1 = 2^k - 1$

Suma progressió geomètrica

$S_n = a_1\ ((r^n-1)\ /(r-1))$

2. Arbres binaris

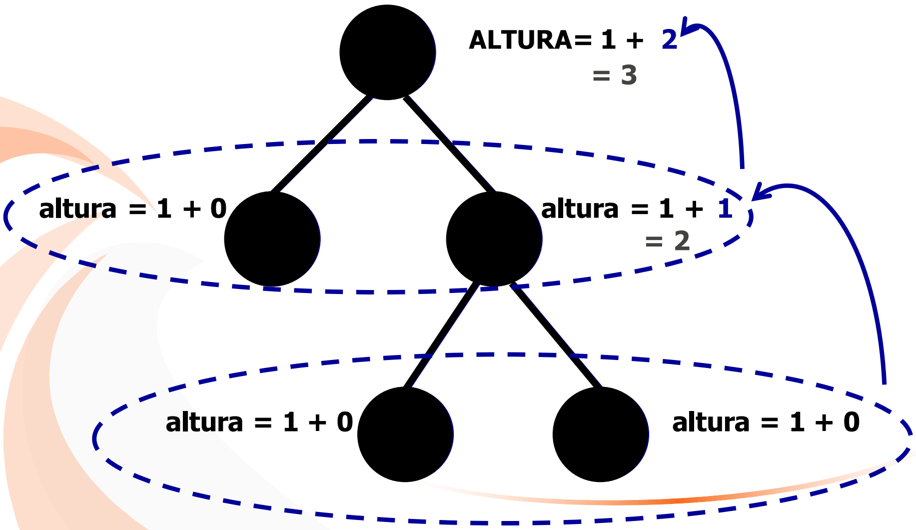
ESPECIFICACIÓ ALGEBRAICA (I)

```
MÒDUL ARBRES_BINARIS
   USA BOOL, NATURAL

   PARAMETRE TIPUS item
        OPERACIONS
                error_item( ) → item
   FPARAMETRE
   TIPUS arbin
   OPERACIONS
        crea_arbin( ) → arbin
        enraizar( arbin, item, arbin ) → arbin
        raiz( arbin ) → item
        esvacio( arbin ) → bool
        hijoiz, hijode( arbin ) → arbin
        altura( arbin ) → natural

VAR i, d: arbin; x: item;
ECUACIONS
raiz( crea_arbin( ) ) = error_item( )
raiz( enraizar( i, x, d ) ) = x
hijoiz( crea_arbin( ) ) = crea_arbin( )
hijoiz( enraizar( i, x, d ) ) = i
hijode( crea_arbin( ) ) = crea_arbin( )
hijode( enraizar( i, x, d )) = d
esvacio( crea_arbin( ) ) = CIERTO
esvacio( enraizar( i, x, d ) ) = FALSO
altura( crea_arbin( ) ) = 0
altura( enraizar( i, x, d ) ) =
   1 + max ( altura( i ), altura( d ) )

FMÒDUL
```

2. Arbres binaris

ESPECIFICACIÓ ALGEBRAICA (II)

altura(crea_arbin( )) = 0
altura = 1 + max(altura(hijoiz), altura(hijode))



2. Arbres binaris

EXERCICIS *nodosHoja*

2) Siga un arbre binari. Especificar la sintaxi i semàntica de les operacions:

nodosHoja, que torna el nombre de nodes fulla d'un arbre binari

Podeu utilitzar *esVacio* i *if*

2. Arbres binaris

EXERCICIS *simetrics i tots*

3) Siga un arbre binari les etiquetes del qual són nombres naturals. Especificar la sintaxi i semàntica de les operacions:

a) **simètrics**, que comprova que 2 arbres binaris són simètrics:

b) **tots**, que calcula la suma de totes les etiquetes dels nodes de l'arbre

Nota: Especificar la sintaxi de totes les operacions d'arbres binaris usades

2. Arbres binaris

EXERCICIS *transforma*

4) Es defineix l'operació transforma que rep un arbre binari i torna un arbre binari. Explicar què fa aquesta operació detallant el comportament de les dues equacions que apareixen a continuació:

