Algoritmos y Estructuras de Datos

Facultad de Informática - UNLP

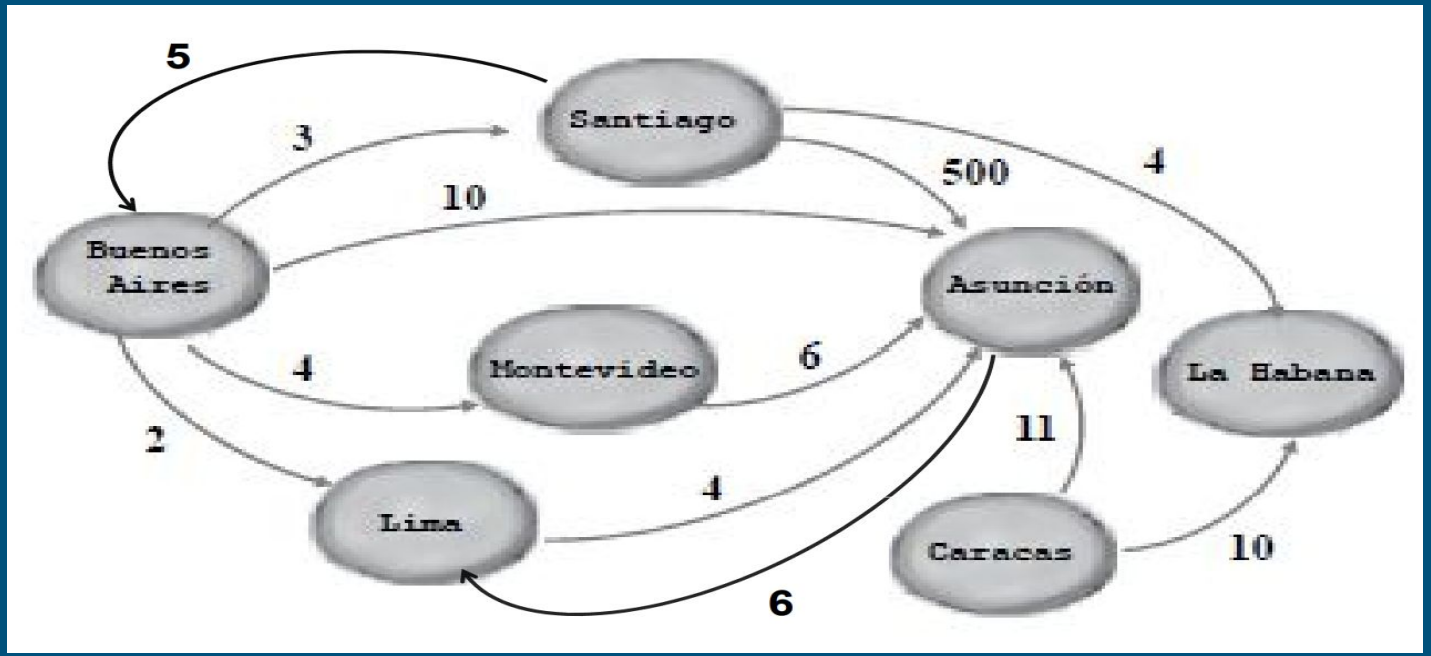
2019 - 2º Semestre

Grafos

¿Qué son los grafos ?

- **Grafo**: (V,E), V es un **conjunto de vértices** o nodos, con una relación entre ellos; E es un conjunto de pares (u,v), u,v Є V , llamados **aristas** o arcos.

¿Qué son los grafos ?



Tipos de Grafos

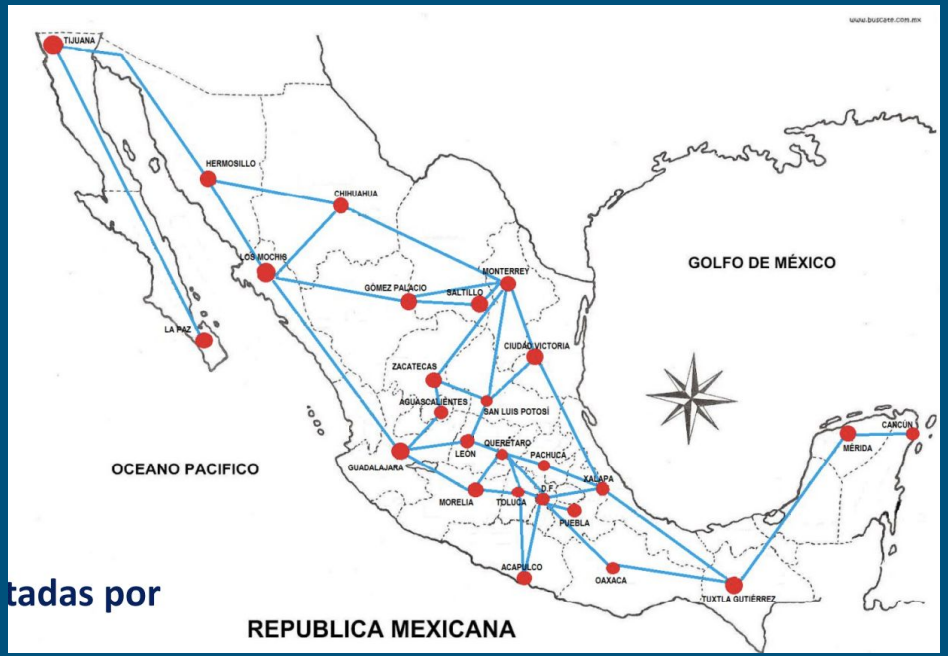
- **Grafo dirigido**: la relación sobre V **NO es simétrica**. Arista ≡ par ordenado (u,v).

- **Grafo no dirigido**: la relación sobre V **es simétrica**. Arista ≡ par no ordenado {u,v}, u,v Є V y u ≠ v.

- **Grafo pesado / con costos / ponderado**: cada arista tiene asociado asociado un valor o etiqueta

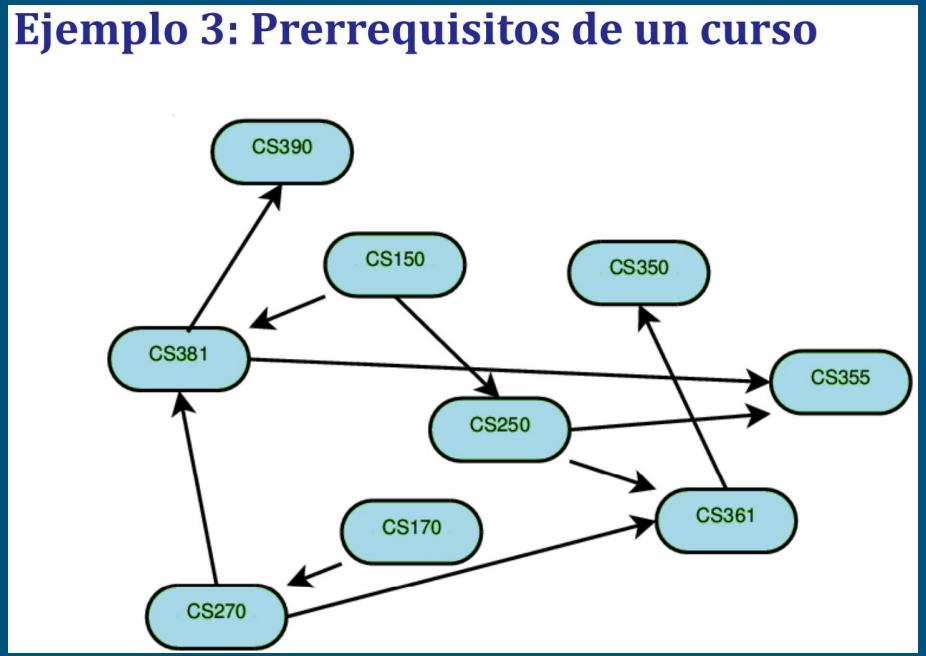
Tipos de Grafos

Es dirigido o no dirigido ?



