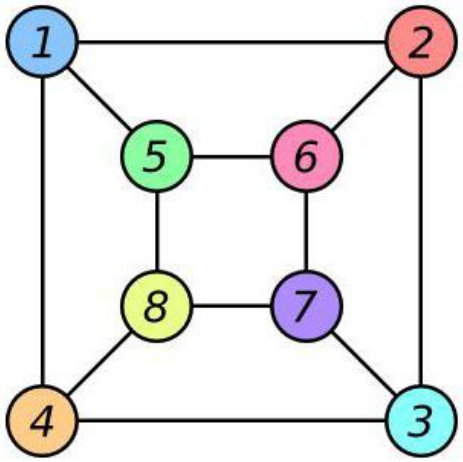
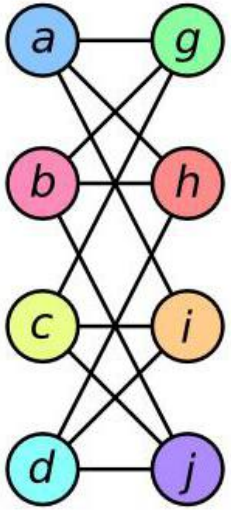
Algoritmos
Estructuras de Datos I

Facultad de Ciencias Exactas y Tecnología
Universidad Nacional de Tucumán
2024

Grafo(1)





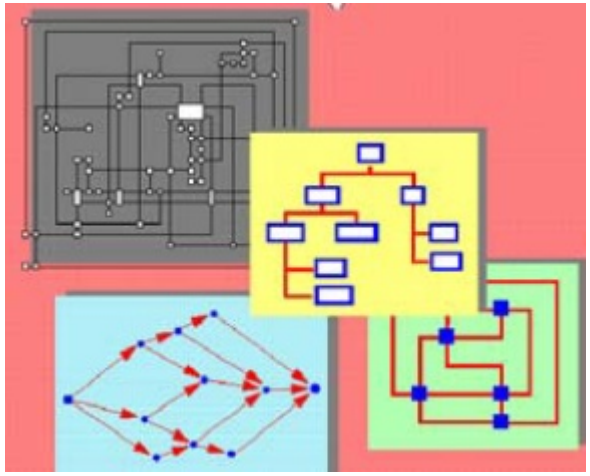
Unidad III - Contenidos

Tipos de datos no lineales - GRAFO.

Grafos. Definiciones. Grafos no dirigidos y dígrafos. El tipo abstracto de datos grafo. Representación con matriz de adyacencia y con listas de adyacencia. Recorrido en un grafo: en profundidad y en amplitud. Árbol de recubrimiento mínimo. Algoritmo de Prim. Ciclo euleriano y ciclo hamiltoniano. Camino en un grafo, el problema de los caminos mínimos. Algoritmos de Dijkstra, Floyd y Warshall.

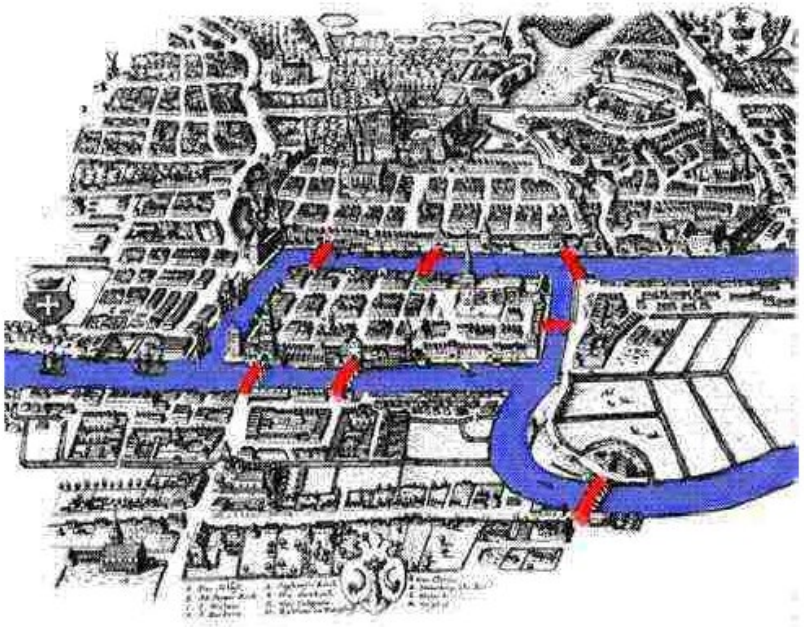
Aplicaciones actuales de grafos

- Redes de comunicaciones, diseño, ruteo.
- Problemas de distribución y ruteo de vehículos.
- Diseño de redes de telefonía móvil.
- Planificación de la producción.
- Redes de tráfico
- Demostración de teoremas.
- Correctitud de programas
- Diseño de circuitos
- Descifrado de Códigos
- Ingeniería de Software
- Bases de datos
- etc.. etc… etc…

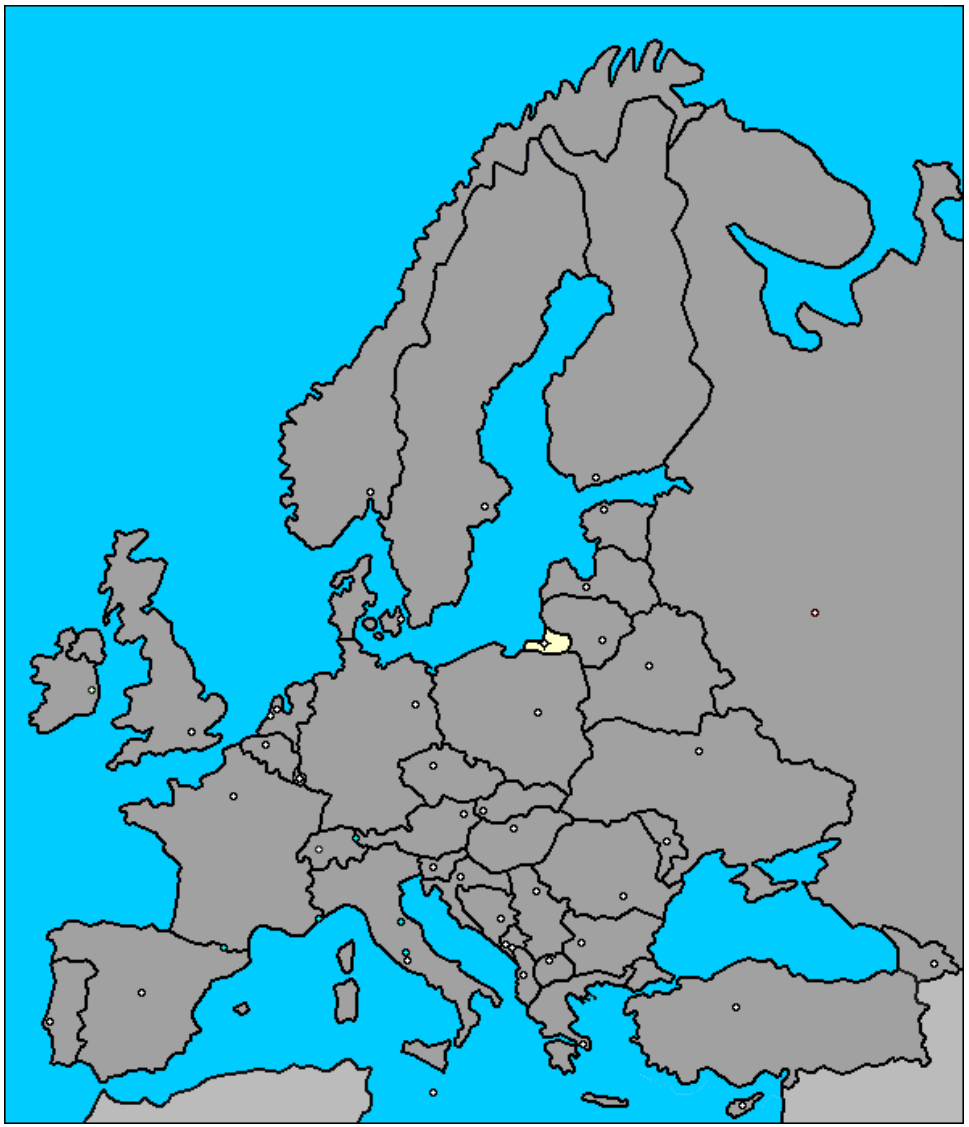


Historia
Puentes de Königsberg

La primera evidencia que se registra del uso de grafos data de 1736 cuando **Leonhard Euler** (1707-1783), matemático del siglo XVIII, suizo de nacimiento) los usó para resolver el ahora clásico ***problema de los puentes de Königsberg***:



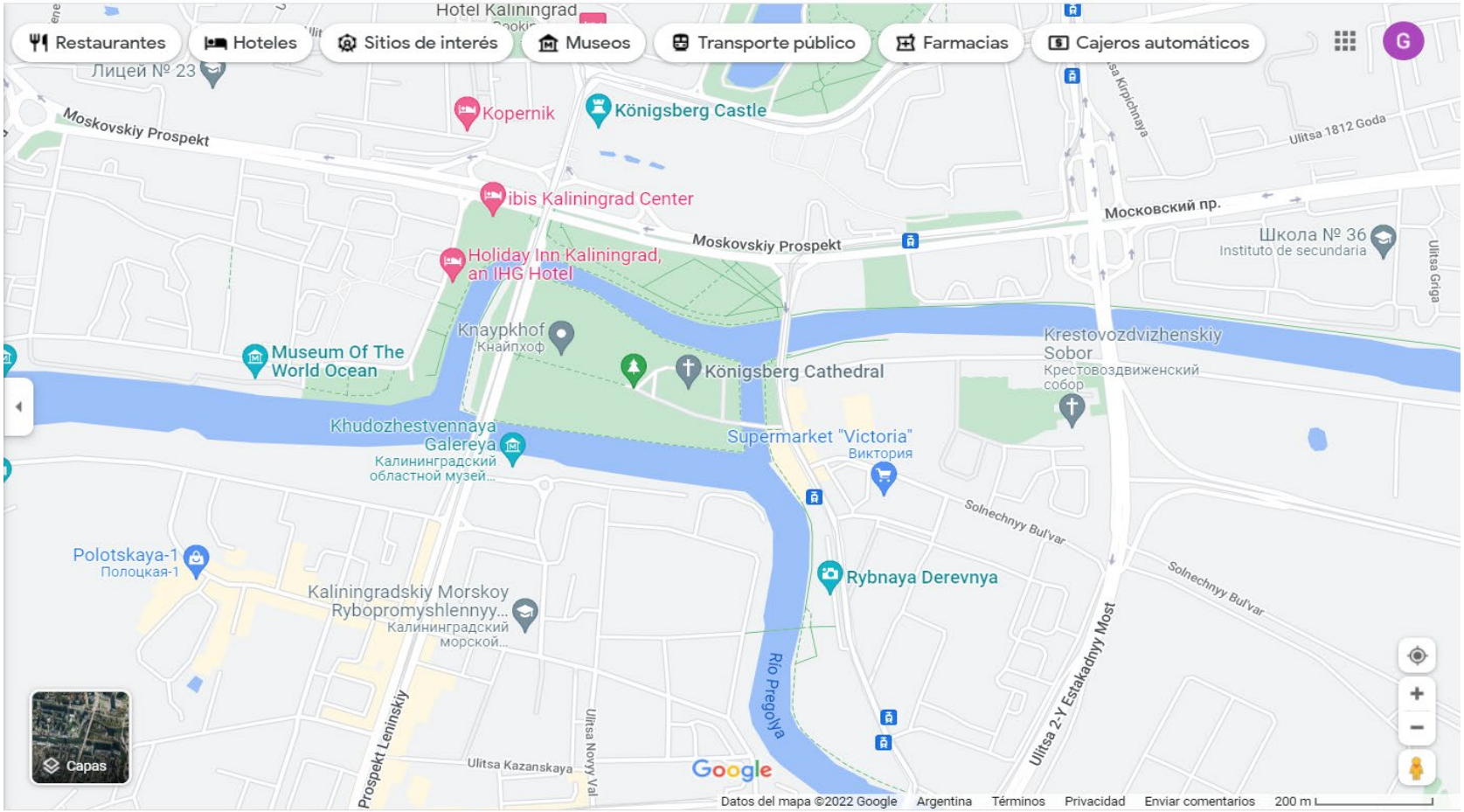
Königsberg hoy Kaliningrado



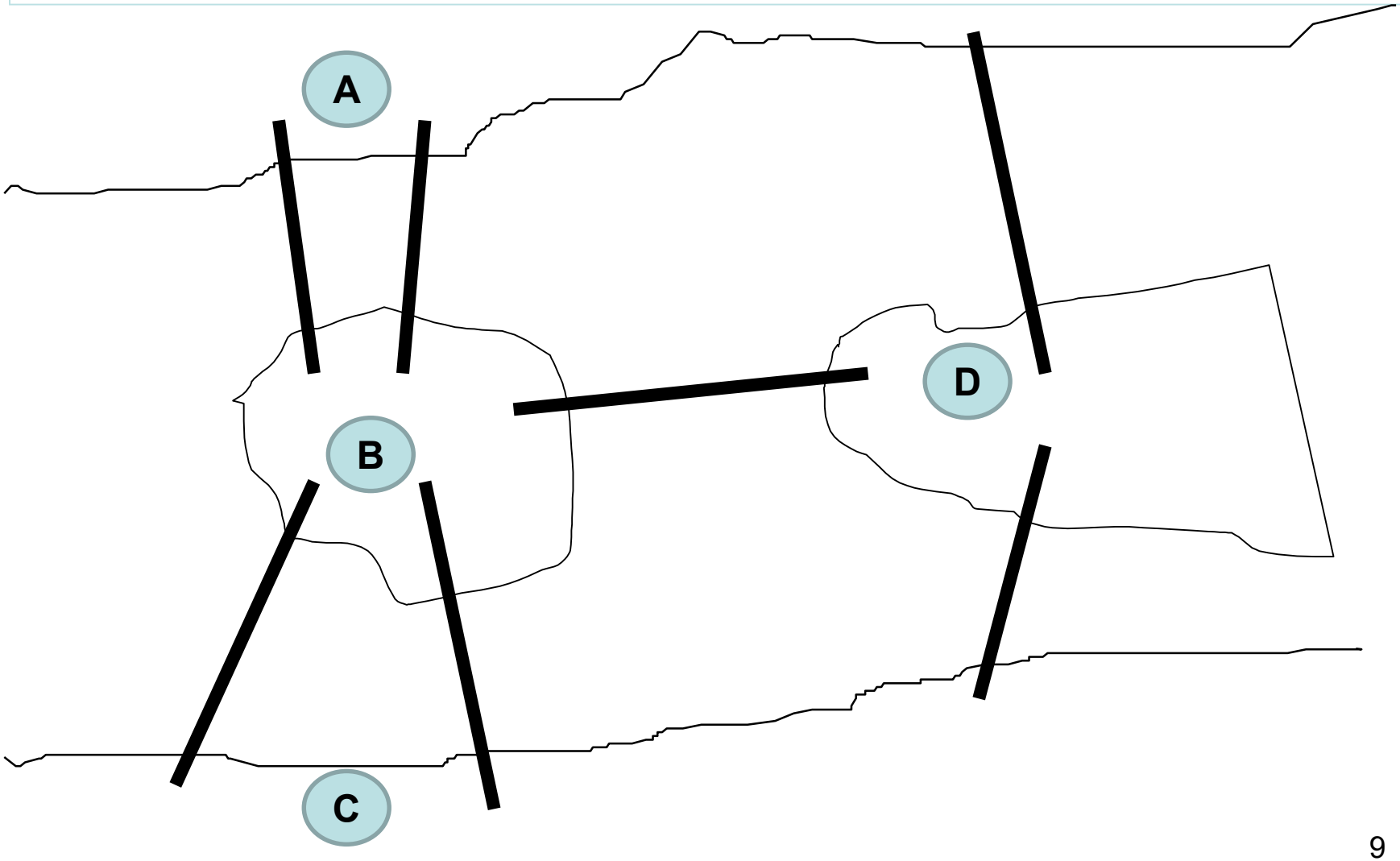
Königsberg hoy Kaliningrado

La antigua Königsberg fundada en 1255 en Prusia, fue posesión prusiana hasta la unificación de Alemania, de la que pasó a formar parte. Hoy es una ciudad portuaria de Europa Oriental perteneciente a Rusia desde 1945. A pesar de ser una parte aislada de Rusia, es un puerto estratégico.

