Análisis de Algoritmos y Estructura de Datos

TDA grafo

Prof. Violeta Chang C

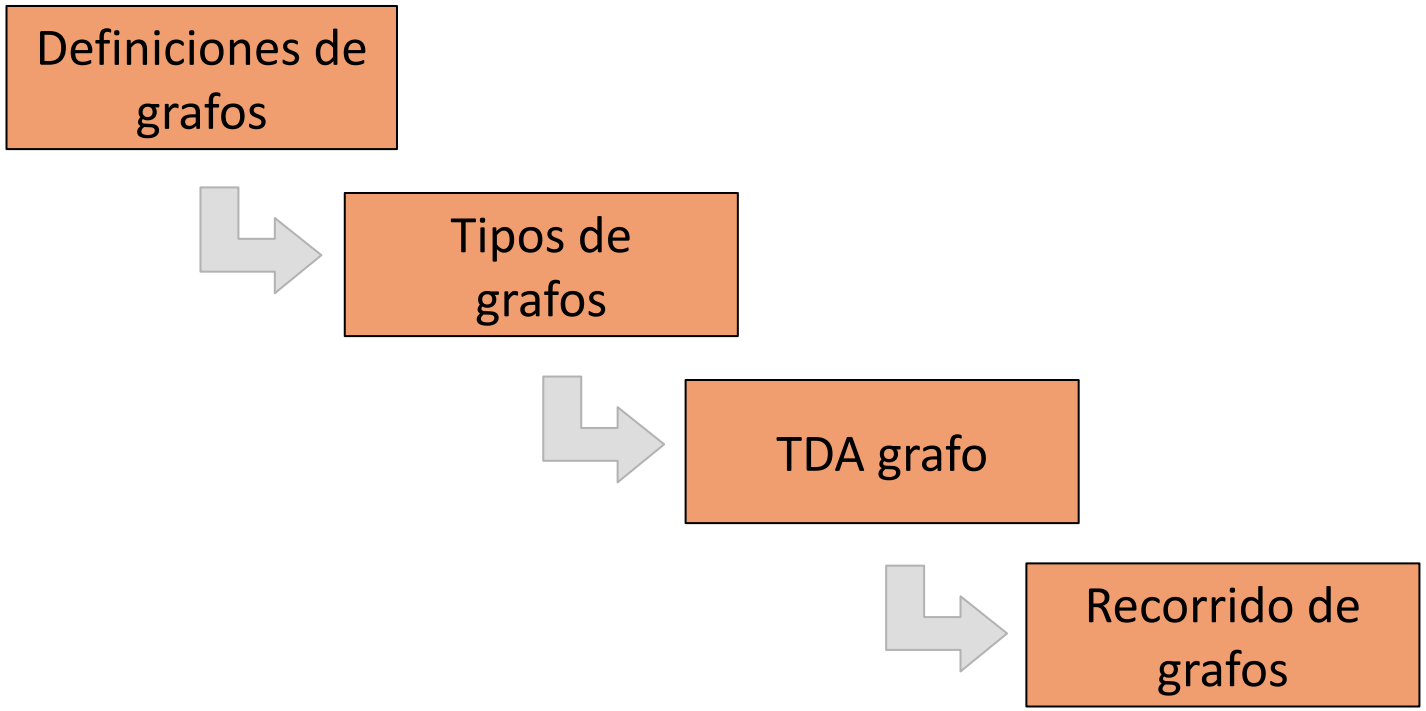
Semestre 2 – 2023

DEPARTAMENTO DE
INGENIERÍA
INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

TDA grafo

- **Contenidos**:
  - Terminología de grafos
  - Representación de estructura de datos grafo
  - Definición formal de TDA grafo
  - Algoritmos de recorrido en grafos
- **Objetivos**:
  - Comprender terminología relacionada a grafos
  - Comprender definición formal de TDA grafo
  - Conocer formas de representación de estructura de datos de TDA grafos
  - Entender funcionamiento de algoritmo de recorrido en grafos

