

Estructuras de Datos

Clase 11 – Árboles Generales (segunda parte)



Dr. Sergio A. Gómez
<http://cs.uns.edu.ar/~sag>



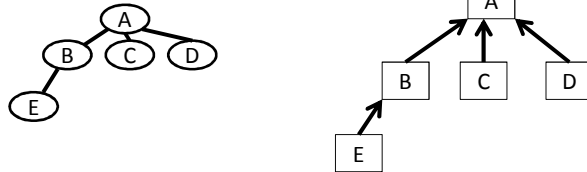
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
Bahía Blanca, Argentina

Representaciones de árboles

- Del padre
- Lista de hijos
- Goodrich & Tamassia: Del padre + Lista de hijos
- Hijo extremo izquierdo – hermano derecho
- Variantes de las de más arriba usando arreglos con cursores (de interés histórico o para modelar árboles en almacenamiento externo)

Representación del padre

- Cada nodo conoce su rótulo y a su padre.



- Aplicación: Directorio padre en sistemas de archivos
- El directorio padre se denota con .. (dos puntos consecutivos).

El directorio corriente es c:\windows
El comando `cd..` lleva al directorio padre.

```
C:\Windows\system32\cmd.exe - dir /ad /p
C:\Windows>dir /ad /p
. Volume in drive C has no label.
Volume Serial Number is 0A37-3572

Directory of C:\Windows

27/02/2013 03:14 p.m. <DIR>      .
27/02/2013 03:14 p.m. <DIR>      ..
14/07/2009 02:32 a.m. <DIR>      addins
14/07/2009 12:20 a.m. <DIR>      AppCompat
19/03/2013 08:30 a.m. <DIR>      AppPatch
```

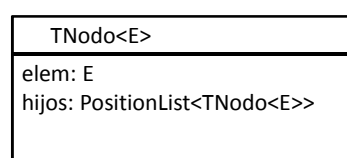
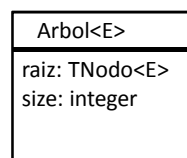
“..” es la referencia al directorio padre

Estructuras de datos - Dr. Sergio A. Gómez

3

Representación de lista de hijos

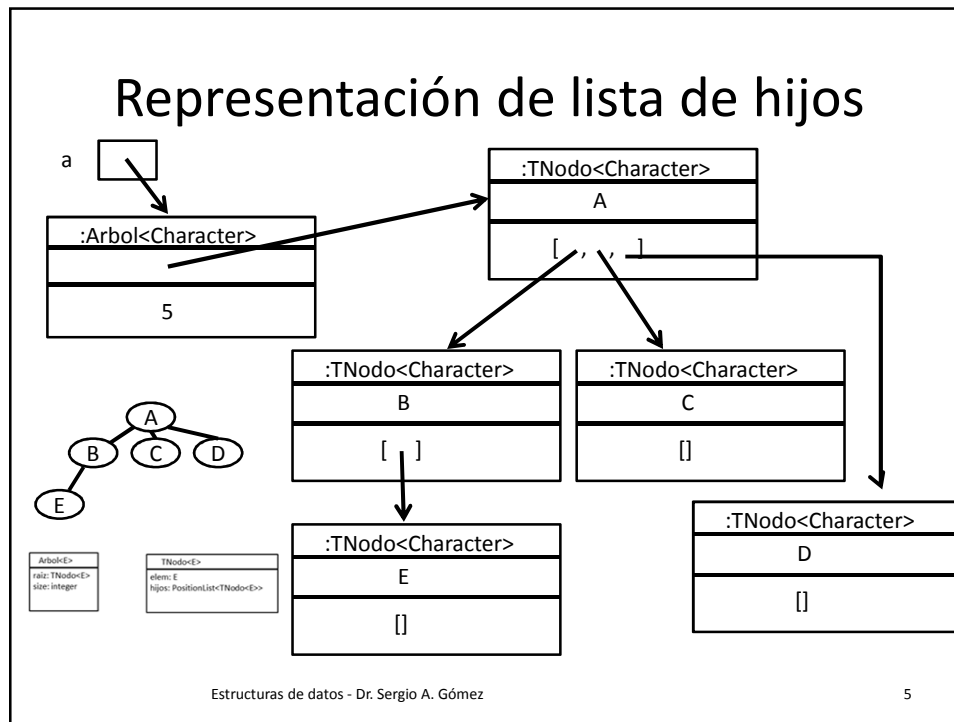
- Cada nodo conoce su rótulo (elemento) y la lista de sus nodos hijos
- El árbol conoce el nodo raíz del árbol



Estructuras de datos - Dr. Sergio A. Gómez

4

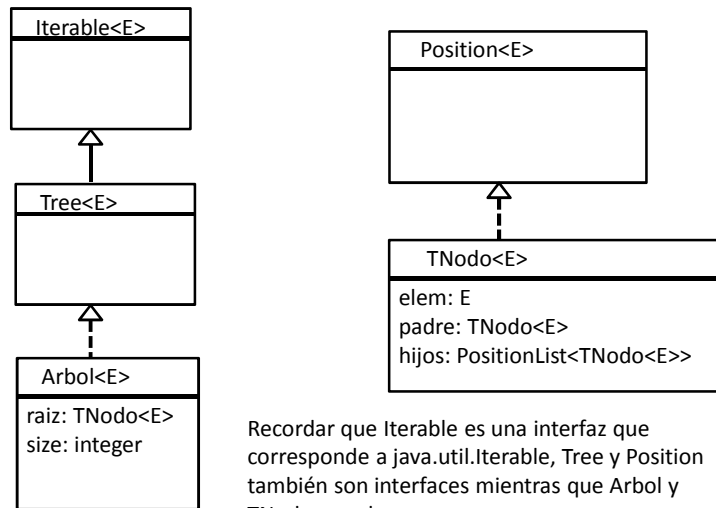
El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente: “Estructuras de Datos. Notas de Clase”. Sergio A. Gómez. Universidad Nacional del Sur. (c) 2013-2019.



Representación en GT: Colección de hijos

- GT = Lista de hijos + padre
- El árbol conoce la raíz
- Cada nodo conoce el rótulo (elemento), la lista de nodos hijos y el nodo padre.
- (Discutiremos la estructura de datos y la implementación de algunas de las operaciones, el resto quedará a cargo del alumno.)

Representación de árboles en GT



Estructuras de datos - Dr. Sergio A. Gómez

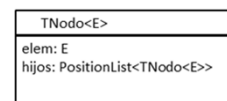
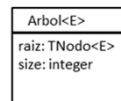
7

Representación de árboles en GT

```

public class TNode<E> implements Position<E>
{
    private E elemento;
    private TNode<E> padre;
    private PositionList<TNode<E>> hijos;

    public TNode( E elemento, TNode<E> padre ) {
        this.elemento = elemento;
        this.padre = padre;
        hijos = new DoubleLinkedList<TNode<E>>();
    }
    public TNode(E elemento ) { this( elemento, null); }
    public E element() { return elemento; }
    public PositionList<TNode<E>> getHijos() { return hijos; }
    public void setElemento( E elemento ) { this.elemento = elemento; }
    public TNode<E> getPadre() { return padre; }
    public void setPadre( TNode<E> padre ) { this.padre = padre; }
}
  
```



Estructuras de datos - Dr. Sergio A. Gómez

8

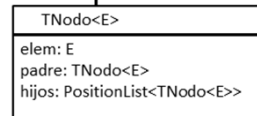
El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente: "Estructuras de Datos. Notas de Clase". Sergio A. Gómez. Universidad Nacional del Sur. (c) 2013-2019.

Representación de árboles en GT

```
public class Arbol<E> implements Tree<E>
{
    protected TNode<E> raiz;
    protected int size;

    public Arbol() { raiz = null; size = 0; }
    public boolean isEmpty() { return raiz == null; }
    public void createRoot( E e ) throws InvalidOperationException {
        if( !isEmpty() )
            throw new InvalidOperationException( "Árbol no vacío" );
        raiz = new TNode<E>( e );
        size = 1;
    }

    public Position<E> root() throws EmptyTreeException {
        if( isEmpty() )
            throw new EmptyTreeException( "Árbol vacío" );
        return raiz;
    }
}
```

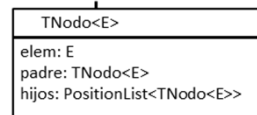


Estructuras de datos - Dr. Sergio A. Gómez

9

```
public boolean isExternal(Position<E> v) throws InvalidPositionException
{
    TNode<E> nodo = checkPosition( v );
    return nodo.getHijos().isEmpty();
}

public Position<E> addFirstChild(Position<E> p, E e) throws
InvalidPositionException {
    TNode<E> padre = checkPosition(p);
    TNode<E> nodo = new TNode<E>( e, padre );
    padre.getHijos().addFirst( nodo );
    size++;
    return nodo;
}
```



Estructuras de datos - Dr. Sergio A. Gómez

10

El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente:
 "Estructuras de Datos. Notas de Clase". Sergio A. Gómez. Universidad Nacional del Sur. (c) 2013-2019.

```

public Position<E> addBefore(Position<E> p, Position<E> rb, E e)
    throws InvalidPositionException {
    TNode<E> padre = checkPosition( p );
    TNode<E> hermanoDerecho = checkPosition( rb );
    TNode<E> nuevo = new TNode<E>( e, padre );
    PositionList<TNode<E>> hijosPadre = padre.getHijos();
    // encuentre: true si encontré ubicación en lista de hijos de su padre
    boolean encuentre = false;
    Position<TNode<E>> pp = hijosPadre.first();
    while( pp != null && !encuentre )
        if( hermanoDerecho == pp.element() )
            encuentre = true
        else
            pp = (pp != hijosPadre.last() ? hijosPadre.next(pp) : null);
    if( !encuentre )
        throw new InvalidPositionException( "p no es padre de rb" );
    hijosPadre.addBefore( pp, nuevo );
    size++;
    return nuevo;
}

