Grafos

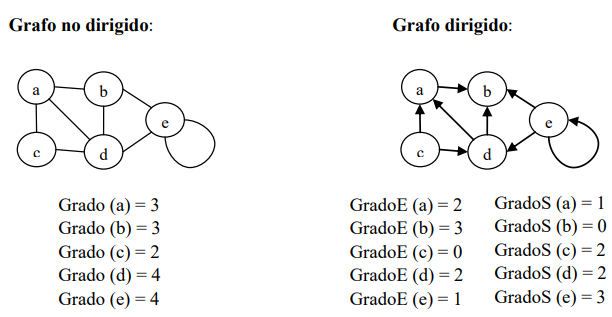
**Términos**

* Islas = grafos
* Varias islas = varios grafos
* Vértices = nodos
* Aristas = relaciones

Definición

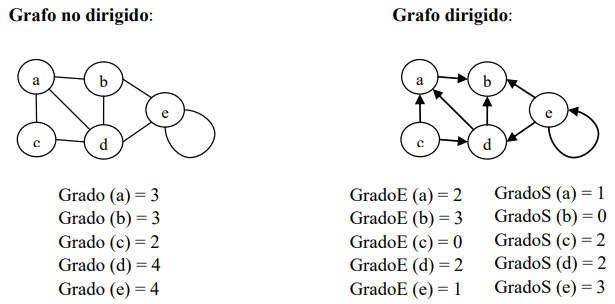
-Grafo es una manera de representar relaciones que existen entre pares de objetos. Así, un grafo es un conjunto de objetos llamados *vértices* y relaciones entre objetos que establecen una conexión entre pares de vértices, representadas por *aristas*

**Grafo/Aristas dirigidos y no dirigidos**

1. Grafo/Aristas no dirigidos: Es aquella que existen un pares los cuales en términos de función (u, v) ≠ (v, u).
2. Grafo/Aristas dirigidos: Es aquella que existen un par de vértices los cuales (u, v) = (v, u)

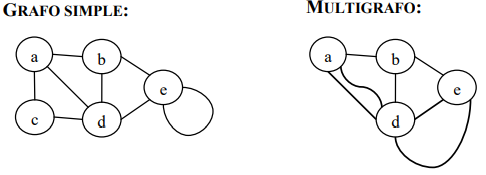
**Incidencia, adyacencia y grado de un vértice**

1. Incidencia: Conexión entre pares. Las aristas (relaciones) 🡪 (u, v) es incidente con los vértices (nodos) u y v
2. Adyacencia: Dos vértices (nodos) son adyacentes (apuntan 1 par de nodos a ellos mismos) si existe una arista cuyos vértices sean u y v
   1. El vértice u es adyacente a v
   2. El vértice v es adyacente desde u
3. Grado: El grado de un vértice (nodo) es el número de vértices adyacentes a u (a un sólo). Para un grafo dirigido , el grado de salida de un vértice u es el número de vértices adyacentes desde u, mientras que el grado de entrada de un vértice u es el número de vértices adyacentes a u



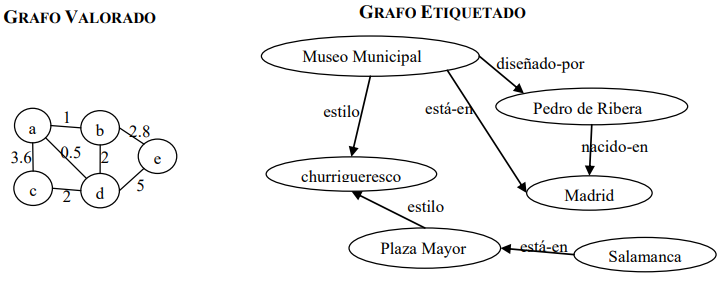
**Grafos simples y multígrafos**

-Un grafo simple es aquél que tiene una sola aristas por un par de nodos, en cambio a un grafo a multígrafos es aquél que puede tener varias aristas por un par de nodos



**Grafos valorados y grafos etiquetados**

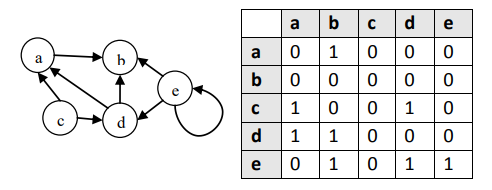
1. Grafos valorados: Es un grafo la cual cada arista tiene un peso denominado “función de coste”. *El peso de un camino en un grafo es la suma de todos los pesos de cada arista*
2. Grafos etiquetados: Es un grafo la cual cada arista tiene una etiqueta no numérica que le da sentido a la arista



**Matriz de adyacencias**

-Una matriz de adyacencias determina las adyacencias (conexiones) entre pares de vértices (nodos). Los vértices se conciben como enteros en la matriz (0, 1, …, n-1). Esto permite almacenar las referencias a las aristas (en las aristas) en las celdas de la matriz.

