DEPARTAMENTO DE
INGENIERÍA
INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

Estructuras de Datos

Clase Coordinada - Unidad nro. III: Grafos

03-05-2022

Contenido

- Grafos: terminología, definiciones y propiedades
- Representación de grafos
- Tareas

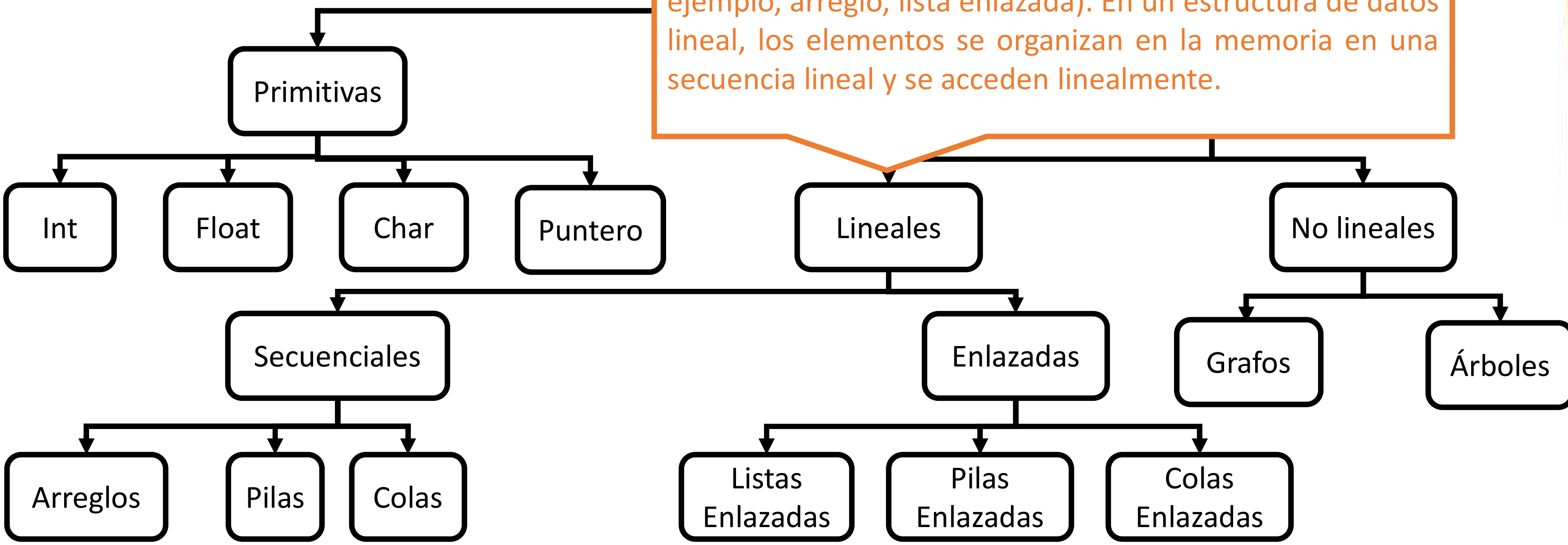
Hasta ahora…

- Programa = Algoritmo (abstracción de procesos) +
  Estructura de Datos (abstracción de datos)

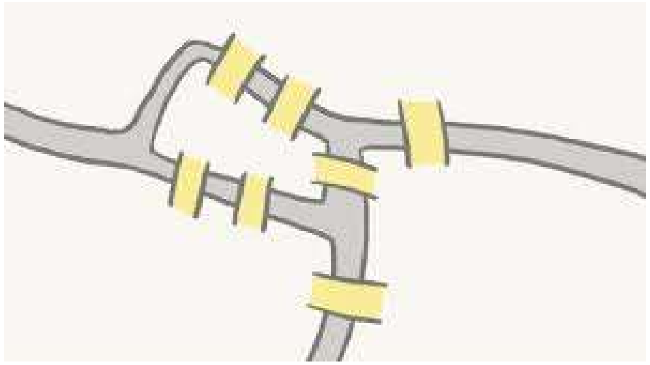
- Las estructuras de datos se puede clasificar en dos categorías:
  - Estructura de datos primitiva: integer, floating point, characters, pointer, boolean, etc.
  - Estructura de datos no primitiva: arrays, structure, stack, queues, linked lists etc.

Clasificación ED

La estructura de datos es lineal si cada elemento está relacionado con los elementos anteriores y siguientes (por ejemplo, arreglo, lista enlazada). En un estructura de datos lineal, los elementos se organizan en la memoria en una secuencia lineal y se acceden linealmente.

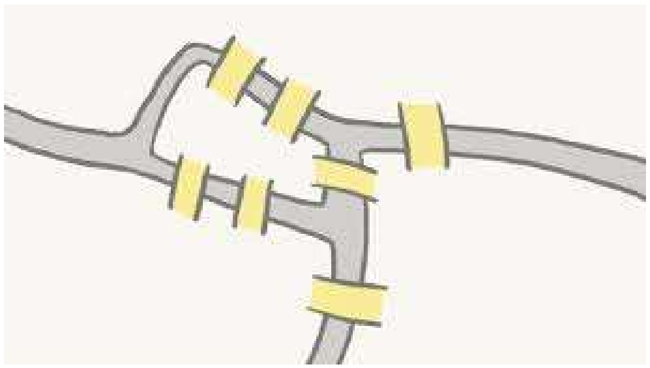


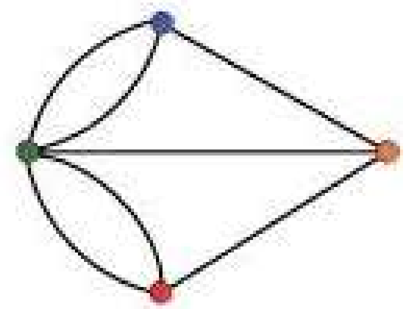
Problema



- ¿Es posible caminar por toda la ciudad cruzando cada uno de los siete puentes exactamente una vez y regresar al mismo punto de partida?

Problema





- ¿Es posible caminar por toda la ciudad cruzando cada uno de los siete puentes exactamente una vez y regresar al mismo punto de partida?
- Respuesta: NO

¿Qué es un grafo?

- Grafo es una estructura de datos abstracta que se utiliza para implementar el concepto matemático de grafos.

- En matemáticas y ciencias de la computación, un grafo (del griego grafos: dibujo, imagen) es un conjunto de objetos llamados vértices nodos unidos por enlaces llamados aristas o arcos, que permiten representar relaciones binarias entre elementos de un conjunto.

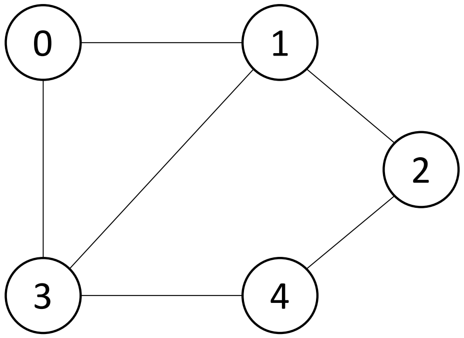
- Formalmente se define como la tupla $G = (V, E)$, donde V son los vértices (o nodos) y E son las aristas (o arcos) del grafo.

