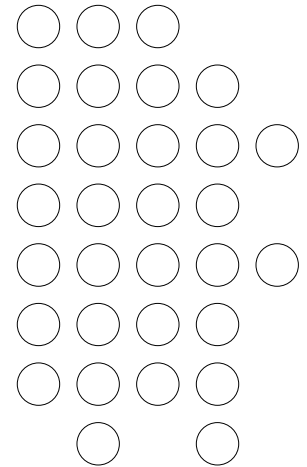


# Grafos. Búsqueda en amplitud

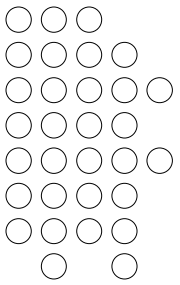


UNIVERSIDAD  
DE LOS ANDES

**Diseño y Análisis de Algoritmos**  
**Cátedra de Programación**  
**Carrera de Ingeniería de Sistemas**  
**Prof. Isabel Besembel Carrera**



# Implementación del DigrafoMat



Marzo, 2005		DigrafoMat()	
		{pos: $g(i, j) = 0 \quad \forall i, j$ }	
1	[[ $g(i, j) = 0$ ] $j = 0, 99$ ] $i = 0, 99$	<b>-g, pos:</b> Definido en DigrafoMat.	
2	pos = 0	<b>-i, j:</b> Entero. Contadores para recorrer el arreglo	

Constructor de DigrafoMat.  $O(N^2)$

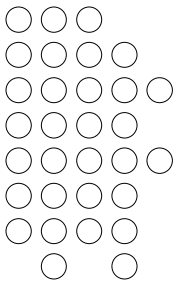
27/04/98		vacíoDigrafo(): Lógico	
		{pre: $g(i, j) \geq 0 \quad \forall i, j$ }	{pos: $g(i, j) \geq 0 \quad \forall i, j$ }
1	regrese ( pos = 0 )	<b>-pos:</b> Definido en DigrafoMat.	

Verifica si el digrafo está vacío  $O(1)$

Marzo, 2005		nuevoNodo(TipoEle: n)	
		{pre: $g(i, j) \geq 0 \quad \forall i, j$ }	{pos: $g(i, j) \geq 0 \quad \forall i, j$ }
1	ele = t.conTabla(n)	<b>-t, pos, dir:</b> Definido en DigrafoMat. <b>-ele:</b> TipoEleTab. Variable auxiliar.	
2	Si (ele.Clave() $\neq$ n) entonces		
	ele.Clave(n)		
	ele.Resto(pos)		
	t.insTabla(ele)		
	dir(pos) = n		
	pos = pos+1		
	fsi		

Ingresar un nuevo nodo al digrafo.  $O(N)$ .

# Implementación del DigrafoMat



Marzo, 2005

nuevoArco(TipoEle: ni, TipoEle nj)

{pre:  $g(i, j) \geq 0 \quad \forall i, j$ }

{pos:  $g(i, j) \geq 0 \quad \forall i, j$ }

<ol style="list-style-type: none"> <li>1 ele = t.conTabla(ni)</li> <li>2 ele1 = t.conTabla(nj)</li> <li>3 Si (ele.Clave() = ni) <math>\wedge</math> (ele1.Clave() = nj) entonces             <ul style="list-style-type: none"> <li>i = ele.Resto()</li> <li>j = ele1.Resto()</li> <li>g(i, j) = 1</li> </ul> </li> </ol> <p>fsi</p>	<p><b>-t, g:</b> Definido en DigrafoMat.  <b>-ele, ele1:</b> TipoEleTab. Variable auxiliar.  <b>-i, j:</b> Entero: variable con la posicion de los nodos en la matriz-</p>
--	--

Ingresa un nuevo arco en el digrafo.  $O(1+N/B)$

Marzo, 2005

eliNodo(TipoEle: n)

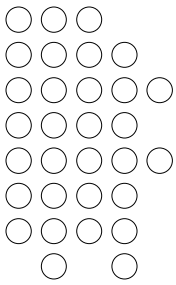
{pre:  $g(i, j) \geq 0 \quad \forall i, j$ }

{pos:  $g(i, j) \geq 0 \quad \forall i, j$ }

<ol style="list-style-type: none"> <li>1 ele = t.conTabla(n)</li> <li>2 Si (ele.Clave() = n)             <ul style="list-style-type: none"> <li>t.elimTabla(n)</li> <li>inicio = ele.Resto()</li> <li>dir(pos) = {TipoNoDef}</li> <li>[g(inicio, j) = 0] j = 0, pos-1</li> <li>[g(j, inicio) = 0] j = 0, pos-1</li> </ul> </li> </ol> <p>fsi</p>	<p><b>-t, g:</b> definido en DigrafoMat.  <b>-ele:</b> TipoEleTab. Variable auxiliar.  <b>-j:</b> Entero: variable con la posicion de los nodos en la matriz</p>
--	--

Elimina un nodo del digrafo.  $O(1+N/B)$

# Implementación del DigrafoMat



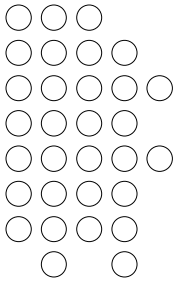
Marzo, 2005		eliArco(TipoEle: ni, TipoEle: nj)
{pre: $g(i, j) \geq 0 \quad \forall i, j$ }		{pos: $g(i, j) \geq 0 \quad \forall i, j$ }
1	ele = t.conTabla(ni)	<b>-t, g:</b> definido en DigrafoMat. <b>-ele, ele1:</b> TipoEleTab. Variable auxiliar. <b>-i, j:</b> Entero: variable con la posición de los nodos en la matriz-
2	ele1 = t.conTabla(nj)	
3	Si (ele.Clave() = ni) $\wedge$ (ele1.Clave() = nj) entonces	
	i = ele.Resto()	
	j = ele1.Resto()	
	g(i, j) = 0	
	fsi	

Elimina un arco del digrafo.  $O(N)$ .

Marzo, 2005		conNodo(TipoEle: n): Lógico
{pre: $g(i, j) \geq 0 \quad \forall i, j$ }		{pos: $g(i, j) \geq 0 \quad \forall i, j$ }
1	ele = t.conTabla(n)	<b>-t:</b> definido en DigrafoMat. <b>-ele:</b> TipoEleTab. Variable auxiliar.
2	regrese (ele.Clave() = n)	

Consultar un nodo del digrafo.  $O(1+N/B)$

# Implementación del DigrafoMat



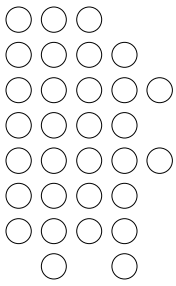
Marzo, 2005		conArco(TipoEle: ni, TipoEle nj):Lógico	
{pre: $g(i, j) \geq 0 \quad \forall i, j$ }		{pos: $g(i, j) \geq 0 \quad \forall i, j$ }	
1	ele, ele1, salir = t.conTabla(ni), t.conTabla(nj), Falso	<b>-t,g:</b> definido en DigrafoMat. <b>-ele,ele1:</b> TipoEleTab. Variable auxiliar. <b>-i,j:</b> Entero: variable con la posicion de los nodos en la matriz- <b>-salir:</b> Lógico. Regresa el resultado de la consulta (1 = verdadero)	
2	Si (ele.Clave() = ni) $\wedge$ (ele1.Clave() = nj) entonces i = ele.Resto() j = ele.Resto() Si (g(i, j) > 0) entonces salir = Verdadero fsi		
3	regresa salir		

Consulta un arco en el digrafo.  $O(1+N/B)$

Marzo, 2005		nodoAdyacente(TipoEle: n):Arreglo[100] de TipoEle	
{pre: $g(i, j) \geq 0 \quad \forall i, j$ }		{pos: $g(i, j) \geq 0 \quad \forall i, j$ }	
1	ele = t.conTabla(n)	<b>-t,g:</b> definido en DigrafoMat. <b>-ele:</b> TipoEleTab. Variable auxiliar. <b>-i,j,k:</b> Entero: variable con la posicion de los nodos en la matriz- <b>-adya:</b> Arreglo[100] de TipoEle. Almacena los nodos que son adyacentes a n.	
2	Si (ele.Clave() = n) entonces i = ele.Resto() [ Si (g(i, j) > 0) entonces adya(k) = dir(k) k = k+ 1 fsi] j = 0, pos-1 fsi		
3	regresa adya		

Busca todos los nodos adyacentes a n.  $O(N)$ .

