

# Diseño de Algoritmos

## Sesión 10

Profesores:

Tomás Lara Valdovinos – [t.lara@uandresbello.edu](mailto:t.lara@uandresbello.edu)

Jessica Meza-Jaque – [je.meza@uandresbello.edu](mailto:je.meza@uandresbello.edu)

## OBJETIVOS DE LA SESIÓN

- Conocer el concepto de grafo.
- Identificar problemas que podemos representar usando grafos.
- Especificar características de los elementos de un grafo.
- Aprender a representar grafos





¿Qué es un grafo?

# ¿Qué es un grafo?

- Es una **abstracción** utilizada para **modelar un sistema** que contiene **elementos discretos interconectados**.
- Esos elementos los podemos llamar **nodos** o **vértices**.
- Y las interconexiones podemos llamarlas **aristas** o **arcos**.

# Ejemplo



- Rutas entre ciudades, donde:
- Las ciudades representan los nodos.
- Y los caminos entre ellas son las aristas.



# Otro Ejemplo



- Una red social, donde:
- Los nodos representan a las personas.
- Y una arista entre 2 de ellas representa si ellos son “amigos”.

¿Otros Ejemplos?



# Aristas

- En algunos grafos, las aristas poseen diferentes medidas, comúnmente conocidas como **Peso** o **Costo**.
- Estas medidas representan datos tanto **cualitativos** como **cuantitativos**.
- A un grafo compuesto de aristas con estas características lo llamaremos **Grafo Ponderado**.

# Aristas



- En la red social, si el tipo de contacto es familiar o profesional.

- En las rutas entre ciudades este costo podría representar la distancia entre ciudades o el tiempo de viaje.



# Aristas

- Las aristas podrían ser **dirigidas** o **no dirigidas**
- Generando relaciones **unidireccionales** o **bidireccionales**
- Un grafo compuesto de aristas dirigidas lo llamaremos **Grafo Dirigido**
- Un grafo compuesto de aristas no dirigidas lo llamaremos **Grafo no-dirigido**

# Aristas

- La relación de amistad en la red social podríamos representarla con una arista no dirigida:
- Si **A** es amigo de **B**, entonces **B** es amigo de **A**



# Aristas



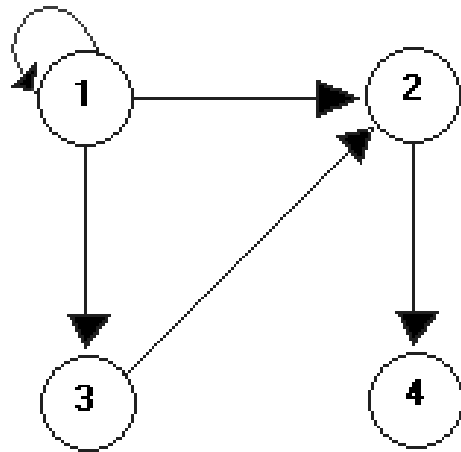
- Y en las rutas podríamos indicar con aristas dirigidas las calles que sólo tienen un sentido.

# Aplicaciones de grafos

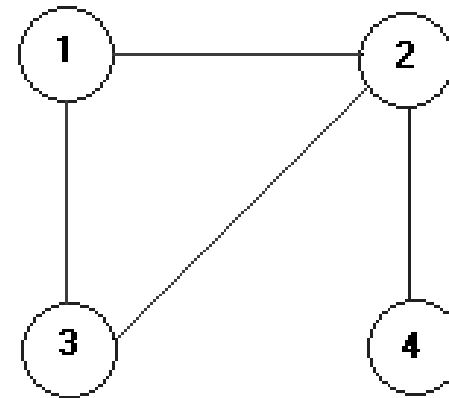
- Un ejemplo útil de resolver un problema utilizando grafos:
  - Reducir el problema del mundo real a una instancia de un problema de grafos.
  - Aplicar un algoritmo de grafos para computar el resultado eficientemente.
  - Interpretar el resultado de la computación en términos de una solución al problema original.

# Representación de grafos

- Gráficamente un grafo lo podemos expresar como círculos u otro elemento para los nodos y líneas uniendo estos círculos como las aristas.