VAR *i, d*: arbin; *x*: item,

transforma(crea_arbin( )) = crea_arbin( )

transforma(enraizar(*i, x, d*)) =

enraizar(transforma(*i*), *x* + todos(*i*) + todos(*d*), transforma(*d*))

Nota: L'operació *todos* calcula la suma de totes les etiquetes dels nodes de l'arbre (nombres naturals)

2. Arbres binaris

EXERCICIS *quita_hojas*

5) Utilitzant exclusivament les operacions *crea_arbin()* i *enraizar(arbin, item, arbin)* definir la sintaxi i la semàntica de l'operació quita_hojas que actua sobre un arbre binari i retorna l'arbre binari original sense les fulles

Tema 3. El tipus arbre

2. Arbres binaris

EXERCICIS *dos_hijos*

6) Especificar la sintaxi i la semàntica de l'operació dos_hijos que actua sobre un arbre binari i retorna CERT si tots els nodes tenen dos fills (excepte els nodes fulla)

2. Arbres binaris

RECORREGUTS

- Recórrer un arbre és visitar cada node de l'arbre una sola vegada
- Recorregut d'un arbre és la llista d'etiquetes de l'arbre ordenades segons es visiten els nodes
- Es distingeixen dues categories bàsiques de recorregut:
  - **recorreguts en profunditat**
  - **recorreguts en amplària** o **per nivells**

2. Arbres binaris

RECORREGUTS EN PROFUNDITAT (I)

- Si representem per I: anar cap a l'esquerra, R: visitar o escriure l'item, D: anar cap a la dreta, hi ha 6 possibles formes de recorregut en profunditat: RID, IRD, IDR, RDI, DRI i DIR. Si només volem fer els recorreguts d'esquerra a dreta queden 3 formes de recorregut:

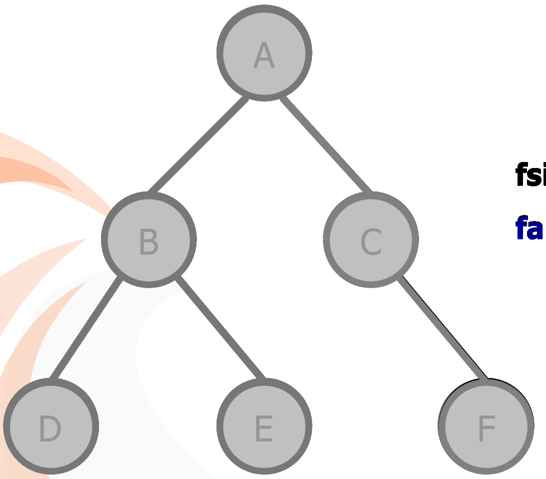
  1. RID o **preordre** (ordre previ)

  2. IRD o **inordre** (ordre simètric)

  3. IDR o **postordre** (ordre posterior)

  (El recorregut en postordre és l'invers especular del recorregut preordre, és a dir, es recorre l'arbre en preordre, visitant primer el subarbre dret abans que l'esquerre, i es considera la llista resultant com l'invers de la solució)

2. Arbres binaris
RECORREGUTS EN PROFUNDITAT (II)



PREORDRE: A B D E C F

```
algoritmo preorden ( a : arbin )
si ( no esvacio( a ) ) entonces
    escribe ( raiz ( a ) )
    preorden ( hijoiz ( a ) )
    preorden ( hijode ( a ) )
fsi
falgoritmo
```


2. Arbres binaris
RECORREGUTS EN PROFUNDITAT (III)

```
algoritmo inorden ( a : arbin )
si ( no esvacio( a ) ) entonces
    inorden ( hijoiz ( a ) )
    escribe ( raiz ( a ) )
    inorden ( hijode ( a ) )
fsi
falgoritmo
```

```
algoritmo postorden ( a : arbin )
si ( no esvacio( a ) ) entonces
    postorden ( hijoiz ( a ) )
    postorden ( hijode ( a ) )
    escribe ( raiz ( a ) )
fsi
falgoritmo
```