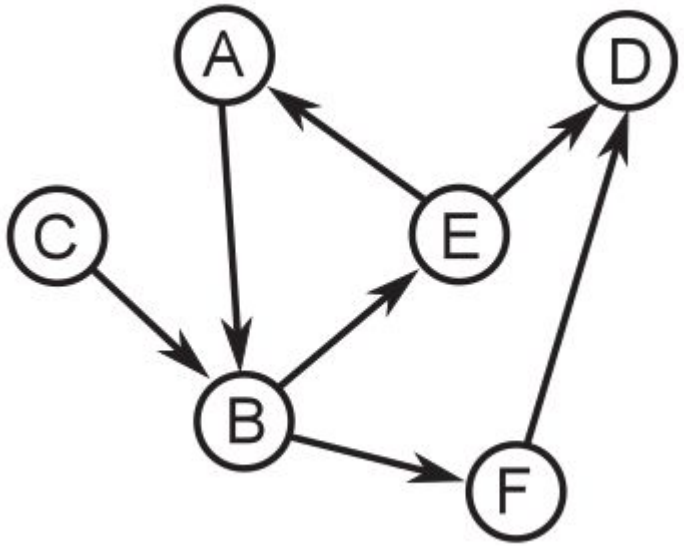
Ruta de la sesión



Definiciones

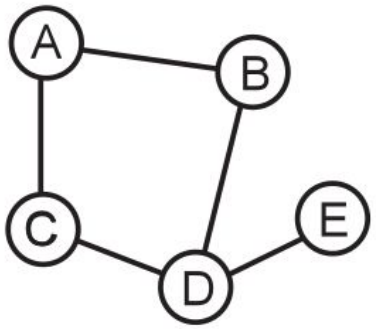
Terminología de grafos

- Un grafo G se define con:
  - **Conjunto de vértices** V={A,B,C,D,E,F}
  - **Conjunto de aristas** A={(A,B),(B,E),(B,F),(C,B),(E,D),(F,D)}

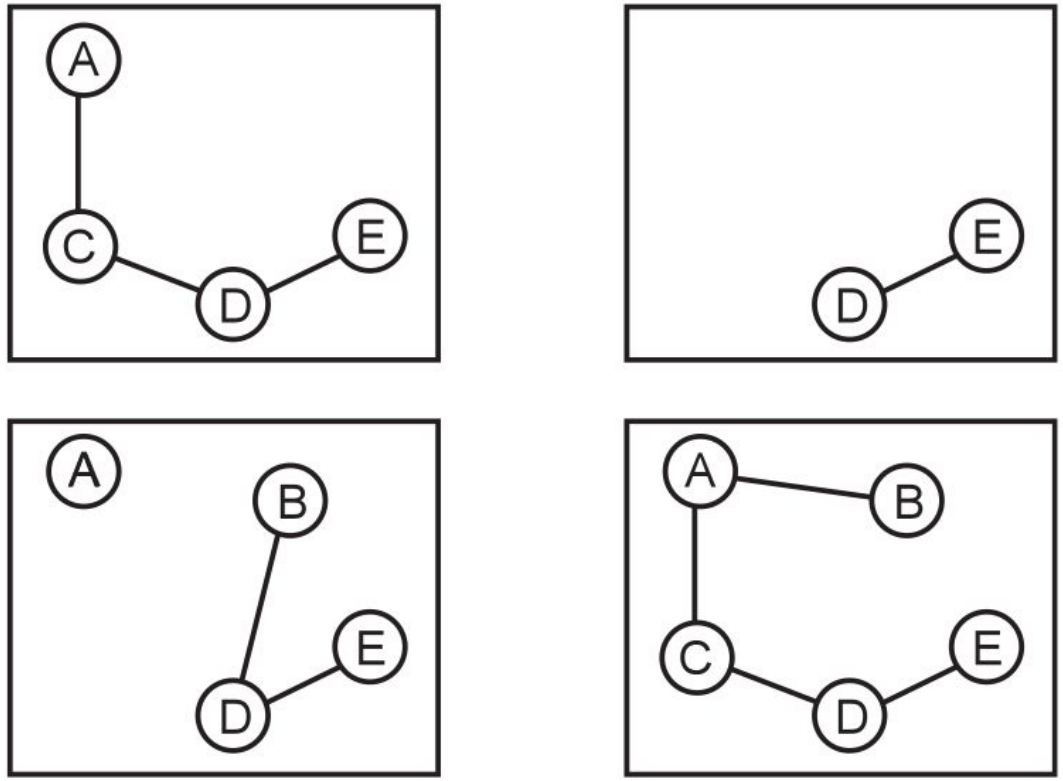


Terminología de grafos

- Subgrafos



Grafo G



Algunos subgrafos de G

¿Qué problemas se pueden resolver con grafos?

- Problemas de transporte y conectividad
- Problemas de administración
- Problemas de planificación logística
- Problemas de la vida cotidiana

Grafo: Definición Formal

- Grafo G=(V, A, W)

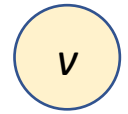
- V Conjunto de vértices o nodos
- V={v_1, v_2, … , v_n} , $|V|=n$ número de vértices

- A, conjunto de arcos o aristas
- A={a_{ij}, …, a_{kl}}, $|A|=m$ número de aristas, $a_{ij}=(v_i, v_j)$

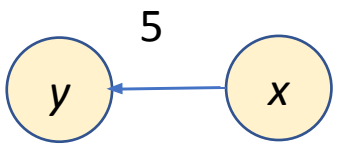
- W, conjunto de pesos
- W={w_{ij},…, w_{kl}}

Definiciones

- **Vértice/Nodo:** Objetos o elementos que forman un grafo.

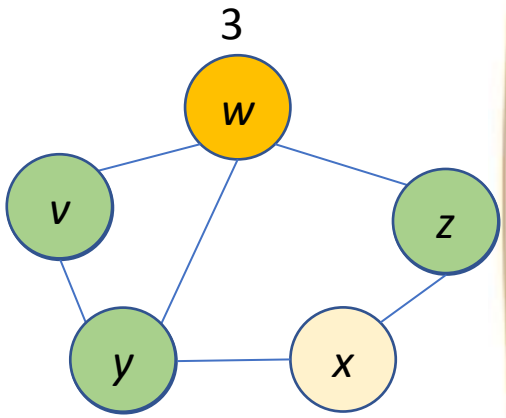


- **Arista/Arco:** Relación entre dos vértices o nodos de un grafo. Se representan como líneas (simples o flechas) que unen los vértices. Pueden tener asociado un peso o valor.

