PROGRAMACION 3

CURSADA 2024

Docente:
Federico Casanova.

Grafos p1

Programación 3 TUDAI

Cursada 2024

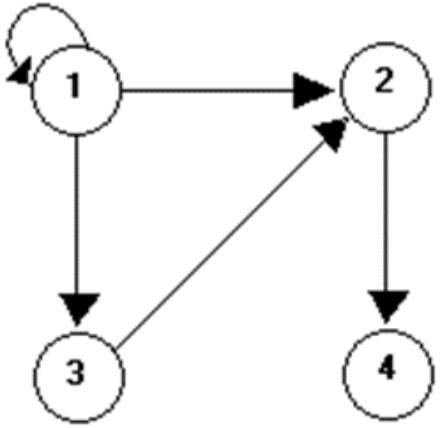
Definición

Un grafo es una estructura no lineal de datos, se compone de vértices y aristas (o conexiones o arcos) entre esos vértices.

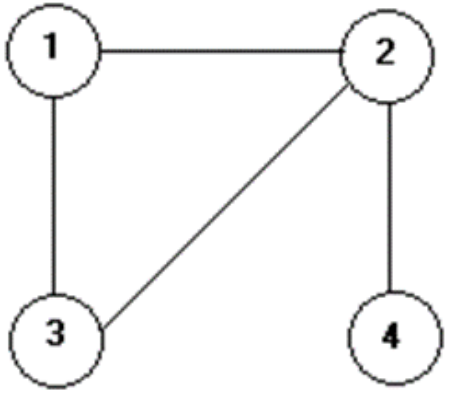
Formalmente definimos el grafo G=(V, A)
Donde V es el conjunto de vértices, y A ⊆ VxV es el conjunto de aristas.

Si consideramos que las aristas tienen dirección (origen, destino), tendremos un Grafo Dirigido (GD).

Si consideramos que las aristas NO tienen dirección, tendremos un Grafo No Dirigido (GND).



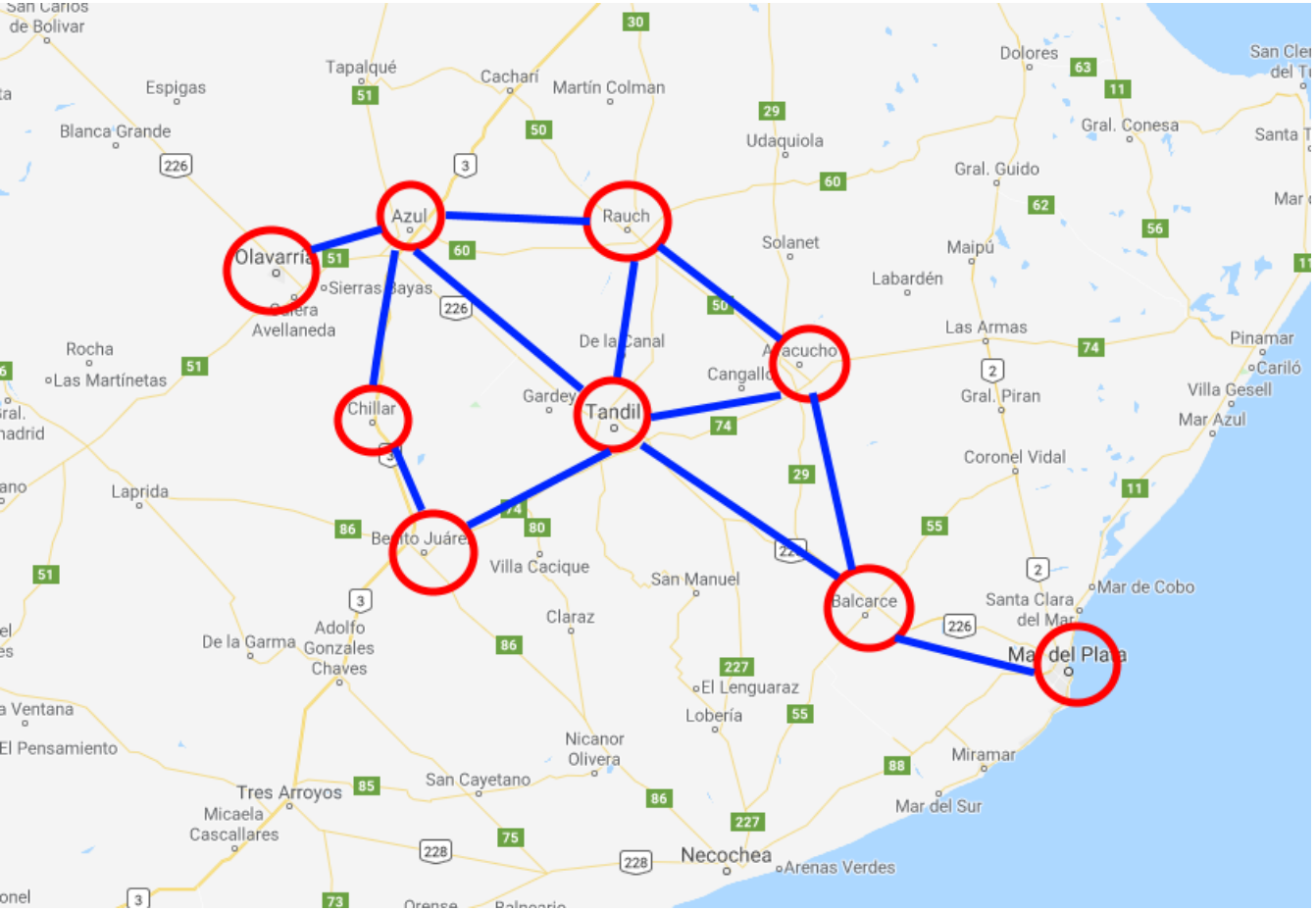
Grafo Dirigido
V={1,2,3,4}
A={(1,1), (1,2), (1,3), (3,2), (2,4)}