Cada fila y columna representa cada vértice (nodo) del grafo y cada posición representa una arista (o la ausencia de ella) cuyo vértice de origen se encuentra en la fila y el vértice final se encuentra en la columna



**Lista de adyacencias**

-En una lista de adyacencias, a cada vértice “i” se le asocia una lista enlazada con todos los vértices (nodos) j adyacentes a “i”. De esta forma solo ocupa memoria necesaria y no crea espacios nulos como pasa en la matriz

El grafo se representa por un número “n” de componentes, siendo “n” el número de vértices (nodos) del grafo, donde cada componente constituye la lista correspondiente a cada uno de los vértices (nodos)

Cada nodo consta de un campo indicando el vértice adyacente. En caso de que el grafo sea etiquetado o valorado, se le añade un segundo campo para guardar el valor del peso o etiqueta

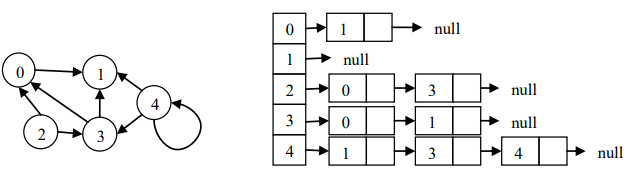


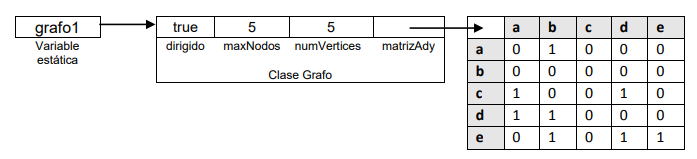
Diagrama de clases

**Class Grafo()**

* agregarVértice(vértice a): *Siempre y cuando no supere el máximo de nodos permitidos*
* eliminarVértice(vértice a)
* agregarArista(vértice a, vértice b)
* eliminarArista(vértice a, vértice b)
* establecerMaxNodos(int a): Establece número máximo de nodos del grafo
* esVacío: Retorna true o false
* existeArista(vértice a, vértice b): Retorna true o false
* gradoEntradaVértice(? a): Devuelve grado de entrada del vértice
* gradoSalidaVértice(? a): Devuelve grado de salida del vértice
* incidencia: Devuelve la incidencia del vértice
* tamaño: Devuelve el tamaño (suma de números de aristas) del grafo
* esDirigido(grafo a): Devuelve true/false si es dirigido
* establecerDirigido(boolean a): Establecer grafo dirigido (true) o no dirigido (false)

**Class GrafoMatriz()**

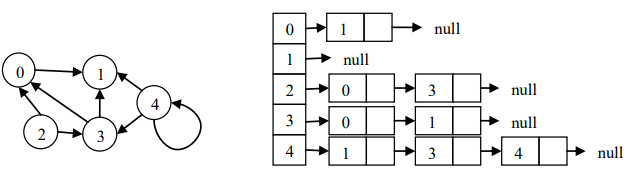
1. Private Dirigido: Boolean
2. Private maxNodos: int
3. Private numVértices: int
4. Private matrizAdy: Boolean [][]



* constructor 1(boolean a)
  + maxNodos = numVértices = 0
  + dirigido = a
* constructor 2(boolean a, int n)
  + dirigido = a
  + maxNodos = n
  + numVértices = 0: No se asigna n porque no se ha agregado ningún nodo
  + matrizAdy = new Boolean[n][n] Se asigna la variable n para establecer la cantidad max de nodos permitidos

**Class GrafoLista()**

1. private dirigido = boolean
2. private maxNodos = int
3. private numVértices = int
4. private listaAdy = []



* Constructor 1(boolean a)
  + maxNodos = numVértices = 0
  + dirigido = a
* Constructor 2(int a, boolean b)
  + dirigido = b
  + maxNodos = a
  + numVértices = 0
  + listaAdy = new Lista[a]

**Class NodoLista()**

1. public clave = int
2. public NodoLista sig;

* Constructor 1(int a, NodoLista b)
  + clave = a
  + sig = b

**Class Lista()**

1. public NodoLista inicio

* Constructor 1()
  + Inicio = null

Class Paths()

* En profundidad (DFS)
* En amplitud (BFS)