Tema 9. Grafos

FUNDAMENTOS DE PROGRAMACIÓN II

Profesora: Raquel Martínez España

Escuela Politécnica

Contenidos

- Introducción
- Terminología
- Representación
- Problemas tipicos
  - Dijkstra
  - Floyd
  - Warshall

Contenidos

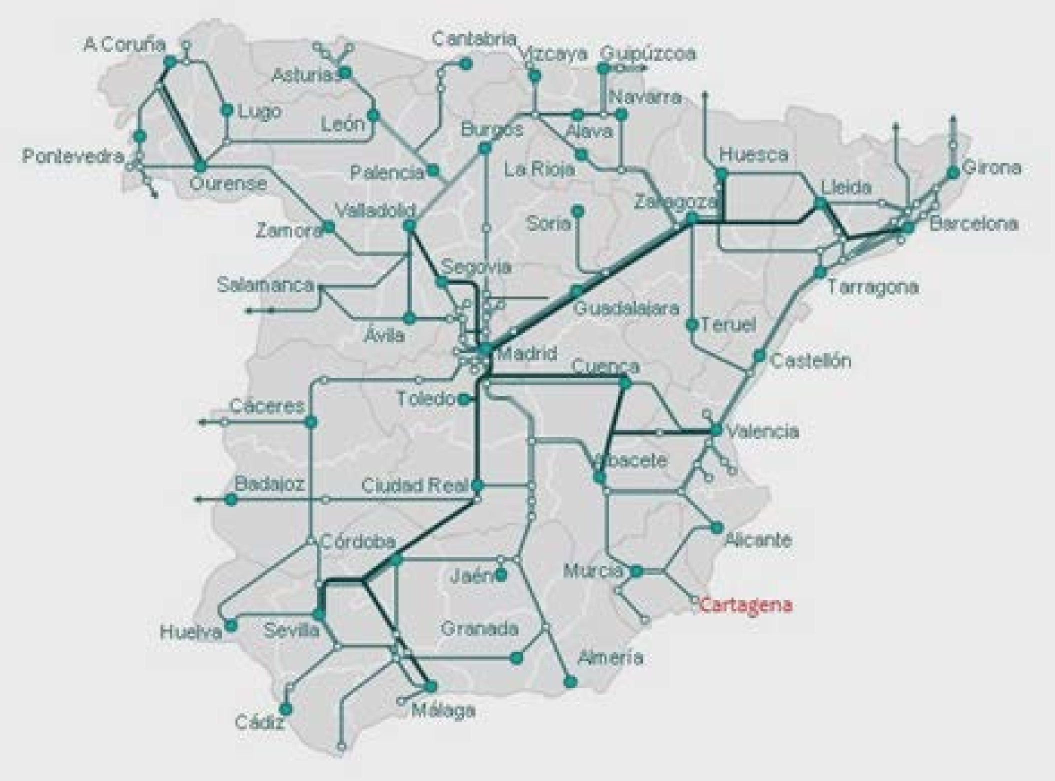
- **Introducción**
- Terminología
- Representación
- Problemas tipicos
  - Dijkstra
  - Floyd
  - Warshall

Grafos

- Estructura de datos no lineal
- Cada nodo tiene
  - Uno o más predecesores
  - Uno o más sucesores
- A los nodos de un grafo también se les conoce como **vértices**.
- Los nodos están unidos por **aristas** o **arcos**.

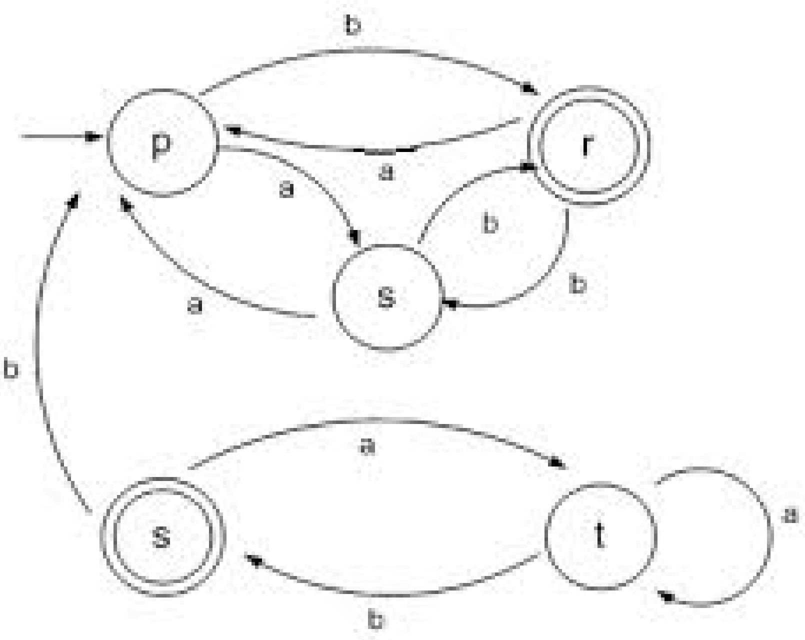
Grafos

- Ejemplo: Red de carreteras de un país.
  - Un arco = carretera entre dos localidades.
  - Se asocia un peso al arco.