— Grafo Dirigido

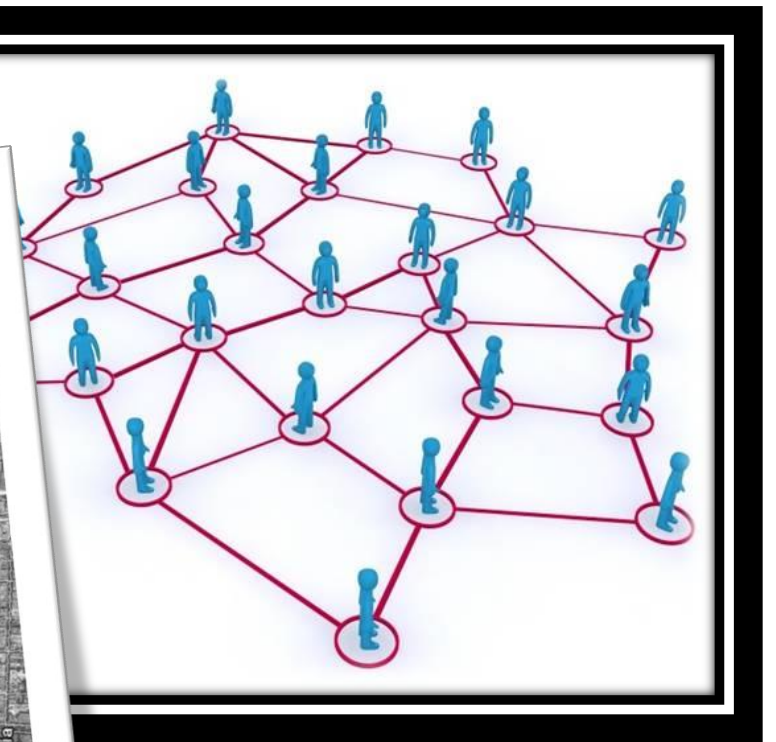
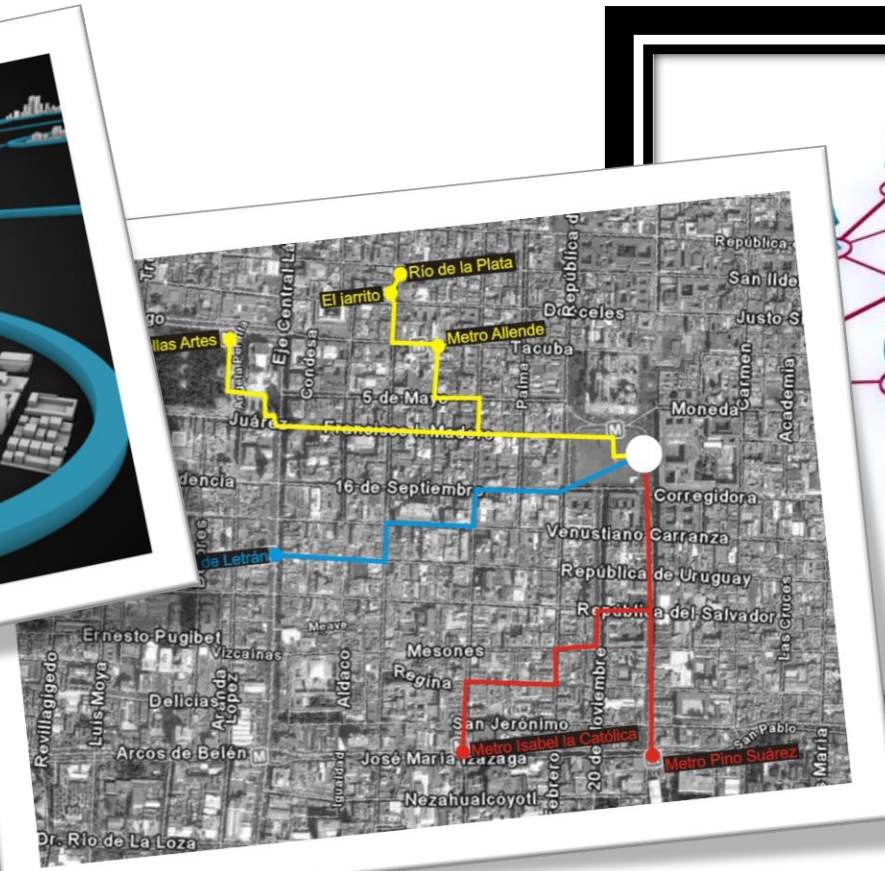


Grafo No dirigido



# Representación de grafos

- Esta representación nos ayuda a definir y entender el problema que estamos representando.