Königsberg


Restaurantes
Hoteles
Sitios de interés
Museos
Transporte público
Farmacias
Cajeros automáticos
Hotel Kaliningrad
Лицей № 23
Kopernik
Königsberg Castle
Moskovskiy Prospekt
ibis Kaliningrad Center
Holiday Inn Kaliningrad, an IHG Hotel
Knaypkhof
Кнайпхоф
Museum Of The World Ocean
Königsberg Cathedral
Krestovozdvizhenskiy Sobor
Крестовоздвиженский собор
Московский пр.
Школа № 36
Instituto de secundaria
Ulitsa 1812 Goda
Ulitsa Grig
Khudozhestvennaya Galereya
Калининградский областной музей...
Supermarket "Victoria"
Виктория
Solnechnyy Bul'var
Polotskaya-1
Полоцкая-1
Kaliningradskiy Morskoy Rybopromyshlennyy...
Калининградский морской...
Rybnaya Derevnya
Río Pregolya
Ulitsa 2-Y Estakadnyy Most
Prospekt Leninskiy
Ulitsa Kazanskaya
Ulitsa Novyy Val
Capas
Google
Datos del mapa ©2022 Google
Argentina
Términos
Privacidad
Enviar comentarios
200 m

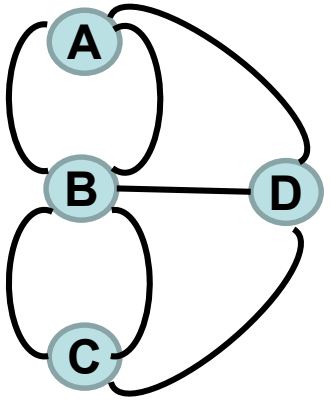
La ciudad de Königsberg tenía en el siglo XVIII siete puentes:



Historia
Puentes de Königsberg

Los habitantes de Königsberg se planteaban el siguiente desafio:

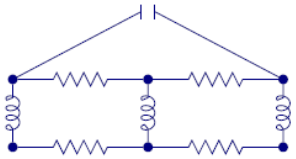
¿Se podría hacer una caminata tal que se cruzaran por todos los puentes sin pasar por ninguno más de una vez, saliendo y llegando al mismo lugar de partida?



- En 1736 **Leonhard Euler** planteó (y resolvió) el problema de cruzar por todos ellos exactamente una vez y volver al punto de partida.
- Euler mostró que el problema no tenía solución y dio una condición necesaria para el caso general.
- Carl Hierholzer (1840-1871) mostro en 1871 que esta condición es también suficiente, y formalizo la demostración.

Historia

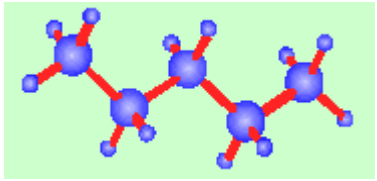
- **Vandermonde, 1771**, Problema de los caballos en un tablero de ajedrez.



- **Wiener, 1873,** Laberintos.

- **Leyes de Kirschhoff (1847)** Introdujo el concepto de árboles para resolver el sistema de ecuaciones lineales que describen el flujo de la corriente eléctrica en cada rama de cada circuito en una red eléctrica. Modeló la red compuesta de resistencias, condensadores, inductancias, etc, con un grafo.

- **Kirkman, 1856**, circuitos en poliedros, pionero en formular el problema del viajante de comercio.



- **Cayley, 1857**: Isómeros Químicos.
  Cuántos compuestos químicos diferentes pueden corresponder a una misma fórmula?

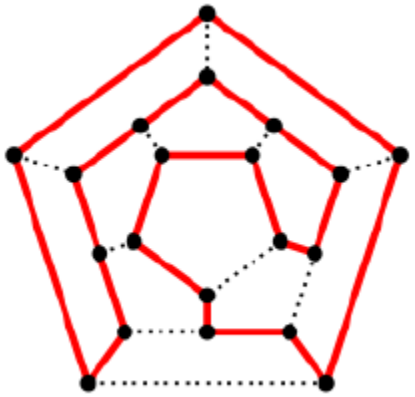
Historia

- **Hamilton, 1858 :** juego, *dar la vuelta "al mundo" sin pasar dos veces por la misma ciudad.*

- Pasar por todos los vértices de un dodecaedro una y sólo una vez y volver a la ciudad de origen.

- Generalización:
  circuitos hamiltonianos.

- Problema del Viajante de Comercio



Historia

Problema de los cuatro colores: *se puede pintar cualquier mapa con cuatro colores sin que dos países que tengan como frontera una línea tengan el mismo color?.*

- Origen impreciso. Primer planteo conocido:

De Morgan 1852. (*Planteado por su alumno Francis Guthrie*)

- Primera "supuesta" demostración, Kempe 1879
- Error descubierto por Heawood, 1890, que demostró el Teorema para 5 colores.
- ***Problema abierto por más de 100 años***
- Avances de la teoría de grafos alrededor de este problema.
- Demostración en 1976, Appel y Haken.
- Uso de la computadora en esta demostración.
- Demostraciones posteriores. 1996 Robertson et all.

Definición de Grafo

Un grafo es una colección de vértices unidos por un conjunto de arcos que se denota como:

$$G = (V , E)$$

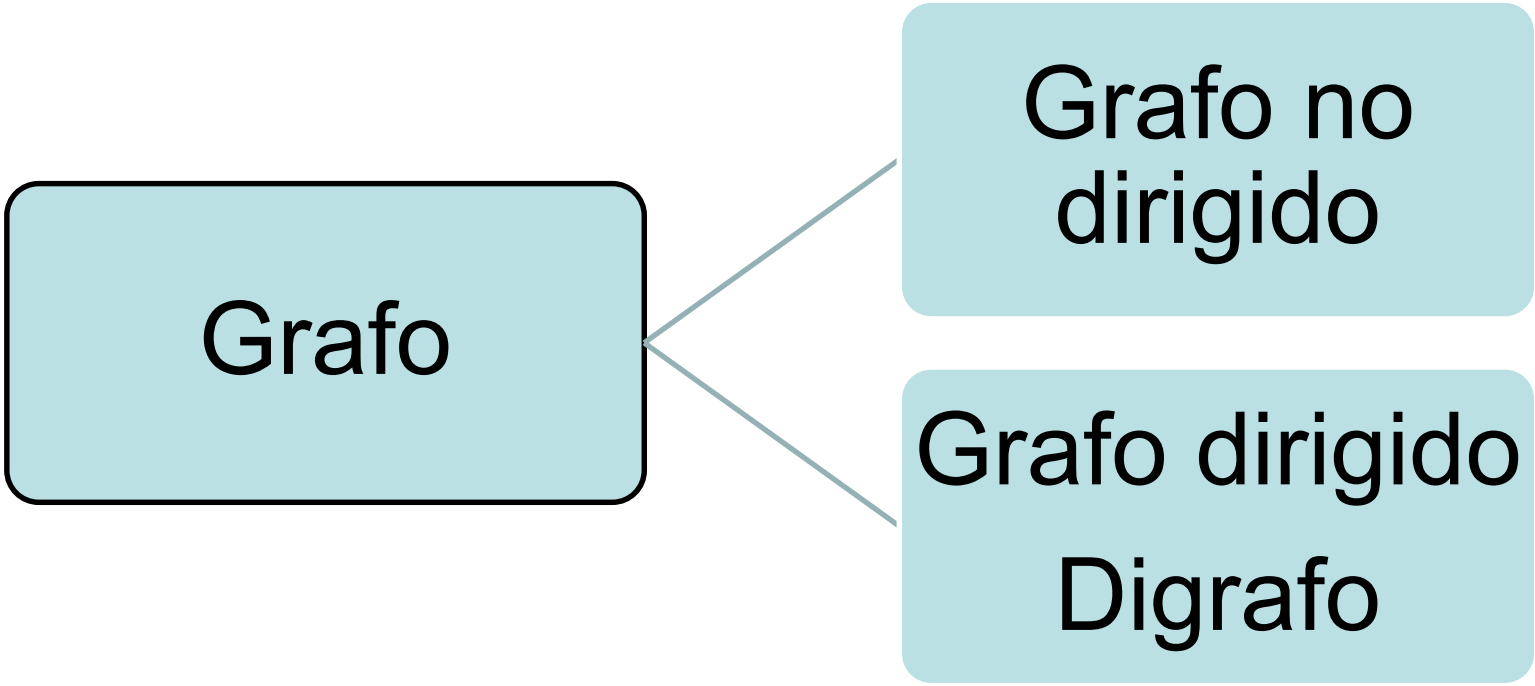
Donde:

V es un conjunto finito de elementos llamados ***vértices*** o ***nodos*** o ***puntos***

E $\subseteq$ VxV: es un conjunto finito de pares (ordenados o no ordenados) de vértices llamados ***aristas*** o ***líneas***.

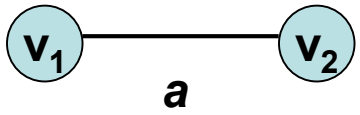
Cada arista tiene la forma (v_1,v_2) donde v_1 y $v_2 \in V$

Tipo de Grafo



Grafo

***Grafo no dirigido*: el par de vértices que representa una arista no tiene orden, de modo que (v_1,v_2) y (v_2,v_1) se refieren a la misma arista.**

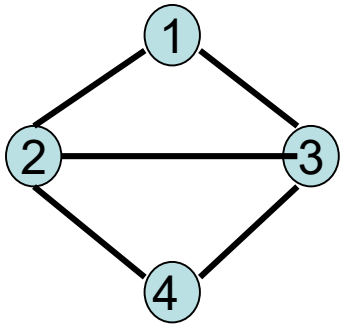


- Si ***a*** = (v_1,v_2) es un arista no dirigida de G, se dice que a une a $\mathbf{v_1}$ con $\mathbf{v_2}$ y que ***a*** incide en $\mathbf{v_1}$ y en $\mathbf{v_2}$.
- Dos vértices son ***adyacentes*** si son extremos de un mismo arco.

Ej.

V_1 = {1,2,3,4}

E_1 = {(1,2),(1,3),(2,3),(2,4),(3,4)}

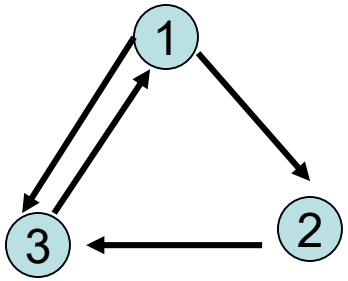


Digrafo

Grafo dirigido o ***digrafo*****:** el par de vértices tiene orden, el par ordenado (v_1,v_2) representa un arco que tiene como origen el vértice $\mathbf{v_1}$ y como destino el vértice $\mathbf{v_2}$.

de modo que $(v_1,v_2)\neq(v_2,v_1)$
En este caso el vértice v_2 es adyacente del vértice v_1 pero no al revés.

Ej.
V_2 = {1,2,3}
E_2 = { (1,2),(2,3),(1,3),(3,1) }



Terminología

Bucle* o *lazo : se denomina a la arista que une un vértice consigo mismo. Es decir es una arista en el que sus dos extremos coinciden en un solo vértice: $a=(v_2,v_1)$, donde $v_2=v_1$.

Un ***grafo simple*** es un grafo sin bucles.

Arcos paralelos: dos arcos son paralelos si tienen los mismos extremos. A un grafo con arcos paralelos se le llama ***multigrafo***.

Un grafo se dice ***sencillo*** si carece de lazos y de aristas paralelas.

Terminología

Grafo:

Grado de un vértice es el numero de aristas que inciden en él, contando cada lazo por 2.

Digrafo:

Grado de entrada: es el número de aristas que inciden en v;

Grado de salida: es el número de aristas que salen de v;

Grado de un vértice v: es la suma de los grados de entrada y salida de v.

El grado de un vértice se denota gr(v)

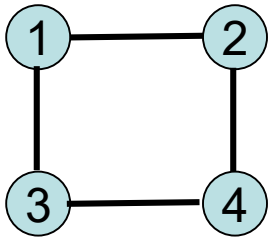
Grado de un grafo *es el máximo grado de sus vértices.*

Un vértice es ***aislado*** si su grado es cero.

Terminología

Un grafo se denomina ***regular*** si todos los vértices tienen el mismo grado.

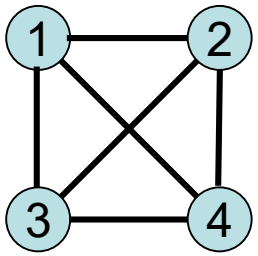
Ej.



Un grafo se dice ***completo*** si cualquier par de vértices distintos son adyacentes, son grafos con todas sus aristas posibles.

Esto es : G=(V,E) en el que $\forall$ u,v $\in$ V, u$\neq$v, (u,v) $\in$ E

Ej.