Definición

- Un grafo $G = (V, E)$ consiste en un conjunto finito no vacío de objetos V, donde $V(G) = \{v_1, v_2, v_3, \dots, v_n\}$ se denominan vértices y otro conjunto E, donde $E(G) = \{e_1, e_2, e_3, \dots, e_m\}$ cuyos elementos se denominan aristas.

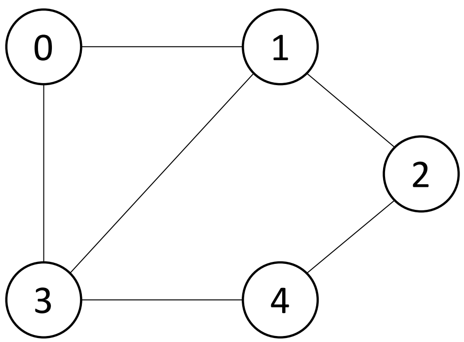
Un grafo

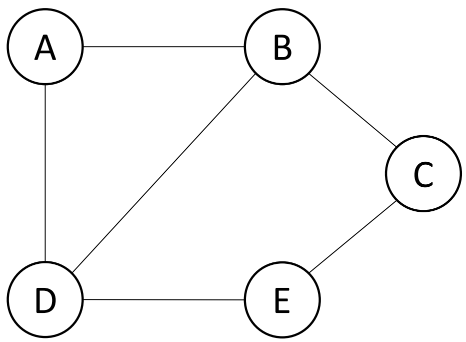
- Cinco vértices $V = \{0, 1, 2, 3, 4\}$
- Seis aristas $E = \{(0, 1), (0, 3), (1, 2), (1, 3), (2, 4), (3, 4)\}$



Un grafo

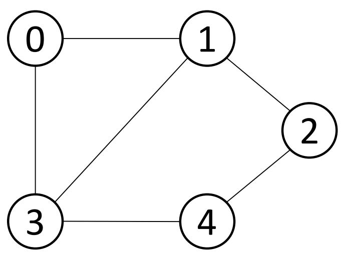
- Es posible dibujar un grafo marcando círculos para los vértices y líneas que los conectan para las aristas. Un dibujo da una intuición sobre la estructura del grafo.
- El grafo se define independiente de la representación.



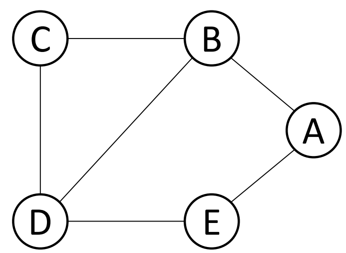


Un grafo

- Una lista de aristas también representa un grafo, dado que un grafo es solo su conjunto (desordenado) de vértices y su conjunto (desordenado) de aristas (pares de vértices).



0 – 1
0 – 3
1 – 2
1 – 3
2 – 4
3 – 4

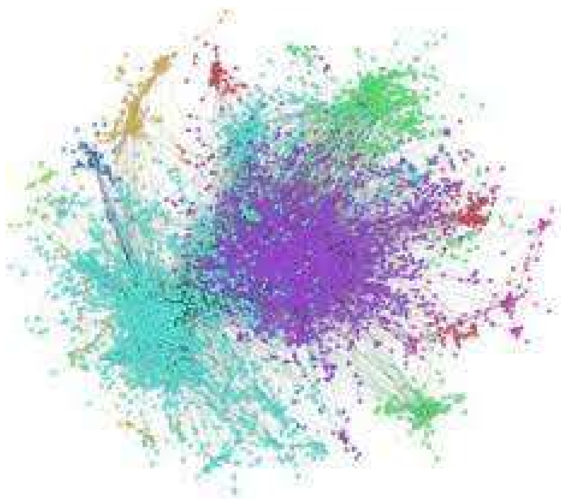


A – B
A – E
B – C
B – D
C – D
D – E

DEPARTAMENTO DE INGENIERÍA INFORMÁTICA UNIVERSIDAD DE SANTIAGO DE CHILE

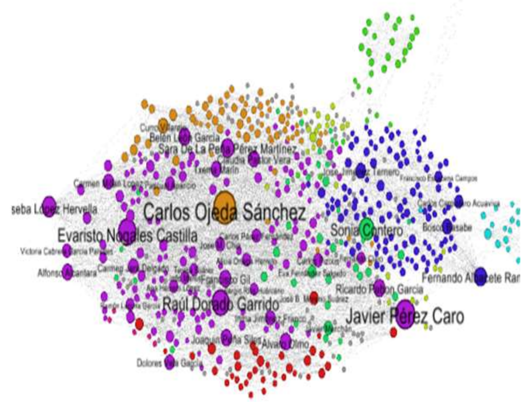
¿Para qué un grafo?

- Se utilizan para modelar redes complejas, por ejemplo, redes de información, redes de transporte, redes biológicas, etc. y una variedad de otros sistemas, donde la relación entre los objetos en el sistema juegan un papel clave.



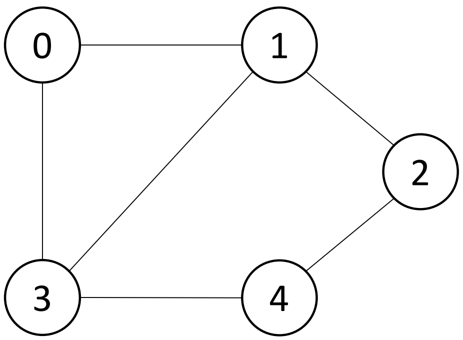
¿Para qué un grafo?

- Los grafos tienen aplicaciones en diversos problemas. Algunos de ellos son:
  - Redes sociales
  - Redes eléctricas
  - Caminos en grillas
  - Optimización de caminos
  - Rutas de vuelo
  - Procesamiento de lenguaje natural



Tipos de grafos

- Grafo no dirigido: en un grafo no dirigido, las aristas no tienen una dirección asociada. Es decir, si hay una arista entre los vértices 1 y 2, entonces los vértices pueden se recorridos tanto de 1 hacia 2, así como de 2 hacia 1. Las aristas representan relaciones simétricas.

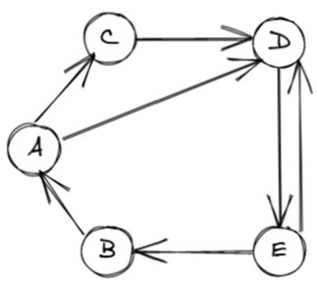


$$\boldsymbol{V} = \{0, 1, 2, 3, 4\}$$

$$\boldsymbol{E} = \{(0,1), (1,2), (0,3), (1,3), (3,4), (2,4)\}$$

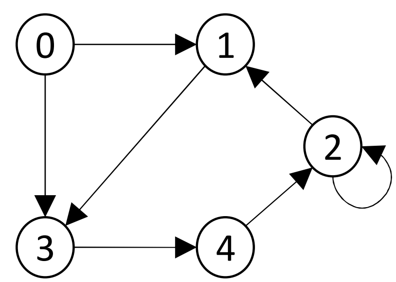
Tipos de grafos

- Grafo dirigido (digrafo): un grafo dirigido $\vec{G}(\mathrm{V},\vec{E})$, consta de un conjunto de vértices V, un conjunto $\vec{E}$ de arcos que son pares ordenados de V. Se usa una flecha apuntando de *u* a *v* para indicar la dirección del arco (u,v). Si existe una arista de *u* a v, entonces existe un camino de *u* a v pero no necesariamente de v a *u*. La arista (u,v) comienza en el vértice *u* y termina en el vértice v.