Grafo No Dirigido
V={1,2,3,4}
A={(1,2), (1,3), (3,2), (4,2)}

Ejemplo de uso

Representar rutas entre ciudades



Problemas que se pueden resolver

¿Cuál es el camino más corto entre Rauch y Balcarce?

¿Existe una forma de llegar entre todos los pares de ciudades?

¿Cuál es la ciudad más lejana de Mar del Plata?

¿Cuál es la ciudad más céntrica?

¿Cuántos caminos distintos existen de Olavarría a Mar del Plata?

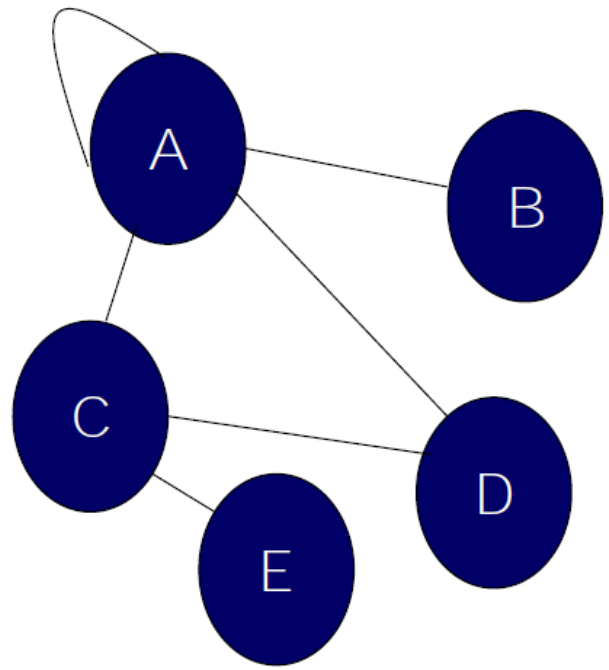
¿Cómo hacer un recorrido por todas las ciudades gastando lo menos que se pueda de nafta?

Ejemplos de Aplicaciones de Grafos

Aplicación	Vértices	Aristas
Redes de computadoras	Computadoras y equipos de telecomunicaciones	Medios de comunicación
Sistemas de transporte	Ciudades, puertos, aeropuertos, etc.	Autopistas, rutas de navegación, etc.
Redes eléctricas	Subestaciones	Cables
Facebook	Cuentas (persona)	Relaciones (Amistad, parentezco, etc.)
Distribución de fluidos	Estaciones de bombeo	Tuberías

Grafo No Dirigido

- 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

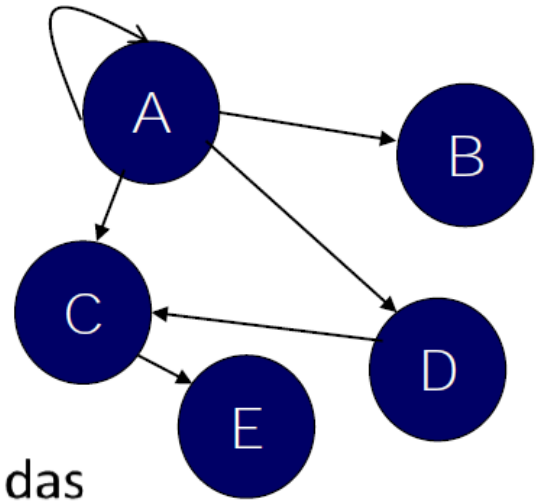


- Arista es par no ordenado de Vértices
- $(\boldsymbol{u}, \boldsymbol{v}) = (\boldsymbol{v}, \boldsymbol{u})$