2. Arbres binaris

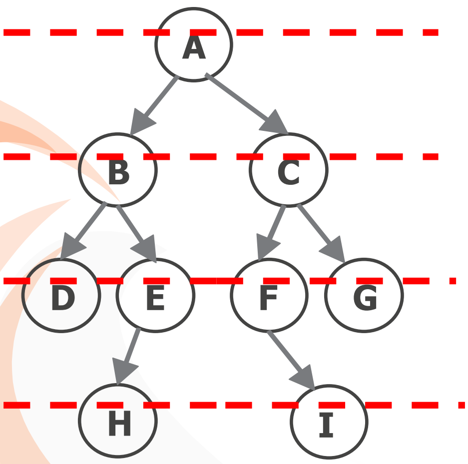
RECORREGUT EN AMPLÀRIA (NIVELLS)

- Consisteix en visitar els nodes des de l'arrel cap a les fulles, i d'esquerra a dreta dins de cada nivell

```
algoritmo niveles ( a : arbin )
var c: cola de arbin; aux: arbin; fvar
encolar(c, a)
mientras no esvacia(c) hacer
    aux := cabeza(c)
    escribe (raiz(aux))
    desencolar(c)
    si no esvacio(hijoiz(aux)) entonces encolar(c, hijoiz(aux))
    si no esvacio(hijode(aux)) entonces encolar(c, hijode(aux))
fmientras
falgoritmo
```


2. Arbres binaris

EXEMPLE DE RECORREGUTS (I)

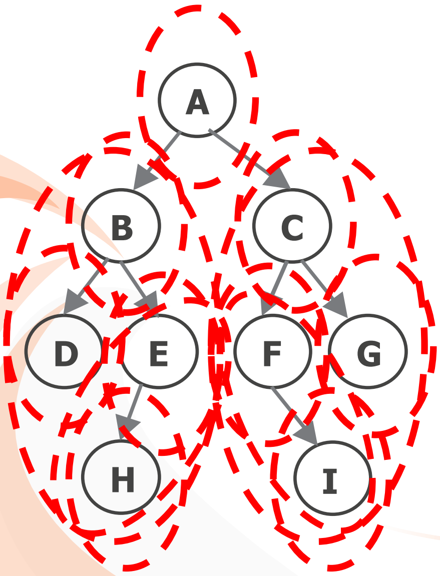


- Nivells:

a b c d e f g h i

2. Arbres binaris

EXEMPLE DE RECORREGUTS (II)

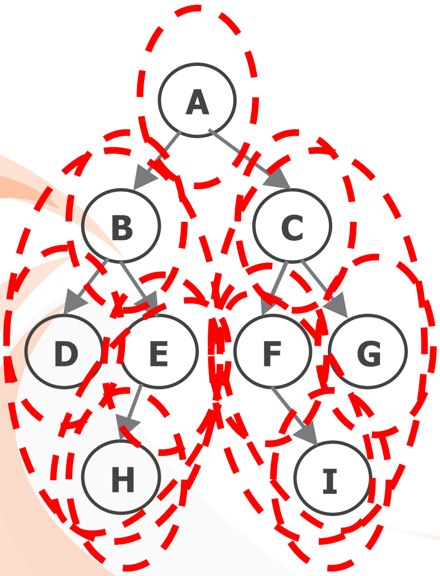


- Inordre:

d b h e a f i c g

2. Arbres binaris

EXEMPLE DE RECORREGUTS (III)