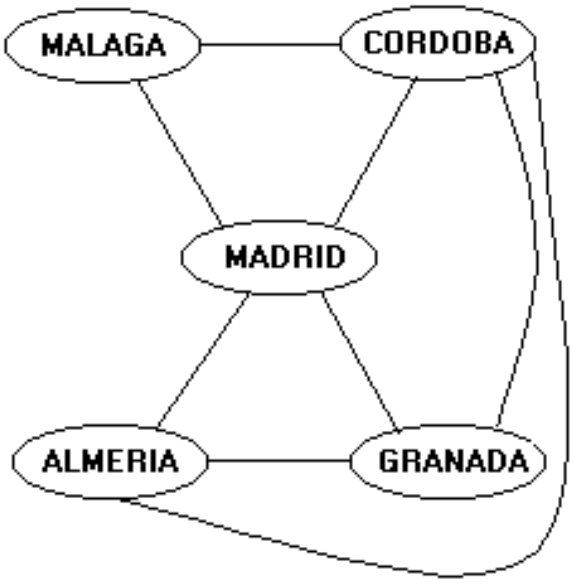


Terminología

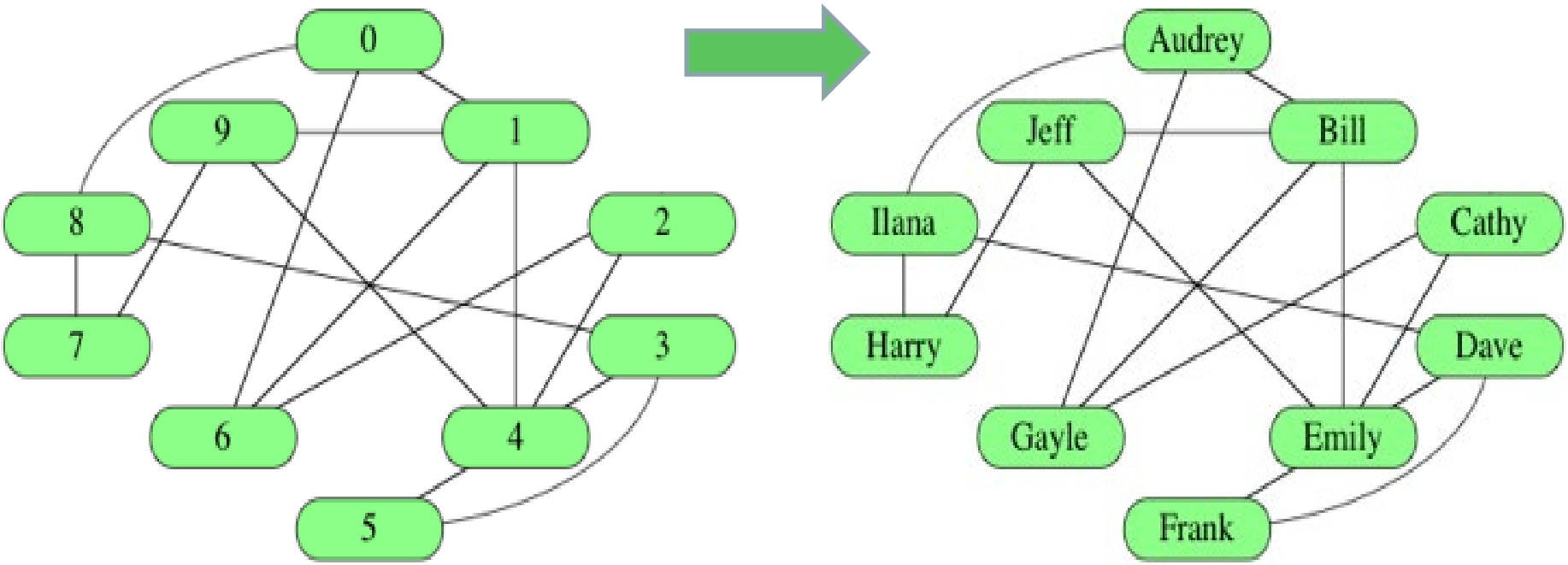
Un ***grafo etiquetado*** es aquel en que los vértices llevan cierta información.

Se acompaña al grafo G=(V,E) de una función: **g : V → $Tipo_V$**, donde $Tipo_V$ es un conjunto cuyas componentes se denominan etiquetas.

Ejemplo: en un mapa de rutas se incluye el nombre de ciudades.



Grafo Etiquetado
Ejemplo:



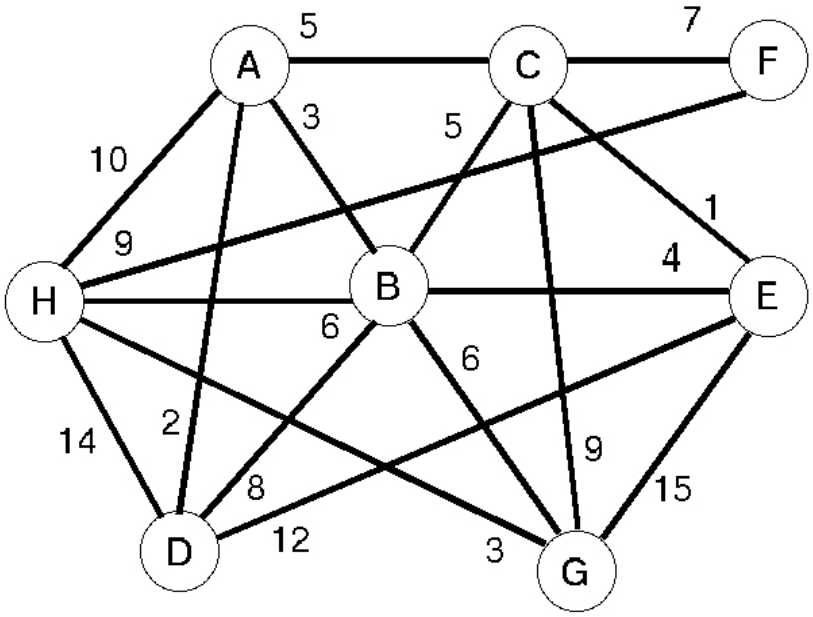
Grafo Etiquetado
Ejemplo:

TERMINOLOGIA

Un ***grafo valuado*** o ***ponderado*** o con ***costos*** o con **pesos** es aquel en que se agrega información a las aristas.

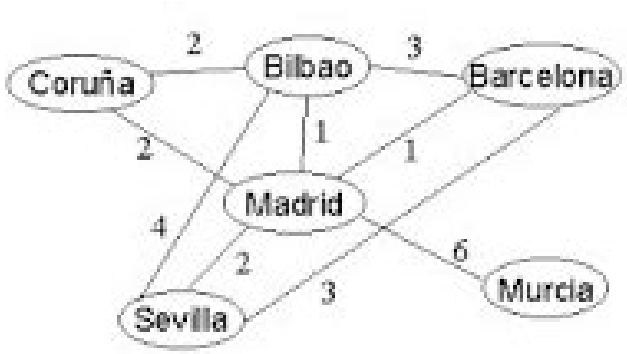
Es un grafo G=(V,E) acompañado de una función **f : E → Tipo$_E$**, donde Tipo$_E$ es un conjunto cuyas componentes se denominan costos.

Ejemplo:

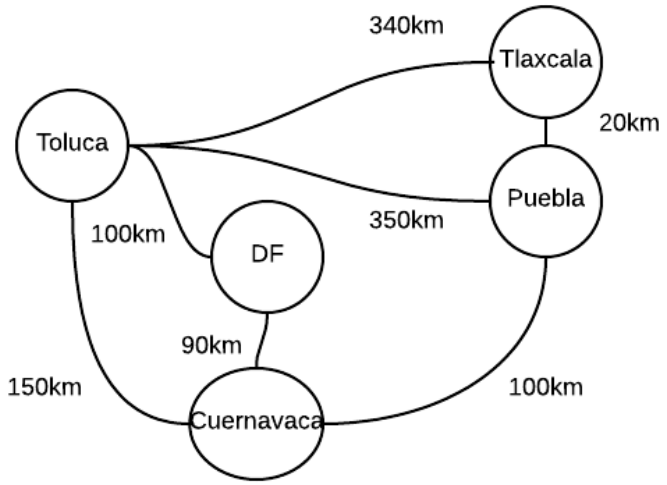


Grafo con costos
Ejemplo:

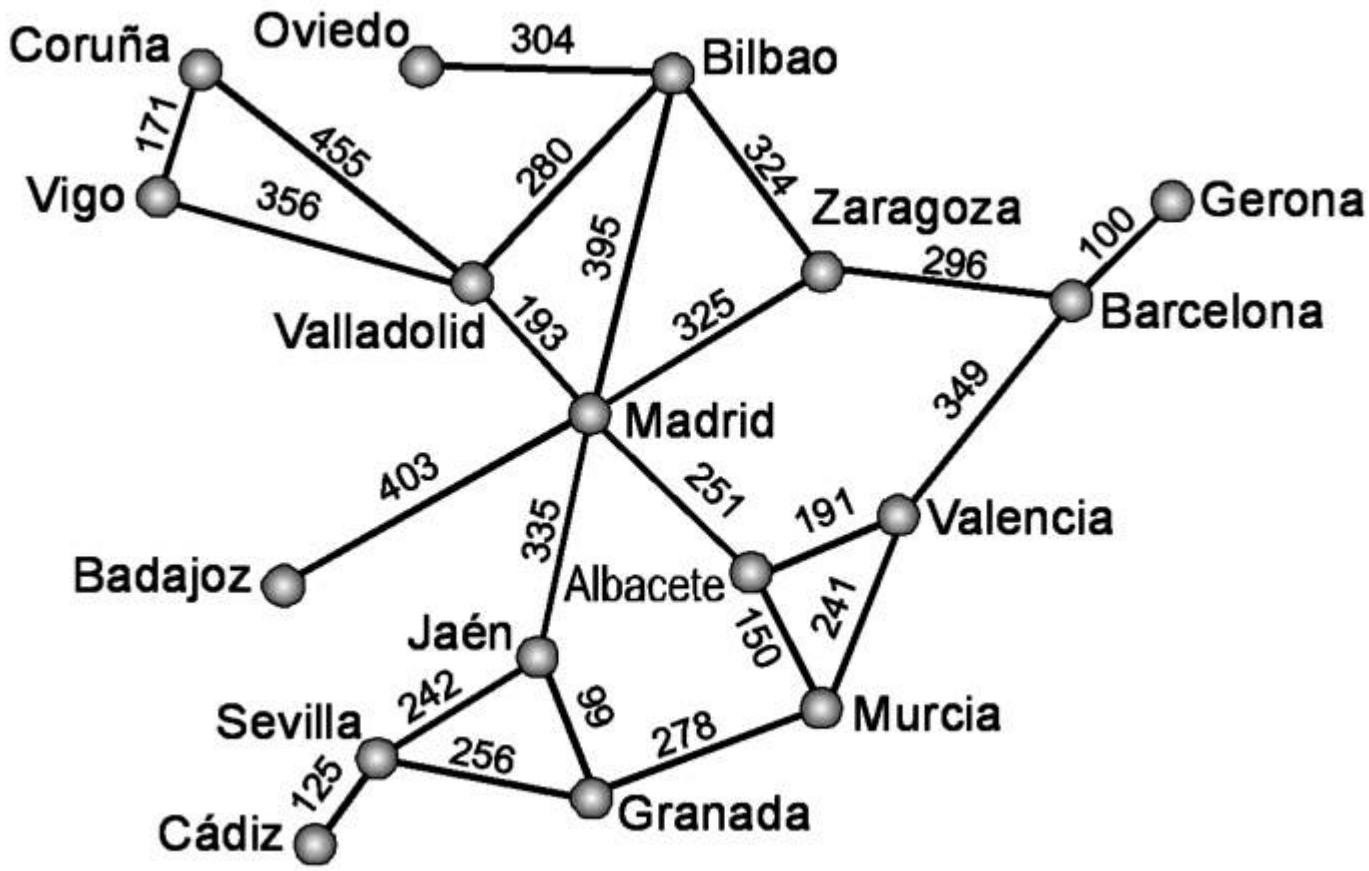
Ejemplo: tiempo de vuelo entre ciudades



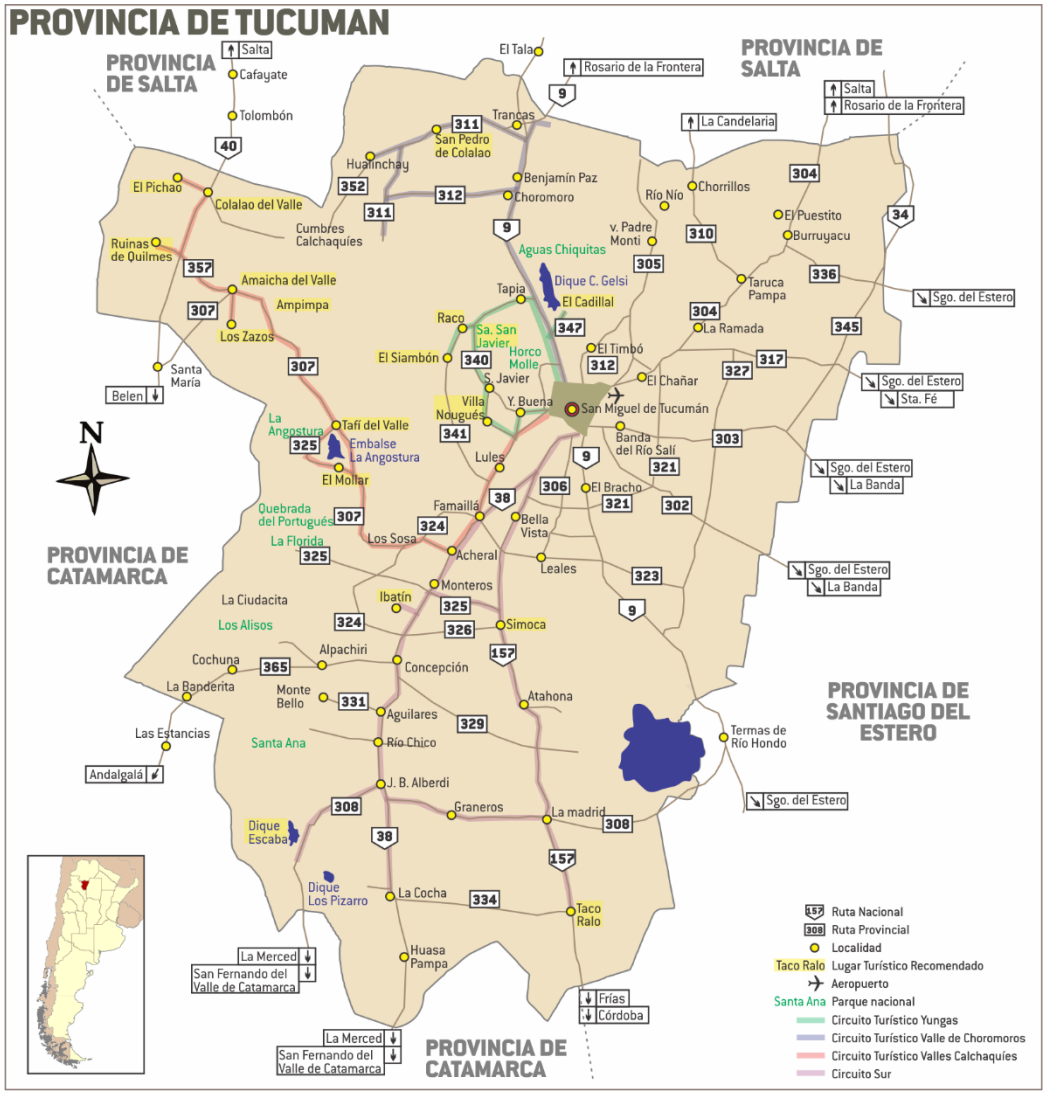
Ejemplo: en el mapa se agrega la distancia entre 2 ciudades.



Grafo Etiquetado y Ponderado
Ejemplo:



Grafo Etiquetado y Ponderado
Ejemplo:



TERMINOLOGIA

Orden de un grafo: es el numero de vértices del grafo, se denota con la letra ***n***

Tamaño de un grafo: es el numero de aristas o arcos del grafo, se denota con la letra ***a***

Grafo denso: es aquel en que el numero de arcos se aproxima al numero de vértices al cuadrado. Esto es $a \approx n^2$.

Grafo esparcido **o** ***Grafo disperso***: es un grafo con pocos arcos.

Cuando a<n logn.