Clasificación de grafos

Grafo Dirigido

- Un grafo dirigido es un par $G = (V, A)$



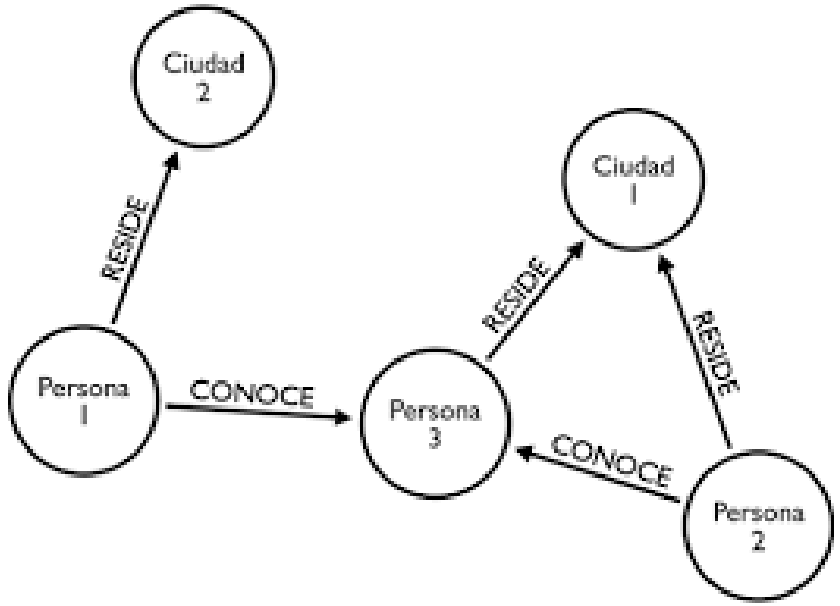
  - V es un conjunto finito de Vértices
  - A es un conjunto de Aristas (o Arcos) Dirigidas

- Arista es par ordenado de Vértices $(\boldsymbol{u}, \boldsymbol{v})$

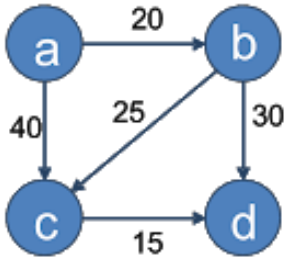
- Es importante la dirección del arco, o sea, el nodo origen del arco y el nodo destino.

Grafo Etiquetado y Grafo Ponderado

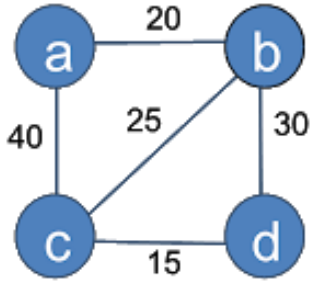
Un **Grafo Etiquetado** es un grafo $G = (V, A)$ sobre el que se define una función $f: A \rightarrow E$, dónde E es un conjunto cuyas componentes se llaman Etiquetas.



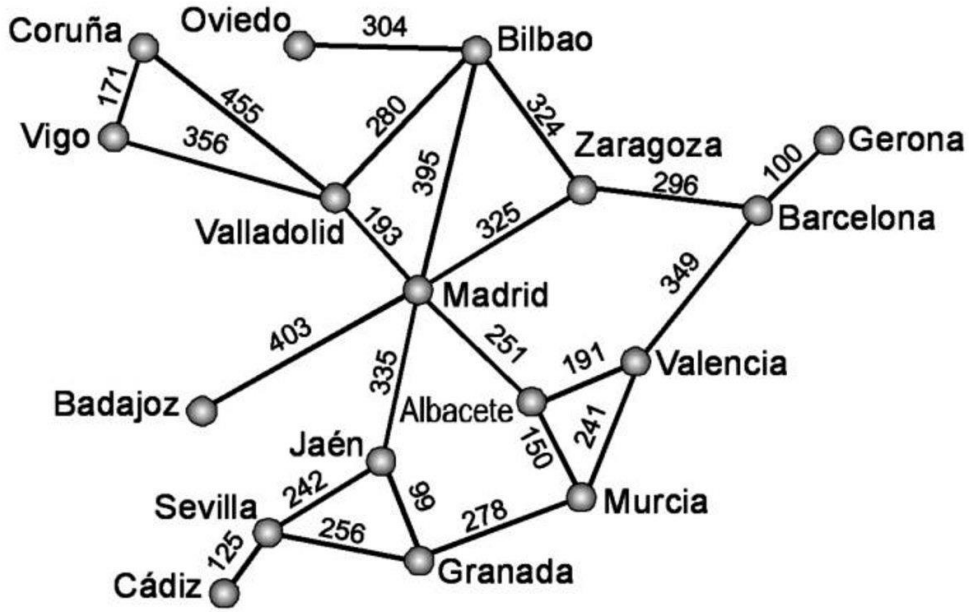
Un **Grafo Ponderado** es un Grafo Etiquetado (sus Aristas) con números Reales.
También se lo conoce como **Grafo Rotulado o Pesado**



GD Ponderado

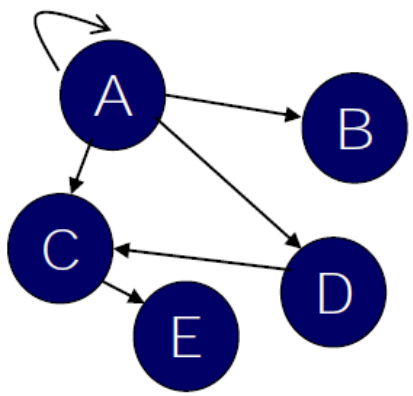


GND Ponderado



Relaciones de Adyacencia

- Sea G=(V,A) un grafo.
- Si $(v_n, v_m) \in A$, decimos que el vértice v_m es adyacente al vértice v_n



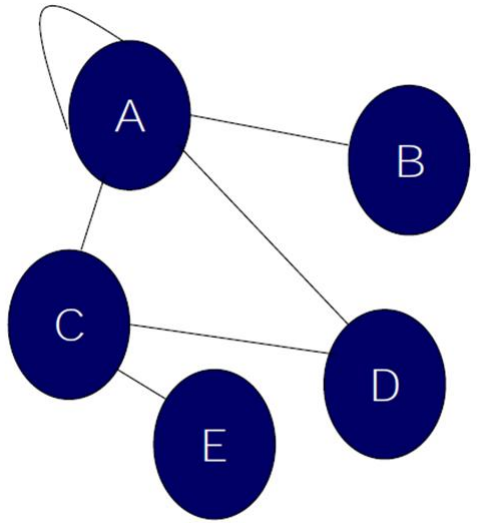
B es adyacente a A
C es adyacente a A
C es adyacente a D
D es adyacente a A
A es adyacente a A
E es adyacente a C

Dicho de otra manera, los adyacentes del vértice A, serán todos aquellos vertices a los que puedo llegar partiendo desde A haciendo un solo paso (un solo arco)

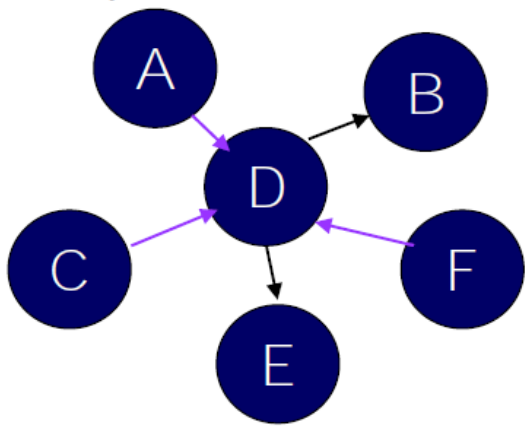
En un grafo no dirigido la relación es simétrica.

Grado de un vértice

- El **grado de un vértice en un GND** es el número de aristas que inciden sobre él (vértices adyacentes).
- El **grado de un vértice en un GD** es la suma de:
  - el número de Aristas que salen de él (Grado de Salida)
  - el número de Aristas que entran en él (Grado de Entrada)



Grado de A es 4



El grado del vértice D es 5

Grado de Entrada de D es 3

Grado de Salida de D es 2

El Grado de un Grafo es el de su Vértice de máximo Grado.

Caminos sobre Grafos

Un **camino de longitud k** desde el vértice ***a*** al ***b*** en un grafo G=(V,A),
Es una secuencia de vértices $<v_0, v_1, v_2, ..., v_k>$ tal que:

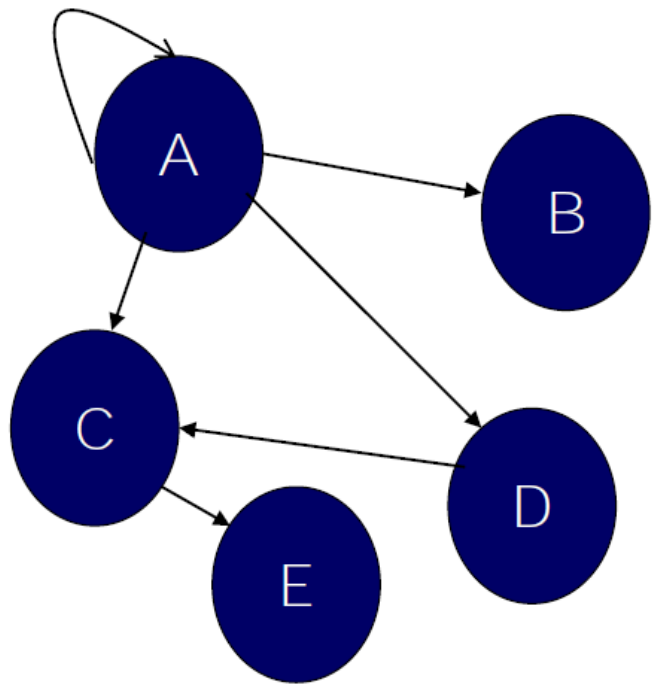
- $v_0 = a$ y $v_k = b$
- Para todo i entre 1 y k, la arista $(v_{i-1}, v_i) \varepsilon A$

La **longitud k** del camino:

- Sin rótulos es la cantidad de aristas.
- Con rótulos, es la suma de los rótulos de las aristas del camino.

Si hay un camino P desde ***a*** hasta ***b***, decimos que ***b*** es **alcanzable** desde ***a*** vía el camino P.

