



# 4.2 GRAFOS

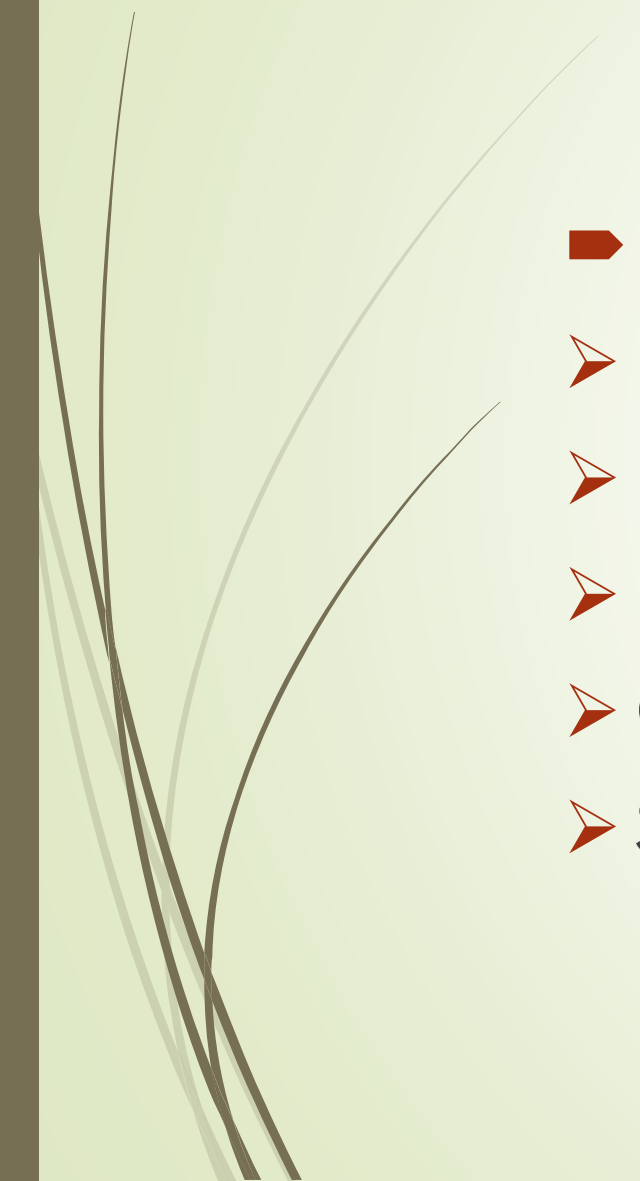
TEMA 4

# Definición

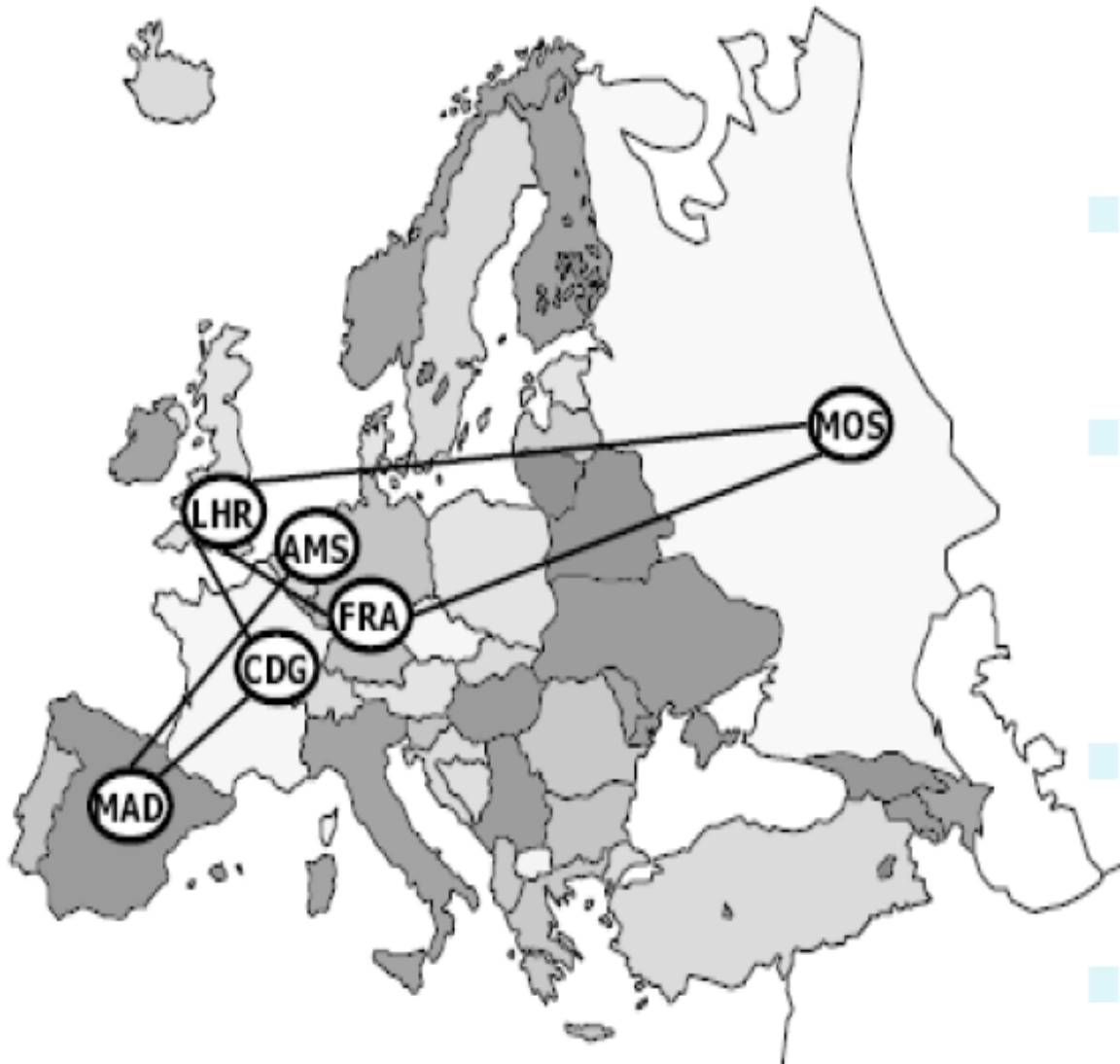
- Un grafo se define como un conjunto de nodos (también llamados (vértices) y un conjunto de arcos (aristas) que establecen relaciones entre los nodos.
- Un grafo consiste en un par  **$G = (V, A)$** , donde
  - **$V$  es un conjunto finito no vacío de vértices**
  - **$A$  es un conjunto de pares de vértices de  $V$ , es decir, las aristas.**



# Aplicaciones de grafos

- Para modelar:
    - Red de vuelos
    - Diagramas de flujo
    - Red de computadoras
    - Circuitos eléctricos
    - Secuenciación de tareas
- 