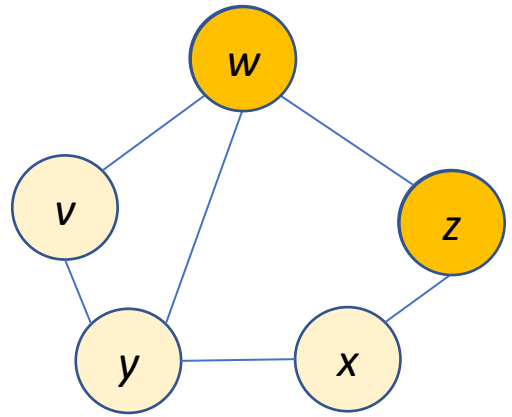


Definiciones

- **Grado:** Cantidad de aristas que confluyen en el vértice. Cantidad de vecinos de un vértice.

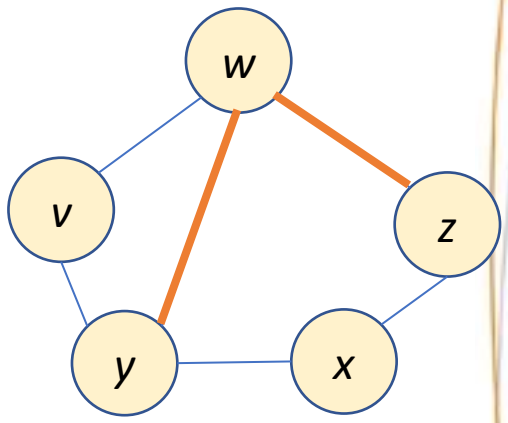


- **Vértices adyacentes:** Vértices unidos mediante una arista. Dada una arista que conecta dos vértices u y v, decimos que u es adyacente a v (y que v es adyacente a u). También se les nombra como vecinos.

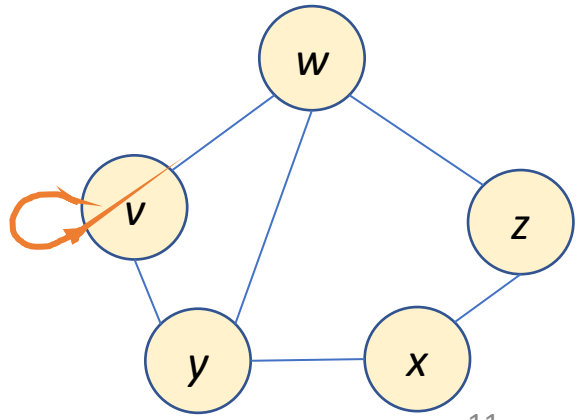


Definiciones

- **Aristas adyacentes:** Dos aristas que convergen en el mismo vértice.



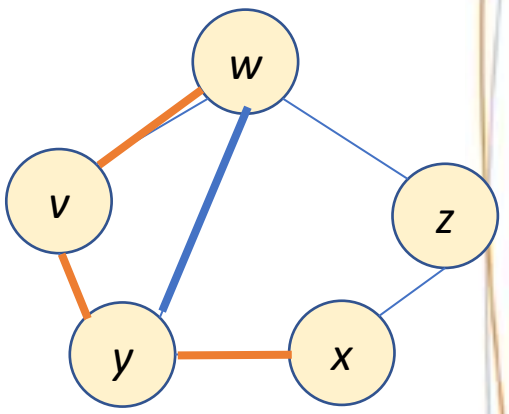
- **Arista cíclica/Bucle/lazo:** Arista que parte de un vértice para llegar al mismo.