Caminos sobre grafos



Caminos entre los vértices A y C:

Camino de longitud 1: (A,C)

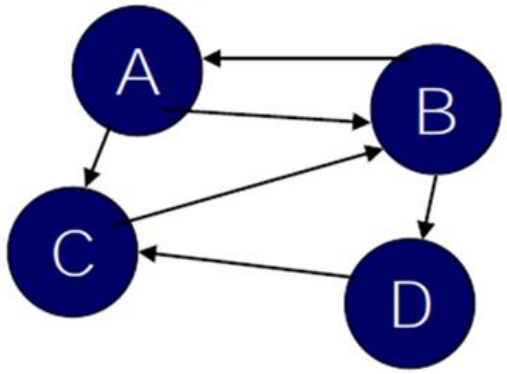
Camino de longitud 2: (A,D,C)

Camino de longitud 2: (A,A,C)

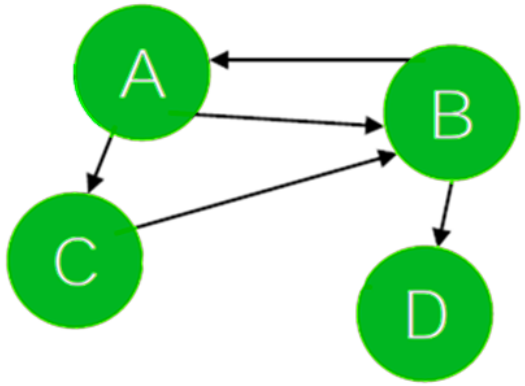
Camino de longitud 3: (A,A,D,C) ó C es **alcanzable** desde A vía $\langle A, A, D, C \rangle$

Grafo Conectado o Conexo

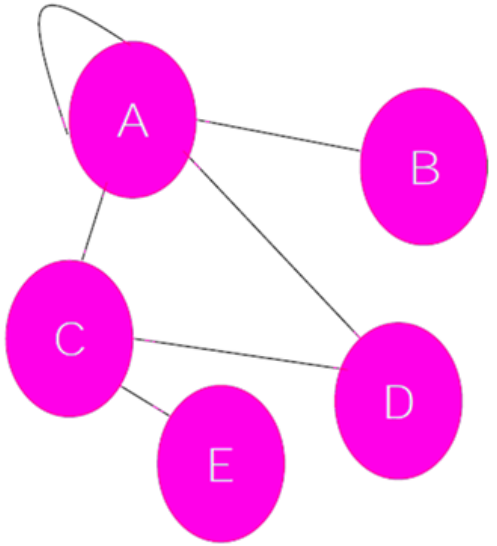
Es un grafo en el cual **existe un camino entre cada par de vértices**.



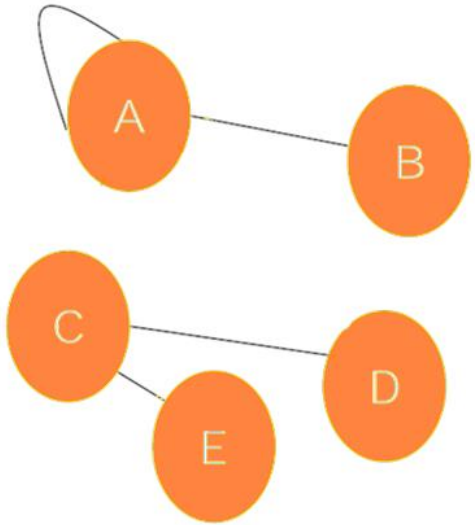
GD conectado (o conexo)



GD no conectado



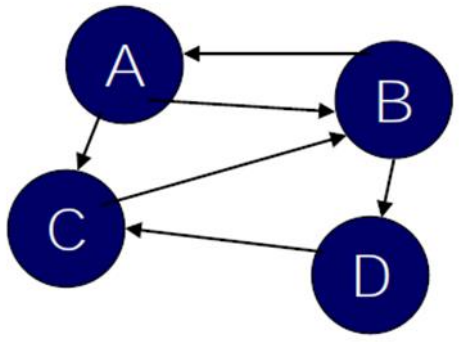
GND conectado



GND no conectado

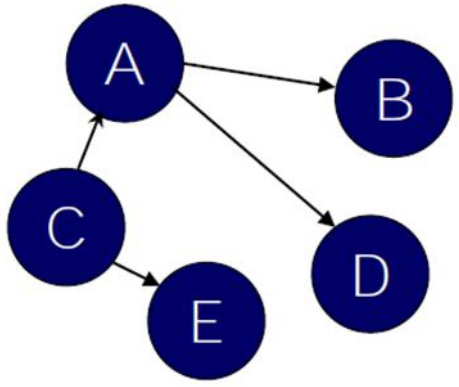
Ciclos

- Un **ciclo** es un **camino cerrado sin aristas repetidas (**el origen es igual a su destino). O sea, es un camino simple sin aristas repetidas v1, v2, ... , vk tal que v1=vk.
  - **Bucle**: Un bucle es una arista que conecta a un vértice consigo mismo. Es un ciclo de longitud 1.
- Un grafo es **Acíclico** si no contiene ciclos, de lo contrario se llama grafo **Cíclico**.
- Este concepto se aplica tanto para GD como para GND.