# Implementación del DigrafoMat



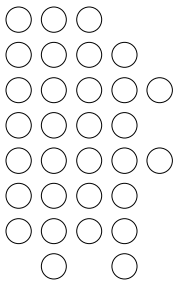
Marzo, 2005		incidenciaEnt(TipoEle: n):Entero
{pre: $g(i, j) \geq 0 \quad \forall i, j$ }		{pos: $g(i, j) \geq 0 \quad \forall i, j$ }
1	ele = t.conTabla(n)	<b>-t, g:</b> Definido en DigrafoMat. <b>-ele:</b> TipoEleTab. Variable auxiliar. <b>-i, j, k:</b> Entero: variable con la posición de los nodos en la matriz-
2	Si (ele.Clave() = n) entonces j = ele.Resto() [Si (g(i, j) > 0) entonces k = k+1 fsi] i = 0, pos-1 fsi	
3	regresa k	

Calcula el grado de incidencia de entrada del nodo n. O(N).

Marzo, 2005		incidenciaSal(TipoEle: n):Entero
{pre: $g(i, j) \geq 0 \quad \forall i, j$ }		{pos: $g(i, j) \geq 0 \quad \forall i, j$ }
1	ele = t.conTabla(n)	<b>-t, g:</b> Definido en DigrafoMat. <b>-ele:</b> TipoEleTab. Variable auxiliar. <b>-i, j, k:</b> Entero: variable con la posición de los nodos en la matriz-
2	Si (ele.Clave() = n) entonces j = ele.Resto() [Si (g(i, j) > 0) entonces k = k+1 fsi] j = 0, pos-1 fsi	
3	regresa k	

Calcula el grado de incidencia de salida del nodo n. O(N).

# Implementación del DigrafoMat

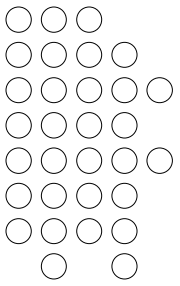


Marzo, 2005		numNodos():Entero	{pre: $g(i, j) \geq 0 \quad \forall i, j$ }	{pos: $g(i, j) \geq 0 \quad \forall i, j$ }
1	regrese pos	<b>-pos:</b> Definido en DigrafoMat.		

Regresa el número de nodos almacenados en el digrafo O(1)

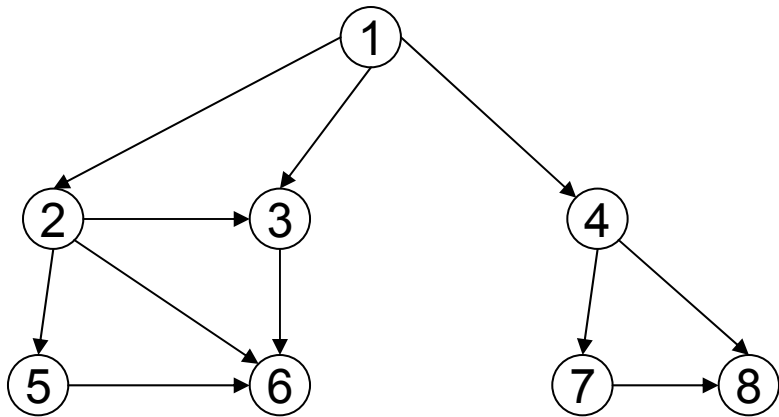
Marzo, 2005		Especificación Cola[TipoEle]
1	<b>Especificación sintáctica:</b> creaCola() Cola, enCola(Cola, TipoEle) Cola, desenCola(Cola) Cola, primero(Cola) TipoEle, vaciaCola(Cola) Lógico, destruyeCola(Cola) .	<b>-creaCola():</b> Crea una cola vacía. <b>-enCola():</b> Ingresa un nuevo elemento a la cola por el fin de la misma. <b>-desenCola():</b> Elimina el elemento que está actualmente en el inicio de la cola. Si la cola está vacía no hace ninguna eliminación. <b>-destruyeCola():</b> Destruye la cola. <b>-vaciaCola():</b> Regresa Verdadero si la cola está vacía. <b>-primero():</b> Devuelve el elemento que se encuentra actualmente en el inicio de la cola. Si la cola está vacía devuelve un valor especial que lo indica.
2	<b>Declaraciones:</b> TipoEle: e, {TipoEleNoDef}	
3	<b>Especificación semántica:</b> vaciaCola(creaCola())=Verdadero vaciaCola(enCola(creaCola(), e))=Falso primero(creaCola())=TipoEleNoDef desenCola(creaCola())=creaCola()	

# Búsqueda en amplitud del DigrafoMat

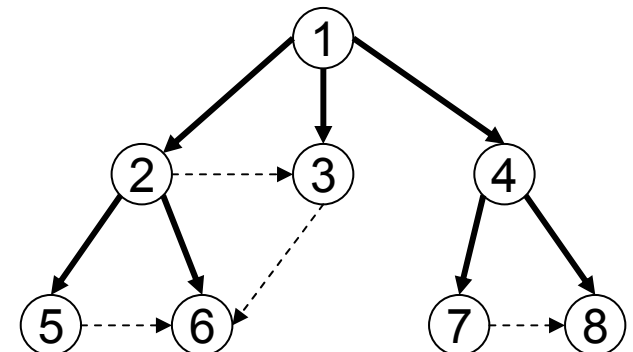


## ➤ Usos:

- ❖ Para recorridos parciales de grafos infinitos o tan grandes que son inmanejables
- ❖ Encontrar los caminos más cortos desde un nodo origen hasta los otros nodos

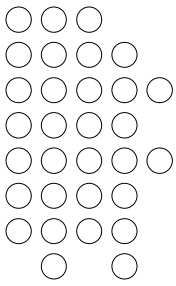


Digrafo DG



Recorrido en amplitud:  
arcos con líneas continuas

# Búsqueda en amplitud del DigrafoMat

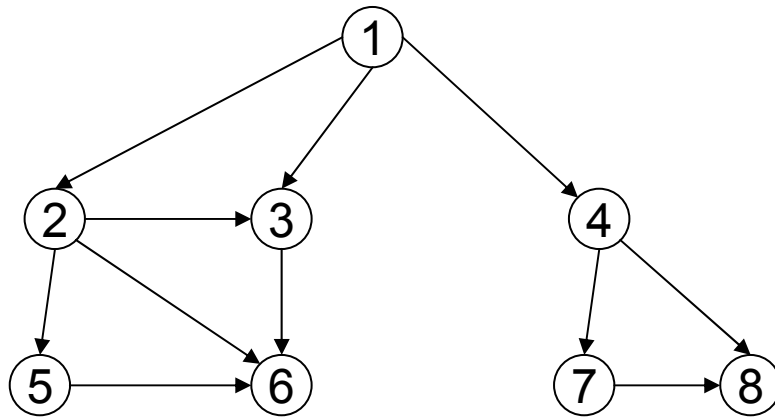
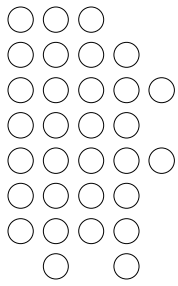


Marzo, 2005		
recorridoEnAmp( )		
{pre: $g(i, j) \geq 0 \quad \forall i, j$ } <span style="float: right;">{pos: <math>\text{marca}(i) = \text{Verdadero} \quad \forall i</math> }</span>		
1	[ $\text{marca}(i) = \text{falso}$ ] $i = 1, \text{pos}$	<b>-pos:</b> Definido en DigrafoMat. <b>-marca:</b> Arreglo[100]De Lógico. Marca que indica si el nodo fue visitado (Verdadero) o no (Falso). <b>-i:</b> Entero: Variable con la posición de los nodos en la matriz
2	[ Si ( $\neg \text{marca}(i)$ ) entonces busAmp( $i, \text{marca}$ ) fsi ] $i = 1, \text{pos}$	

Marzo, 2005		
busAmp(Entero+: na, Arreglo[100]De Lógico: mar )		
{pre: $1 \leq \text{na} \leq \text{pos}$ } <span style="float: right;">{pos: <math>\text{marca}(i) = \text{Verdadero} \quad \forall i</math> }</span>		
1	mar( na ) = Verdadero	<b>-pos, nodoAdyacente( ):</b> Definidos en DigrafoMat. <b>-mar:</b> Arreglo[100]De Lógico. Marca que indica si el nodo fue visitado (Verdadero) o no (Falso). <b>-i:</b> Entero+: Variable con la posición de los nodos en la matriz <b>-co:</b> Cola[Entero+]. Cola que mantiene los nodos no tratados. <b>-nadya:</b> Arreglo[100]De Entero+. Vector auxiliar que contiene los nodos adyacentes <b>-enCola(), primero(), desenCola(), vaciaCola().</b> Operaciones de la clase Cola.
2	co.enCola( na )	
3	( $\neg \text{co.vaciaCola}()$ ) [ na = co.primeros( ) co.desenCola( ) nadya = nodoAdyacente( na ) [ Si ( $\neg \text{mar}( \text{nadya}(i) )$ ) entonces mar( nadya( i ) ) = Verdadero fsi co.enCola( nadya( i ) ) ] $i = 1, \text{pos}$ ]	

$$T(n) = O(N + A) = \Theta(\text{máx}(N, A))$$

# Recorrido en amplitud del DigrafoMat



g

0	1	1	1	0	0	0	0
0	0	1	0	1	1	0	0
0	0	0	0	0	1	0	0
0	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0

Condiciones  
iniciales

nadya 

2	3	4
---	---	---

marca 

F	F	F	F	F	F	F	F
---	---	---	---	---	---	---	---

na = 1

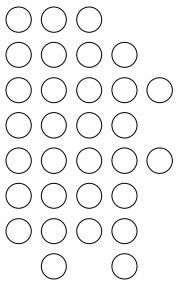
Estado intermedio luego de  
tener los nodos adyacentes  
al nodo 1

marca 

V	V	V	V	V	V	V	V
---	---	---	---	---	---	---	---

Condiciones  
finales

# Implementación del DigrafoLis



Marzo, 2005

vacíoDigrafo(): Lógico

{pre:  $g.n \geq 0$ }

{pos:  $g.n \geq 0$ }

1

regrese g.vacíaLista()

-g: Definido en DigrafoLis.

Verifica si el digrafo esta vacío  $O(1)$

Marzo, 2005

nuevoNodo(TipoEle: n)

{pre:  $g.n \geq 0$ }

{pos:  $g.n \geq 0$ }

1

Si ( $\neg$ conNodo(n) ) entonces  
ele.Infor( n )  
g.insLista( ele )  
fsi

-g: Definido en DigrafoLis.  
-ele: Nodo1[TipoEle]. Variable auxiliar.

Ingresa un nuevo nodo al digrafo.  $O(N)$ .

Marzo, 2005

nuevoArco(TipoEle: ni, TipoEle: nj)

{pre:  $g.n \geq 0$ }

{pos:  $g.n \geq 0$ }

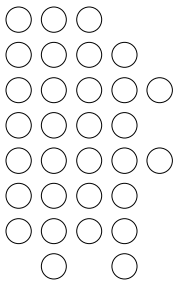
1

Si (conNodo(nj)  $\wedge$  conNodo(ni)) entonces  
g.conLista().ListaAdy insLista( nj )  
fsi

-g: Definido en DigrafoLis.

Ingresa un nuevo arco en el digrafo.  $O(N)$

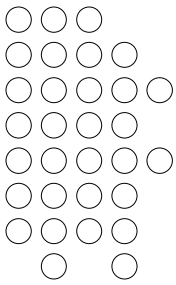
# Implementación del DigrafoLis



Marzo, 2005		
eliNodo(TipoEle: n)		
{pre: g.n ≥ 0 }	{pos: g.n ≥ 0 }	
1	<p>Si ( conNodo(n) ) entonces</p> <p>g.conLista().ListaAdya().cursorAllInicio()</p> <p>(¬ g.conLista().ListaAdya().vacíaLista() ) [g.conLista().ListaAdya().eliLista() ]</p> <p>g.eliLista()</p> <p>g.cursorAllInicio()</p> <p>[ g.conLista().ListaAdya().cursorAllInicio()</p> <p>  i = 1</p> <p>  (g.conLista().ListaAdya().conLista().Info() ≠ n ∧ i ≤ g.conLista().ListaAdya().numEle())</p> <p>    [ i = i+1</p> <p>      g.conLista().ListaAdya().cursorAlProximo() ]</p> <p>  Si ( i ≤ g.conLista().ListaAdya().numEle() ) entonces</p> <p>    g.conLista().ListaAdya().eliLista()</p> <p>  fsi</p> <p>  g.cursorAlProximo()</p> <p>  ] j = 1, g.numEle()</p> <p>fsi</p>	<p><b>-g:</b> Definido en DigrafoLis.</p> <p><b>-i, j:</b> Entero: variable auxiliar.</p>

Elimina un nodo del digrafo.  $O(|N| + |A|)$

# Implementación del DigrafoLis



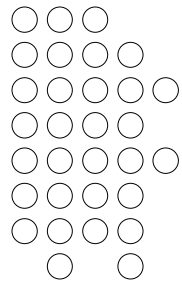
Marzo, 2005		eliArco(TipoEle: ni, TipoEle: nj)	
{pre: g.n ≥ 0 }		{pos: g.n ≥ 0 }	
1	Si ( conNodo(nj) ∧ conNodo(ni) ) entonces i = 1 (g.conLista().ListaAdya().conLista().Info() ≠ nj ∧ i ≤ g.conLista().ListaAdya().numEle()) [ i = i + 1 g.conLista().ListaAdya().cursorAlProximo() Si ( i ≤ g.conLista().ListaAdya().numEle() ) entonces g.conLista().ListaAdya().eliLista() fsi fsi	<b>-g:</b> Definido en DigrafoLis. <b>-i:</b> Entero. Variable con auxiliar.	

Elimina un arco del digrafo. O(N).

Marzo, 2005		conNodo(TipoEle: n): Lógico	
{pre: g.n ≥ 0 }		{pos: g.n ≥ 0 }	
1 2 3 4	g.cursorAlInicio() i = 1 ( i ≤ g.numEle() ) [ Si ( g.conLista().Infor() = n ) entonces regrese Verdadero sino g.cursorAlProximo() i = i + 1 fsi ] regrese Falso	<b>-g:</b> Definido en DigrafoLis. <b>-i:</b> Entero: Variable con auxiliar para recorrer los nodos del digrafo.	

Consulta un nodo del digrafo. O(N).

# Implementación del DigrafoLis



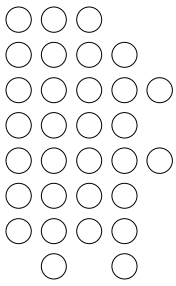
Marzo, 2005		
conArco(TipoEle: ni, TipoEle: nj): Lógico		
{pre: g.n ≥ 0 }		{pos: g.n ≥ 0}
1 2	<p>Si ( conNodo( ni ) ) entonces</p> <p>    i = 1</p> <p>    ( i ≤ g.conLista().ListaAdya().numEle() )</p> <p>        [ Si ( g.conLista().ListaAdya().conLista().Info() = nj ) entonces</p> <p>            regrese Verdadero</p> <p>            sino</p> <p>                i = i + 1</p> <p>                g.conLista().ListaAdya().cursorAlProximo()</p> <p>        fsi]</p> <p>fsi</p> <p>regrese Falso</p>	<p><b>-g:</b> Definido en DigrafoLis.</p> <p><b>-i:</b> Entero: variable auxiliar</p>

Consulta un arco en el digrafo. O(N)

Marzo, 2005		
nodoAdyacente(TipoEle: n): Lista[TipoEle]		
{pre: g.n ≥ 0 }		{pos: g.n ≥ 0}
1 2	<p>Si ( conNodo(n) ) entonces</p> <p>    adya = g.conLista().ListaAdya()</p> <p>fsi</p> <p>regresa adya</p>	<p><b>-g:</b> definido en Digrafo.</p> <p><b>-adya:</b> Lista[TipoEle]. Almacena los nodos que son adyacentes a n.</p>

Busca todos los nodos adyacentes a n. O(N).

# Implementación del DigrafoLis



Marzo, 2005

incidenciaEnt(TipoEle: n): Entero

{pre:  $g.n \geq 0$ }

{pos:  $g.n \geq 0$ }

```

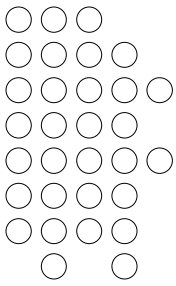
1  k = 0
2  g.cursorAlInicio()
   [ g.conLista().ListaAdya().cursorAlInicio()
     i = 1
     salir = Falso
     ( i ≤ g.conLista().ListaAdya().numEle() ∧ ¬salir )
       [ Si (g.conLista().ListaAdya().conLista().Info() = n) entonces
         salir = Verdadero
         k = k + 1
       sino
         i = i + 1
         g.conLista().ListaAdya().cursorAlProximo()
       fsi ]
     g.cursorAlProximo()
   ] j = 1, g.numEle()
3  regresa k

```

-**g**: Definido en DigrafoLis.  
 -**salir**: Lógico. Variable auxiliar para controlar la búsqueda.  
 -**i**: Entero: Variable auxiliar.  
 -**k**: Entero: Grado de incidencia de entrada del nodo.

Calcula el grado de incidencia de entrada del nodo n.  $O(|N|+|A|)$ .

# Implementación del DigrafoLis

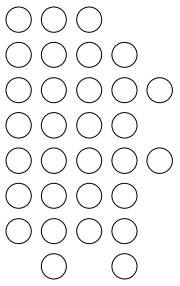


Marzo, 2005		incidenciaSal(TipoEle: n): Entero	{pre: g.n ≥ 0 }	{pos: g.n ≥ 0}
1	Si ( conNodo(n) ) entonces regrese g.conLista().ListaAdya().numEle()	-g: Definido en DigrafoLis.		
2	fsi regresa -1			

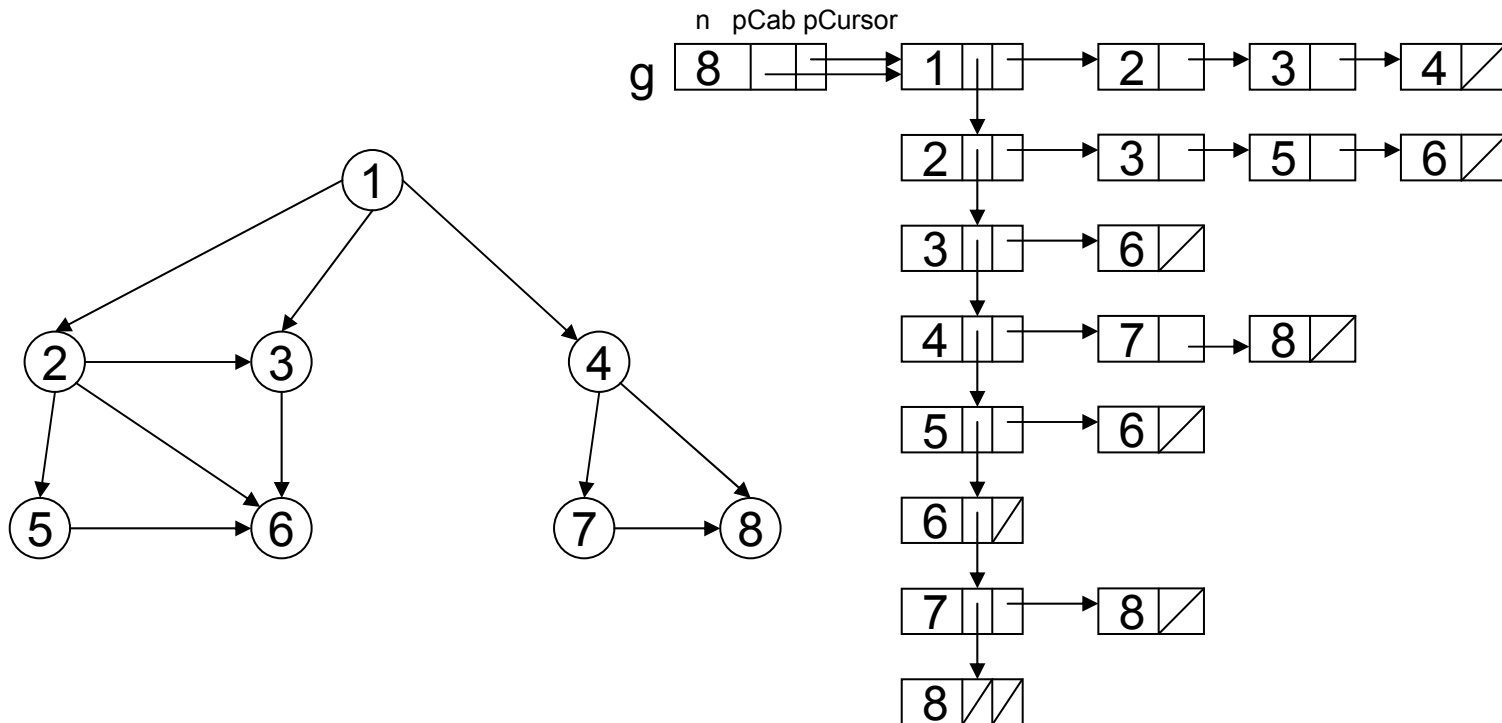
Calcula el grado de incidencia de salida del nodo n.  $O(N)$ .

Para la búsqueda en amplitud utilizando listas de adyacencia, se utiliza también la clase **Cola**, con el propósito de ir almacenando los nodos no tratados que son adyacentes al nodo actual (na).

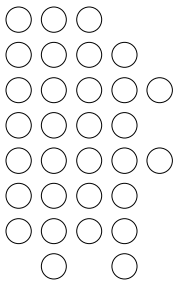
# Búsqueda en amplitud del DigrafoLis



- Se recorre la lista de adyacencia de cada nodo



# Búsqueda en amplitud del DigrafoLis



Marzo, 2005

recorridoEnAmp( )

{pre:  $g.n \geq 0$  }

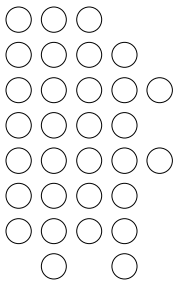
{pos:  $\text{marca}(i) = \text{Verdadero} \forall i$ }

1	[ $\text{marca}(i) = \text{falso}$ ] $i = 1, g.\text{numEle}()$
2	[ Si ( $\neg \text{marca}(i)$ ) entonces $\text{busAmp}(i, \text{marca})$ fisi ] $i = 1, g.\text{numEle}()$

- $g$ : Definido en DigrafoLis
- **marca**: Arreglo[100]De Lógico. Marca que indica si el nodo fue visitado (Verdadero) o no (Falso).
- **i**: Entero: Variable auxiliar para los nodos

$$T(n) = O(N + A) = \Theta(\text{máx}(N, A))$$

# Búsqueda en amplitud del DigrafoLis



Marzo, 2005

busAmp(Entero+: na, Arreglo[100]De Lógico: mar )

{pre: g.n ≥ 0 }

{pos: marca ( i ) = Verdadero ∀ i}

```

1  mar( na ) = Verdadero
2  co.enCola( na )
3  ( ¬ co.vaciaCola( ) ) [ na = co.primerO( )
    co.desenCola( )
    nadya = nodoAdyacente( na )
    nadya.cursorAlInicio( )
    [ Si (¬mar(nadya.conLista().Info())) entonces
      mar(nadya.conLista().Info()) = Verdadero
    fsi
    co.enCola( nadya.conLista().Info())
    nadya.cursorAlProximo( )
    ] i = 1, nadya.numEle( )
  ]

```

**-g, nodoAdyacente( ):** Definidos en DigrafoLis

**-mar:** Arreglo[100]De Lógico. Marca que indica si el nodo fue visitado (Verdadero) o no (Falso).

**-i:** Entero+: Variable con la posición de los nodos en la matriz

**-co:** Cola[Entero+]. Cola que mantiene los nodos no tratados.

**-nadya:** Lista[Entero+]. Lista auxiliar que contiene los nodos adyacentes

**-enCola( ), primero( ), desenCola( ), vaciaCola( ).** Operaciones de la clase Cola.

**-cursorAlInicio( ), numEle( ), conLista( ), cursorAlProximo( ).** Definidos en Lista