GRAFO(TipoVertice,TipoArco)

Especificación algebraica - OPERACIONES:

Sintaxis:

GRAFOVACIO **:** → GRAFO
AGREGAVERTICE **:** GRAFO X TipoVertice → GRAFO
AGREGAARCO **:** GRAFO X TipoArco → GRAFO
VERTICES **:** GRAFO → CONJUNTO(TipoVertice)
ARCOS **:** GRAFO → CONJUNTO(TipoArco)
ADYACENTES **:** GRAFO X TipoVertice → CONJUNTO(TipoVertice)
BORRAVERTICE **:** GRAFO X TipoVertice → GRAFO
BORRAARCO **:** GRAFO X TipoArco → GRAFO
EXISTEARCO **:** GRAFO X TipoArco → BOOL

B) Semántica: $\forall$ g de GRAFO, $\forall$ i,j,v,w de TipoVertice, $\forall$[i,j],[v,w] de TipoArco

VERTICES(GRAFOVACIO) ≡ CONJVACIO
VERTICES(AGREGAVERTICE(g,v)) ≡ INSERTAR(VERTICES(g),v)
VERTICES(AGREGAARCO(g,[i,j])) ≡ INSERTAR(INSERTAR(VERTICES(g),i),j)

ARCOS(GRAFOVACIO) ≡ CONJVACIO
ARCOS(AGREGAVERTICE(g,v)) ≡ ARCOS(g)
ARCOS(AGREGAARCO(g,[i,j])) ≡ INSERTAR(ARCOS(g),[i,j])

ADYACENTES(GRAFOVACIO,v) ≡ CONJVACIO
ADYACENTES(AGREGAVERTICE(g,w),v) ≡ ADYACENTES(g,v)
ADYACENTES(AGREGAARCO(g,[i,j]),v) ≡
si v=i entonces INSERTAR(ADYACENTES(g,v),j)
sino si v=j entonces INSERTAR(ADYACENTES(g,v),i)
sino ADYACENTES(g,v)

BORRAVERTICE(GRAFOVACIO,v) ≡ GRAFOVACIO

BORRAVERTICE(AGREGAVERTICE(g,w),v) ≡ si v=w entonces g
sino AGREGAVERTICE(BORRAVERTICE(g,v),w)

BORRAVERTICE(AGREGAARCO(g,[i,j]),v) ≡ si v=i or v=j entonces
BORRAVERTICE(g,v)
sino AGREGAARCO(BORRAVERTICE(g,v),[i,j]))

BORRAARCO(GRAFOVACIO,[i,j]) ≡ GRAFOVACIO

BORRAARCO(AGREGAARCO(g,[i,j]),[v,w]) ≡ si [v,w]=[i,j] entonces g
sino AGREGAARCO(BORRARCO(g,[v,w]),[i,j])

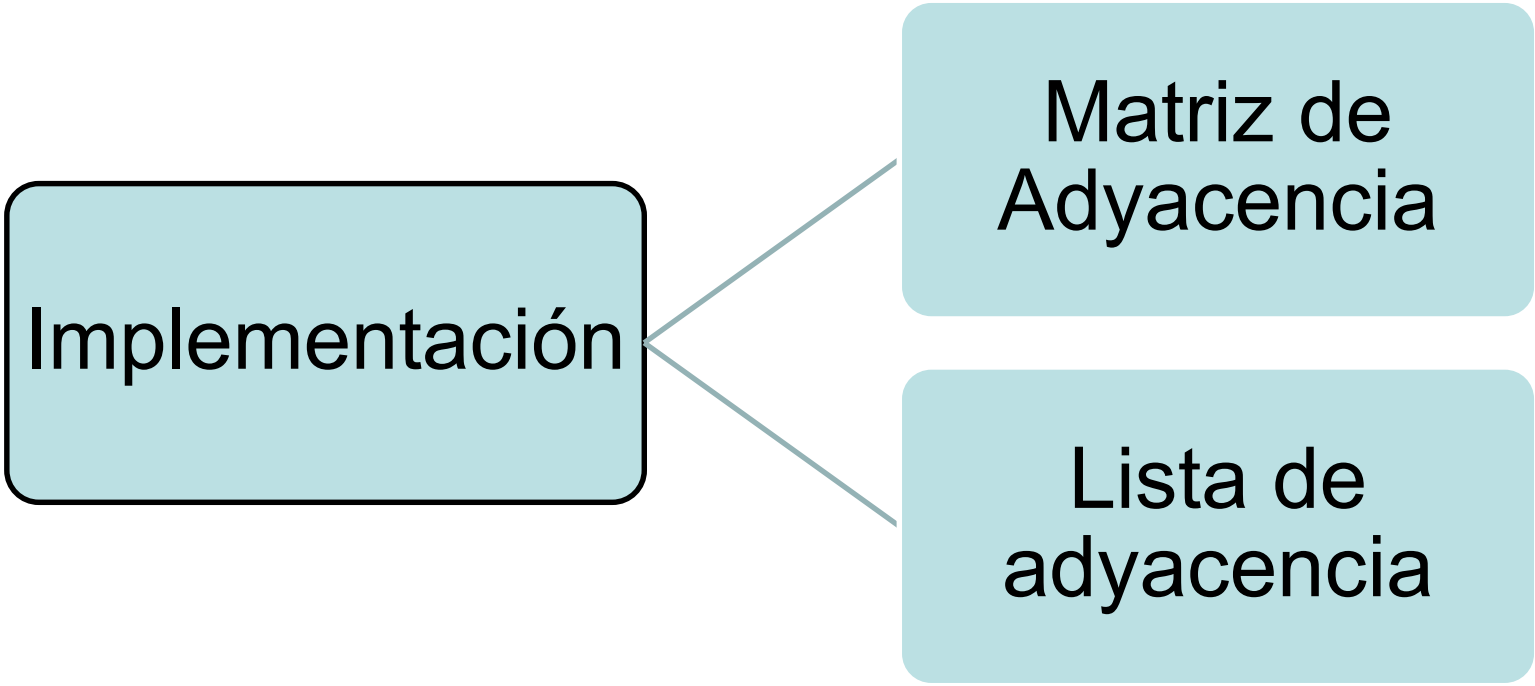
BORRAARCO(AGREGAVERTICE(g,i) ,[v,w]) ≡
AGREGAVERTICE(BORRAARCO(g,[v,w]),i)

EXISTEARCO(GRAFOVACIO,[i,j]) ≡ FALSE

EXISTEARCO(AGREGAVERTICE(g,i),[v,w]) ≡ EXISTEARCO(g,[v,w])

EXISTEARCO(AGREGAARCO(g,[i,j]),[v,w]) ≡ si [v,w]=[i,j] entonces
TRUE
sino EXISTEARCO(g,[v,w])

Implementación de un Grafo



Ventajas y Desventajas?
Costos?

Grafo – Implementación con Matriz de Adyacencia

La **matriz de adyacencia** es la representación mas directa de un grafo .

Sea G=(V,E) un grafo donde V= $\{v_1, v_2, ..., v_n\}$, la matriz de adyacencia de G es una ***matriz A cuadrada de orden n***, de elementos booleanos, donde:

A(i,j) = true si y solo si existe un arco de v_i a v_j.

A(i,j) = false si y solo si no existe un arco de v_i a v_j.

También se puede representar A con ceros y unos:

A(i,j) = 1 si y solo si existe un arco de v_i a v_j.

A(i,j) = 0 si y solo si no existe un arco de v_i a v_j.

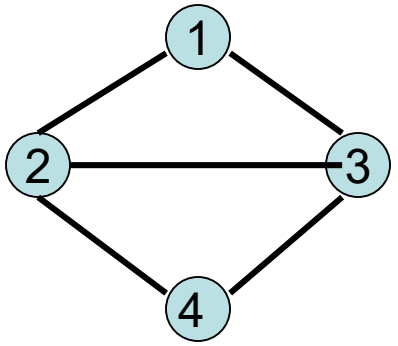
La matriz A es simétrica cuando se trata de un grafo no dirigido.

En el caso de un digrafo no se puede hacer ninguna afirmación.

Grafo – Implementación con Matriz de Adyacencia

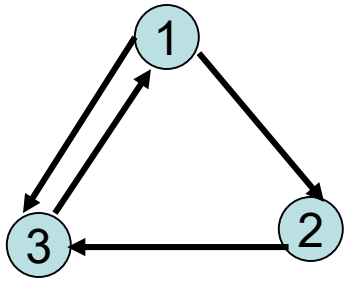
Ej1.

$$A_1 = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$



Ej2.

$$A_2 = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$



Grafo – Implementación con Matriz de Adyacencia con costo

La **matriz de adyacencia** se puede usar para almacenar un valor de otro tipo en lugar de un valor booleano.