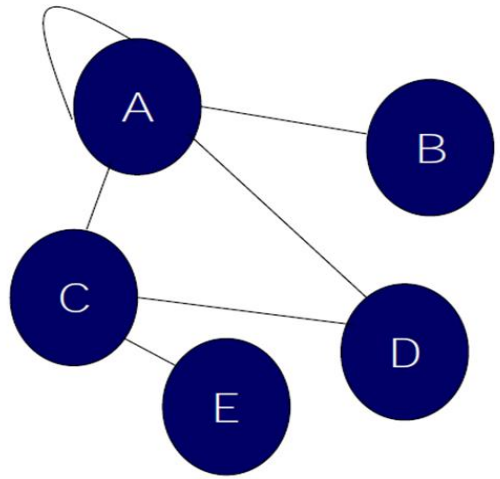


Grafo cíclico

ciclo (A,C,B,A) de longitud 3

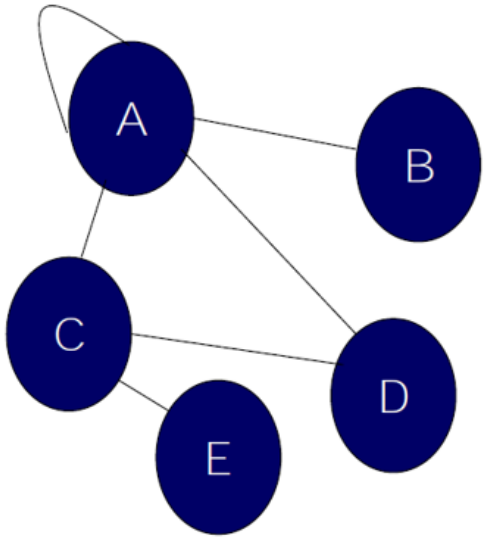


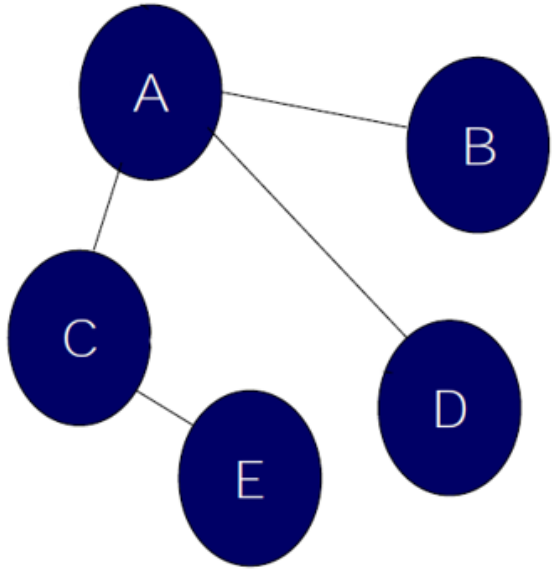
Grafo acíclico



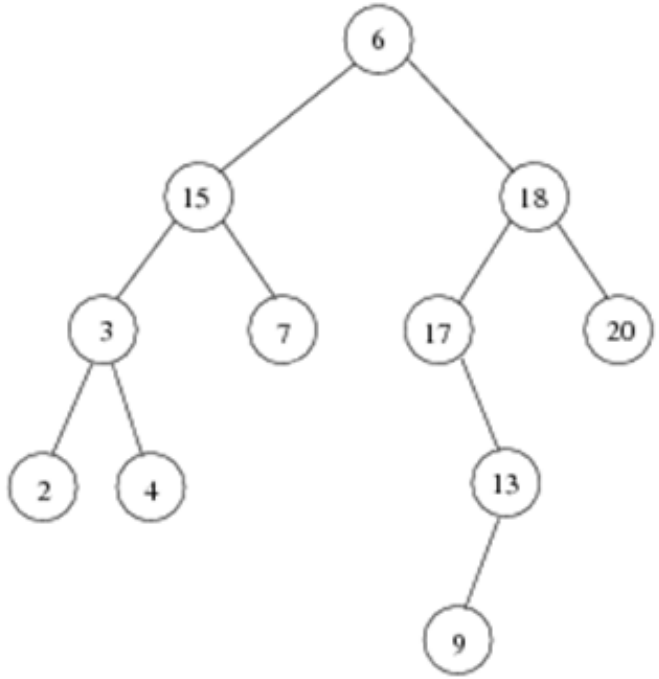
Grafo cíclico

Grafo cíclico y acíclico





- ¿Podemos deducir alguna propiedad para los GND que indique cuándo hay ciclos?



- Un árbol es un GND acíclico y conexo.

Implementación de Grafos

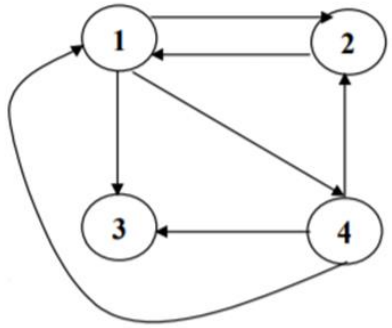
Matriz de Adyacencia

Con la matriz de adyacencia, los arcos de un grafo G=(V,A) se representan como elementos de una matriz de *n x n*, donde *n* es |V| (la cardinalidad de V, o sea la cantidad de vértices del conjunto V).