Definiciones

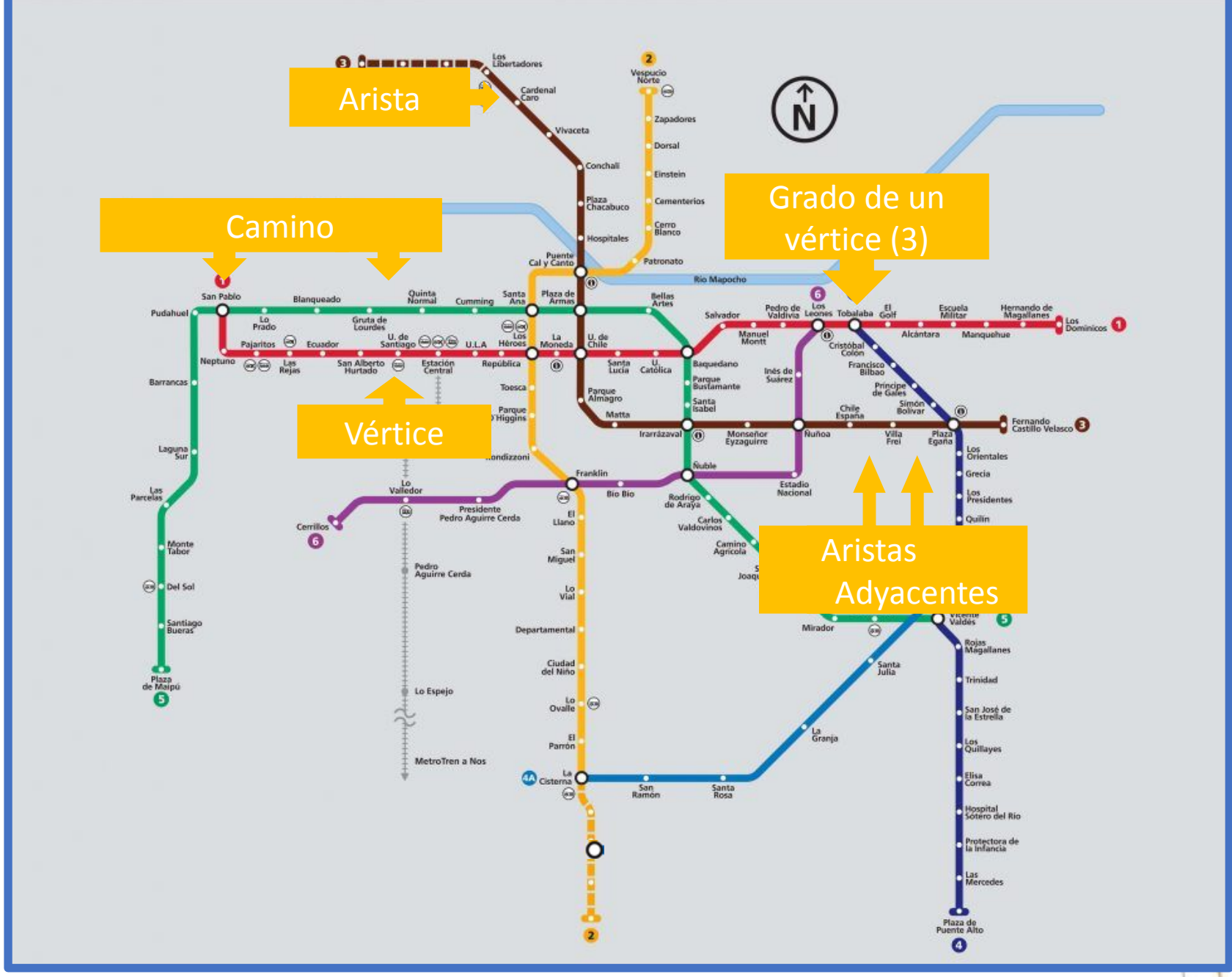
- **Camino:** Conjunto de vértices interconectados por aristas. Sucesión finita de vértices tal que de cada uno de sus vértices existe una arista hacia el vértice sucesor. Dos vértices están conectados si hay un camino entre ellos.

- **Longitud :** Número de arcos/aristas que forman el camino.



Longitud: 3

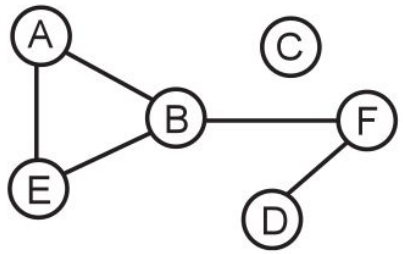
- Vértice
- Grado de un vértice
- Vértice Adyacente
- Arista
- Aristas adyacentes
- Camino
- Longitud de Camino
- Camino Simple