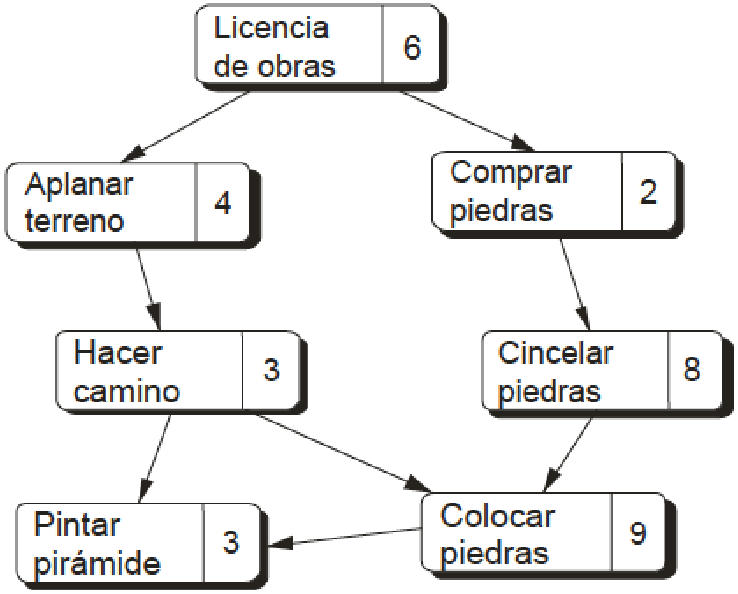
Grafos

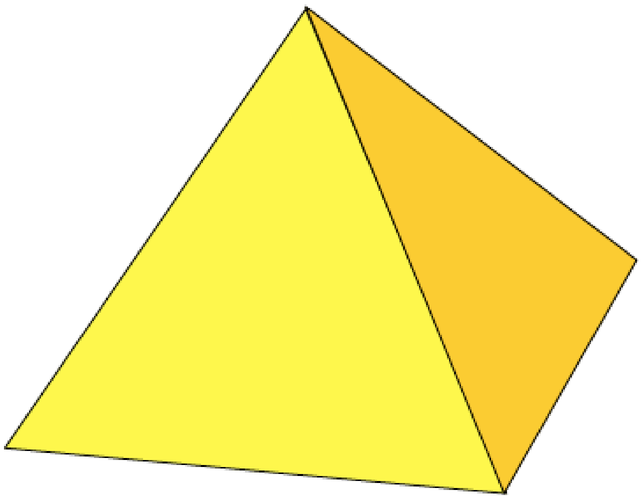
- Ejemplo: Un autómata finito determinista
  - Un arco = transición entre dos estados
  - Se asocia el evento que genera la transición al arco



Grafos

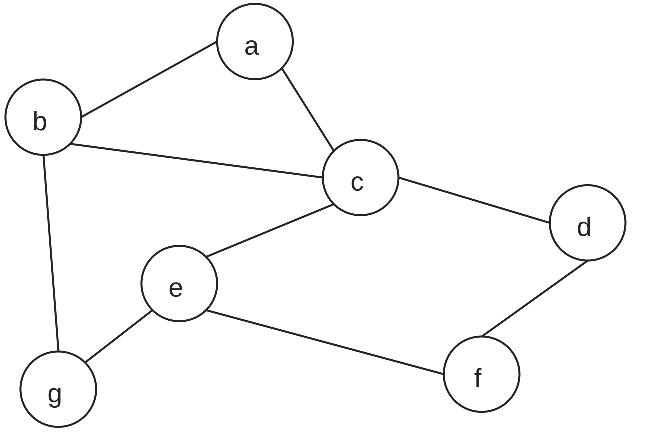
- Ejemplo: Planificación de tareas
  - Cada nodo es una tarea a realizar
  - Un arco indica relación de dependencia entre tareas.



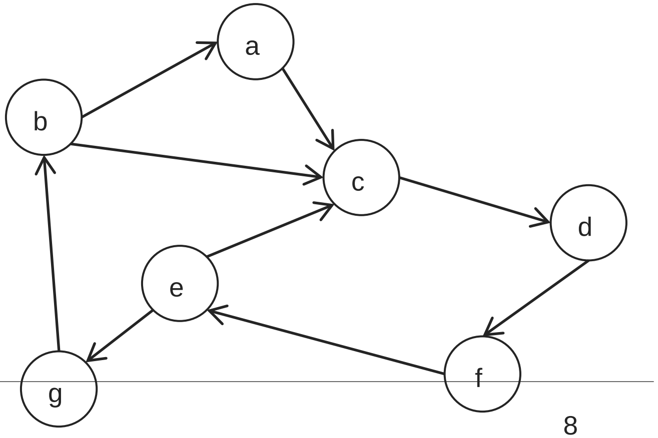


Grafos

- Tipos de grafos:
  - **No dirigido**
    - No existe dirección entre los nodos



  - **Dirigido**
    - Existe direccionalidad entre los nodos



Contenidos

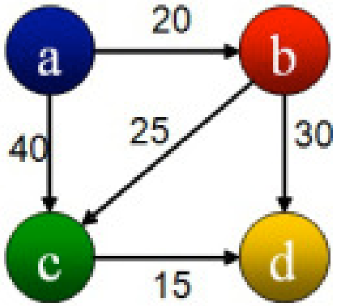
- Introducción
- **Terminología**
- Representación
- Problemas tipicos
  - Dijkstra
  - Floyd
  - Warshall