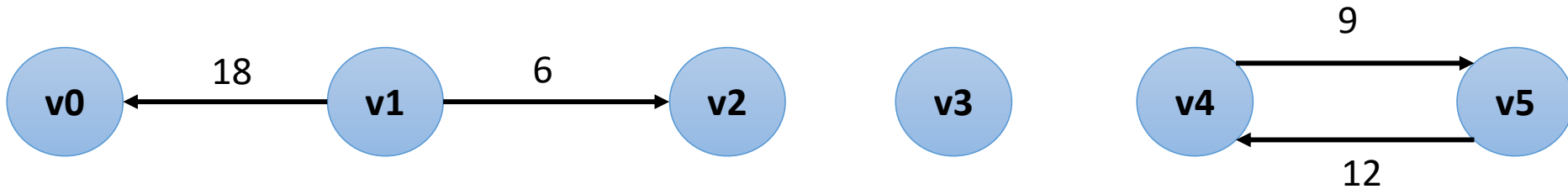


# Representación de grafos

- En código existen diversas estructuras de datos para representar un Grafo:
- Por ejemplo:
  - Listas de adyacencias
  - Matrices de adyacencias

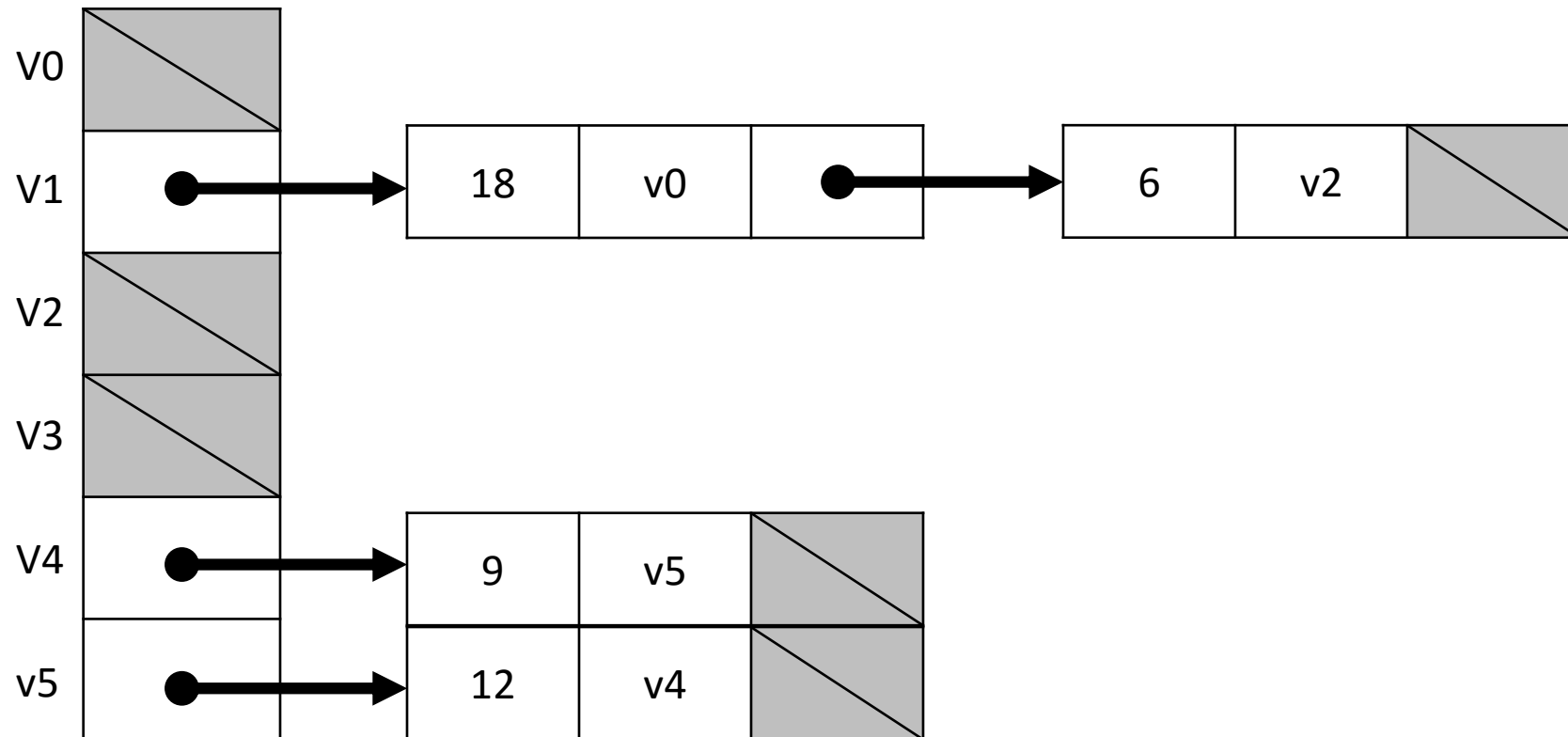
# Listas de adyacencias

- Sea el siguiente grafo ponderado y dirigido



# Listas de adyacencia

- Representaremos el grafo con listas de adyacencia de la siguiente manera:




# Listas de adyacencia

- Para el gráfico anterior:
  - La lista vertical representa todos los nodos en el grafo
  - Las listas contenidas (Lista horizontal) corresponde a las aristas como una estructura de datos conteniendo el peso/costo, el nodo relacionado y la siguiente relación.
- **Puede ser usado tanto para grafos dirigidos como no-dirigidos**

# Matriz de adyacencia

- Utilizando el mismo grafo del ejemplo anterior, podemos representarlo como matriz de adyacencia de la siguiente manera:

	v0	v1	v2	v3	v4	v5
v0	0	0	0	0	0	0
v1	18	0	6	0	0	0
v2	0	0	0	0	0	0
v3	0	0	0	0	0	0
v4	0	0	0	0	0	9
v5	0	0	0	0	12	0

 A[4][5]

# Matriz de adyacencia

- Esta representación de grafos utiliza una matriz de  $n$ -por- $n$  enteros
- La entrada  $A[i][j]$  almacena el peso de la arista desde el vértice  $v_i$  al  $v_j$
- Cuando  $A[i][j] = 0$  no existe relación entre ambos vértices.
- **Puede ser usado tanto para grafos dirigidos como no-dirigidos**



# Consideraciones especiales

- Obtener la relación entre 2 vértices **usando matriz de adyacencia** es llevado a cabo con una complejidad constante,  $O(1)$ .
- Las listas de adyacencia requieren el **espacio en memoria acotado** para almacenar los vértices y relaciones.
- Para grafos con muchísimos vértices el uso de matriz de adyacencia requiere gran cantidad de memoria, ya que la cantidad de almacenos tiene el orden de  $n^2$ , donde  $n$  es la cantidad de vértices.

# Consideraciones especiales

- Para grafos no-dirigidos el uso de listas de adyacencia puede reducir en gran medida el uso de almacenamiento.
- Para grafos en los cuales los vértices tienen gran cantidad de relaciones, llamados **Grafos densos**, el uso de matrices de adyacencia suele ajustar mejor el almacenamiento requerido.
- Para grafos con baja cantidad de relaciones, llamados **Grafos dispersos**, el uso de memoria se optimiza usando listas de adyacencia.

# Otros ejemplos

- Ahora que hemos especificado el concepto de grafo,


**¿Qué otros usos podríamos darle a esta estructura de dato?**

## CHECK - OBJETIVOS DE LA SESIÓN

- Conocer el concepto de grafo.
- Identificar problemas que podemos representar usando grafos.
- Especificar características de los elementos de un grafo.
- Aprender a representar grafos

CHECK





# Diseño de Algoritmos

## Sesión 10

Profesores:

Tomás Lara Valdovinos – [t.lara@uandresbello.edu](mailto:t.lara@uandresbello.edu)

Jessica Meza-Jaque – [je.meza@uandresbello.edu](mailto:je.meza@uandresbello.edu)