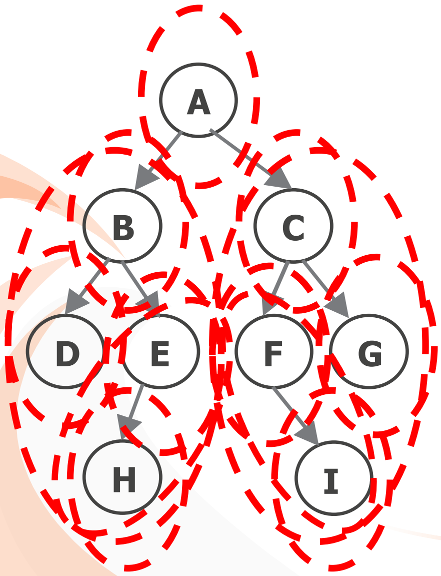


- Postordre:

d h e b i f g c a

2. Arbres binaris

EXEMPLE DE RECORREGUTS (IV)



- Preordre:

a b d e h c f i g

2. Arbres binaris

ENRIQUIMENT DE L'ESPECIFICACIÓ

OPERACIONS

preorden, inorden, postorden(arbin) → lista
nodos (arbin) → natural
eshoja (arbin) → bool

VAR i, d: arbin; x: item;

ECUACIONS

preorden(crea_arbin( )) = crea_lista( )
preorden(enraizar(i, x, d)) = concatenar(insiz(x, preorden(i)), preorden(d))
inorden(crea_arbin( )) = crea_lista( )
inorden(enraizar(i, x, d)) = concatenar(insde(inorden(i), x), inorden(d))
postorden(crea_arbin( )) = crea_lista( )
postorden(enraizar(i, x, d)) = insde(concatenar(postorden(i), postorden(d)), x)
nodos(crea_arbin( )) = 0
nodos(enraizar(i, x, d)) = 1 + nodos(i) + nodos(d)
eshoja(crea_arbin( )) = FALSO
eshoja(enraizar(i, x, d)) = esvacio(i) Λ esvacio(d)

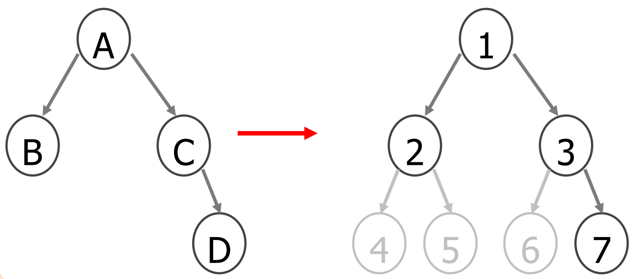
2. Arbres binaris

REPRESENTACIÓ SEQÜENCIAL I ENLLAÇADA (I)

- Representació seqüencial
  Es numeren seqüencialment els nodes de l'arbre hipotèticament ple des de l'arrel a les fulles per nivells (començant pel nivell 1, després el nivell 2, etc.) i d'esquerra a dreta en cada nivell. La representació seqüencial es pot fer usant un vector unidimensional:
  - L'arrel es guarda en l'adreça 1
  - si un node *n* està en l'adreça *i*, llavors el seu fill esquerre estarà en l'adreça *2i* i el seu fill dret en l'adreça *2i + 1*

2. Arbres binaris

REPRESENTACIÓ SEQÜENCIAL I ENLLAÇADA (II)



A	B	C				D
1	2	3	4	5	6	7

2. Arbres binaris

REPRESENTACIÓ SEQÜENCIAL I ENLLAÇADA (III)

- Representació enllaçada

```
typedef int TItem;
class TNodo;

class TArbin{
    public:
        TArbin ( );                                   //CONSTRUCTOR
        TArbin (const TArbin & origen);               //CONSTRUCTOR DE COPIA
        ~TArbin ( );                                  //DESTRUCTOR
        TArbin & operator = (const TArbin & a);       //ASSIGNACIÓ
        TItem & Raiz ( );
        TArbin HijoIz ( ); TArbin HijoDe ( );
        bool EsVacio ( );
        int Altura ( );
    private:
        void Copiar (const TArbin & origen);
        TNodo *farb;
};
```


2. Arbres binaris

REPRESENTACIÓ SEQÜENCIAL I ENLLAÇADA (IV)

```
class TNodo{
    friend class TArbin;
    private:
        TItem fitem;
        TArbin fiz, fde;
};

/* ------------------------------------------------------------------------------------------------*/
TArbin::TArbin ( ) {farb = NULL; }
/* ------------------------------------------------------------------------------------------------*/
TArbin::TArbin (const TArbin & origen){
    Copiar (origen);
}
/* ------------------------------------------------------------------------------------------------*/
void
TArbin::Copiar (const TArbin & origen){
    if (origen.farb != NULL){
        TNodo *aux = new TNodo( );
        aux → fitem = origen.farb → fitem;
        farb = aux;
        (farb → fiz).Copiar (origen.farb → fiz);
        (farb → fde).Copiar (origen.farb → fde);
    }
    else farb = NULL;
}
```

2. Arbres binaris

REPRESENTACIÓ SEQÜENCIAL ENLLAÇADA (V)

```
TArbin::~TArbin ( ) {
    if (farb != NULL){
        delete farb;
        farb = NULL;}
}
/* ------------------------------------------------------------------------------------------*/
TArbin &
TArbin::operator = (const TArbin & a){
    this → ~TArbin( );
    Copiar (a);
  return *this;
}
```


2. Arbres binaris

REPRESENTACIÓ SEQÜENCIAL I ENLLAÇADA (VI)

```
TItem
TArbin::Raiz ( ){
    TItem vacio;
    if (farb != NULL) return farb → fitem;
    else return vacio;
}
```

2. Arbres binaris

REPRESENTACIÓ SEQÜENCIAL I ENLLAÇADA (VII)

```
bool
TArbin::EsVacio ( ){
    return (farb == NULL)
}

/* --------------------------------------------------------------------------------*/

int
TArbin::Altura ( ){
    int a1, a2;

    if (farb != NULL){
        a1 = (farb → fiz).Altura( );
        a2 = (farb → fde).Altura( );
        return (1 + (a1 < a2 ? a2 : a1));
    }
  else return 0;
}
```


2. Arbres binaris

REPRESENTACIÓ SEQÜENCIAL I ENLLAÇADA (VIII)

```
/* Programa de prova

int
main ( ){

    TArbin a, b, c;

    a.Enraizar (b, 1, c);
    b.Enraizar (a, 2, c);
    cout << "el fill esquerra de l'arbre té " << (b.HijoIz( )).Raiz( );
  // ESCRIBE 1
    cout << "la altura de l'arbre es " << b.Altura() << endl;
    // ESCRIBE 2
}
```

2. Arbres binaris

REPRESENTACIÓ SEQUENCIAL I ENLLAÇADA (IX)

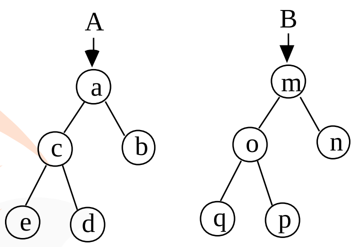
¿Constructor i destructor de TNodo?

```
TNodo::TNodo( ):fiz(),fde(){
    fitem=0;
}

TNodo::~TNodo( ){
    fitem=0;
}
```


2. Arbres binaris

ALTRES OPERACIONS INTERESSANTS (I)

- A més de totes les operacions vistes anteriorment, utilitzarem les operacions d'assignació i "moviment" d'arbres i iteradors:



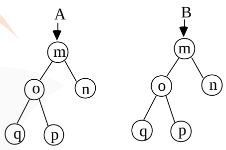
I, J = Iteradors

2. Arbres binaris

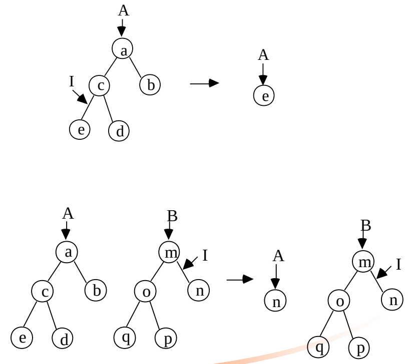
ALTRES OPERACIONS INTERESSANTS (II)

- a) *Assignació (còpia) entre arbres i iteradors*:

  - a1) A = B. Fa una còpia de B en A



- a2) A = I. Fa una còpia sobre l'arbre A, de la branca de l'arbre a què apunta l'iterador I

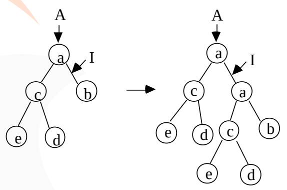


2. Arbres binaris

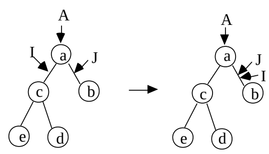
ALTRES OPERACIONS INTERESSANTS (III)

- a) *Assignació (còpia) entre arbres i iteradors*:

  - a3) I = A. Fa una còpia sobre la branca de l'arbre a què apunta l'iterador I de l'arbre A



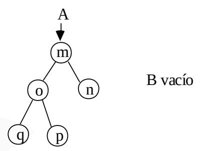
- a4) I = J. Serveix per a inicialitzar l'iterador I de manera que apunte al mateix node a què apunta l'iterador J



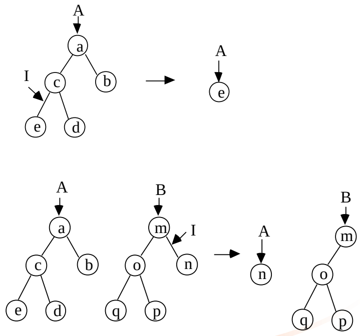
2. Arbres binaris

ALTRES OPERACIONS INTERESSANTS (IV)

- b) *Moviment de branques entre arbres i iteradors*:
  - b1) Moure (A, B). Mou l'arbre B a l'arbre A. B es queda buit



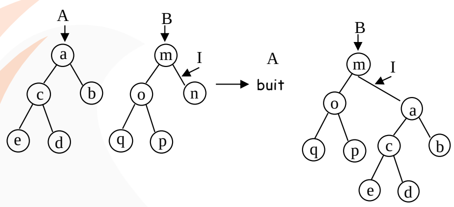
- b2) Moure (A, I). Mou la branca de l'arbre a què apunta l'iterador I a l'arbre A



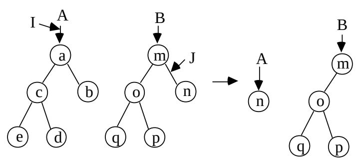
2. Arbres binaris

ALTRES OPERACIONS INTERESSANTS (V)

- b) *Moviment de branques entre arbres i iteradors*:
  - b3) Moure (I, A). Mou l'arbre A a la branca de l'arbre a què apunta l'iterador I



- b4) Moure (I, J). Mou la branca de l'arbre a què apunta l'iterador J a la branca de l'arbre a què apunta l'iterador I

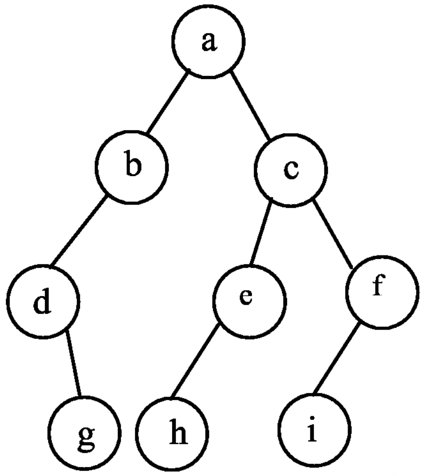


2. Arbres binaris

EXERCICIS *recorreguts*

1a) Donat el següent arbre binari, calculeu els recorreguts preordre, postordre, inordre i nivells

1b) Es pot reconstruir un arbre binari donant-ne només el recorregut inordre? Quants recorreguts com a mínim són necessaris? Quins?