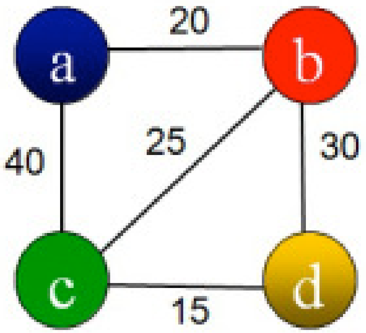
Terminología de grafos

- **Nodos adyacentes** a un nodo v:
  - Todos los nodos unidos a v mediante una arista.
  - En grafos dirigidos:
    - Nodo adyacente a v: todos los nodos w tal que existe un arco de w a v (v ➔w)
    - Nodo adyacente de v: todos los nodos p tal que existe un arco de w a v (p ➔v)

Terminología de grafos

- **Grafo etiquetado**:
  - Si cada arista tiene asociada una etiqueta o valor de cierto tipo.
- **Grafo con pesos**:
  - Es un grafo etiquetado donde cada etiqueta representa un valor numérico.





Terminología de grafos

- **Camino de un nodo w_1 a w_q**:
  - Secuencia de nodos $w_1, w_2, \ldots, w_q$ tal que existe una arista entre cada w_i y w_{i+1}
- **Longitud de un camino**:
  - Número de aristas que conforman el camino
  - También nº de nodos – 1

Terminología de grafos

- **Camino simple**:
  - Aquel en el que todos lo nodos son distintos (excepto el primero y el último que pueden ser iguales)
- **Ciclo**:
  - Camino en el cual el primer y último vértice son iguales.
  - Si el grafo no es dirigido, las aristas deben ser diferentes.

Terminología de grafos

- **Nodos conectados**:
  - Dos nodos de un grafo se dice que están conectados si existe un camino entre ellos.
- **Grafo conexo**:
  - Un grafo es conexo (o conectado) si hay un camino entre cualquier par de vértices.
- **Grafo completo**:
  - Si existe una arista entre cualquier par de nodos.

Terminología de grafos

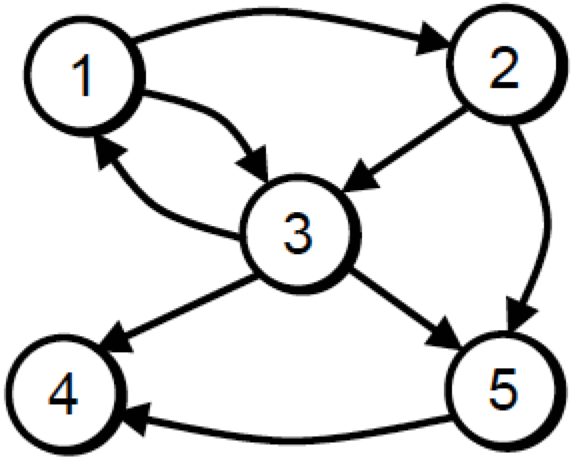
- **Grado de un vértice**:
  - Número de arcos que inciden en él.
  - Para grafos dirigidos:
    - *Grado de entrada*: número de aristas dirigidas hacia v
    - *Grado de salida*: número de aristas dirigidas desde v.

Contenidos

- Introducción
- Terminología
- **Representación**
- Problemas tipicos
  - Dijkstra
  - Floyd
  - Warshall

Aspectos a tener en cuenta

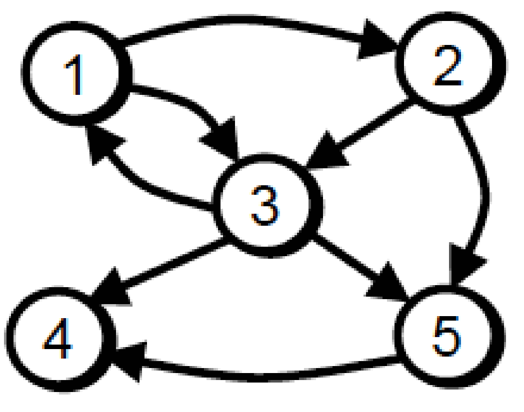
- Necesitamos representar los nodos y las aristas que existen en un grafo.
- Distinguir entre grafos dirigidos y no dirigidos.



Matrices de adyacencia

- Matriz de adyacencia A
- A[i,j] = true si existe un arco entre el nodo 'i' y el 'j'
- Se pueden definir las etiquetas de los arcos mediante estructuras en la posición de la matriz.

M	1	2	3	4	5
1		T	T		
2			T		T
3	T			T	T
4					
5				T	



Matrices de adyacencia

- Matriz de adyacencia (no dirigido)

	a	b	c	d	e	f	g
a		T	T				
b	T		T				T
c	T	T		T	T		
d			T			T	
e			T			T	T
f				T	T		
g		T			T		

Matrices de adyacencia

- Matriz de adyacencia (no dirigido)

	a	b	c	d	e	f	g
a		T	T				
b	T		T				T
c	T	T		T	T		
d			T			T	
e			T			T	T
f				T	T		
g		T			T		

Matrices de adyacencia

- Matriz de adyacencia (no dirigido)

	a	b	c	d	e	f	g
a		T	T				
b	T		T				T
c	T	T		T	T		
d			T			T	
e			T			T	T
f				T	T		
g		T			T		

Matrices de adyacencia

- Matriz de adyacencia (no dirigido)

	a	b	c	d	e	f	g
a		T	T				
b	T		T				T
c	T	T		T	T		
d			T			T	
e			T			T	T
f				T	T		
g		T			T		

Matrices de adyacencia

- Matriz de adyacencia (no dirigido)