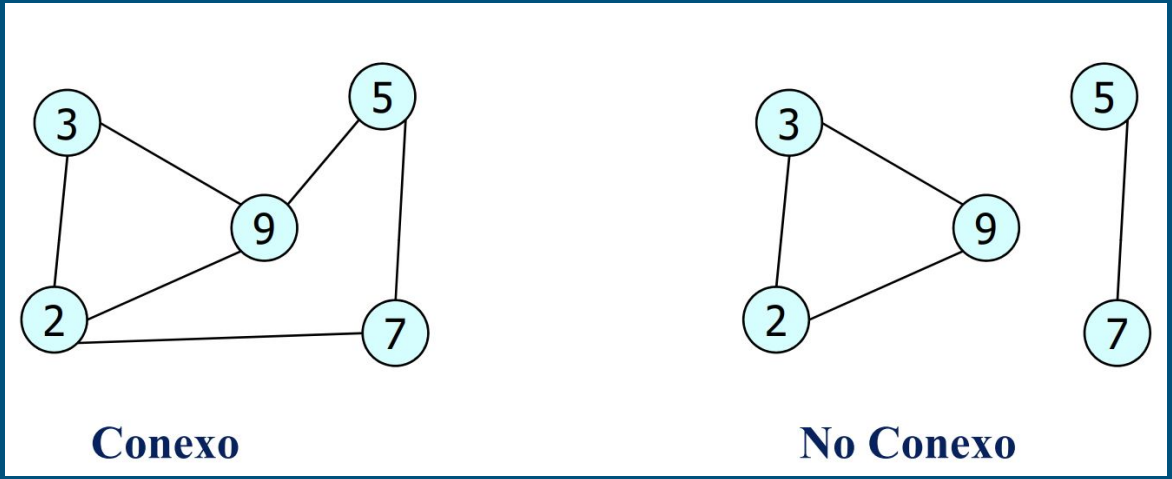
Tipos de Grafos

Es dirigido o no dirigido ?



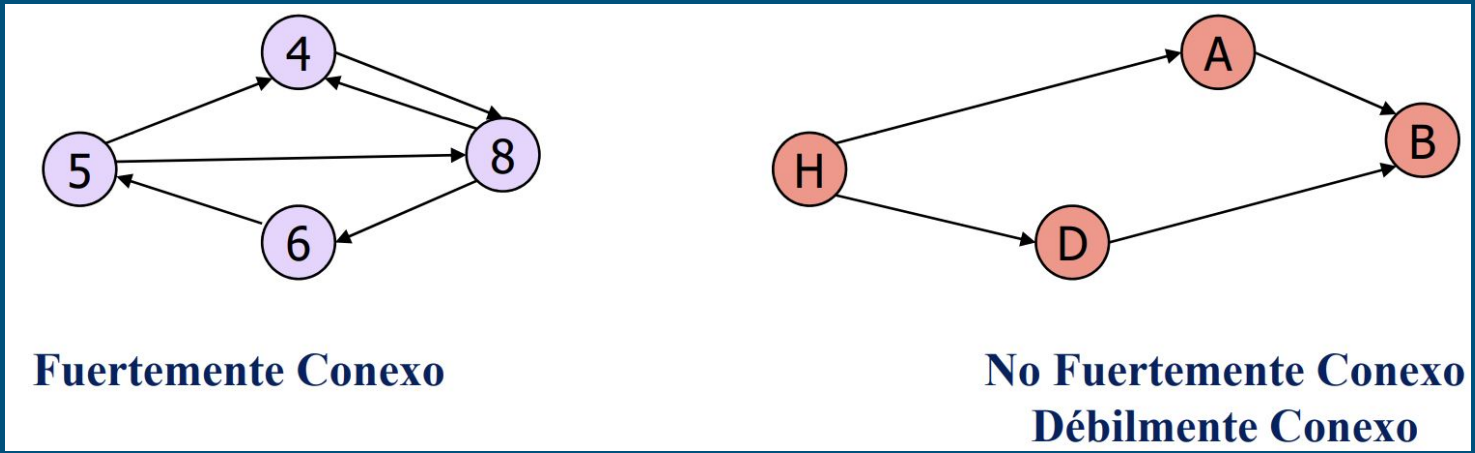
Tipos de Grafos

- **Grafo conexo**: Un grafo es conexo si hay un camino entre cada par de vértices.



Tipos de Grafos

- **Fuertemente Conexo**: Un grafo dirigido se denomina fuertemente conexo si *existe un camino desde cualquier vértice a cualquier otro vértice*



Tipos de Grafos - Más ejemplos

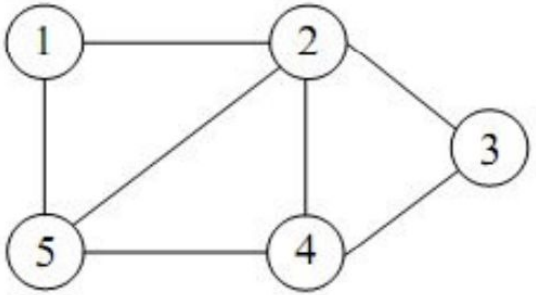
- Redes Sociales (Instagram vs Facebook)
- Rutas
- Vuelos
- Encomiendas

- **Un árbol, es un grafo ?**

Representaciones

- **Matriz de adyacencias**: Un grafo G=(V,A) se representa como una matriz de booleanos de |V| x |V| donde

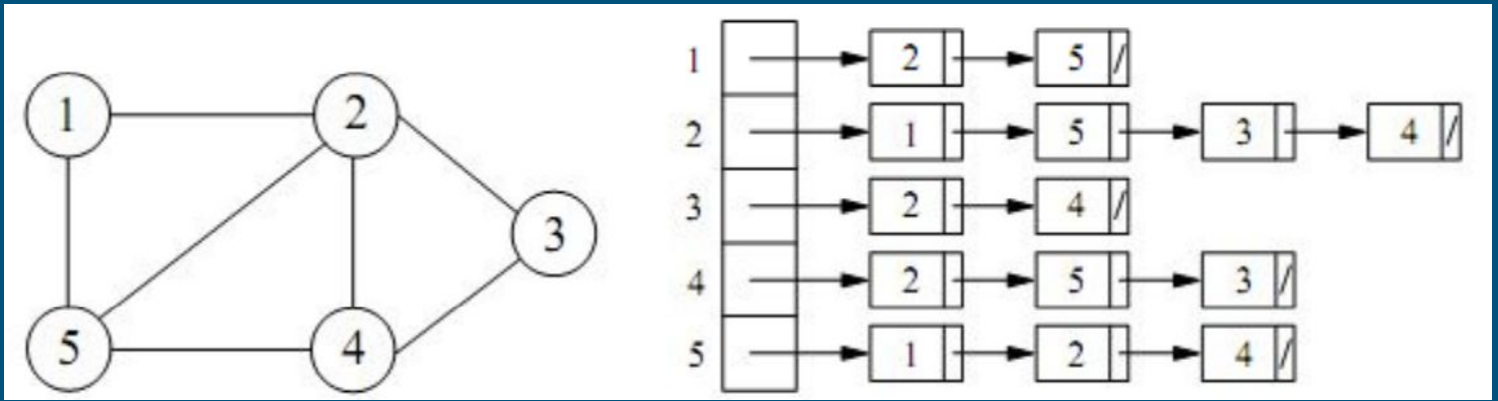
$$a_{ij} = \begin{cases} 1 & \text{si } (i,j) \in E \\ 0 & \text{en cualquier otro caso} \end{cases}$$



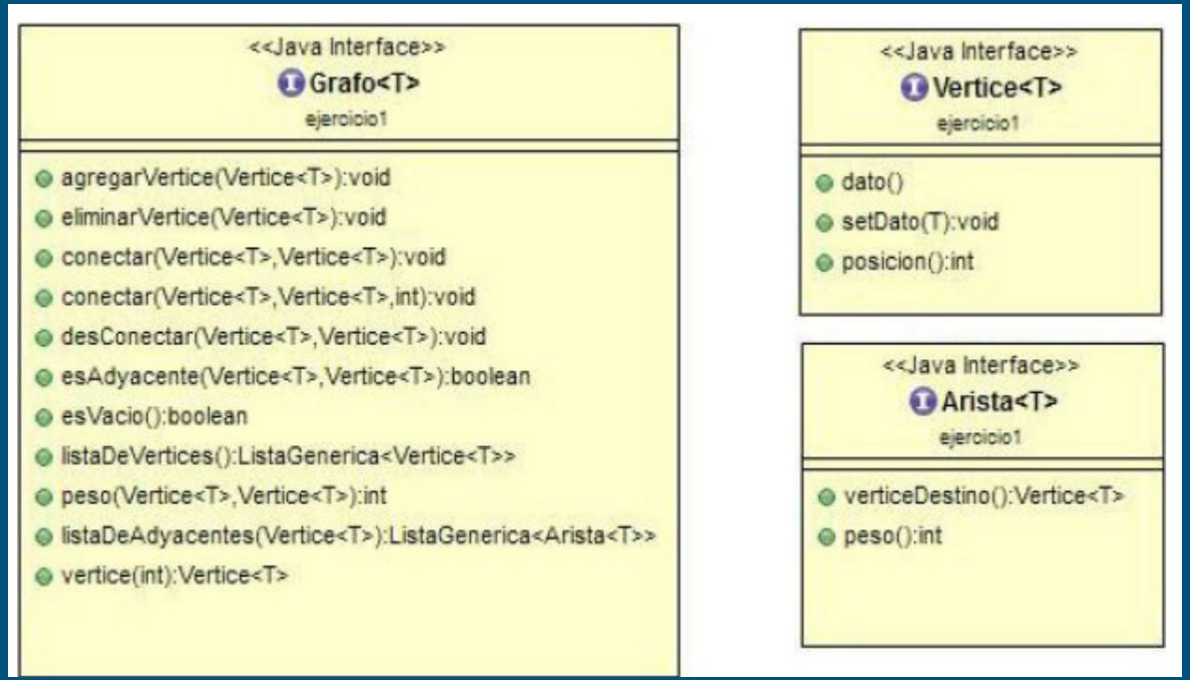
	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

Representaciones

- **Lista de Adyacencias**: Un grafo G=(V,A) se representa como un arreglo/lista de tamaño |V| de vértices.



Interfaz - Diagrama UML



Recorridos en Grafos -DFS

- **DFS (Depth First Search):** Es un algoritmo de recorrido de grafos en profundidad. Generalización del recorrido preorden de un árbol.

- Tiempo de ejecución en DFS: El tiempo es de O(|V|+|E|), deja de ser en función de un “N”, ahora hay dos variables, la cantidad de vértices (V) y de aristas (E). T(|V|,|E|)

Recorridos en Grafos - DFS

Estrategia, esquema **recursivo**: Dado un grafo G = (V , A)

1. Marcar todos los vértices como no visitados.
2. Elegir vértice “u“ (no visitado) como punto de partida.
3. Marcar “u” como visitado.
4. Para todo vértice “v” adyacente a “u” repetir 3 y 4.

* Finalizar cuando se hayan visitado todos los nodos alcanzables desde “u”.

* Si desde “u” no fueran alcanzables todos los nodos del grafo: volver a (2), elegir un nuevo vértice de partida “v” no visitado, y repetir el proceso hasta que se hayan recorrido todos los vértices.

Recorridos en Grafos - DFS (pseudocódigo)

```
main:dfs (grafo) {
        inicializar marca en false (arreglo de booleanos);
        para cada vértice w del grafo:
                si w no está visitado dfs(v);
}

dfs(v: Vertice) {
        marca[v]:= visitado;
        para cada nodo w adyacente a v:
                si w no está visitado dfs(w);
}
```

Recorridos en Grafos -DFS (JAVA)

```
public class Recorridos<T> {

 public void dfs(Grafo<T> grafo){
   boolean[] marca = new boolean[grafo.listaDeVertices().tamanio()];
   for(int i=0; i<grafo.listaDeVertices().tamanio();i++){
      if (!marca[i]) // si no está marcado
        this.dfs(i, grafo, marca);
   }
 }

 private void dfs(int i,Grafo<T> grafo, boolean[] marca){
    marca[i] = true;
    Arista <T> arista=null;
    int j=0;
    Vertice<T> v = grafo.listaDeVertices().elemento(i);
    System.out.println(v);
    ListaGenerica<Arista<T>> ady = grafo.listaDeAdyacentes(v);
    ady.comenzar();
    while(!ady.fin()){
       arista=ady.proximo();
       j = arista.getVerticeDestino().getPosicion();
       if(!marca[j])
           this.dfs(j, grafo, marca);
    }
 }
}
```

Recorridos en Grafos -BFS

- **BFS (Breadth First Search):** Es un algoritmo de recorrido de grafos en anchura. Generalización del recorrido recorrido por niveles niveles de un árbol.

- Tiempo de ejecución en BFS: El tiempo es de O(|V|+|E|), deja de ser en función de un “N”, ahora hay dos variables, la cantidad de vértices (V) y de aristas (E). T(|V|,|E|)

Recorridos en Grafos - BFS

Estrategia, esquema **iterativo**: Dado un grafo G = (V , A)

1. Encolar el vértice origen “u”.
2. Marcar el vértice “u” como visitado.
3. Procesar Procesar la cola.
    4. Desencolar un vértice “u” de la cola
    5. Para cada vértice “w” adyacente a “u”
        6. si “w” no ha sido visitado
            7. encolar y visitar “w”

* Si desde “u” no fueran alcanzables todos los nodos del grafo: volver a (1), elegir un nuevo vértice de partida no visitado,, y repetir el proceso hasta que se hayan recorrido todos los vértices.

Recorridos en Grafos BFS (JAVA)

```java
private void bfs(Vertice<T> vInicial, boolean[] visitados, ListaGenerica<T> resultado, Grafo<T> grafo) {
        ColaGenerica<Vertice<T>> cola = new ColaGenerica<Vertice<T>>();
        cola.encolar(vInicial);
        visitados[vInicial.getPosicion()] = true;
        while(!cola.esVacia()) {
                Vertice<T> vActual = cola.desencolar();
                resultado.agregarFinal(vActual.dato());
                ListaGenerica<Arista<T>> listaDeAdyacentes = grafo.listaDeAdyacentes(vActual);
                listaDeAdyacentes.comenzar();
                while(!listaDeAdyacentes.fin()) {
                        Vertice<T> vSiguiente = listaDeAdyacentes.proximo().verticeDestino();
                        if(!visitados[vSiguiente.getPosicion()]) {
                                visitados[vSiguiente.getPosicion()] = true;
                                cola.encolar(vSiguiente);
                        }
                }
        }
}
```

Preguntas ?

- Puedo ir al baño ?
- Soy un grafo ?
- Existen más recorridos ?