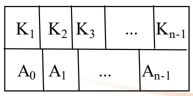
2. Arbres binaris

Preguntes de tipus test: Vertader vs. Fals

- El nivell d'un node en un arbre coincideix amb la longitut del camí desde l'arrel a eixe node
- Donat un únic recorregut d'un arbre binari ple, és possible reconstruir eixe arbre
- Un arbre binari complet amb n nodes i alçària k és un arbre binari ple per a eixa mateixa alçària

3. Arbres de cerca(I)

- Arbres de cerca= Arbres n-aris de cerca = Arbres multicamí de cerca
- Són un tipus particular d'arbres que poden definir-se quan el tipus dels elements de l'arbre posseeix una relació $\leq$ d'ordre total
- Un arbre multicamí de cerca T és un arbre n-ari buit o que cumpleix les propietats següents:
  - 1. L'arrel de T conté $A_0, \ldots, A_{n-1}$ subarbres i $K_1, \ldots, K_{n-1}$ etiquetes
  - 2. $K_i < K_{i+1}$, $1 \leq i < n-1$
  - 3. Totes les etiquetes del subarbre A_i són:
    menors que K_{i+1} $\quad 0 \leq i < n-1$
    majors que K_i $\quad 0 < i \leq n-1$
  - 4. Els subarbres A_i, $0 \leq i \leq n-1$ són també arbres multicamí de cerca



3. Arbres de cerca(II)

- Algoritme de busca
  - Per a buscar un valor x l'arbre, primer es mira el node arrel i es fa la comparació següent:
  - $x < K_i$ o bé $x > K_i$ o bé $x = K_i$ ($1 \leq i \leq n-1$)
  - 1) En el cas que $x = K_i$, la busca ja s'ha completat
  - 2) Si $x < K_i$, aleshores per la definició d'arbre multicamí de busca, x ha d'estar en el subarbre A_{i-1}, si aquest existeix en l'arbre
  - 3) Si $x > K_{n-1}$, x ha d'estar en A_{n-1}
- Els arbres multicamí de cerca són útils quan la memòria principal és insuficient per a utilitzar-la com a emmagatzemament permanent
- En una representació enllaçada d'aquests arbres, els punters poden representar adreces de disc en compte de adreces de memòria principal. Quantes vegades s'accedeix al disc quan es fa una cerca? Com es pot reduir el nombre d'accessos al disc?

3.1. Arbres binaris de cerca

ESPECIFICACIÓ ALGEBRAICA (I)

- Propietats
  - tots els elements en el subarbre esquerre són ≤ que l'arrel,
  - tots els elements en el subarbre dret són ≥ que l'arrel,
  - els dos subarbres son binaris de cerca
    - en algunes variants no es permet la repetició d'etiquetes

MÒDUL ARBRE_BIN_CERCA USA BOOL, ARBRES_BINARIS
  PARÀMETRE TIPUS item
    OPERACIONS
      <, ==, >: item, item → bool
      error_item( ) → item
  FPARAMETRE

  OPERACIONS
    insertar(arbin, item) → arbin
    buscar(arbin, item) → bool
    borrar(arbin, item) → arbin
    min(arbin) → item

3.1. Arbres binaris de cerca

ESPECIFICACIÓ ALGEBRAICA (II)

VAR i, d: arbin; x, y: item;
ECUACIONS
  insertar(crea_arbin( ), x) =
    enraizar(crea_arbin( ), x, crea_arbin( ))
  si (y < x) **entonces**
    insertar(enraizar(i, x, d), y) =
      enraizar(insertar(i, y), x, d)
  si no si (y > x) insertar(enraizar(i, x, d), y) =
      enraizar(i, x, insertar(d, y))   **fsi**

  buscar(crea_arbin( ), x) = FALSO
  si (y < x) **entonces**
    buscar(enraizar(i, x, d), y) = buscar(i, y)
  si no si (y > x) **entonces**
    buscar(enraizar(i, x, d), y) = buscar(d, y)
  si no buscar(enraizar(i, x, d), y) = CIERTO   **fsi**

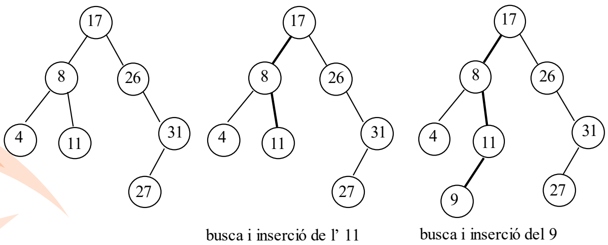
borrar(crea_arbin( ), x) = crea_arbin( )
si (y < x) **entonces**
  borrar(enraizar(i, x, d), y) =
    enraizar(borrar(i, y), x, d)
si no si (y > x) **entonces**
  borrar(enraizar(i, x, d), y) =
 enraizar(i, x, borrar(d, y))   **fsi**
si (y==x) **y** esvacio(d) **entonces**
  borrar(enraizar(i, x, d), y) = i   **fsi**
si (y==x) **y** esvacio(i) **entonces**
  borrar(enraizar(i, x, d), y) = d   **fsi**
si (y==x) **y** no esvacio(d) **y** no esvacio(i) **entonces**
  borrar(enraizar(i, x, d), y) =
    enraizar(i, min(d), borrar(d, min(d)))   **fsi**
min(crea_arbin( )) = error_item( )
si esvacio(i) **entonces** min(enraizar(i, x, d)) = x
si no min(enraizar(i, x, d)) = min(i)   **fsi**

FMÒDUL

3.1. Arbres binaris de cerca

OPERACIONS BASIQUES (I)

- Cerca i inserció d'un element

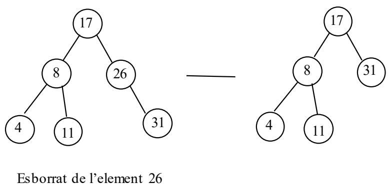


- Recorreguts en inordre: totes les etiquetes ordenades ascendentement
- Quin és el cost de les operacions de cerca i inserció en el ABB? Qué passa si inserim una sèrie d'elements ordenats en un ABB inicialment buit?

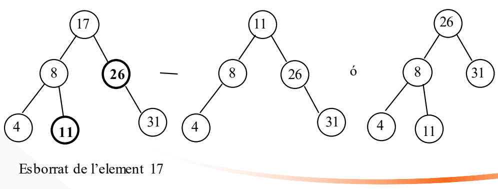
3.1. Arbres binaris de cerca

OPERACIONS BASIQUES (II)

- Esborrat d'un element
  - El node on es troba és una fulla
  - El node on es troba té un únic fill. El node a eliminar és substituït pel seu fill



  - El node on es troba té dos fills



3.1. Arbres binaris de cerca

EXERCICIS *inserció* i esborrat

1) En un arbre binari de cerca inicialment buit,
   a) Inserir els elements següents: 20, 10, 30, 40, 5, 15, 50, 22, 25, 24, 26, 3, 35, 38, 39, 37

   b) Sobre l'arbre resultant, esborrar els elements següents: 5, 3, 30, 22, 39 (utilitzar el criteri de substituir pel menor de la dreta)

3.1. Arbres binaris de cerca

Preguntes de tipus test: Vertader vs. Fals

– A l'esborrat d'un element que es troba a un node amb dos fills no buits en un arbre binari de cerca, hem d'intercanviar l'element a esborrar pel menor del subarbre de l'esquerra o pel major del subarbre de la dreta
– El menor element en un arbre binari de cerca sempre es troba a un node fulla
– El cost temporal (en el seu pitjor cas) d'inserir una etiqueta en un arbre binari de cerca és lineal respecte al nombre de nodes de l'arbre