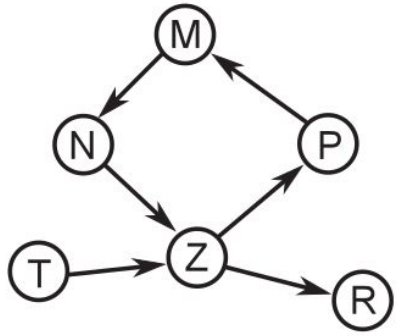
Tipos de grafos

Tipos de grafos

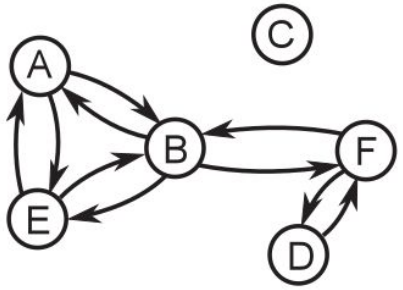
- Ponderado/dirigido



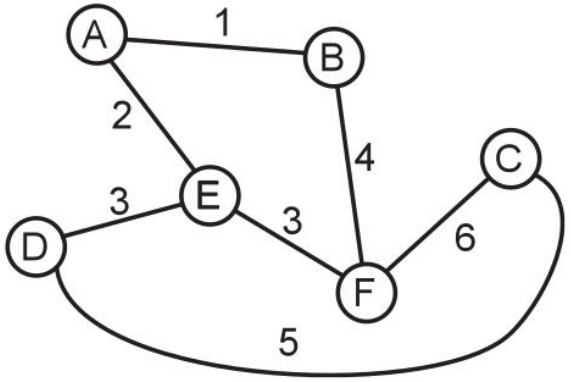
Grafo no dirigido G1



Grafo dirigido G2



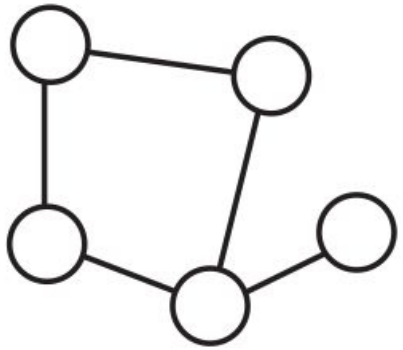
Versión dirigida de grafo G1



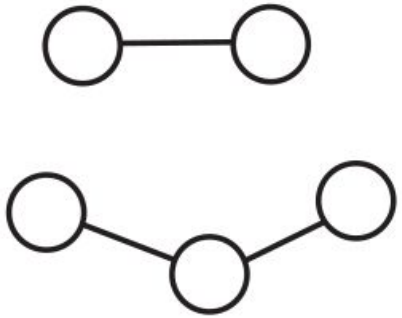
Grafo no dirigido con pesos en las aristas

Tipos de grafos

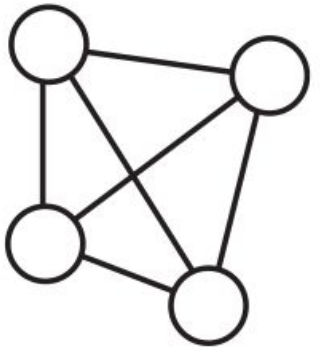
- Conexión en grafos



Grafo conexo



Grafo desconexo

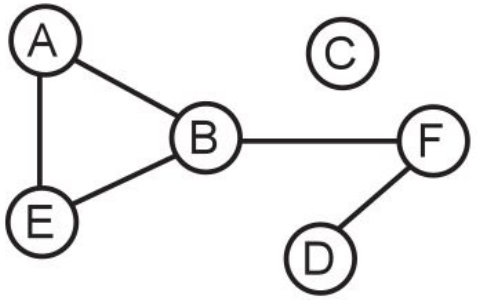


Grafo completo

Representación de grafos

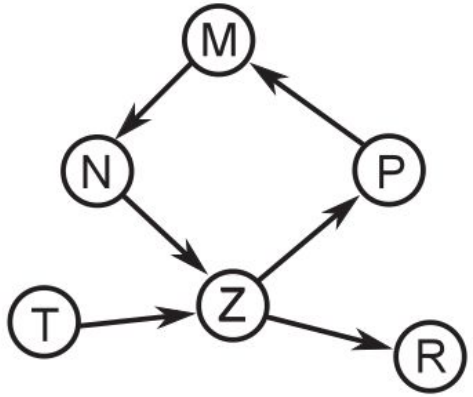
Representación de grafos

- Matriz de adyacencia



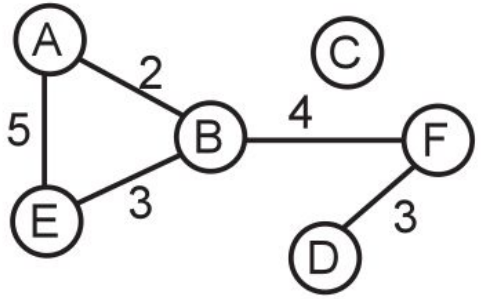
	A	B	C	D	E	F
A	0	1	0	0	1	0
B	1	0	0	0	1	1
C	0	0	0	0	0	0
D	0	0	0	0	0	1
E	1	1	0	0	0	0
F	0	1	0	1	0	0

Matriz de adyacencia de un grafo no dirigido



	M	N	P	R	T	Z
M	0	1	0	0	0	0
N	0	0	0	0	0	1
P	1	0	0	0	0	0
R	0	0	0	0	0	0
T	0	0	0	0	0	1
Z	0	0	1	1	0	0

Matriz de adyacencia de un grafo dirigido

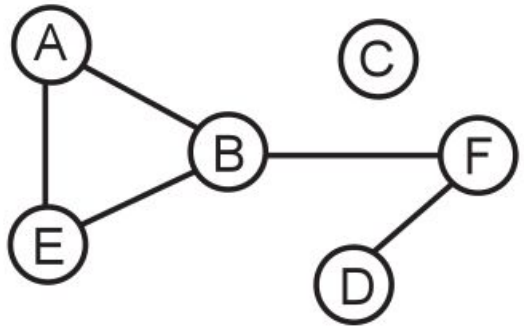


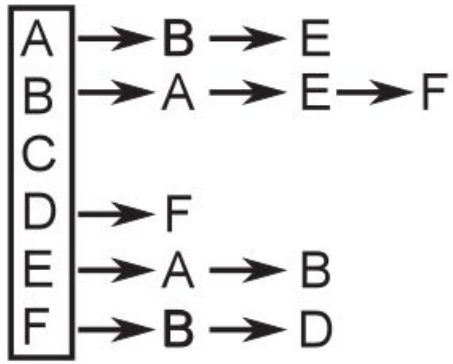
	A	B	C	D	E	F
A	0	2	0	0	5	0
B	2	0	0	0	3	4
C	0	0	0	0	0	0
D	0	0	0	0	0	3
E	5	3	0	0	0	0
F	0	4	0	3	0	0