$\boldsymbol{V}$ = {A, B, C, D, E}

$\vec{\boldsymbol{E}}$ = {(B,A),(A,C),(A,D),(C,D),(E,B),(D,E),(E,D) }



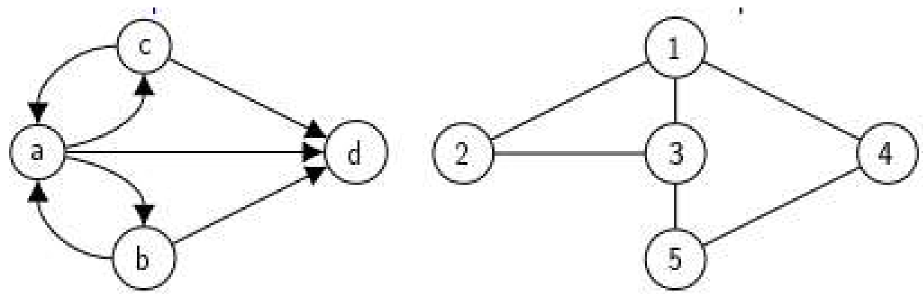
$\boldsymbol{V}$ = {0, 1, 2, 3, 4}

$\vec{\boldsymbol{E}}$ = {(0,1),(0,3),(2,1),(1,3),(2,2),(4,2),(3,4) }

DEPARTAMENTO DE
INGENIERÍA
INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

Tipos de grafos

- Grafo simple: un grafo simple $G = (V, E)$ es un grafo en el cual cada arista conecta dos diferentes vértices y donde dos aristas no conectan el mismo par de vértices. No tiene ni aristas paralelas ni bucles.

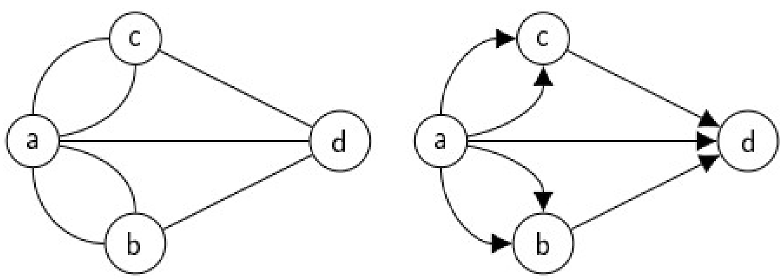


Propiedad: Un grafo simple con n vértices tiene como máximo ***n * (n – 1) / 2*** aristas.

DEPARTAMENTO DE INGENIERÍA INFORMÁTICA UNIVERSIDAD DE SANTIAGO DE CHILE

Tipos de grafos

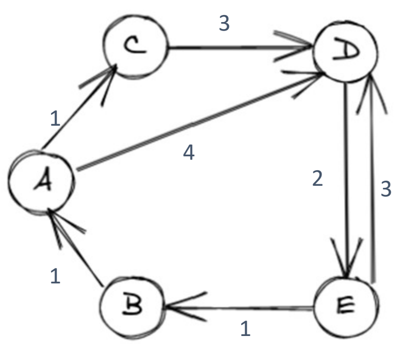
- Multigrafo: un multigrafo $G = (V, E)$ consta de un conjunto de vértices V, un conjunto E de aristas y una función f de E en { {u, v} / u, v ∈ V, u ≠ v}. Se dice quelas aristas e_1 y e_2 son aristas múltiples o paralelas si $f(e_1) = f(e_2)$.

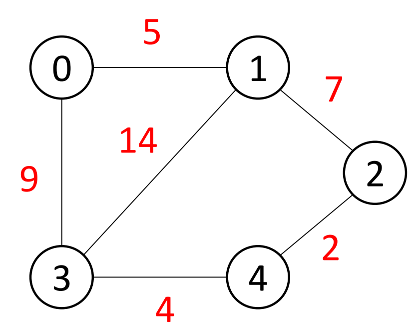


- Pseudografo: un pseudografo $G = (V, E)$ consta de un conjunto de vértices V, un conjunto E de aristas y una función f de E en { {u, v} / u, v ∈ V}. Una arista e es un bucle o lazo si $f(e)$ = {u, v} = {u} para algún $u \in V$.

Tipos de grafos

- Grafo dirigido/no dirigido ponderado (o valuado): un grafo ponderado se asocia un número (peso) a cada arista, que generalmente representa una distancia o un costo.





$\boldsymbol{V}$ = {A, B, C, D, E}

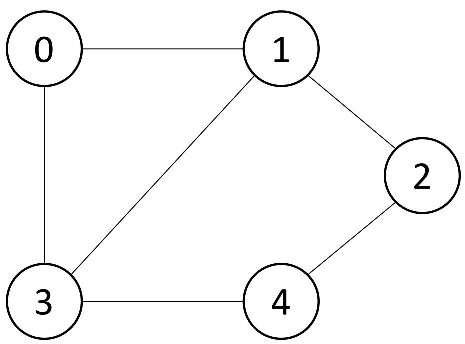
$\vec{\boldsymbol{E}}$={(B,A,1),(A,C,1),(A,D,4),(C,D,3),(E,B,1),(D,E,2),(E,D,3) }

$\boldsymbol{V} = \{0, 1, 2, 3, 4\}$

$\boldsymbol{E} = \{(0, 1,5), (1, 2,7), (0, 3,9), (1, 3,14), (3, 4,4), (2, 4,2)\}$

Terminología de grafos

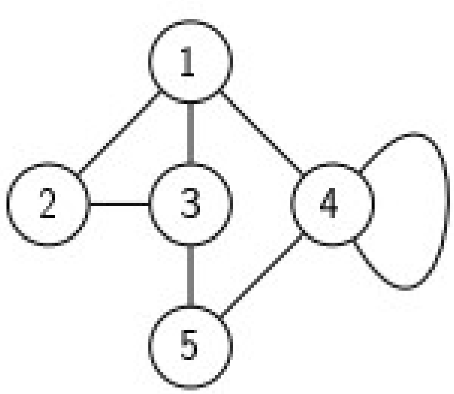
- Vértices adyacentes o vecinos: para cada arista *e = (u, v)*, que conecta los vértices *u* y *v*, los vértices u y v son puntos finales y se dice que son vértices vecinos o adyacentes. También se dice que la arista *(u, v)* incide sobre u y v.



ady(0) = {1, 3}
ady(1) = {0, 2, 3}
ady(2) = {1, 4}
ady(3) = {0, 1, 4}
ady(4) = {2, 3}

Terminología de grafos

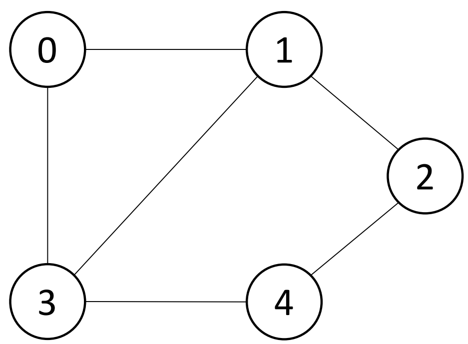
- Aristas o arcos adyacentes: dos aristas son adyacentes si tienen un vértice en común, por ejemplo en la figura (1,3) y (2,3) son aristas adyacentes.