```

Estructuras de datos - Dr. Sergio A. Gómez

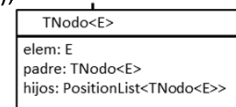
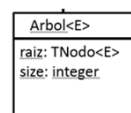
11

```

public void removeExternalNode (Position<E> p)
    throws InvalidPositionException {
    if( isEmpty() ) throw new InvalidPositionException( "Arbol vacío" );
    TNode<E> n = checkPosition( p );
    if( !n.getHijos().isEmpty() )
        throw new InvalidPositionException( "p no es hoja" );
    TNode<E> padre = n.getParent();
    PositionList<TNode<E>> hijosPadre = padre.getHijos();
    boolean encuentre = false; Position<TNode<E>> pp = null;
    Iterable<Position<TNode<E>> posiciones = hijosPadre.positions();
    Iterator<Position<TNode<E>> it = posiciones.iterator();
    while( it.hasNext() && !encuentre ) {
        pp = it.next();
        if( pp.element() == n ) encuentre = true;
    }
    if( !encuentre )
        throw new InvalidPositionException( "p no aparece en la lista de"
            + "hijos de su padre--- árbol corrupto???" );
    hijosPadre.remove( pp );
    size--;
}

```

Estructuras de datos - Dr. Sergio A. Gómez



El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente: "Estructuras de Datos. Notas de Clase". Sergio A. Gómez. Universidad Nacional del Sur. (c) 2013-2019.

Árboles generales: children

```

public Iterable<Position<E>> children( Position<E> v )
    throws InvalidPositionException
{
    if( v == null )
        throw new
            InvalidPositionException( "Children: Posición inválida");
    TNode<E> p = checkPosition(v);
    PositionList<Position<E>> lista =
        new ListaDoblementeEnlazada<TNode<E>>();
    for( TNode<E> n : p.getHijos() )
        lista.addLast(n);
    return lista;
} El tiempo de ejecución es del orden de la cantidad de hijos de v

```

Estructuras de datos - Dr. Sergio A. Gómez

13

Árboles generales: Iterador

```

public Iterable<Position<E>> positions() {
    PositionList<Position<E>> l = new ListaDoblementeEnlazada<Position<E>>();
    if( !isEmpty() ) pre( l, raiz );
    return l;
}  $T_{\text{positions}}(n) = O(n)$  si árbol this tiene n nodos

private void pre(PositionList<Position<E>> l, TNode<E> r) {
    l.addLast( r );
    for( TNode<E> h : r.getHijos() )
        pre( l, h );
} El tiempo de ejecución es lineal en la cantidad de nodos del árbol

public Iterator<E> iterator() { //  $T_{\text{iterator}}(n) = O(n)$  si árbol this tiene n nodos
    PositionList<E> l = new ListaDoblementeEnlazada<E>();
    for( Position<E> p : positions() )
        l.addLast( p.element() );
    return l.iterator();
}

```

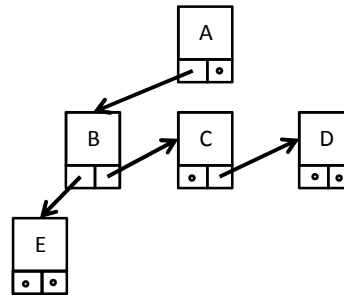
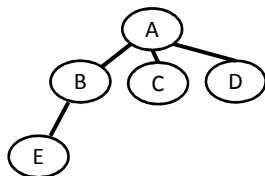
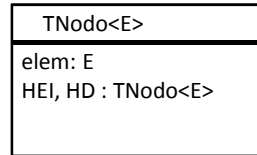
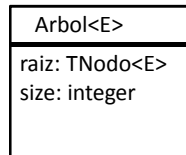
Estructuras de datos - Dr. Sergio A. Gómez

14

El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente:
 "Estructuras de Datos. Notas de Clase". Sergio A. Gómez. Universidad Nacional del Sur. (c) 2013-2019.

Representación HEI-HD

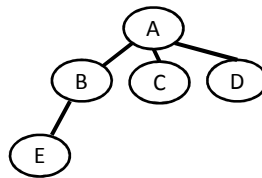
Cada nodo conoce su rótulo y la identidad de su hijo extremo izquierdo y de su hermano derecho (se puede combinar con la representación del padre de ser necesario)



Estructuras de datos - Dr. Sergio A. Gómez

15

Representaciones con arreglos



	0	1	2	3	4	5
rótulos	A	B	C	D	E		
padres	-1	0	0	0	1		
HEI	1	4	-1	-1	-1		
HD	-1	2	3	-1	-1		
size	5						
raiz	0						

Estructuras de datos - Dr. Sergio A. Gómez

16

El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente: "Estructuras de Datos. Notas de Clase". Sergio A. Gómez. Universidad Nacional del Sur. (c) 2013-2019.

Recorrido preorden

```
public void preorden() {
    pre( raiz );
}

// r representa el cursor de la raíz del subárbol listado actualmente
private void pre( int r ) {
    System.out.println( rotulos[r] );
    int h = HEI[r];    // h representa el hijo actual procesado
    while( h != -1 ) {
        pre( h );
        h = HD[h];
    }
}
```