Matriz de adyacencia de un grafo no dirigido con pesos

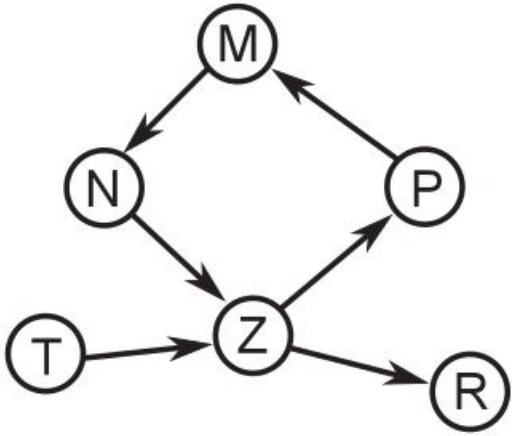
Representación de grafos

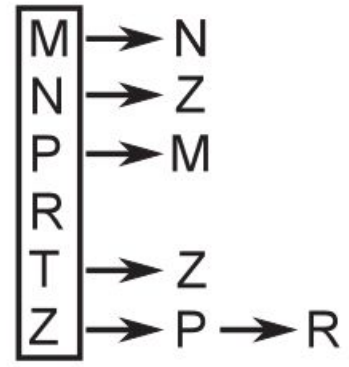
- Lista de adyacencia



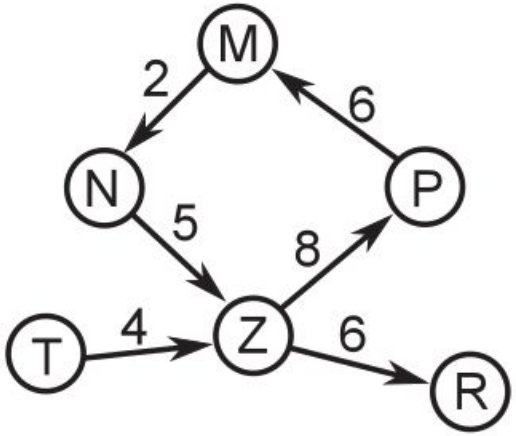


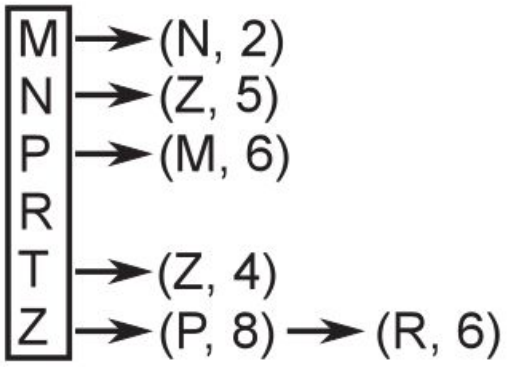
Lista de adyacencia de un grafo no dirigido





Lista de adyacencia de un grafo dirigido

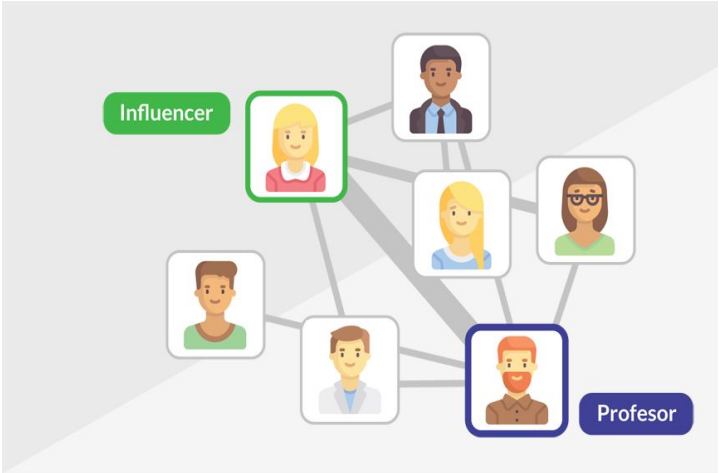




Lista de adyacencia de un grafo dirigido con pesos

Modelamiento de problemas con grafos

¿Cómo utilizar grafos para modelar un problema?