ady(1,2) = {(1,3), (1,4), (2,3)}
ady(1,3) = {(1,2), (1,4), (2,3)}
ady(1,4) = {(1,2), (1,3), (4,4), (4,5)}
ady(2,3) = {(1,2), (1,3), (3,5)}
ady(3,5) = {(2,3), (1,3), (4,5)}
ady(4,4) = {(1,4), (4,5)}
ady(4,5) = {(1,4), (4,4), (3,5)}

Terminología de grafos

- **Grado:** el grado de un vértice es el número de aristas que inciden en el vértice. El grado de un vértice *v* se denota por *deg(v)*. Si grados *deg(v)* = 0, significa que v no tiene vecinos y dicho vértice se conoce como vértice aislado. Por ejemplo *deg(1)* = 3.



Terminología de grafos

- Grado máximo de un gráfico G, denotado por $\Delta(G)$ y grado mínimo de un gráfico, denotado por $\delta(G)$, son el grado máximo y el mínimo de sus vértices.

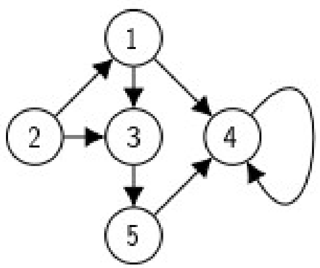
Propiedad (lema del apretón de manos)

Para un grafo no dirigido G = (V, E)

$$\sum_{v \in V} deg(v) = 2|E|$$

Terminología de grafos

- Grado de un vértice para un grafo dirigido: el grado de entrada de un vértice ***v***, denotado por **δ⁻**(v), es el número de aristas que tienen a ***v*** como ***vértice final***. El grado de salida de un vértice ***v***, denotado por **δ⁺**(v), es el número de aristas que tienen a ***v*** como ***vértice inicial***.



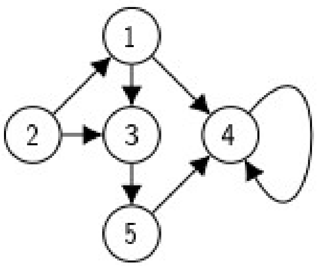
δ −(1) = 1	**δ** +(1) = 2
δ −(2) = 0	**δ** +(2) = 2
δ −(3) = 2	**δ** +(3) = 1
δ −(4) = 3	**δ** +(4) = 1
δ −(5) = 1	**δ** +(5) = 1

Propiedad:

Sea G = (V, E) un grafo dirigido. Entonces: $\sum_{v \in V} \delta^-(v) = \sum_{v \in V} \delta^+(v) = |E|$

Terminología de grafos

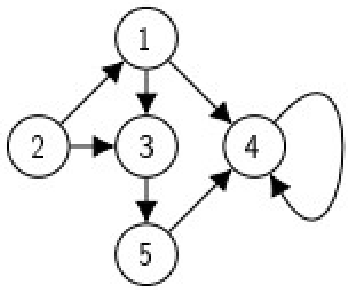
- Antecesor y Sucesor de un vértice: en un grafo dirigido, si *(v,w)* ∈ $\vec{E}$, se dice que v es ***antecesor*** de w, y que w es ***sucesor*** de v. Si un nodo posee un bucle, entonces se cuenta a sí mismo como un antecesor y como un sucesor.



ant(1) = {2}	suc(1) = {3, 4}
ant(2) = ∅	suc(2) = {1, 3}
ant(3) = {1, 2}	suc(3) = {5}
ant(4) = {1, 4, 5}	suc(4) = {4}
ant(5) = {3}	suc(5) = {4}

Terminología de grafos

- Vértice Fuente: nodo *u* de un grafo dirigido que solo posee nodos sucesores y ningún nodo antecesor, es decir: $\delta^-(u) = 0$ y $\delta^+(u) \neq 0$
- Vértice Sumidero: nodo *u* de un grafo dirigido que solo posee nodos antecesores y ningún nodo sucesor, es decir: $\delta^-(u) \neq 0$ y $\delta^+(u) = 0$



Fuente(G) = {2}
Sumidero(G) = {4}

Terminología de grafos

- Densidad de un grafo: sea *G=(V, E)* un grafo simple (sin bucles) con ***n = |V|*** vértices y con ***m = |E|*** aristas, la densidad de un grafo es la relación entre la cantidad máxima de aristas que puede tener un grafo y la cantidad que realmente posee. La densidad está dada por las siguientes ecuaciones para un grafo ***no dirigido*** *y* **dirigido** respectivamente. La densidad de un grafo toma valores entre 0 y 1.

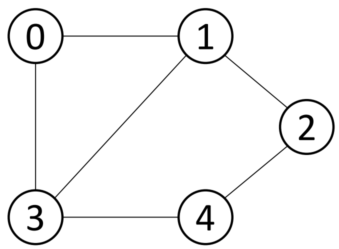
$$d(g) = \frac{2m}{n(n-1)} \qquad d(g) = \frac{m}{n(n-1)}$$

- Se dice que un grafo es denso cuando d(g) es cercano a 1, y se dice que es disperso cuando es cercano a 0.

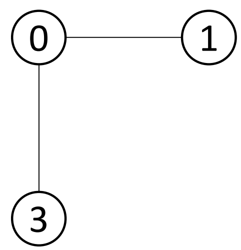
Terminología de grafos

- **Subgrafo:** Si $G = (V, E)$ es un grafo (dirigido o no), $G_1 = (V_1, E_1)$ entonces es un subgrafo de G, si $V_1 \subseteq V$ y $E_1 \subseteq E$ y $E_1 \subseteq V_1^2$

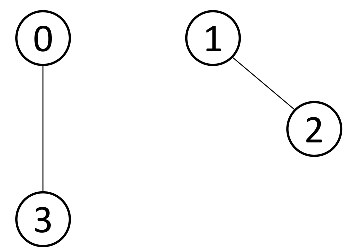
Grafo G



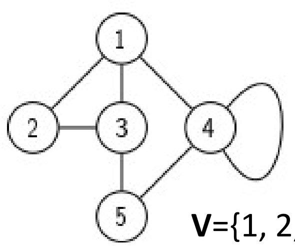
subgrafo de G



subgrafo de G



Grafo G



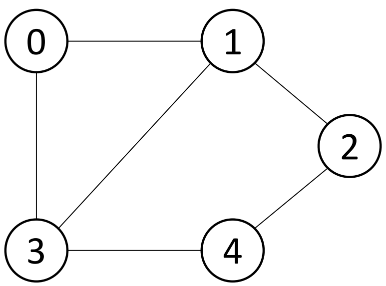
V={1, 2, 3, 4, 5}
E={(1,2), (1,3), (2,3), (1,4), (3,5), (4,4), (4,5)}

subgrafo de G

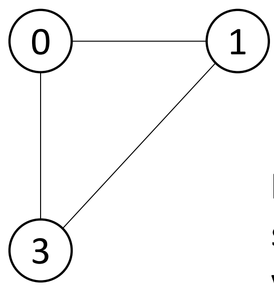
G1(V1, A1):
V1={3, 4, 5}
A1={(3,5), (4,4), (4,5)}

Terminología de grafos

- Subgrafo inducido: Para cualquier subconjunto *W* de vértices de un grafo G=(V,E), se llama subgrafo inducido por W, denotado por <W>, al subgrafo de *G* que se obtiene tomando los vértices de *W* y las aristas de *G* que son incidentes con ellos.



Grafo G



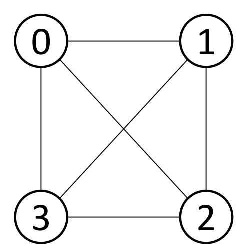
H1

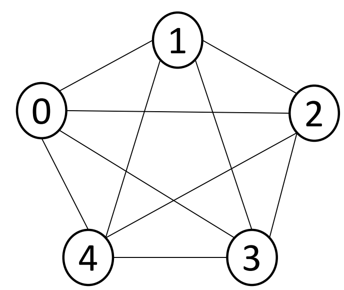
H1 es un subgrafo inducido ya que para W = {0, 1, 3}, el subgrafo ***H1*** contiene todas las aristas de G incidentes con los vértices de W

Terminología de grafos

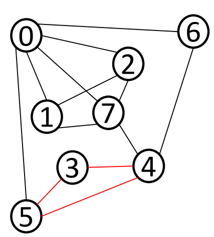
- Grafo completo: El grafo completo de ***n*** vértices, es el grafo simple que contiene exactamente una arista entre cada par de vértices distintos. Cada nodo posee ***n – 1 aristas***. En un grafo completo con n nodos, el número de arcos es:

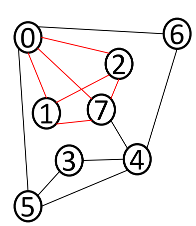
$$m = \frac{n \cdot (n-1)}{2}$$





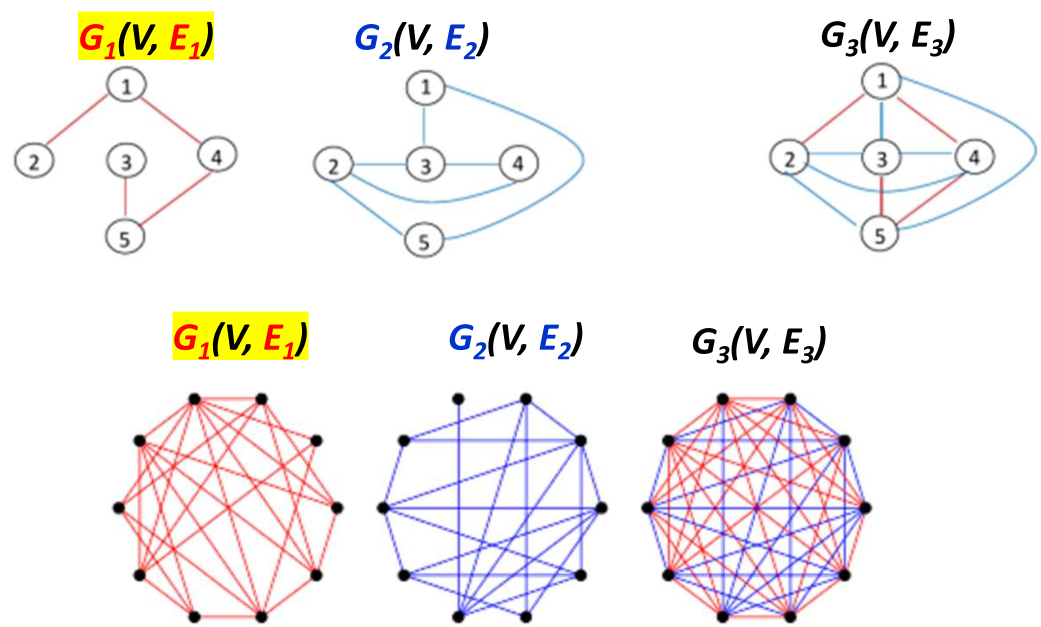
- Un subgrafo completo es una clique. Debe ser de, a lo menos, orden 3.





Terminología de grafos

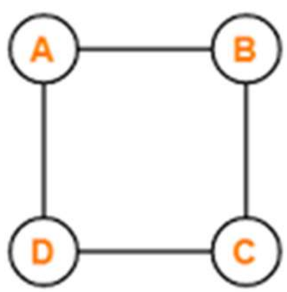
- **Grafo complemento:** Sea G3 un grafo completo, entonces $G_1(V, E_1)$ es el grafo complemento de $G_2(V, A_2)$ si y solo si $G_3(V, E_3)$, es un grafo completo, al considerar: $E_3 = E_1 \cup E_2$ y $E_1 \cap E_2 = \emptyset$.



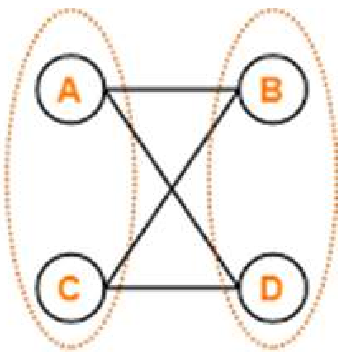
DEPARTAMENTO DE INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

Terminología de grafos

- Grafo bipartito o bigrafo: Se dice que un grafo simple G es bipartito si su conjunto de vértices ***V*** se puede dividir en dos conjuntos disjuntos V_1 y V_2 (es decir, $V_1 \cap V_2 = \emptyset$) tales que cada arista del grafo conecta un vértice V_1 con un vértice de V_2 (de manera que no haya ninguna arista que conecte entre sí dos vértices de V_1 ni tampoco dos vértices de V_2).



Grafo G

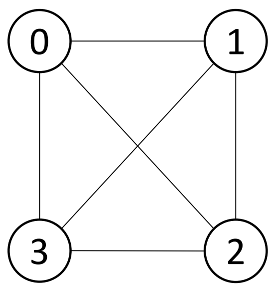


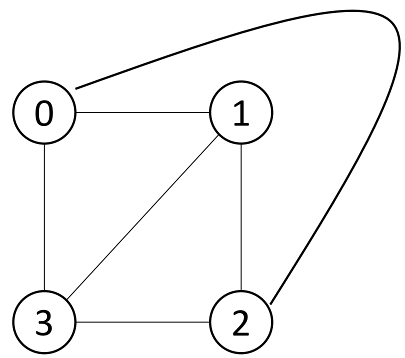
El grafo **G** es bipartito donde $\mathbf{V_1}$ = {A, C} y $\mathbf{V_2}$ = {B, D} entre los vértices de $\mathbf{V_1}$ y $\mathbf{V_2}$ no existen aristas que los comunique.

DEPARTAMENTO DE INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

Terminología de grafos

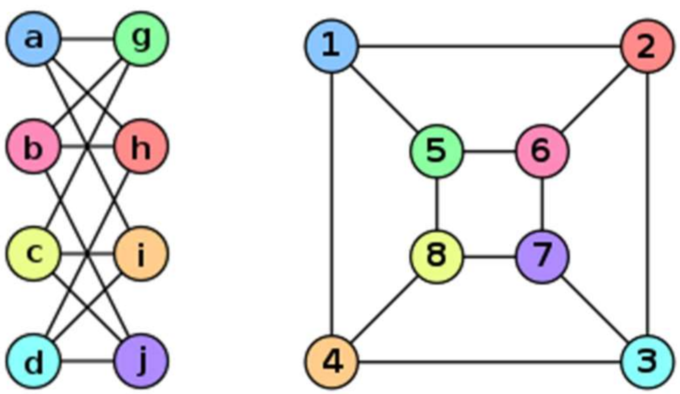
- Grafo planar: es aquel que se puede dibujar en un plano sin cruce de aristas.





Terminología de grafos

- **Isomorfismo de grafos:** los grafos simples $G_1 = (V_1, E_1)$ y $G_2 = (V_2, E_2)$ son isomorfos si hay una función biyectiva ***f*** de V_1 en V_2 con la propiedad de que, para cada par de vértices $u, v \in V_1$, u y v son adyacentes en G_1 si, y solo si, $f(u)$ y $f(v)$ son adyacentes en G_2. Se dice que esta función f es un isomorfismo.



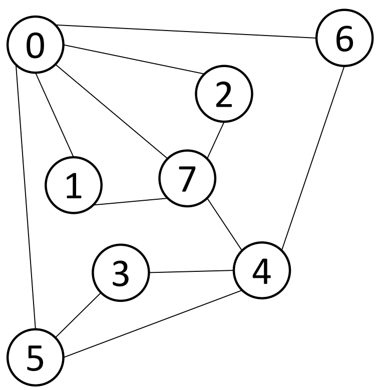
La funcion f con:
f(a) = 1, f(b) = 6, f(c) = 8, f(d) = 3, f(g) = 5, f(h) = 2, f(i) = 4, f(j) = 7

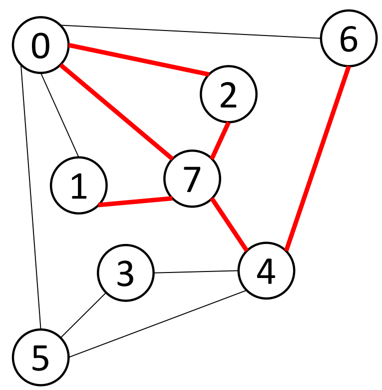
(*)Una función es biyectiva si todos los elementos del conjunto de salida tienen una imagen distinta en el conjunto de llegada, y a cada elemento del conjunto de llegada le corresponde un elemento del conjunto de salida

DEPARTAMENTO DE INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

Terminología de grafos

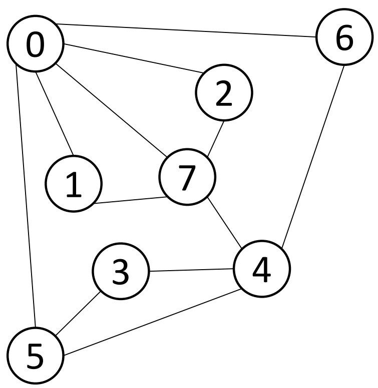
- Camino en un grafo es una secuencia de vértices en la que cada sucesivo vértice (después del primero) es adyacente a su predecesor en el camino.

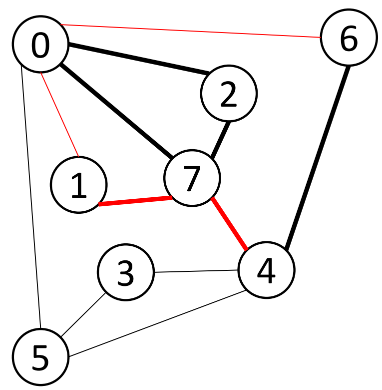




Terminología de grafos

- En un camino simple, los vértices y las aristas son todos distintos.

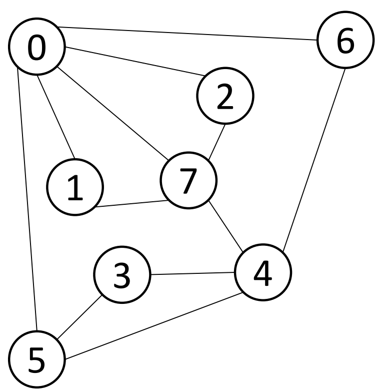


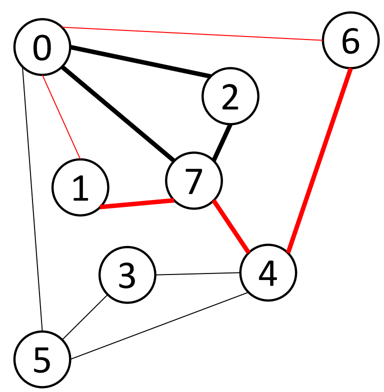


- Largo o longitud de un camino es el número de aristas del camino. Un camino de longitud n debe tener n+1 vértices.

Terminología de grafos

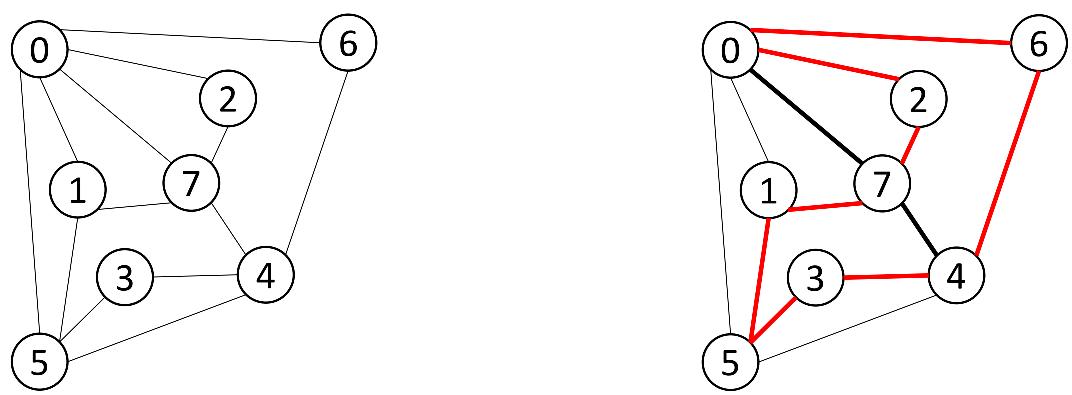
- Un ciclo es un camino simple, siendo el primer y el último vértice el mismo. Cada vértice es un camino de longitud 0.





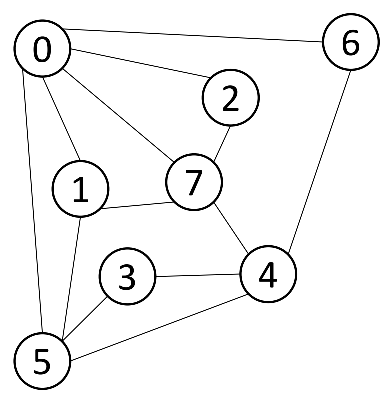
Terminología de grafos

- Un tour es un ciclo que incluye todos los vértices.



