Grafos
Algoritmos básicos

Dra. MPuerto Paule Ruiz

Objetivos

- Hacer algoritmos básicos de grafos
- Pruebas unitarias

Clase Graph

- **private double[][] weights;// Pesos de las arista**
- **private boolean[][] edges;// Matriz de aristas**
- **private T[] nodes;// Nodos: Lista de nodos**
- **private int size;**

Constructor, getSize, getNode, existNode

- **public Graph (int dimension)**
- **public int getSize()**
- **protected int getNode(T node)→** Devuelve la posición del nodo pasado como parámetro dentro del vector de nodos y -1 si el nodo no existe
- **public boolean existsNode(T node)→** indica si existe (true) o no (false) el nodo en el grafo

addnode, existEdge, getEdge

- **public int addNode(T node)** → Inserta un nuevo nodo que se le pasa como parámetro. Si lo inserta devuelve 0. Si ya existe y además no cabe, no lo inserta y devuelve -3. Si ya existe, pero sí cabría, no lo inserta y devuelve -1. Si no existe, pero no cabe, no lo inserta y devuelve -2. Si el nodo a insertar no es válido, devuelve –4 en cualquier caso.
- **public boolean existEdge(T source, T target)** → Devuelve true si existe una arista entre los nodos origen y destino, false en caso contrario o no existen los nodos
- **public double getEdge(T source, T target)** → Devuelve el peso de la arista que une el nodo origen y el nodos destino. devuelve –1, -2 y –3 si no existe origen, destino, ambos. Devuelve –4 si no existe la arista.

addEdge, removeEdge, removeNode

- **public int addEdge(T source, T target, double weight)→**Inserta una arista con el peso indicado (> 0) entre dos nodos. Devuelve 0 si la inserta. Devuelve -1, -2 y -3 si no existe nodos origen, destino o ambos respectivamente. Devuelve -4 si ya existe y -8 si el peso no es válido
- **public int removeEdge(T source, T target)→**Borra una arista del grafo que conecta dos nodos. Devuelve -1, -2 y -3 si no existe nodos origen, destino o ambos respectivamente. Devuelve -4 si no existe la arista. Devuelve 0 si la borra
- **public int removeNode(T node)→**Devuelve 0 si borra el nodo correctamente y -1 en caso contrario

toString()

```
public String toString() {
   DecimalFormat df = new DecimalFormat("#.##");
   String cadena = "";

   cadena += "NODOS\n";
   for (int i = 0; i < numNodes; i++) {
      cadena += nodes[i].toString() + "\t";
   }
   cadena += "\n\nARISTAS\n";
   for (int i = 0; i < numNodes; i++) {
      for (int j = 0; j < numNodes; j++) {
         if (edges[i][j])
            cadena += "T\t";
         else
            cadena += "F\t";
      }
      cadena += "\n";
   }
   cadena += "\nPESOS\n";
   for (int i = 0; i < numNodes; i++) {
      for (int j = 0; j < numNodes; j++) {
         cadena += (edges[i][j]?df.format(weights[i][j]):"-") + "\t";
      }
      cadena += "\n";
   }
   return cadena;
}
```

Tarea a realizar

- Pruebas unitarias exhaustivas para cada método