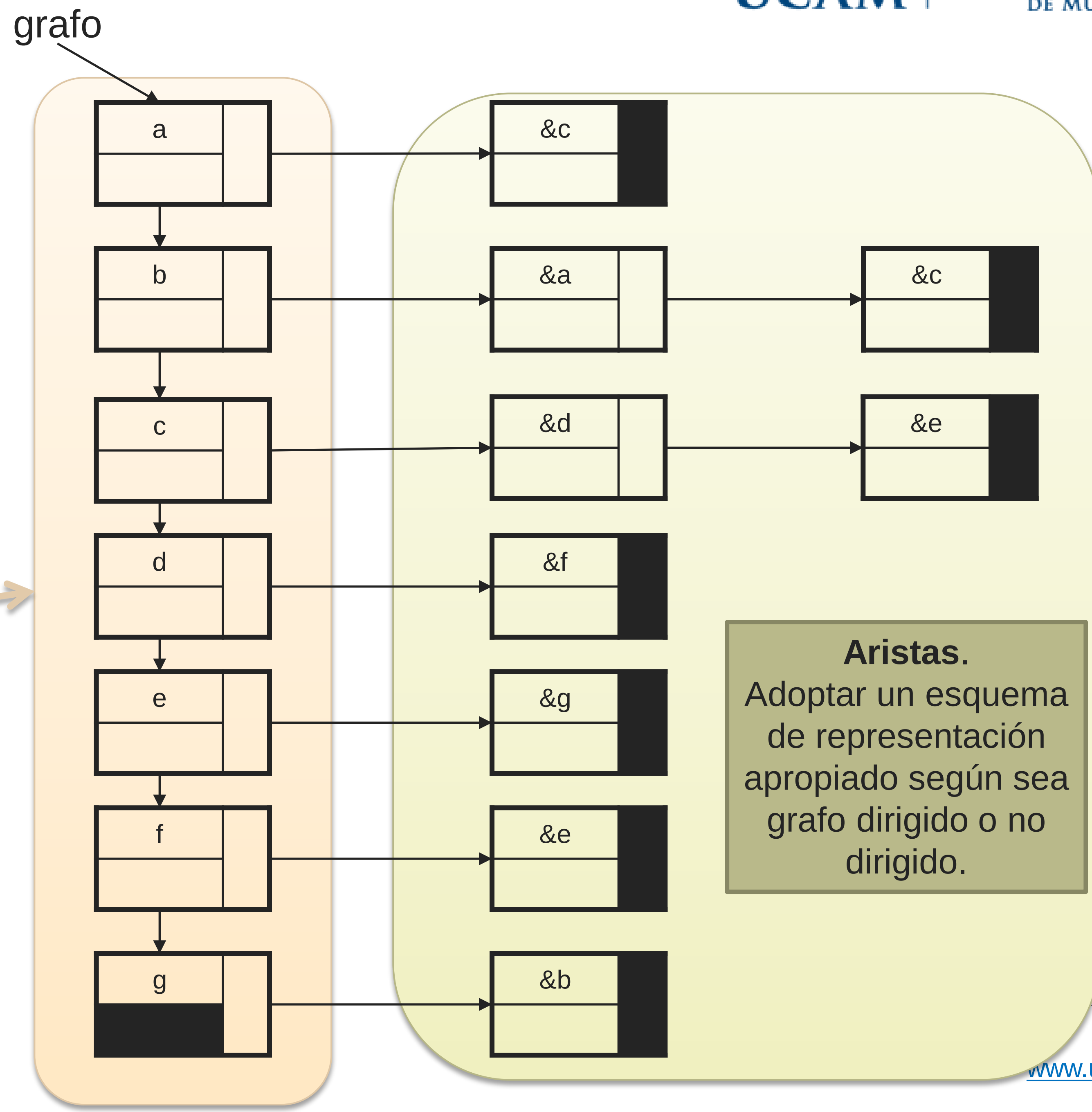
	a	b	c	d	e	f	g
		T	T				
			T				T
				T	T		
						T	
						T	T

Hay una redundancia de información
“Sobra” toda esta parte

Listas de adyacencia

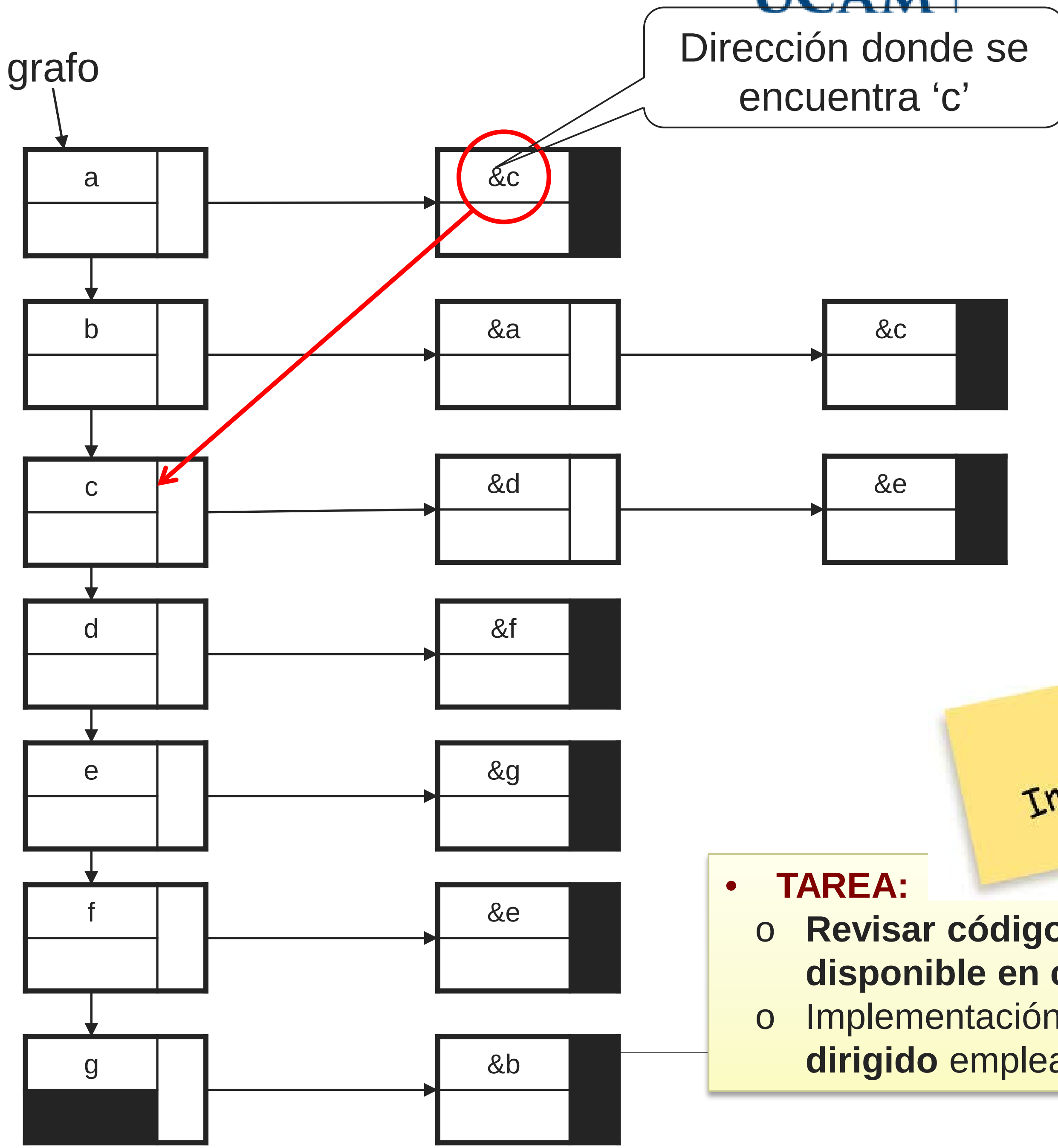
- Un grafo también se puede representar mediante listas
  - Es irrelevante el orden en que aparecen los nodos en la lista principal

(*Ver siguiente transparencia*)



Nodos del grafo

Aristas.
Adoptar un esquema de representación apropiado según sea grafo dirigido o no dirigido.



- **TAREA:**
  - **Revisar código disponible en campus**
  - Implementación de **grafo dirigido** empleando listas

Contenidos

- Introducción
- Terminología
- Representación
- **Problemas tipicos**
  - Dijkstra
  - Floyd
  - Warshall

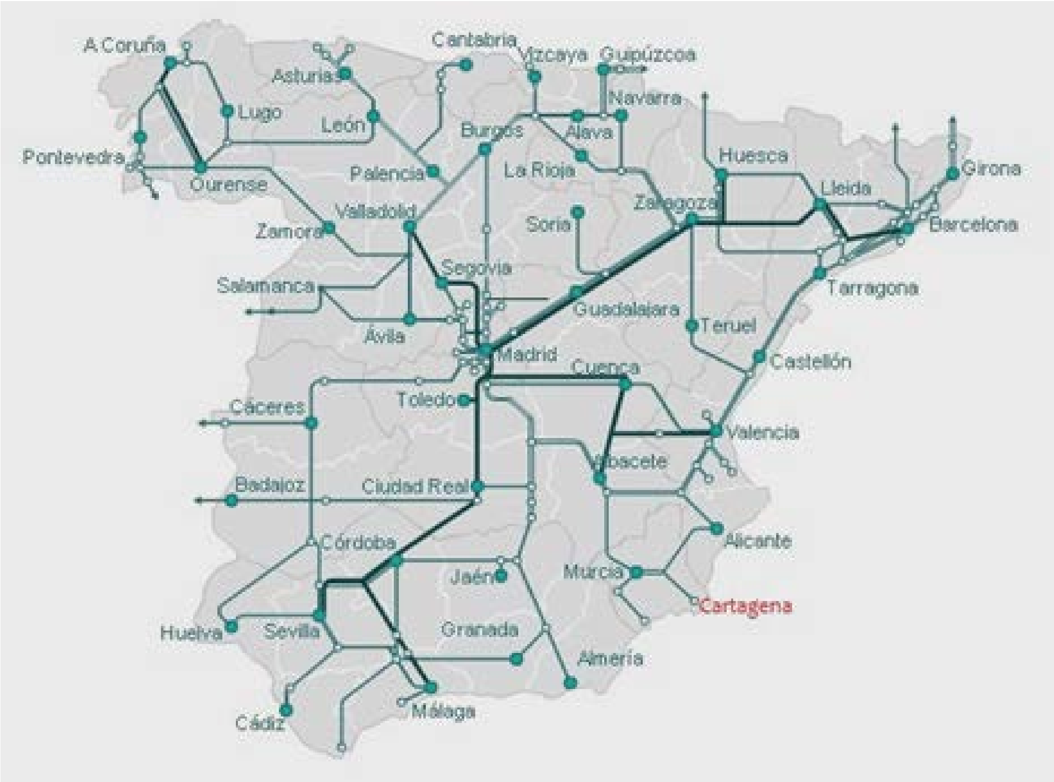
Resolución de problemas sobre grafos

- Los grafos son estructuras de datos sobre las que se presentan problemas recurrentes:
  - Recorridos óptimos
  - Encontrar caminos mínimos
  - Determinar si un grafo es conexo
  - Etc.

Resolución de problemas sobre grafos

- Ejemplo: grafo de carreteras entre ciudades

- ¿Cuál es el camino más corto de Murcia a Badajoz?
- ¿Cuál es la ciudad más lejana de Barcelona?
- ¿Existe un camino entre todos los pares de ciudades?
- ¿Cuántos caminos distintos existen de Sevilla a Zaragoza?
- ¿Cómo hacer un tour por todas las ciudades en el menor tiempo posible?



Problemas de caminos mínimos

- Algoritmos:
  - **Dijkstra**: caminos más cortos desde un origen dado.
  - **Floyd**: encontrar los caminos mínimos entre todos los pares de nodos del grafo.
  - **Warshall**: cierre transitivo de la conectividad.

Contenidos

- Introducción
- Terminología
- Representación
- Problemas tipicos
  - **Dijkstra**
  - Floyd
  - Warshall

Grafos: Dijkstra

- Problema a resolver:
  - Encontrar la forma más económica de moverse desde un nodo origen a cualquier otro.
- Aplicable sobre grafo etiquetado con **pesos positivos**.
- Necesita como entrada un **nodo origen**.

Grafos: Dijkstra

- Elementos:
  - **N nodos** numerados de 0 a n-1. Siendo <u>*0 el nodo origen*</u>.
  - **Matriz de costes C[i,j]**: coste de ir del nodo ‘i’ al ‘j’.
    - No negativos.
    - Si no se puede ir del nodo ‘i’ al ‘j’ entonces C[i,j] = ∞
  - **Vector de costes D[i]**: array con costes mínimos para ir del nodo origen al nodo ‘i’.
    - Inicialmente D[i] = C[0,i]

Grafos: Dijkstra

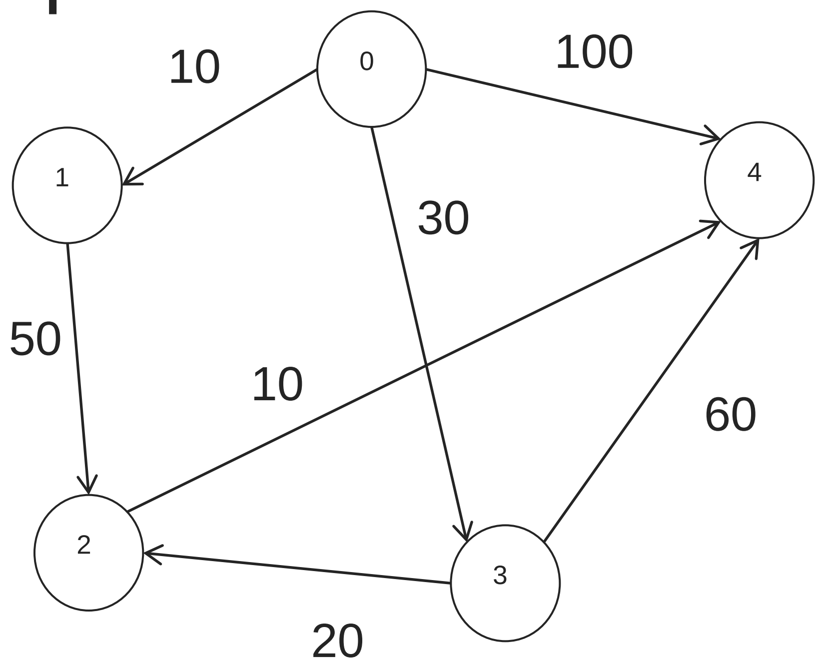
- Esquema del algoritmo (*pseudo*-*C*)

```
void Dijkstra(unsigned int c[][], unsigned int d[], unsigned int
nNodos){
        ConjuntoDeEnteros U; //conjunto universal de nodos
        ConjuntoDeEnteros S; //Conjunto universal de nodos
        int v, w;                  //Números de nodos

        U={};
        for (i = 1; i< nNodos; i++)
                D[i] = C[0,i];
        repetir (nNodos -1) veces{
                w = elegir un nodo en U-S sea mínimo
                S = S ∪ { w };
                para cada nodo v en U-S
                        D[v] = min(D[v], D[w] + C[w,v]);
        }
}
```

Grafos: Dijkstra

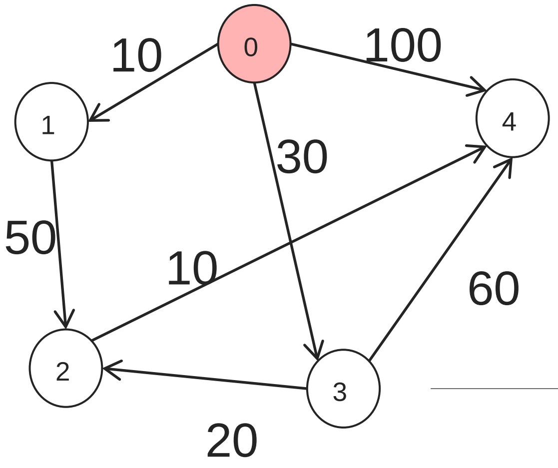
- Ejemplo



Grafos: Dijkstra

- Ejemplo

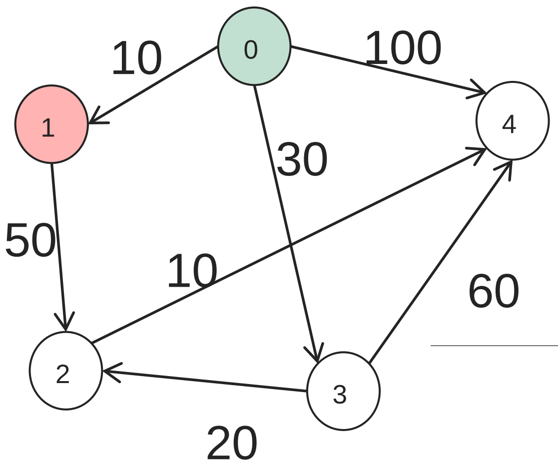
Iteración	S	w	D[1]	D[2]	D[3]	D[4]
inicial	{0}	-	10	∞	30	100



Grafos: Dijkstra

- Ejemplo

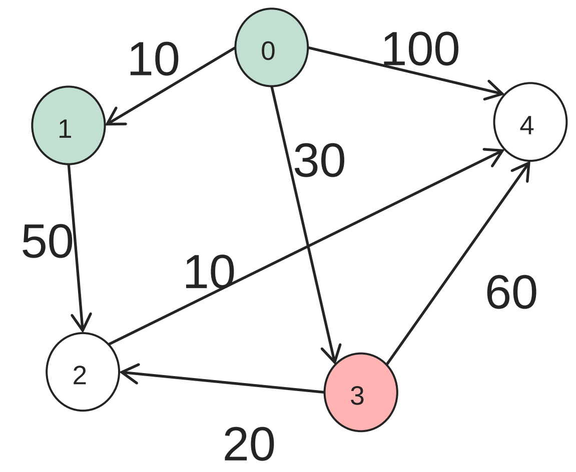
Iteración	S	w	D[1]	D[2]	D[3]	D[4]
inicial	{0}	-	10	∞	30	100
1	{0,1}	1	10	60	30	100



Grafos: Dijkstra

- Ejemplo

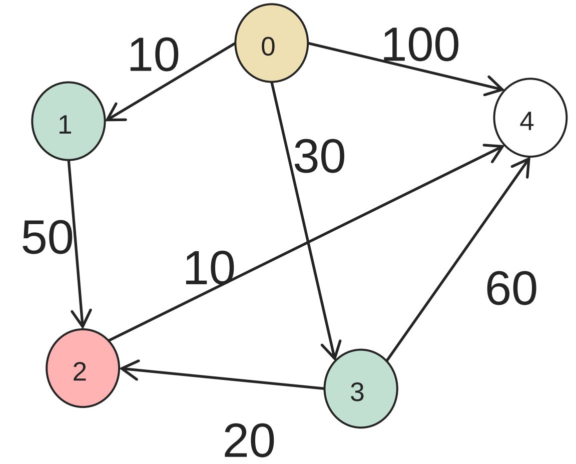
Iteración	S	w	D[1]	D[2]	D[3]	D[4]
inicial	{0}	-	10	∞	30	100
1	{0,1}	1	10	60	30	100
2	{0,1,3}	3	10	50	30	90



Grafos: Dijkstra

- Ejemplo

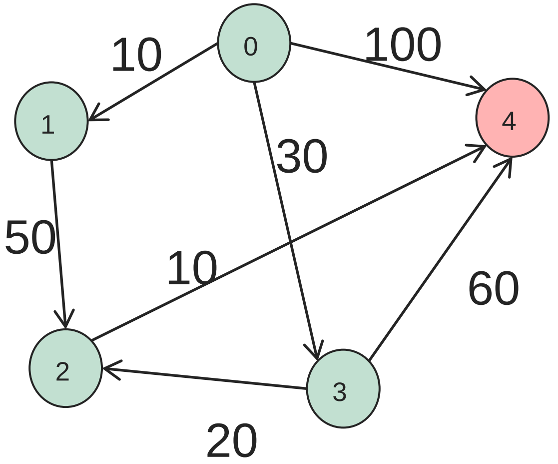
Iteración	S	w	D[1]	D[2]	D[3]	D[4]
inicial	{0}	-	10	∞	30	100
1	{0,1}	1	10	60	30	100
2	{0,1,3}	3	10	50	30	90
3	{0,1,3,2}	2	10	50	30	60



Grafos: Dijkstra

- Ejemplo

Iteración	S	w	D[1]	D[2]	D[3]	D[4]
inicial	{0}	-	10	∞	30	100
1	{0,1}	1	10	60	30	100
2	{0,1,3}	3	10	50	30	90
3	{0,1,3,2}	2	10	50	30	60
4	{0,1,3,2,4}	4	10	50	30	60



Contenidos

- Introducción
- Terminología
- Representación
- Problemas tipicos
  - Dijkstra
  - **Floyd**
  - Warshall

Grafos: Floyd

- Problema: calcular los caminos mínimos entre todos los pares de nodos del grafo.
- Elementos:
  - **Matriz de costes no negativos C[i,j].**
  - **Matriz de costes caminos mínimos A[i,j]**: coste mínimo entre cualquier par de nodos ‘i’ y ‘j’.
  - **Matriz de nodos P[i,j]**: número de un nodo de paso que se encuentra en algún punto intermedio del camino mínimo de “i” a “j”

Grafos: Floyd

```
void mas_corto(unsigned int c[][], unsigned int a[][], int P[][],
unsigned int nNodos){
        int i,j,k;

        for (i = 0; i < nNodos; i++){
            for(j=0; j < nNodos; j++){
                // Inicializamos con el coste de los caminos directos
                A[i][j] = C[i][j];
                 P[i][j] = -1;
            }
        }
        for (k = 0; k < nNodos; k++)
            for (i = 0; i < nNodos; i++)
                for (j=0; j< nNodos; j++)
                    if (A[i][k]+A[k][j] < A[i][j]){
                        A[i][j] = A[i][k] + A[k][j];
                        P[i][j] = k;
                    }
}
```

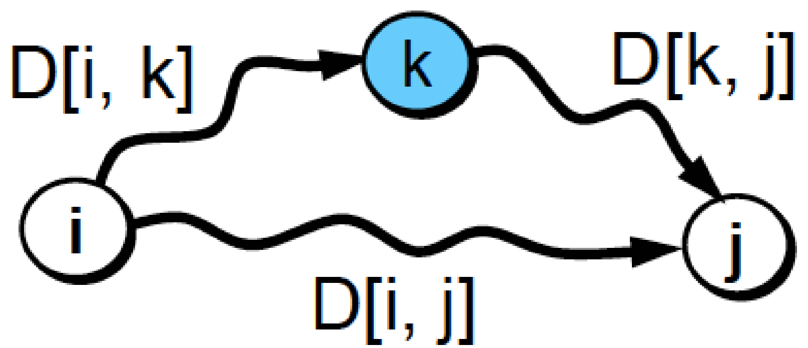
¿En qué se basa el algoritmo de Floyd?

- En cada paso **k**, la matriz **A** almacena los caminos mínimos entre todos los pares, pudiendo pasar por los **k** primeros nodos.
- **Inicialización**: A almacena caminos directos
- **Paso 1**: Caminos mínimos pudiendo pasar por el 1.
- **Paso 2:** Caminos mínimos pudiendo pasar por el 1 y el 2.
- …
- **Paso n**: Caminos mínimos pudiendo pasar por cualquier nodo ➔ ¡ Lo que buscamos !

- En cada paso **k**, el nodo **k** actúa de *pivote*.

¿En qué se basa el algoritmo de Floyd?

- Camino mínimo entre el nodo "**i**" y el nodo "**j**", en el paso **k**:
  - Sin pasar por **k**: D[i,j]
  - Pasando por **k**: D[i,k]+D[k,j]
  - Nos quedamos con el menor



Grafos: Floyd

- Función para obtener el listado de nodos que componen el camino más corto entre un nodo “**i**” y otro “**j**”

```
void camino (int P[][], int i, int j){
        int k;

        if ((k=P[i][j])== -1)
                return;
        camino(i,k);
        printf(“%d “,k);
        camino(k,j);
}
```

Contenidos

- Introducción
- Terminología
- Representación
- Problemas tipicos
  - Dijkstra
  - Floyd
  - **Warshall**

Grafos: Warshall

- Problema: conocer si existe conectividad entre dos nodos
  - No interesa el coste.
  - No interesa el camino.
- Elementos:
  - **Matriz de conexiones directas C[i,j]**: indica la existencia de conexión directa (arista) del nodo “**i**” al “**j**”.
  - **Matriz de conexiones A[i,j]:** indica la existencia de un camino que conecta el nodo ‘**i**’ y ‘**j**’.

Grafos: Warshall

- Esquema del algoritmo

```
#define boolean int

void warshall (boolean c[][], boolean a[][], unsigned int nNodos){
    int i,j,k;

    for (i = 0; i < nNodos; i++)
       for (j=0; j< nNodos; j++)
           A[i][j] = C[i][j];

    for (k = 0; k < nNodos; k++)
       for (i = 0; i < nNodos; i++)
           for (j=0; j< nNodos; j++)
               A[i][j] = A[i][j] || A[i][k] && A[k][j];
               //Si hay un camino de 'i' a 'j' o si hay
               //un camino de 'i' a 'j' pasando por 'k'
}
```

Adaptación del algoritmo de Floyd a valores booleanos

Conclusiones

- **Caminos mínimos**: problema fundamental en grafos. Diferentes problemas, con diversas aplicaciones.
- Desde un origen hasta los demás nodos
  - Aplicar **Algoritmo de Dijkstra**
  - Idea: Nodos escogidos y candidatos
- Entre todos los pares
  - Aplicar **Algoritmo de Floyd**
  - Idea: Pivotar sobre cada nodo
- Ambos algoritmos pueden modificarse para resolver otros problemas.

Bibliografía

- King, K.N. **C Programming. A modern approach.** 2ªed. Ed. W.W. Norton & Company. Inc. 2008.
- Khamtane Ashok**. Programming in C**. Ed. Pearson. 2012.
- Ferraris Llanos, R. D. **Fundamentos de la Informática y Programación en C**. Ed. Paraninfo. 2010.