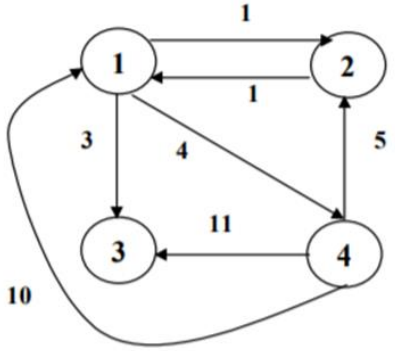
Para un grafo no ponderado, el valor (i,j) de la matriz será 1 (o verdadero) si el arco (i,j) pertenece a A. Si el arco no pertenece, el valor será 0 (o falso).

Si el grafo es ponderado, se sustituyen los 1 por el valor de los rótulos y los 0 (o sea cuando no hay arco) por un valor especial (∞, o un valor que no tenga sentido para nuestro problema).

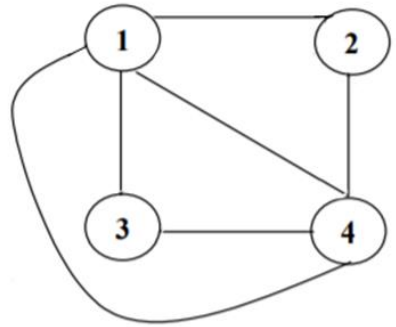
Cuando el grafo en no dirigido: si existe la arista (i,j) entonces también existe la (j,i).



	1	2	3	4
1	0	1	1	1
2	1	0	0	0
3	0	0	0	0
4	1	1	1	0



	1	2	3	4
1	∞	1	3	4
2	1	∞	∞	∞
3	∞	∞	∞	∞
4	10	5	11	∞



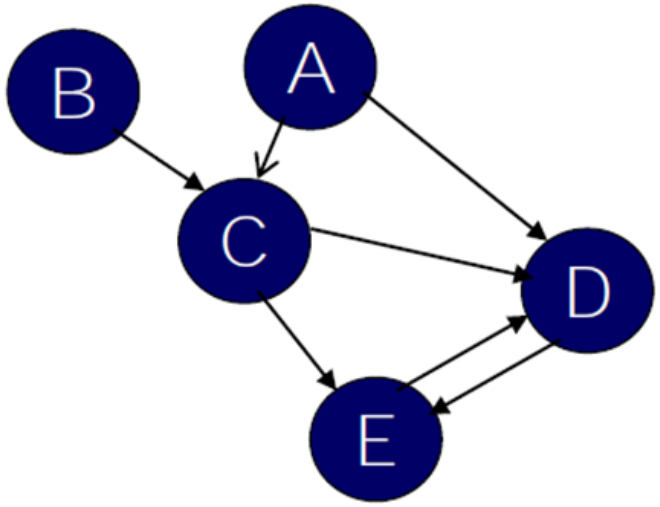
	1	2	3	4
1	0	1	1	1
2	1	0	0	1
3	1	0	0	1
4	1	1	1	0

Contra: Si el grafo no es denso (conectado todos con todos), desperdicia espacio.

GND: Matriz simétrica.

Implementación de Grafos

Matriz de Adyacencia



A	B	C	D	E
0	1	2	3	4

Mapeo

	0	1	2	3	4
0	0	0	1	1	0
1	0	0	1	0	0
2	0	0	0	1	1
3	0	0	0	0	1
4	0	0	0	1	0

Matriz de Adyacencia

Operaciones (para un grafo de n vertices):

- Listar los adyacentes a un vértice **O(n)**
- Decidir si un vértice es adyacente a otro **O(1)**

Implementación de Grafos

Matriz de Adyacencia

Algunas desventajas:

- Se debe conocer la cantidad de vértices al crear la matriz.
- Agregar o quitar vértices es una operación cara, sería crear nueva matriz y copiar todos los datos existentes de la anterior a la nueva O(n^2)).
- Si el grafo es poco conectado (no es denso), se desperdicia mucho espacio.

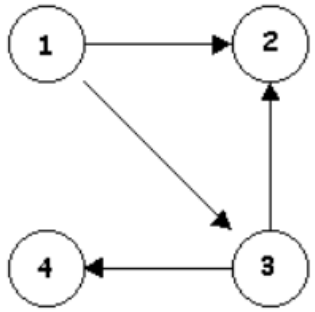
Algunas ventajas:

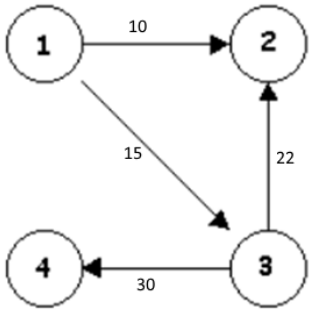
- Ver si existe una arista entre dos vertices es O(1).
- Listar los adyacentes a un vértice es O(n).

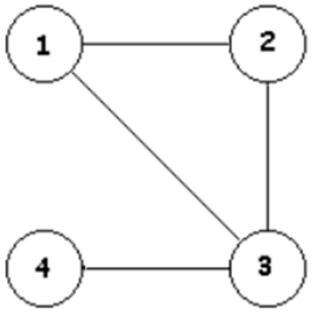
Implementación de Grafos

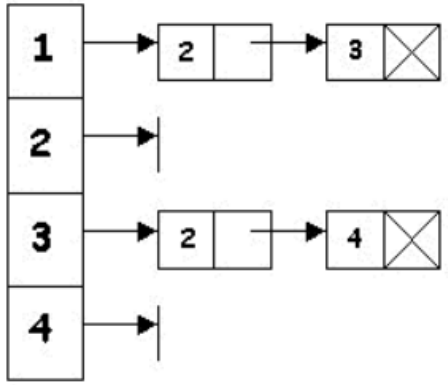
Lista de Adyacencia

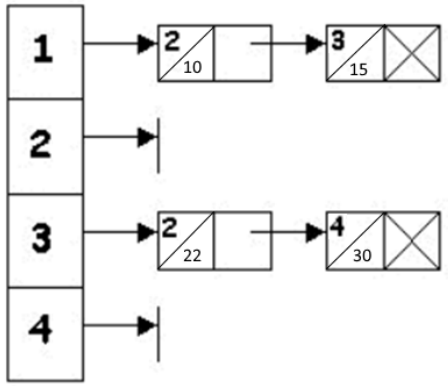
Definimos un array de vértices del grafo.
Cada elemento del array contendrá además del vértice, la referencia a una lista simplemente vinculada de nodos con sus vértices adyacentes, y si el grafo es ponderado el nodo contendrá también el valor del rótulo.

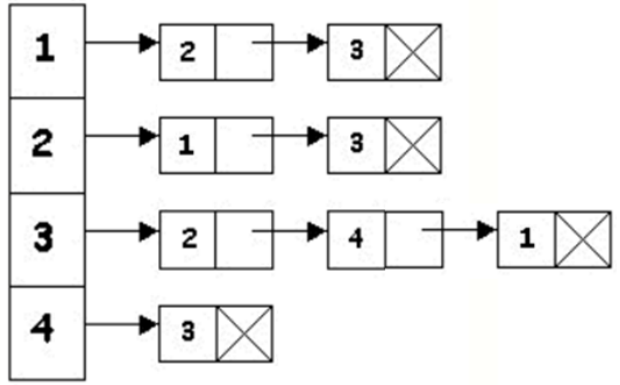












Operaciones (para un grafo de n vértices):

- Listar los adyacentes a un vértice **O(n)**
- Decidir si un vértice es adyacente a otro **O(n)**

Implementación de Grafos

Lista de Adyacencia

Algunas desventajas:

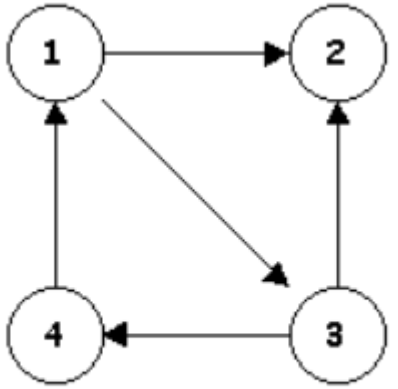
- Se debe conocer la cantidad de vértices al crear el array.
- Agregar o quitar vértices es una operación cara (ya que se debe redimensionar el array de vértices).
- Incremento de complejidad temporal en algunas operaciones.

Posible mejora a algunas de esas cuestiones:

En vez de utilizar un array de vértices, usar una lista simplemente vinculada de vértices. Esto permitirá agregar y quitar vértices a bajo costo, y no tendremos la necesidad de conocer la cantidad de vértices previamente.

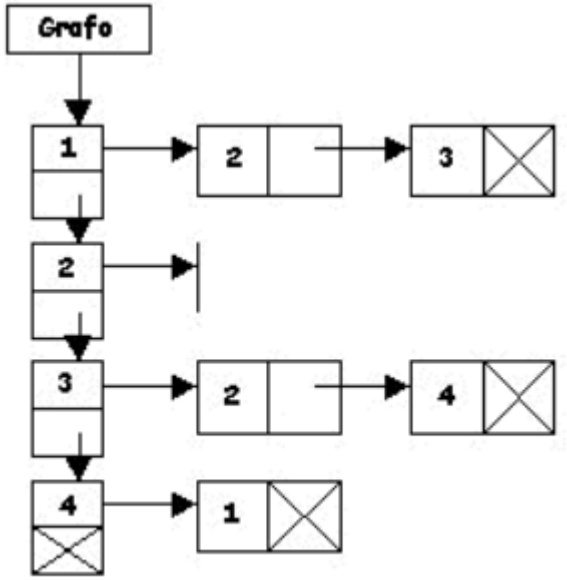
Implementación de Grafos

Lista de Listas de Adyacencia



Operaciones (para un grafo de n aristas):

- Listar los adyacentes a un vértice **O(n) + O(n) ε O(n)**
- Decidir si un vértice es adyacente a otro **O(n)**



No es necesario conocer de antemano la cantidad de vertices, y agregar o quitar vertices es simplemente agregar o quitar nodos de listas O(n).

La cantidad total de nodos de las listas va a ser |V| + |A|