- Identificar los objetos.
- Identificar cómo se relacionan los objetos
- Modelar el grafo
- Transformar el problema en una pregunta puntual sobre el grafo resultante.
- Por ejemplo:
  - Encontrar el camino mínimo entre dos nodos
  - Determinar la cantidad de nodos en una componente conexa desde algún nodo en particular
  - Saber si dos nodos están conectados
  - Verificar si el grafo es de cierto tipo

TDA grafo

Definición de TDA grafo

- **Estructura de datos**
  - Un grafo es una colección no lineal de elementos homogéneos que se entiende como un conjunto de vértices y un conjunto de aristas que conectan dichos vértices. Las aristas pueden tener un peso asociado.
  - La estructura de datos que representa un grafo consiste de una de las siguientes alternativas:
    - Número de vértices + listas de adyacencia
    - Número de vértices + matriz de adyacencia

Definición de TDA grafo

- **Operaciones**
  - ***crearGrafo(V,A)***
  - ***agregarVertice(v)***
  - ***agregarArista(v1,v2)***
  - ***eliminarVertice(v)***
  - ***eliminarArista(v1,v2)***
  - **obtenerAdyacentes(v)**
  - **recorrerGrafo(g)**
  - **obtenerCaminoMinimo(g,v1,v2)**
  - **obtenerArbolCoberturaMinimo(g)**
  - **calcularMaximoFlujo(g)**

Operaciones de TDA grafo

- **ObtenerAdyacentes(grafo G, num v): lista**

  Devuelve una lista simplemente enlazada con todos los vértices adyacentes a v

Operaciones de TDA grafo

- **ObtenerAdyacentes(grafo G, num v): lista**

usando matriz de adyacencia

Devuelve una lista simplemente enlazada con todos los vértices adyacentes a v

```
obtenerAdyacentesVertice(grafo G,num v): lista
    adyacentes←crearListaVacia()
    n←calcularLargoArreglo(G→V)
    para i←1 hasta n
        si G→A[v,i]>0 entonces
            insertarNodoInicio(adyacentes,G→V[i])
    devolver(adyacentes)
```

Operaciones de TDA grafo

- **ObtenerAdyacentes(grafo G, num v): lista**

usando matriz de adyacencia

Devuelve una lista simplemente enlazada con todos los vértices adyacentes a v

```
obtenerAdyacentesVertice(grafo G,num v): lista
    adyacentes←crearListaVacia()
    n←calcularLargoArreglo(G→V)
    para i←1 hasta n
        si G→A[v,i]>0 entonces
            insertarNodoInicio(adyacentes,G→V[i])
    devolver(adyacentes)
```

Complejidad: O(n)

Operaciones de TDA grafo

- **ObtenerAdyacentes(grafo G, num v): lista**

usando listas de adyacencia

Devuelve una lista simplemente enlazada con todos los vértices adyacentes a v

```
obtenerAdyacentesVertice(grafo G,num v): lista
    adyacentes←crearListaVacia()
    aux←G→A[v]→puntero
    mientras aux<>NULO hacer
        insertarNodoInicio(adyacentes,aux→dato)
        aux←aux→puntero
    devolver(adyacentes)
```

Operaciones de TDA grafo

- **ObtenerAdyacentes(grafo G, num v): lista**

usando matriz de adyacencia

Devuelve una lista simplemente enlazada con todos los vértices adyacentes a v

```
obtenerAdyacentesVertice(grafo G,num v): lista
    adyacentes←crearListaVacia()
    aux←G→A[v]→puntero
    mientras aux<>NULO hacer
        insertarNodoInicio(adyacentes,aux→dato)
        aux←aux→puntero
    devolver(adyacentes)
```

Complejidad: O(n)

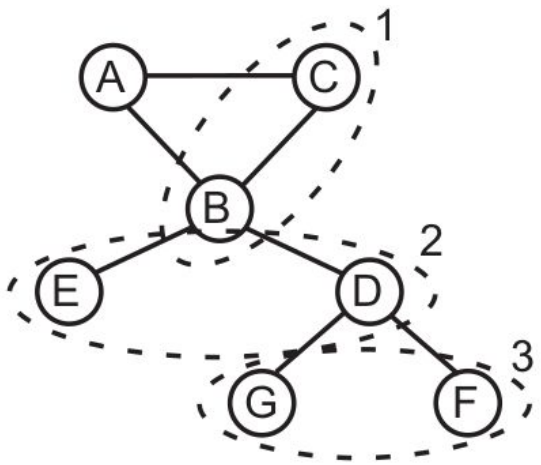
Recorrido de grafos

- Recorrer un grafo consiste en **“visitar” cada uno de los vértices** a través de las aristas del mismo. Se trata de realizar **recorridos de grafos de manera eficiente**.
- Para ello, se pondrá una **marca en un vértice** en el momento en que es visitado, de tal manera que, inicialmente, no está marcado ningún vértice del grafo.
- Para grafos dirigidos y no dirigidos el proceso es análogo, sólo cambia el significado del **concepto de adyacencia**.
- Dos enfoque básicos:
  - Recorrido (o búsqueda) en anchura (breadth-first search, **BFS**): Se visita a todos los vecinos del vértice inicial, luego a los vecinos de los vecinos, etc.
  - Recorrido (o búsqueda) en profundidad (depth-first search, **DFS**): La idea es visitar el primer vecino y visitar su vecino avanzando lo más posible desde el vértice inicial (sin repetir vértices), luego devolverse un paso e intentar lo mismo por otro camino.

Recorrido en amplitud (BFS)

- Recorrido en amplitud BFS (Breadth-First Search)
  - Visita todos los vértices posibles desde el punto de partida antes de avanzar
  - En un recorrido en amplitud, se distingue a un vértice en particular (que en un principio puede ser cualquiera) donde comienza el recorrido del grafo.
  - Se comienza visitando el vértice inicial, luego se visita a todos los vecinos del vértice inicial (que son los vértices que están a distancia 1 de él)
  - Después se visita a los vecinos de estos vecinos (que son los vértices que están a distancia 2 del vértice inicial), y así sucesivamente
  - De alguna forma, se va expandiendo ordenadamente en todas las direcciones a la vez desde el vértice de origen.

Recorrido en amplitud (BFS)



	A	B	C	D	E	F	G
A	0	1	1	0	0	0	0
B	1	0	1	1	1	0	0
C	1	1	0	0	0	0	0
D	0	1	0	0	0	1	1
E	0	1	0	0	0	0	0
F	0	0	0	1	0	0	0
G	0	0	0	1	0	0	0

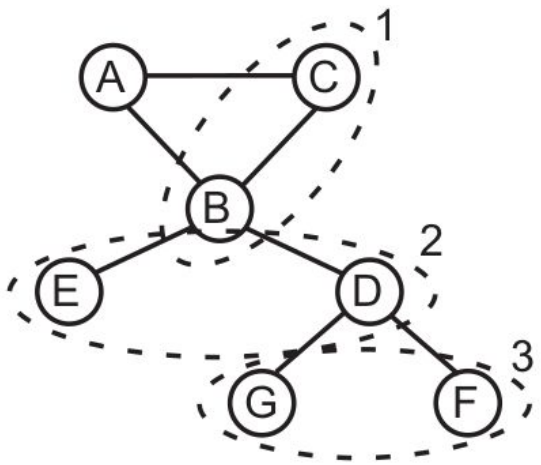
Orden de visita BFS

Frente de la cola

Los vértices agrupados por los óvalos punteados representan su distancia desde el vértice inicial

A	→ B → C
B	→ A → C → D → E
C	→ A → B
D	→ B → F → G
E	→ B
F	→ D
G	→ D

Recorrido en amplitud (BFS)



Los vértices agrupados por los óvalos punteados representan su distancia desde el vértice inicial

	A	B	C	D	E	F	G
A	0	1	1	0	0	0	0
B	1	0	1	1	1	0	0
C	1	1	0	0	0	0	0
D	0	1	0	0	0	1	1
E	0	1	0	0	0	0	0
F	0	0	0	1	0	0	0
G	0	0	0	1	0	0	0

A	→ B → C
B	→ A → C → D → E
C	→ A → B
D	→ B → F → G
E	→ B
F	→ D
G	→ D

Orden de visita BFS

Frente de la cola

	–
	A
A	B C
B	C D E
C	D E
D	E F G
E	F G
F	G
G	–

Recorrido en amplitud (BFS)

```
BFS(grafo g, num inicio)
    cola←crearColaVacia(calcularLargo(G→V))
    encolar(cola,inicio)
    mientras no(esColaVacia(cola)) hacer
        frenteCola←frente(cola)→dato
        descolar(cola)
        marcarVisitado(frenteCola)
        adyacentes←obtenerAdyacentes(G,frenteCola)
        mientras adyacentes<>NULO hacer
            w←adyacentes→dato
            si no(estaVisitado(w)) entonces
                encolar(cola,w)
            adyacentes←adyacentes→puntero
```

Recorrido en amplitud (BFS)

```
BFS(grafo g, num inicio)
    cola←crearColaVacia(calcularLargo(G→V))
    encolar(cola,inicio)
    mientras no(esColaVacia(cola)) hacer
        frenteCola←frente(cola)→dato
        descolar(cola)
        marcarVisitado(frenteCola)
        adyacentes←obtenerAdyacentes(G,frenteCola)
        mientras adyacentes<>NULO hacer
            w←adyacentes→dato
            si no(estaVisitado(w)) entonces
                encolar(cola,w)
            adyacentes←adyacentes→puntero
```

Recorrido en amplitud (BFS)

```
BFS(grafo g, num inicio)
    cola←crearColaVacia(calcularLargo(G→V))
    encolar(cola,inicio)
    mientras no(esColaVacia(cola)) hacer
        frenteCola←frente(cola)→dato
        descolar(cola)
        marcarVisitado(frenteCola)
        adyacentes←obtenerAdyacentes(G,frenteCola)
        mientras adyacentes<>NULO hacer
            w←adyacentes→dato
            si no(estaVisitado(w)) entonces
                encolar(cola,w)
            adyacentes←adyacentes→puntero
```

Complejidad: $O(n^2)$

Actividad de cierre

- Ir a menti.com e ingresar código 3673 8101

Próximas fechas...

- Resumen de la semana:
  - Definiciones y tipos de grafos
  - TDA grafo
  - Modelamiento usando grafos
  - Algoritmo de recorrido BFS

  ~~cátedra~~ – refuerzo – ~~laboratorio~~

- Sub-siguiente semana:
  - Algoritmo de recorrido DFS
  - Algoritmo de camino mínimo Dijkstra
  - Algoritmo de flujo máximo Ford-Fulkerson

U3 - S8