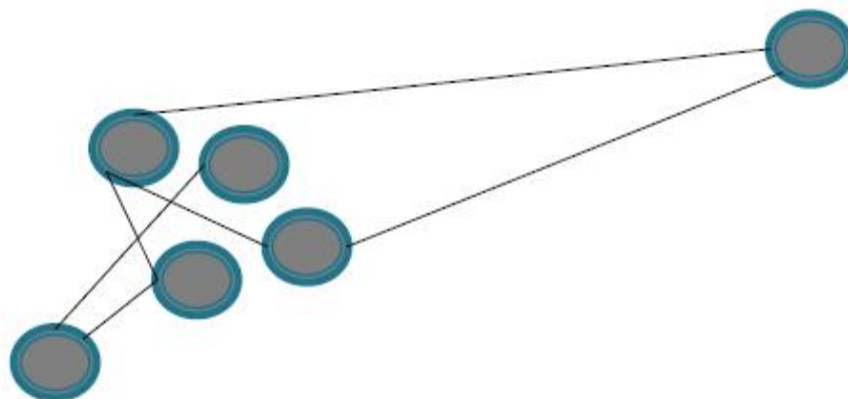
# Aplicaciones de grafos Ejemplo: red de vuelos



- Aeropuerto de Londres–Heathrow LHR
- Paris Charles De Gaulle Airport Guide – Paris CDG
- Aeropuerto de Ámsterdam AMS
- Frankfurt Airport FRA

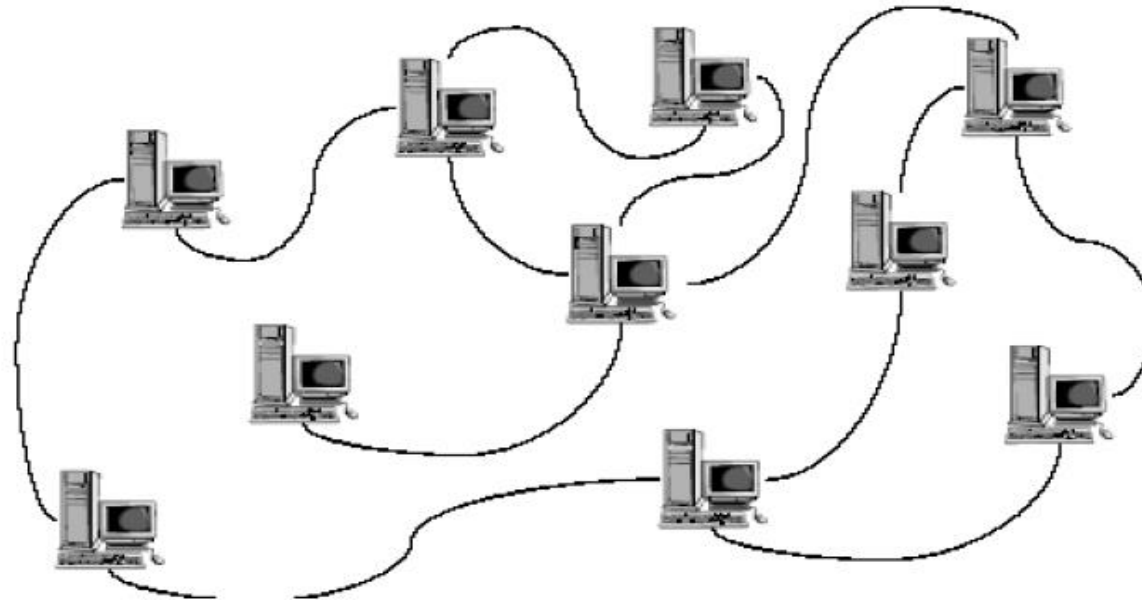
# Aplicaciones de grafos

## Ejemplo: red de vuelos



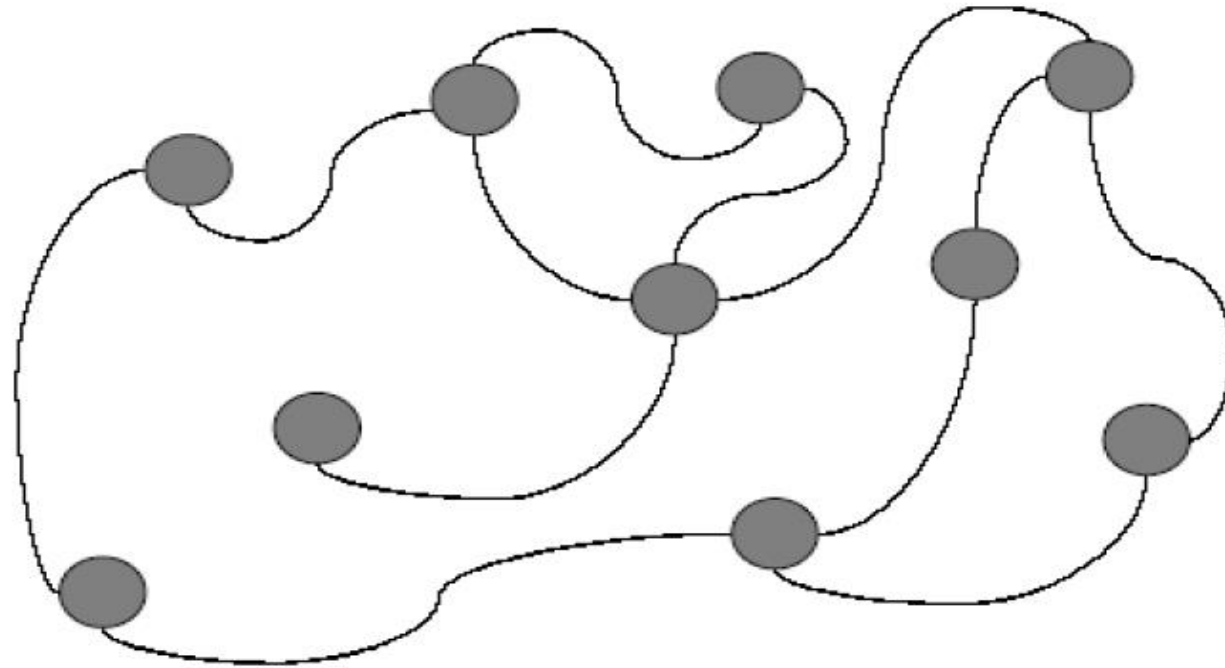
# Aplicaciones de grafos

## Ejemplo: red de computadoras



# Aplicaciones de grafos

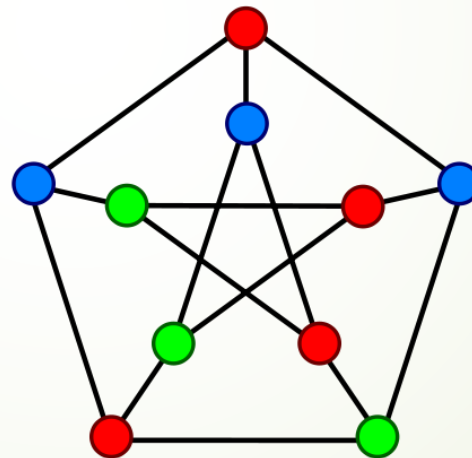
## Ejemplo: red de computadoras





# Coloreo de grafos

- El coloreo de grafos es un caso especial de grafos etiquetados, tales que los vértices adyacentes y las aristas coincidentes deben tener diferentes etiquetas

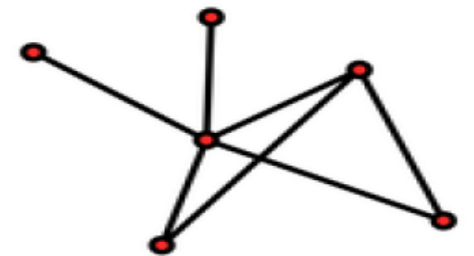


Una coloración por vértices para un grafo de Petersen, cada color representa una etiqueta.

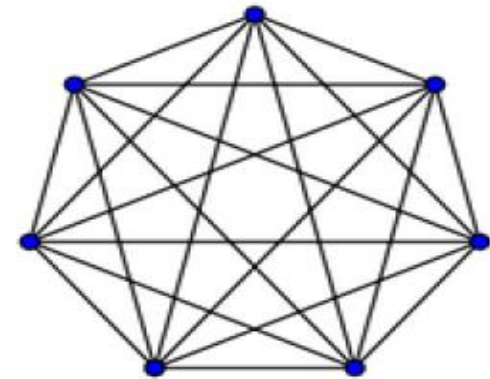


## Tipos de grafos( Simples, completos, planos, conexos, ponderados)

**Grafos simples.-** Un grafo es simple si a lo más existe una arista uniendo dos vértices cualesquiera. Esto es equivalente a decir que una arista cualquiera es la única que une dos vértices específicos. Un grafo que no es simple se denomina multigrafo.

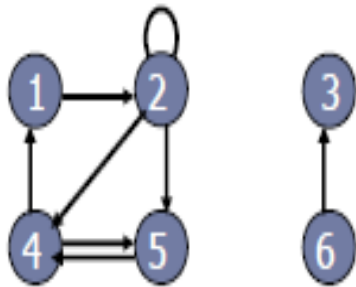


- ▶ **Grafo completo.**– Un grafo es *completo* si existen aristas uniendo *todos* los pares posibles de vértices. Es decir, todo par de vértices (a, b) debe tener una arista *e* que los une. El conjunto de los grafos completos es denominado usualmente *K*, siendo *K<sub>n</sub>* el grafo completo de *n* vértices. Un *K<sub>n</sub>*, es decir, grafo completo de *n* vértices tiene exactamente  $n(n-1)/2$  aristas.



# Ejemplos

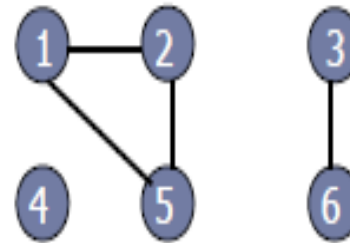
- ▶ Un **Grafo Dirigido** (GD) es un Par  $G = (V, A)$ 
  - ▶  $V$  es un conjunto finito de Vértices (o Nodos o Puntos)
  - ▶  $A$  es un conjunto de Aristas (o Arcos) dirigidas
- ▶ Arista: Par **ordenado** de Vértices  $(u, v)$



$$V = \{1, 2, 3, 4, 5, 6\} \quad |V| = 6$$

$$A = \{ (1, 2), (2, 2), (2, 4), (2, 5), (4, 1), (4, 5), (5, 4), (6, 3) \} \quad |A| = 8$$

- ▶ Un **Grafo No Dirigido** (GND) es un Par  $G = (V, A)$ 
  - ▶  $V$  es un conjunto finito de Vértices
  - ▶  $A$  es un conjunto de Aristas no Dirigidas
- ▶ Arista: Par **no ordenado** de Vértices  $(u, v) = (v, u)$



$$V = \{1, 2, 3, 4, 5, 6\} \quad |V| = 6$$

$$A = \{ (1, 2), (1, 5), (2, 5), (3, 6), (4, 1), (4, 5), (5, 4), (6, 3) \} \quad |A| = 8$$

**Cardinalidad** de un conjunto es el número de elementos que posee ese conjunto.  
El símbolo que representa la cardinalidad de un conjunto  $A$ , es:  $|A|$ .



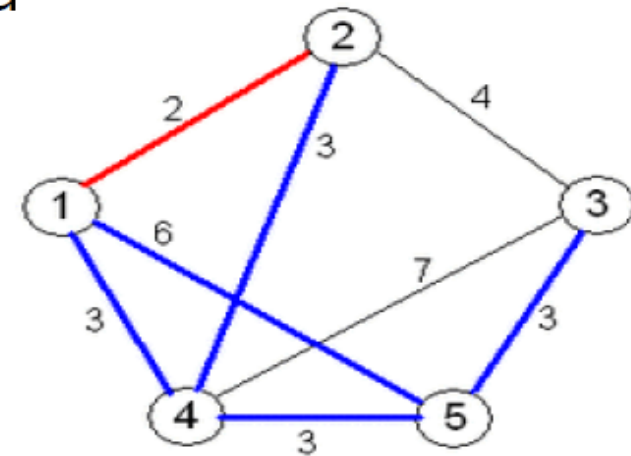
# Grafo etiquetado y grafo ponderado

---

- ▶ Un **Grafo Etiquetado** es un grafo  $G = (V, E)$  sobre el que se define una función  $f: E \rightarrow A$ , donde  $A$  es un conjunto cuyas componentes se llaman Etiquetas.
- ▶ Un **Grafo Ponderado** es un Grafo Etiquetado (sus Aristas) con números Reales.
- ▶ También es posible definir la función de etiquetado para los Vértices, con lo que podemos asignar un nombre a cada Vértice.



- ▶ **Grafos ponderados.**– Llamamos grafos ponderados a los grafos en los que se asigna un numero a cada una de las aristas. Este numero representa un peso para el recorrido a través de la arista. Este peso podrá indicar, por ejemplo, la distancia, el costo monetario o el tiempo invertido, entre otros. Definimos la longitud de un camino en un grafo ponderado como la suma de los pesos de las aristas de ese camino.



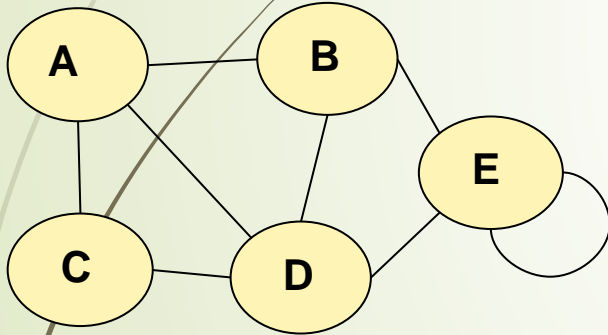
## Ejemplo de grafo ponderado



# Terminología de grafos

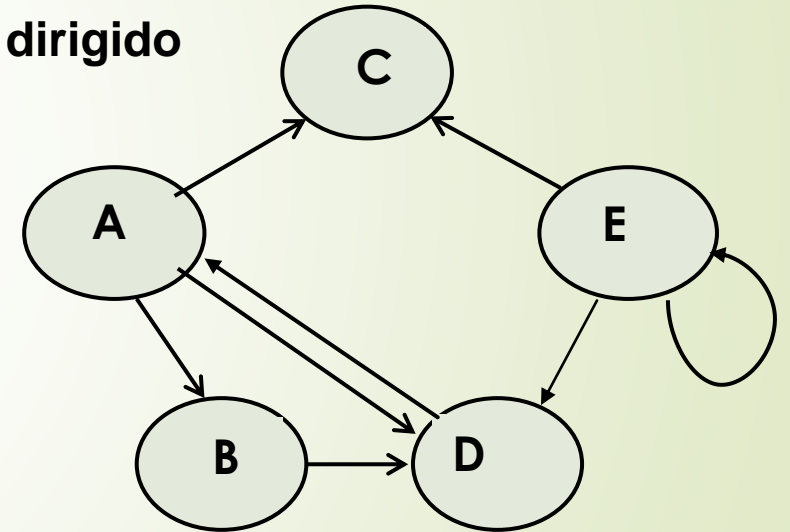
- **Camino:** Un camino en el grafo  $G$  es una sucesión de vértices y arcos:  $v_0, a_1, v_1, a_2, v_2, \dots, a_k, v_k$ ; tal que los extremos del arco  $a_i$  son los vértices  $v_{i-1}$  y  $v_i$ .
- **Longitud de camino:** Es el número de arcos que componen el camino.
- **Bucle:** Es un camino simple de longitud uno que empieza y termina en el mismo vértice.

Grafo no dirigido



- $\langle A, B, E, D, C \rangle$ : camino simple de longitud 4.
- $\langle A, C, D, A, B, E \rangle$ : camino de longitud 5.
- $\langle A, E \rangle$ : no es un camino.
- $\langle E, E \rangle$ : camino, bucle y ciclo

Grafo dirigido

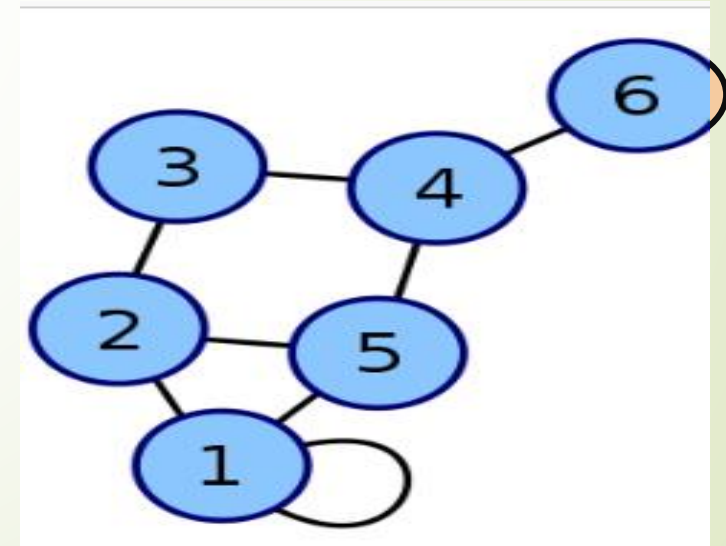


- $\langle A, B \rangle$ : camino simple de longitud 1.
- $\langle E, D, A, B \rangle$ : camino de longitud 3.
- $\langle A, C, D \rangle$ : no es un camino.
- $\langle E, E \rangle$ : camino, bucle y ciclo



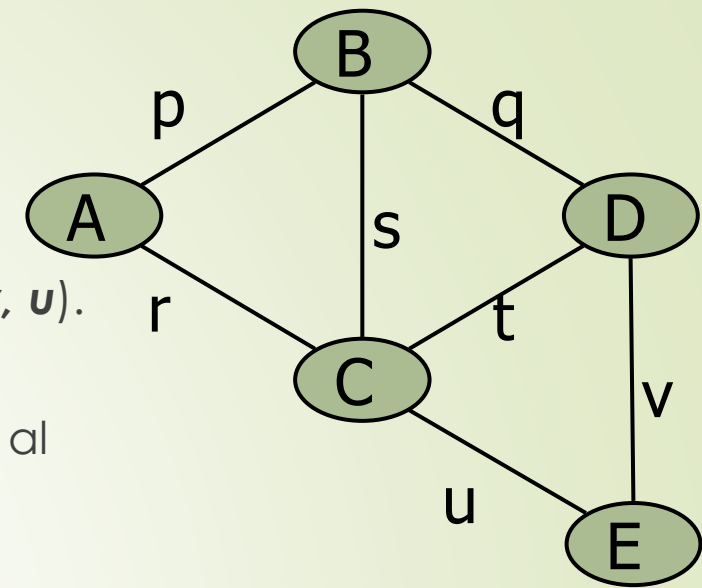
# Terminología de grafos

- **Ciclo:** Un ciclo (o circuito) es un camino cerrado que empieza y acaba , es decir, un conjunto de vértices unidos en el que ÚNICAMENTE se pasa 1 VEZ por cada uno excepto el primer nodo que se pasaría dos veces (la de inicio y la de fin).
- Ej: camino: (2,3,4,5,2).
- (6,4,3,2,5,4,6) no es un camino

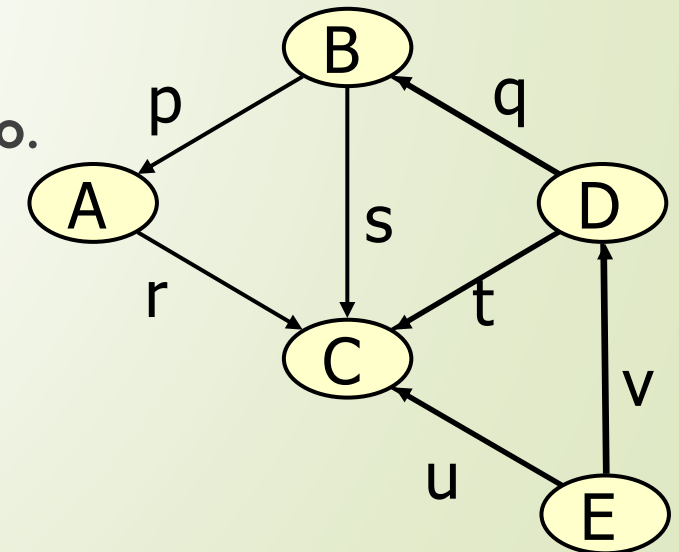


# Terminología de grafos

- **Adyacencia:** Dos vértices  $u$  y  $v$  son adyacentes si existe la arista  $(u, v)$  o  $(v, u)$ .
  - Ay B son adyacentes, B y E no son adyacentes
- **Grado de un vértice:** Determinado por el número de vértices adyacentes al nodo.
  - Grado de D = 3
- **Incidencia:** La arista  $(u, v)$  es incidente con los vértices  $u$  y con  $v$ . De forma que:
  - Aristas p, s, y q son incidentes en B

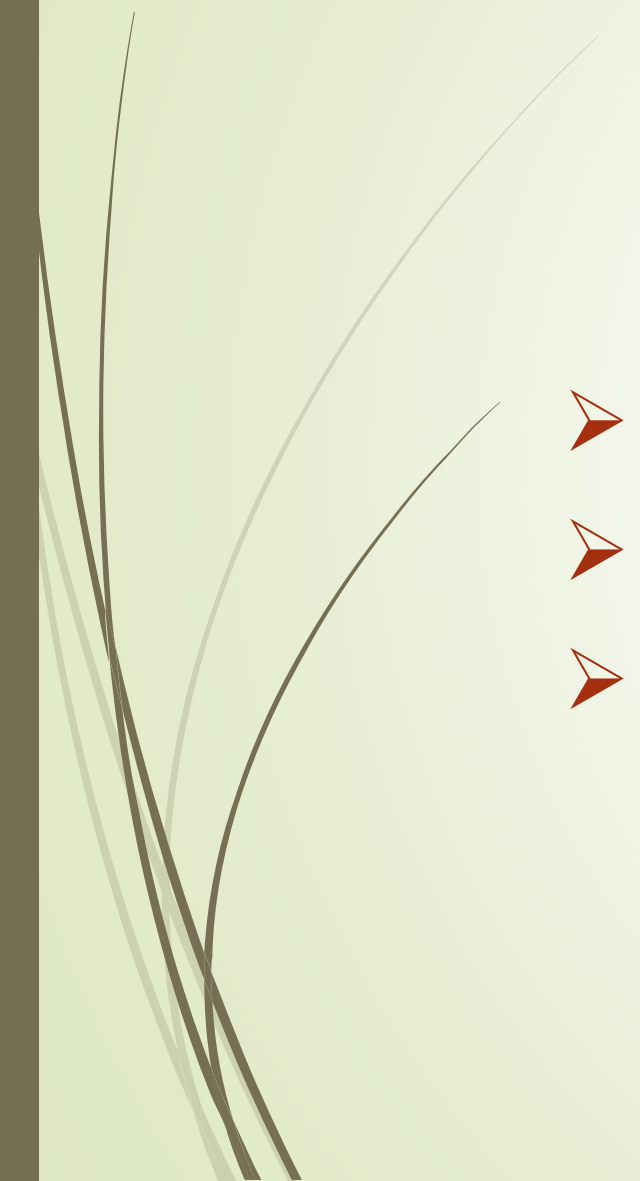


- **En un grafo dirigido:**
  - **Grado de salida:** número de vértices adyacentes desde el nodo.
    - Grado de salida de C = 0
    - Grado de salida de E = 2
  - **Grado de entrada:** número de vértices adyacentes al nodo.
    - Grado de entrada de C = 4
    - Grado de entrada de E = 0






# Representación de grafos

- matriz de adyacencias
  - lista de adyacencias
  - multilistas de adyacencia
- 

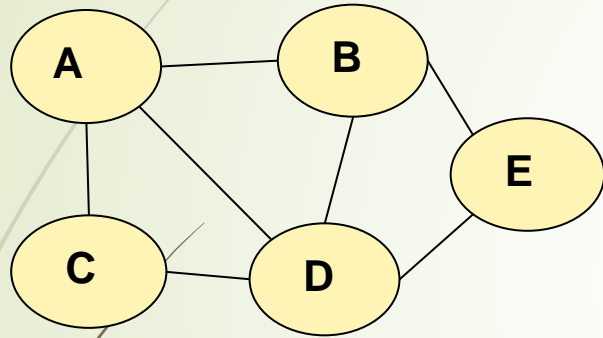


# Matriz de adyacencias

- Arreglo bidimensional que guarda las adyacencias entre pares de vértices de un grafo.
  - Se asocia cada fila y cada columna a cada nodo(vértice) del grafo, cada posición representa una arista, cuyo vértice origen se encuentra en la fila y vértice final se encuentra en la columna.
  - Tomando los valores de 1 si existe la arista y 0 en caso contrario.
- 

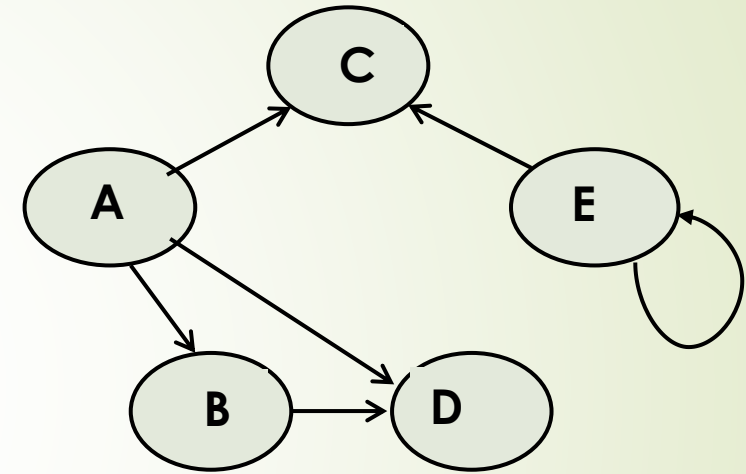
# Matriz de adyacencias - Ejemplos

Grafo no dirigido



	A	B	C	D	E
A	0	1	1	1	0
B	1	0	0	1	1
C	1	0	0	1	0
D	1	1	1	0	1
E	0	1	0	1	0

Grafo dirigido



	A	B	C	D	E
A	0	1	1	1	0
B	0	0	0	1	0
C	0	0	0	0	0
D	0	0	0	0	0
E	0	0	1	0	1



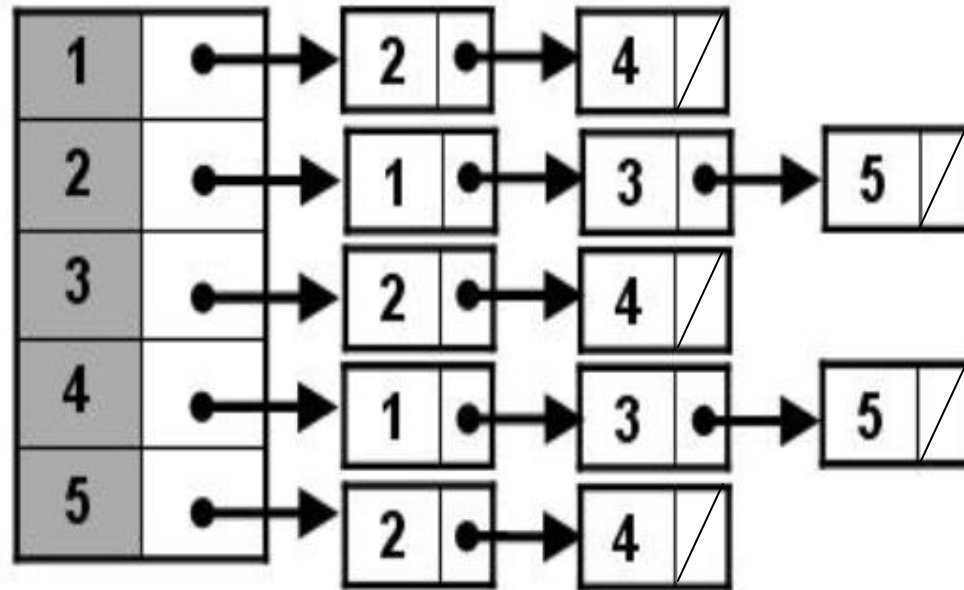
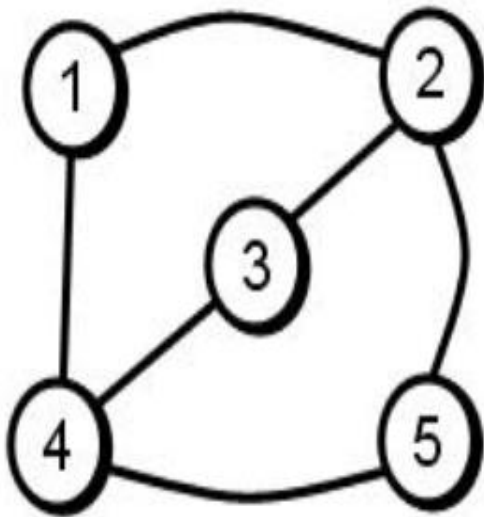
# Lista de adyacencias

- En esta representación se asocia a cada nodo del grafo una lista que contenga todos aquellos nodos que sean adyacentes a él.
- Sólo se reserva memoria para almacenar las aristas adyacentes al nodo.
- El grafo se representa por medio de un arreglo de  $n$  elementos, ( $n$  = número de vértices del grafo) donde cada elemento constituye la lista de adyacencias correspondiente a cada vértice del grafo.
- Cada nodo de la lista consta de un atributo indicando el vértice adyacente.
- Si el grafo fuese etiquetado o valorado, habría que añadir un segundo atributo para mostrar el valor de la etiqueta o el peso de la arista.



# Lista de adyacencias – ejemplo:

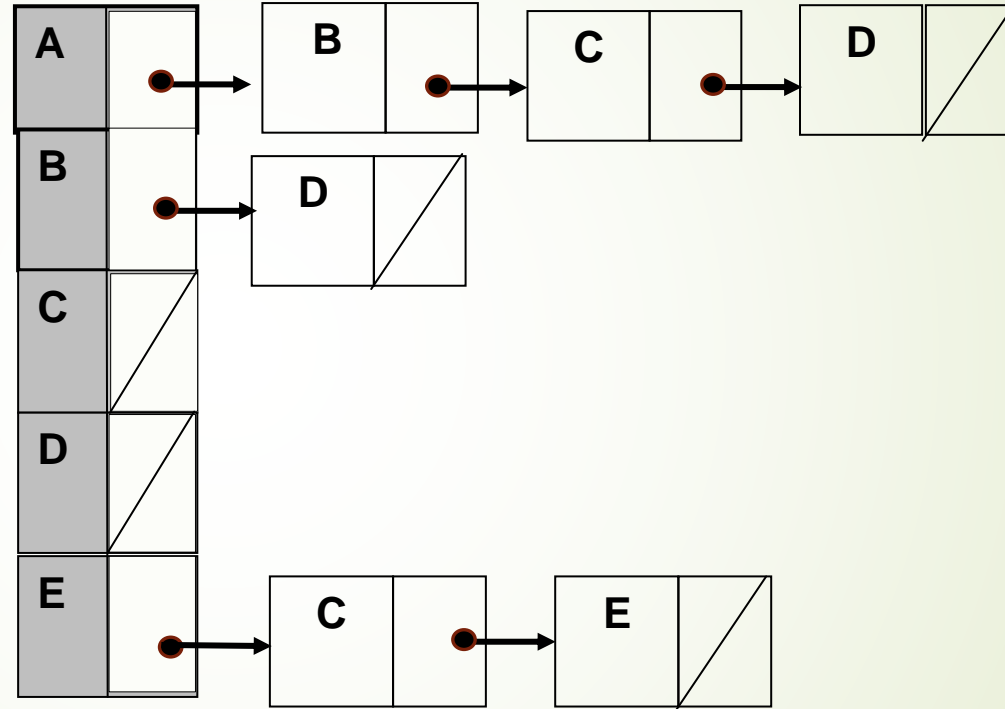
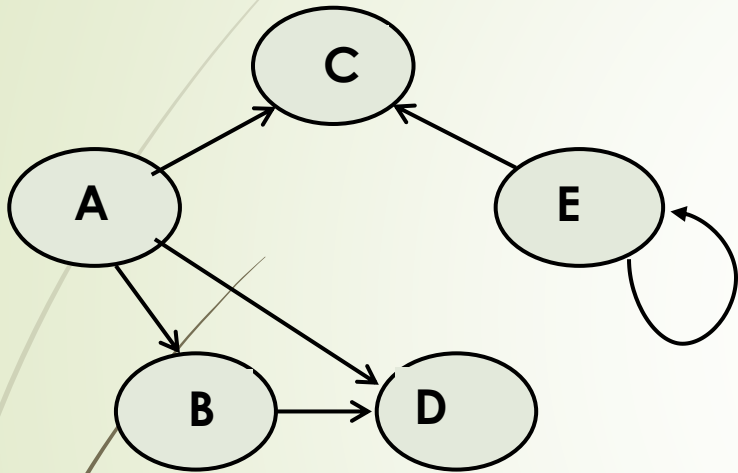
Grafo no dirigido





# Ejemplo

## Grafo dirigido



# Multilista de adyacencias

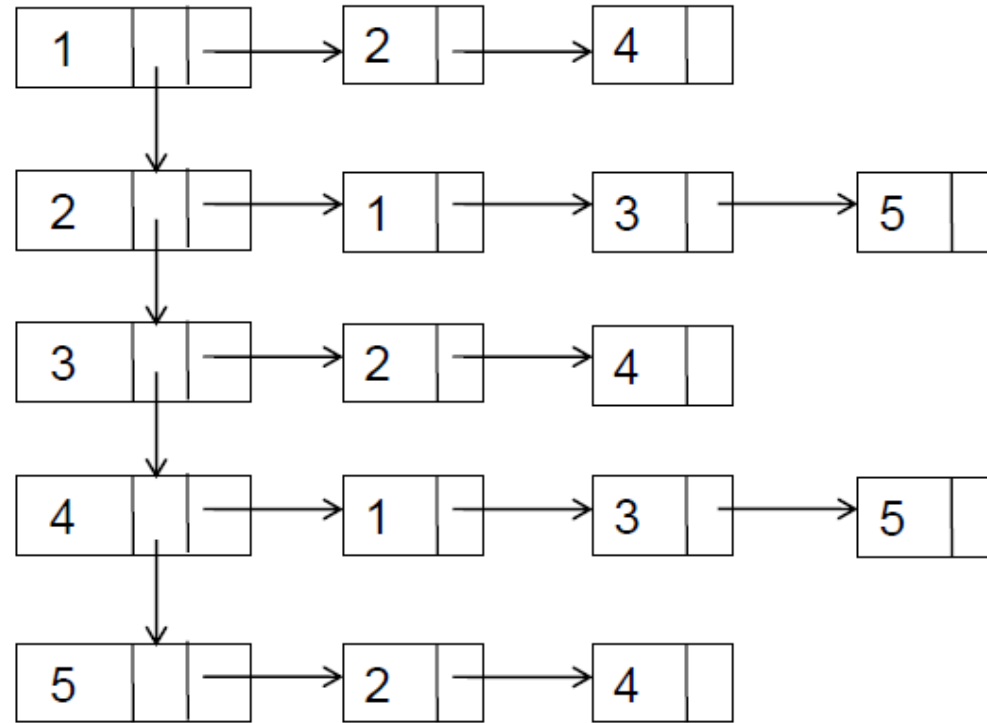
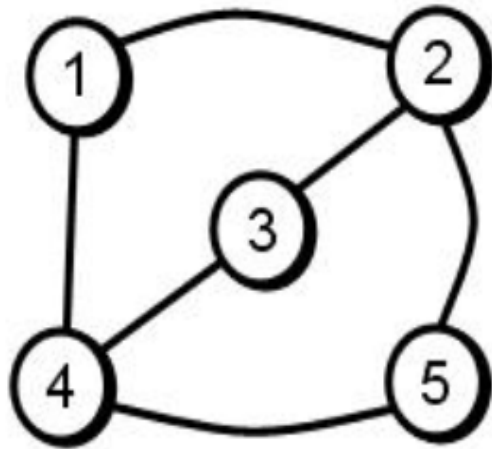
- Los nodos de una multilista de adyacencia tienen enlaces a varias listas, de ahí su nombre.

En esta representación se tiene por cada nodo una lista enlazada con los nodos destinos y otra lista enlazada con los nodos origen.

La ventaja de la multilista es su ahorro de memoria.

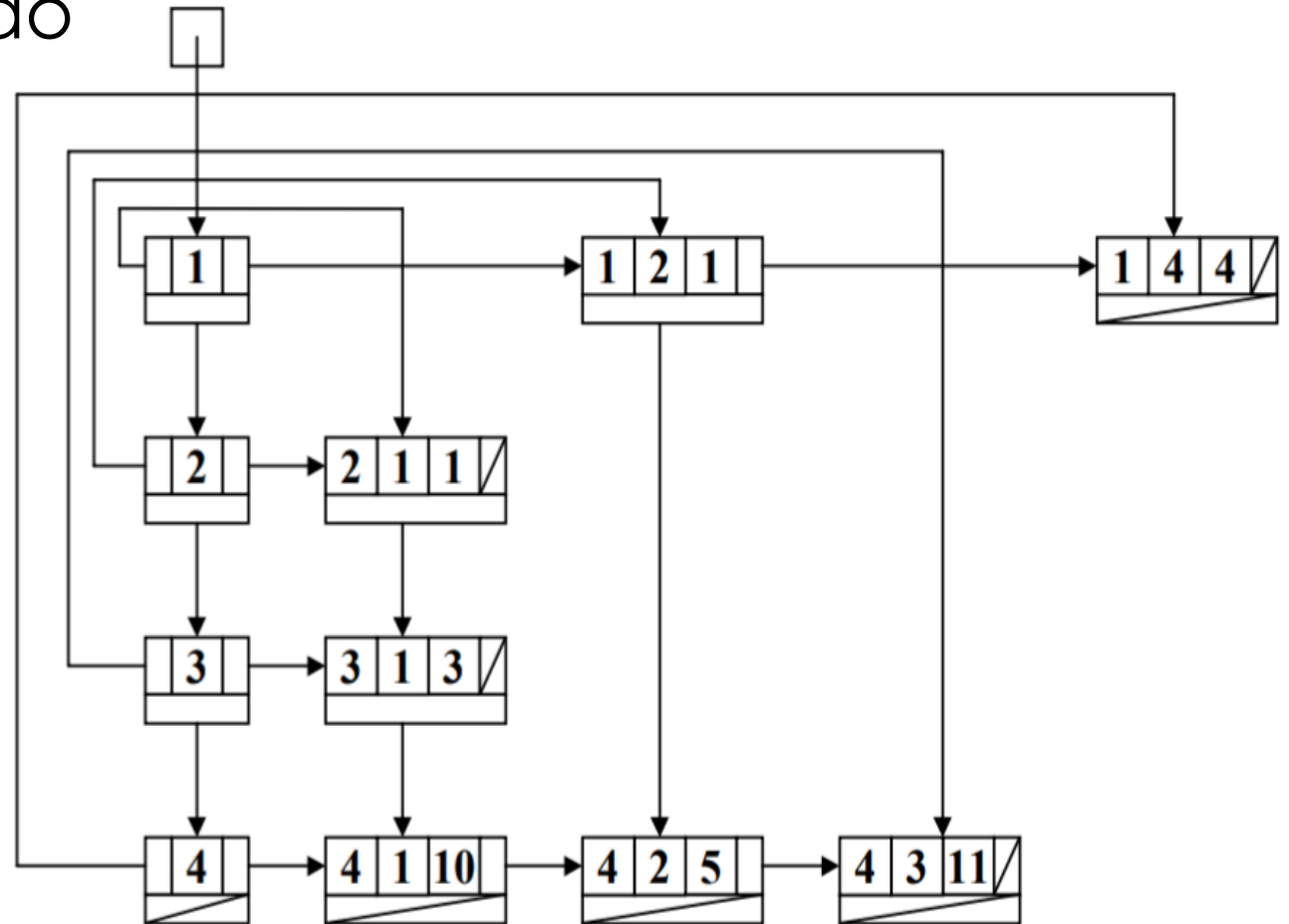
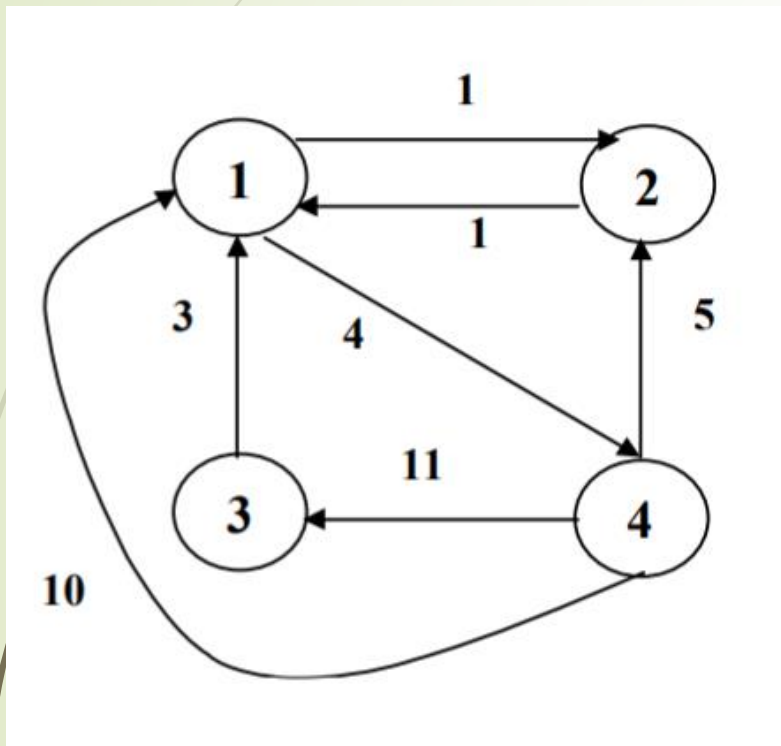
# Ejemplo

Grafo no dirigido



# Ejemplo Multilista de adyacencias

Grafo dirigido ponderado





# Grafos

## Operaciones básicas



- **Crear un grafo vacío:** Devuelve un grafo vacío. (Crea la matriz de adyacencias vacía)
- **Inicializar el grafo:** carga los valores 0 ó 1 en la matriz.
- **Insertar aristas.** Dado un grafo, añade una relación entre dos nodos de dicho grafo, la inserción de una arista  $(i, j)$  en la matriz supone asignar a la celda correspondiente el valor true (1).
  - En un grafo dirigido: las filas representan el vértice origen  $(i)$ . las columnas representan el vértice destino  $(j)$
  - En un grafo no dirigido: La arista  $(i, j)$  es igual a la arista  $(j, i)$  (para que la matriz conserve su simetría)



# Operaciones:

*(métodos de la clase grafo)*

- **Insertar vértices.** Dado un grafo, incluye un nodo en él, en caso en el que no exista previamente.
  - El tratamiento de los vértices implicaría modificar el tamaño de la tabla (o modificar los índices en caso de querer eliminar un vértice):
- Simplificación del método:
  - No se permite añadir vértices si se supera el tamaño máximo del grafo (valor del campo maxNodos).
  - Si el número de nodos es menor al tamaño máximo, se asigna el valor false a las celdas correspondientes y se actualiza el campo numVertices


- 
- 
- **Eliminar vértices:** Devuelve un grafo sin un vértice (nodo) y las aristas relacionadas con él. Si dicho nodo no existe se devuelve el grafo inicial.
  - **Eliminar aristas.** Devuelve un grafo sin la arista indicada. En caso de que la arista no exista devuelve el grafo inicial. La eliminación de una arista  $(i, j)$  en la matriz supone asignar a la celda correspondiente el valor false (0).





# Operaciones:

- **Grafo Vacío:** Comprueba si un grafo no tiene ningún nodo.
- **Búsqueda de un elemento:** Comprueba si un elemento pertenece a un grafo.
- **Adyacencia:** Comprueba si dos nodos tienen una arista que los relacione.
- **Imprimir grafo:** muestra el grafo. (el contenido de la matriz de adyacencias).
- **Recorrido de grafos.**



# Algoritmos para obtener el árbol abarcador de costo mínimo

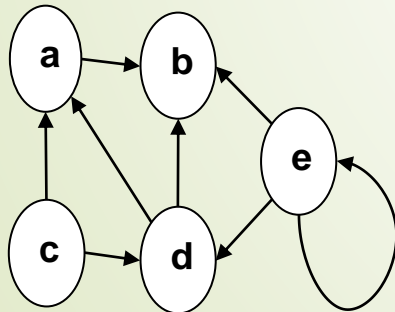
- **Algoritmo de Prim**

- **Algoritmo de Kruskal**

# Matriz de adyacencias - implementación

- Una matriz de adyacencias se implementa como un arreglo bidimensional de  $n \times n$  donde:
  - La celda  $[i, j]$  guarda información referente a la arista  $(v, w)$  donde  $v$  es el vértice con índice  $i$  y  $w$  es el vértice con índice  $j$ .
  - Para grafos no etiquetados, las celdas guardan valores booleanos:
    - true: existe la arista
    - false: no existe la arista

➡ Los vértices se representan mediante índices.




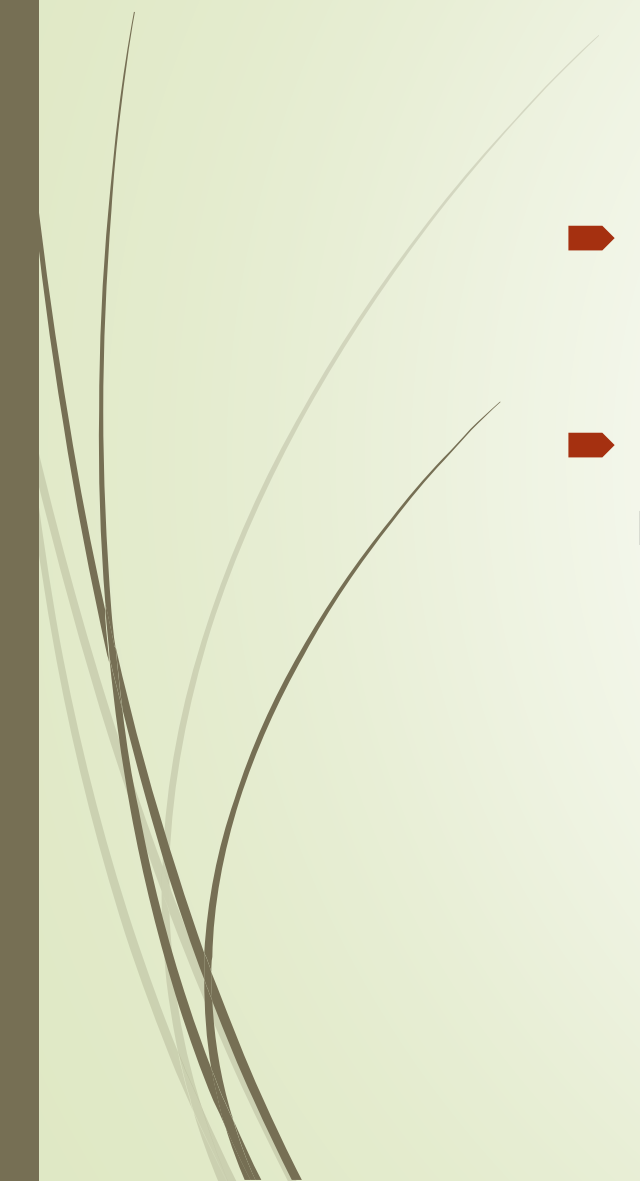
Vértices: a b c d e  
Índices: 0 1 2 3 4

		0	1	2	3	4
		A	B	C	D	E
0	A	0	1	0	0	0
1	B	0	0	0	0	0
2	C	1	0	0	1	0
3	D	1	1	0	0	0
4	E	0	1	0	1	1

# Clase GrafoMatriz

```
public class GrafoMatriz {  
    boolean esdirigido; //indica si el grafo es o no dirigido  
    int maxNodos;  
    int numVertices;  
    boolean matAdy[ ][ ];  
  
    public GrafoMatriz (boolean d) {  
        // construye un grafo vacio  
        maxNodos = numVertices = 0;  
        esdirigido = d;  
    }  
}
```

```
    public GrafoMatriz (int n, boolean d) {  
        // construye un grafo de tamaño n  
        esdirigido = d;  
        maxNodos = n;  
        numVertices = 0;  
        matAdy = new boolean[n][n];  
    }  
    // aquí van los métodos de la clase GrafoMatriz  
} // fin de clase GrafoMatriz
```

- 
- 
- Resuelve el ejercicio gráfico (publicado en Teams).
  - Práctica: implementa el tda grafo y sus operaciones mediante una matriz de adyacencias



# Referencias



- [Estructura de datos. Cairo Guardati](#)
  - <http://informatica.uv.es/iiguia/AED/teoria/apuntes/cuatr2/tema15.pdf>
  - <https://tareasuniversitarias.com/terminologia-en-grafos.html>
- <https://es.stackoverflow.com/questions/262202/cu%C3%A1l-es-la-diferencia-entre-bucle-y-ciclo-en-grafos>