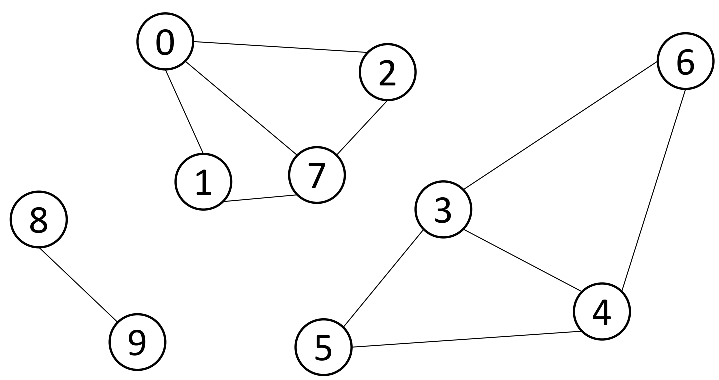
Terminología de grafos

- Un grafo conexo o conectado es un grafo en que todos sus vértices están conectados por un camino (si el grafo es no dirigido).



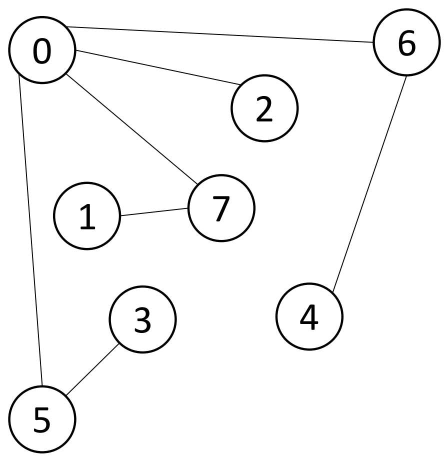
Terminología de grafos

- Un grafo no conexo consta de un conjunto de componentes conexas, que son subgrafos conexos máximos.



Terminología de grafos

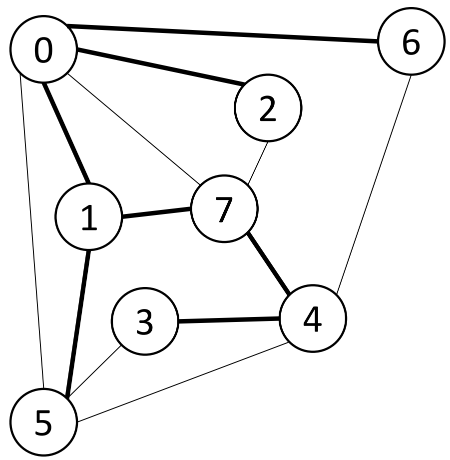
- Un árbol es un grafo conexo acíclico.

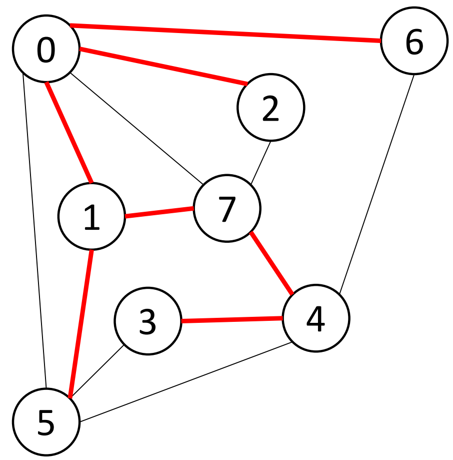


DEPARTAMENTO DE INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

Terminología de grafos

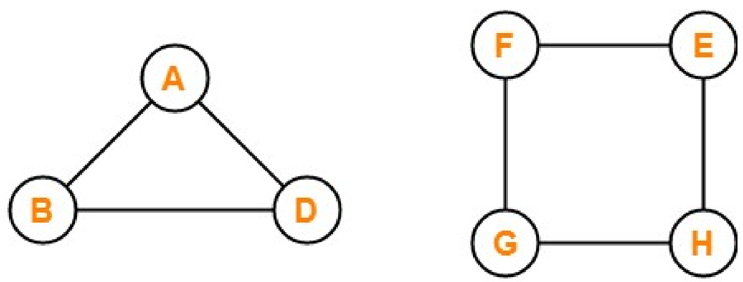
- Un árbol recubridor de un grafo conectado es un subgrafo que contiene todos los vértices del grafo y es un árbol.





Terminología de grafos

- Grafo regular: cada vértice tiene el mismo número de vecinos. Es decir, cada vértice tiene el mismo grado. Un grafo regular con vértices vértice de grado *k* se denomina grafo *k*-regular.



DEPARTAMENTO DE
INGENIERÍA
INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

Representación I

- ¿Cómo representar un grafo?

Representación I

- Una representación de un grafo mediante una matriz de adyacencia es una matriz V por V de valores booleanos, con la entrada en la fila v y la columna w definida como 1 si hay una arista que conecta el vértice v y el vértice w en el gráfico, y 0 en otro caso.

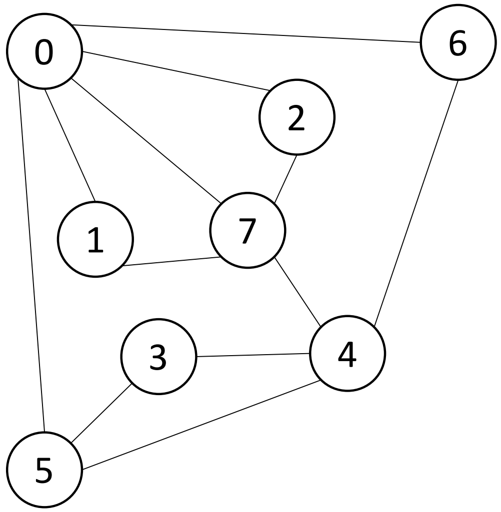
Matriz de adyacencia

- Con una matriz de adyacencia, es posible determinar eficientemente si hay o no una arista desde el vértice i al vértice j, simplemente verificando si la fila i y la columna j de la matriz es diferente de cero.

- Para un grafo no dirigido, si hay una entrada en la fila i y la columna j, entonces también debe haber una entrada en la fila j y la columna i, por lo que la matriz es simétrica.

Matriz de adyacencia

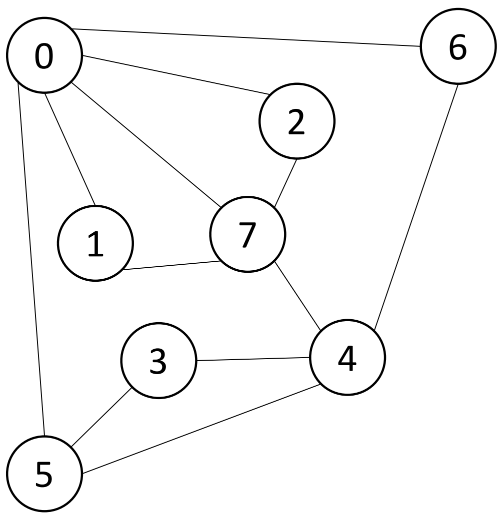
- Matriz de 8 por 8.



	0	1	2	3	4	5	6	7
0	0	1	1	0	0	1	1	1
1	1	0	0	0	0	0	0	1
2	1	0	0	0	0	0	0	1
3	0	0	1	0	1	0	0	0
4	0	0	0	1	0	1	1	0
5	1	0	0	1	1	0	0	0
6	1	0	0	0	1	0	0	0
7	1	1	1	0	1	0	0	0

Matriz de adyacencia

- Grafo arreglo de arreglos



0	0	1	1	0	0	1	1	1
1	1	0	0	0	0	0	0	1
2	1	0	0	0	0	0	0	1
3	0	0	1	0	1	0	0	0
4	0	0	0	1	0	1	1	0
5	1	0	0	1	1	0	0	0
6	1	0	0	0	1	0	0	0
7	1	1	1	0	1	0	0	0

Matriz de adyacencia

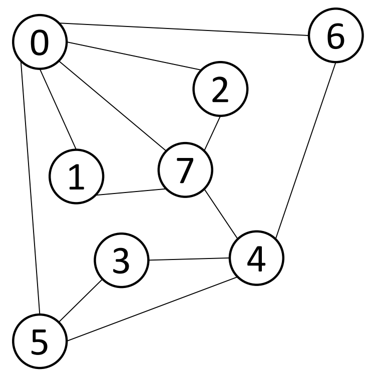
```
int main()
{
      char nombre[200];
      int vertices, aristas, i, j, k;
      printf("Ingrese el nombre del archivo a leer\n");
      scanf("%s", nombre);
      FILE *fp;
      fp = fopen(nombre, "r");
      fscanf(fp, "%d %d", &vertices, &aristas);

      int **adj = (int **) malloc(sizeof(int *) * vertices);
      for(int i = 0;i < vertices; ++i){
                adj[i] = (int*) malloc(sizeof(int) * vertices);
                }

      for (i = 0; i < vertices; i++)
                for (j = 0; j < vertices; j++)
                          adj[i][j] = 0;

      k = 0;
      while (k < aristas){
                fscanf(fp, "%d %d", &i, &j);
                adj[i][j] = 1;
                adj[j][i] = 1;
                k = k + 1;
      }
      return 0;
}
```

Número_vértices Número_aristas
Vértice Vértice
Vértice Vértice
Vértice Vértice
...

DEPARTAMENTO DE INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

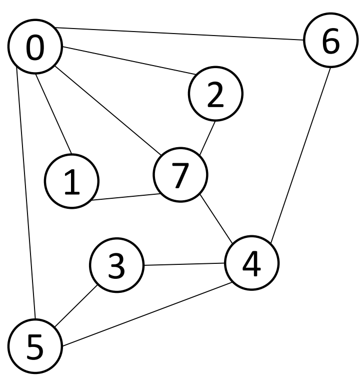
Representación II

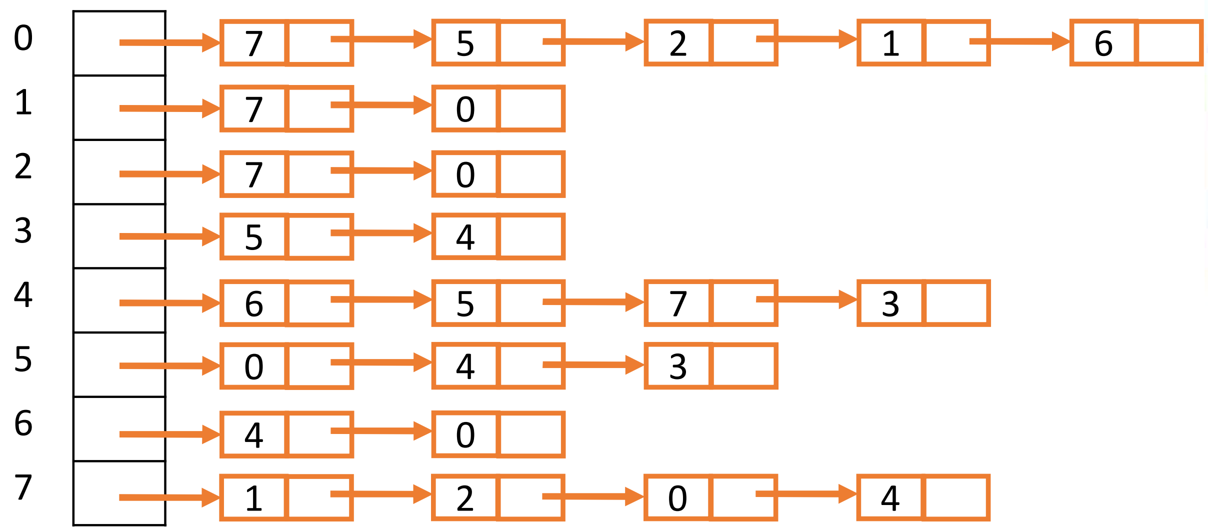
- Es posible representar un grafo mediante una matriz de listas enlazadas, llamadas listas de adyacencia. Se mantiene una lista enlazada para cada vértice, con un nodo para cada vértice conectado a ese vértice. Para un grafo no dirigido, si hay un nodo para j en la lista de i, entonces debe haber un nodo para i en la lista j.



Listas de adyacencia

- Grafo de 8 vértices





Listas de adyacencia

```
typedef struct nodo *enlace;

struct nodo{
     int v;
     enlace siguiente; };

enlace nuevo_nodo(int v, enlace siguiente){
     enlace x = malloc(sizeof *x);
     x->v = v;
     x->siguiente = siguiente;
     return x;
}
```

```
int main(){
     char nombre[200];
     int vertices, aristas, i, j, k;

     printf("Ingrese el nombre del archivo a leer\n");
     scanf("%s", nombre);
     FILE *fp;
     fp = fopen(nombre, "r");
     fscanf(fp, "%d %d", &vertices, &aristas);
     enlace adj[vertices];

     for (i = 0; i < vertices; i++)
               adj[i] = NULL;

     k = 0;
     while (k < aristas){
               fscanf(fp, "%d %d", &i, &j);
               adj[j] = nuevo_nodo(i, adj[j]);
               adj[i] = nuevo_nodo(j, adj[i]);
               k = k + 1;
     }

}
```

Listas de adyacencia

- La principal ventaja de la representación mediante listas de adyacencia sobre la representación mediante matriz de adyacencia es que siempre utiliza un espacio proporcional a $E + V$, en oposición a V^2 en la matriz de adyacencia.

- La principal desventaja es que verificar la existencia de aristas específicas puede tomar un tiempo proporcional a V, a diferencia del tiempo constante en una matriz de adyacencia.

Tareas

- Implementar el TDA grafo, con operaciones:
  - ***crear_grafo***(número vértices) → grafo
  - ***insertar_arista***(grafo, arista) → void
  - **mostrar_grafo**(grafo) → void (muestra la matriz o lista de c/vértice según corresponda)
  - ***remover_arista***(grafo, arista) → void
  - ***pertenece_arista***(grafo, arista) → booleano
  - ***crear_arista***(vértice, vértice) → arista
  - ***obtener_aristas***(grafo) → arreglo de aristas
  - ***obtener_grado_vertice***(grafo, vértice) → entero
  - ***obtener_adyacentes_vertice***(grafo, vértice) → arreglo de vértices adyacentes

Tareas

- ¿Cuál representación es mejor? ¿Por qué?
- ¿Cuándo utilizar una u otra?

DEPARTAMENTO DE
INGENIERÍA
INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

CONSULTAS