Con esta representación se puede usar una ***matriz de adyacencia con costo***, donde A(i,j) es el ***rótulo*** o ***valor*** o ***costo*** del arco que une el vértice v_i con el vértice v_j. Si no existe tal arco se hace la convención de usar un valor especial constante por ejemplo infinito.

Sea G=(V,E) un grafo donde V= $\{v_1, v_2, ..., v_n\}$, la matriz de adyacencia con costo de G es una ***matriz A cuadrada de orden n***, donde:

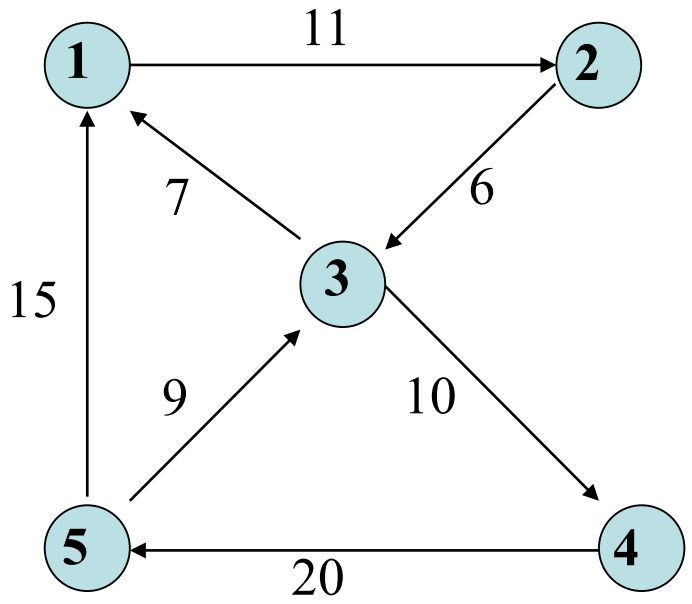
A(i,j) = costo si y solo si existe un arco de v_i a v_j.

A(i,j) = ∞ si y solo si no existe un arco de v_i a v_j.

Grafo – Implementación con Matriz de Adyacencia con costo

Ejemplo:

$$A = \begin{bmatrix} \infty & 11 & \infty & \infty & \infty \\ \infty & \infty & 6 & \infty & \infty \\ 7 & \infty & \infty & 10 & \infty \\ \infty & \infty & \infty & \infty & 20 \\ 15 & \infty & 9 & \infty & \infty \end{bmatrix}$$



Grafo – Implementación con Lista de Adyacencia

La **matriz de adyacencia** es una representación adecuada para grafos densos, esto es, cuando el número de arcos es casi el número de vértices al cuadrado:

$$a \approx n^2$$

La matriz de adyacencia requiere almacenamiento proporcional a $O(n^2)$, aun cuando tenga muchas entradas nulas.

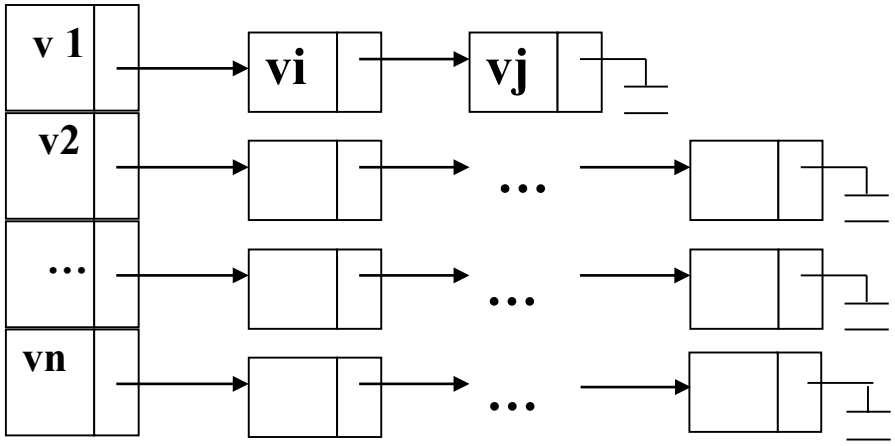
Si n es grande y el numero de arcos es pequeño, habrá que usar una representación más conveniente en cuanto al uso de almacenamiento. Para grafos esparcidos (no densos) es más adecuada una representación con listas de adyacencia.

Grafo – Implementación con Lista de Adyacencia

En la representación con **lista de adyacencia** para cada vértice del grafo se mantiene una lista de todos sus vértices adyacentes.

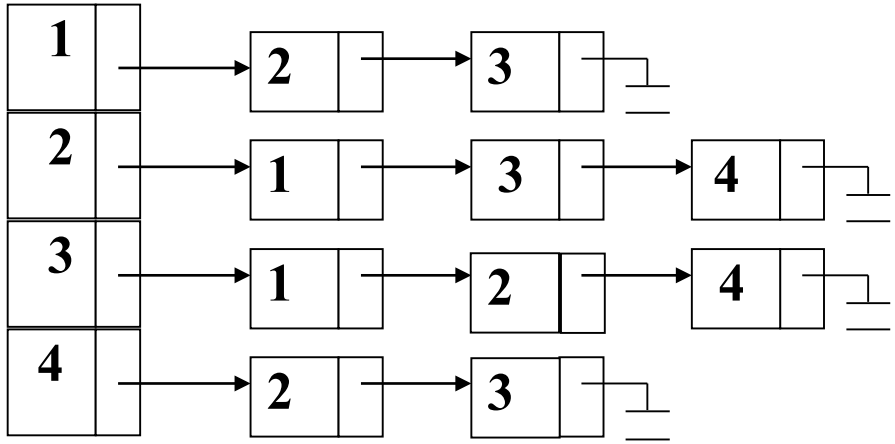
Esa lista se puede implementar con una lista enlazada o con un vector de vértices adyacentes.

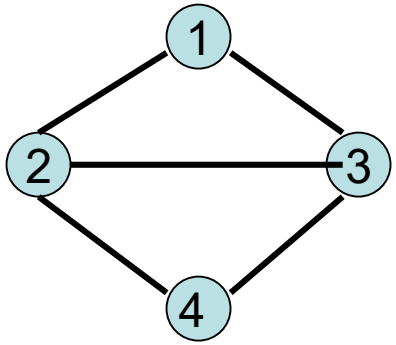
El grafo se representa generalmente por un vector **A**, donde cada entrada **A(k)** es un puntero a la lista del vértice $\mathbf{v_k}$.



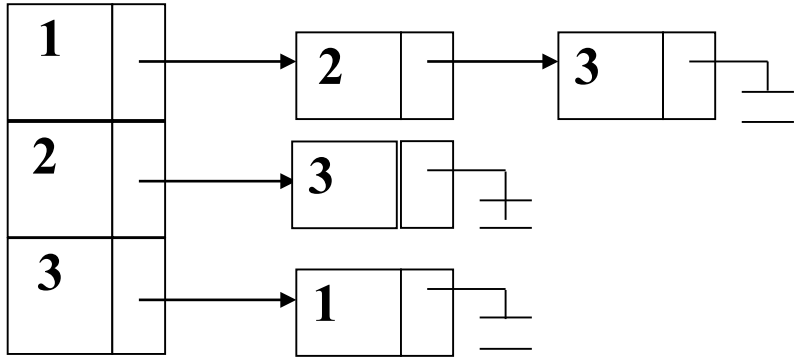
Grafo – Implementación con Lista de Adyacencia

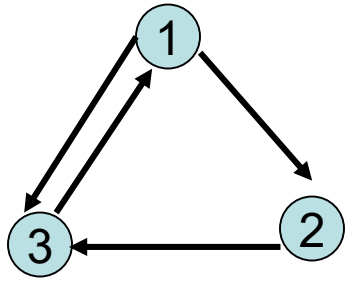
Ej1.





Ej2.





Grafo – Implementación con Listas de Adyacencia con costo

Ejemplo: