Estructuras de datos no lineales

ED - UNIVERSIDAD DE CORDOBA

Árboles (1)

Contenidos

- Concepto y definiciones generales.
- Especificación del TAD Tree[T].
- Ejemplos de procesamiento de un árbol:
  - Cálculo de la altura.
  - Cálculo del tamaño.

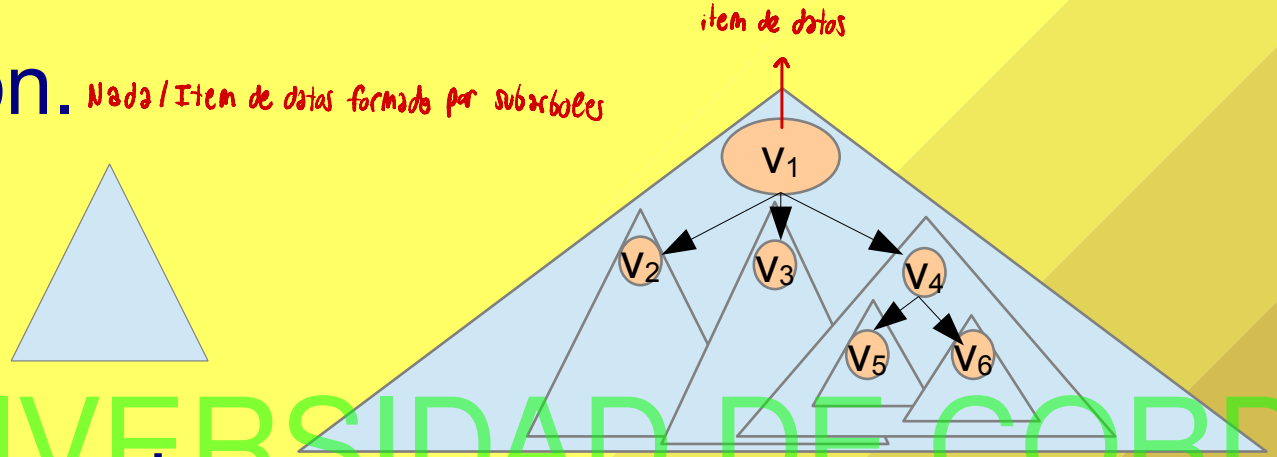
Introducción

- Ideas clave del TAD árbol:
  - Representa relaciones uno-a-muchos.
  - Define una estructura jerárquica.

¿Tendrán los árboles algo en común con las listas?

Introducción

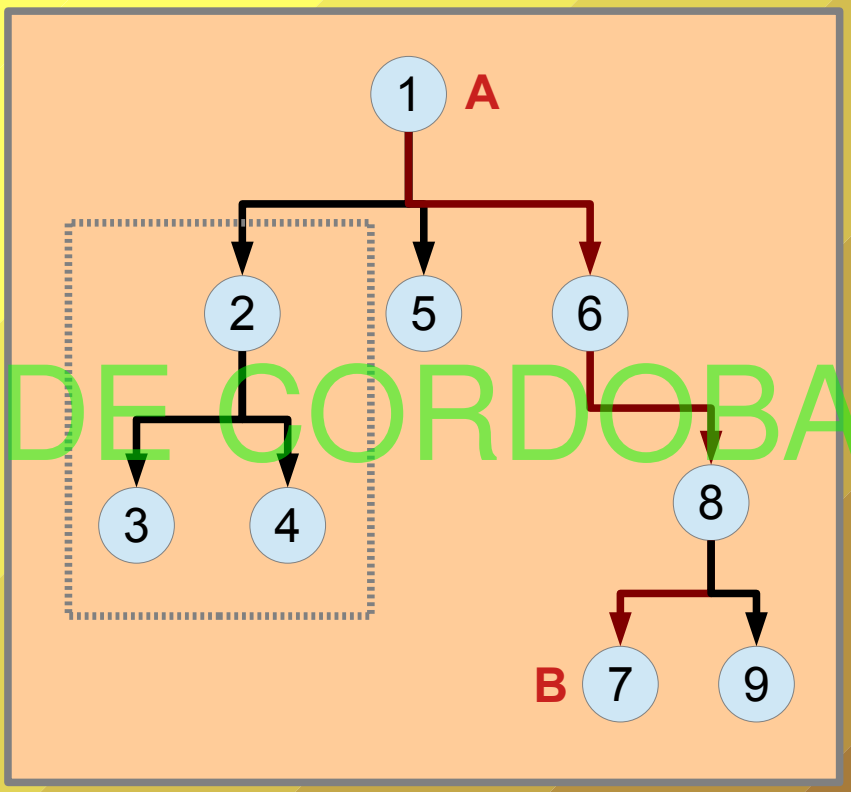
- Definición. Nodo / Item de datos formado por subarboles



- Tipos de nodos.
  - Nodo raíz.
  - Nodo hoja (exterior).
  - Nodo bifurcación (interior).

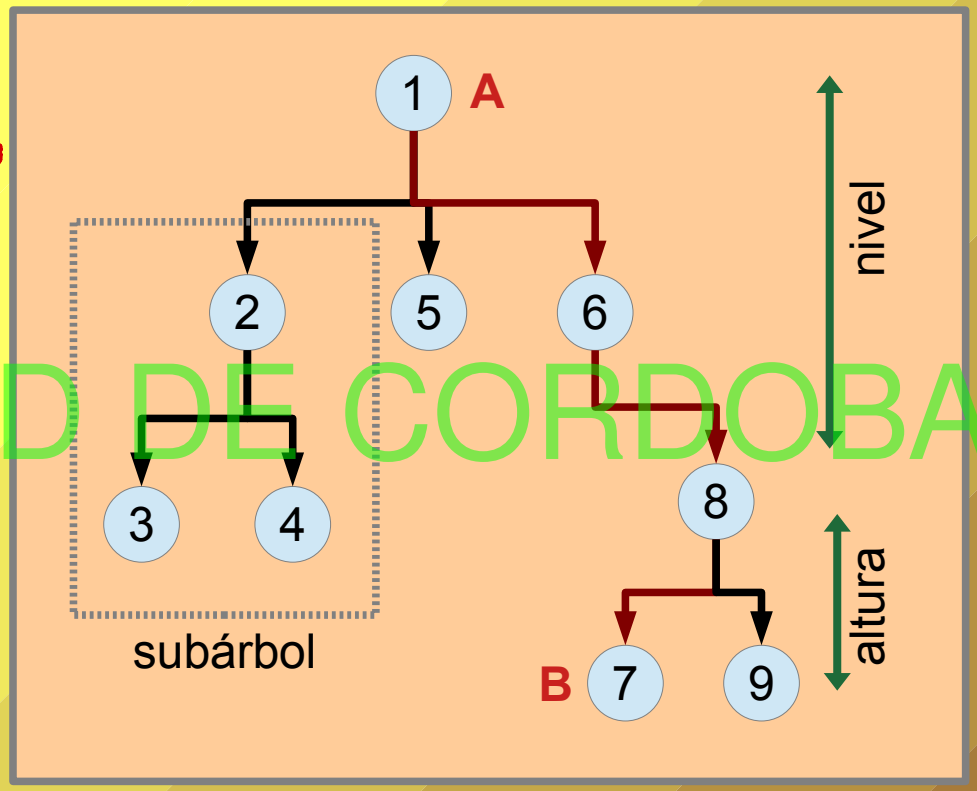
Introducción

- Definiciones:
  - Camino. secuencia de nodos donde la relación es padre-hijo
  - Longitud del camino. nº de enlaces
  - Antecesor y descendiente.
  - Descendientes de nodo (propios).
  - Antecesores de un nodo (propios).
  - Subárbol.



Introducción

- Definiciones:
  - Altura de un nodo. Camino nodo-hoja más largo
  - Altura del árbol. Camino más largo
  - Altura del árbol vacío.
  - Profundidad (nivel) de un nodo. Camino nodo-raíz
  - Grado de un nodo. nº de hijos
  - Grado del árbol. max grado de un nodo
  - Ancho (de un nivel). ancho de nivel
  - Amplitud del árbol. nº de nodos hoja
  - Tamaño de un árbol. nº de nodos



Introducción

- ADT Tree[T]:
  - **Makers**:
    - create():Tree[T] //Makes an empty tree.
      - Post-c: isEmpty()
    - create(v:T):Tree[T] //Makes a leaf.
      - Post-c: not isEmpty()
      - Post-c: item()=v
      - Post-c: nChilds()==0
  - **Observers**:
    - isEmpty():Bool //is an empty tree?
    - item():T //Gets the root's item.
      - Pre-c: not isEmpty().
    - nChildren():Integer //Gets the number of children.
      - Pre-c: not isEmpty()
      - Post-c: not isEmpty() or nChildren>=0
    - child(i:Integer):Tree[T] //The i-sub-tree.
      - Pre-c: not isEmpty().
      - Pre-c: 0<= i< nChildren().
  - **Modifiers**:
    - createRoot(it:T) //create the root.
      - Pre-c: isEmpty()
      - Post-c: not isEmpty()
      - Post-c: item()==it
    - setItem(it:T) //set the root's item.
      - Pre-c: not isEmpty()
      - Post-c: item()==it
    - setChild(t:Tree[T])//Set the i-subtree.
      - Pre-c: not isEmpty()
      - Pre-c: 0<=i<nChilds()
      - Post-c: t==child(i)
    - addChild(t:Tree[T])//add a new subtree.
      - Pre-c: not t.isEmpty()
      - Post-c: nChildren()==(old.nChildren()+1)
      - Post-c: child(old.nChildren())==t

Introducción

- Procesado de un árbol.
  - Altura de un árbol.

```
Algorithm tree_height(t:Tree[T]): Integer
Aux:
   height: Integer #The tree's height
   i     : Integer #loop index.
   maxH  : Integer #maximum of child's heights.
Begin
   height ← -1
   If not t.isEmpty() Then
      maxH ← -1
      For i from 0 to t.nChilds() Increment 1 Do
         maxH ← max(maxH, tree_height(t.child(i))
      End-For
      height ← 1 + maxH
   End-If
   Return height
End.
```

Introducción

- Procesado de un árbol.
  - Tamaño del árbol.

```
Algorithm tree_size(t:Tree[T]): Integer
Aux
   size: Integer #the tree's size.
   i   : Integer #loop index.
Begin
   size ← 0
   If not t.isEmpty() Then
      size ← 1
      For i from 0 to t.nChildren() Increment 1 Do
         size ← size + tree_size(t.child(i))
      End-For
   End-If
   Return size
End.
```

Introducción

- Resumiendo:
  - Representa relaciones (1 - N).
  - Define una jerarquía: padre-hijos.
  - Su definición es recursiva -> algoritmos recursivos.
  - Se suele representar con nodos.
  - Tenemos distintos tipos de nodos: raíz, bifurcación, hoja.

Referencias

- Lecturas recomendadas:
  - Caps. 10, 11 y 12 de “Estructuras de Datos”, A. Carmona y otros. U. de Córdoba. 1999.
  - Caps 9 y 13.5 de *“Data structures and software develpment in an object oriented domain”*, Tremblay J.P. y Cheston, G.A. Prentice-Hall, 2001.
  - Wikipedia: en.wikipedia.org/wiki/Tree_(data_structure)#Terminology