Grafos
Algoritmos básicos

Estos apuntes son notas de clase realizadas por el profesor.
El lector ha de complementarlas con las sesiones impartidas de manera presencial

Dra. MPuerto Paule Ruiz

Objetivos

- Construir la estructura de datos Grafos
- Pruebas unitarias de las operaciones de Grafos

Clase Grafos- Matrices de adyacencias

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

addNode, getNode, existsNode, removeNode

- **public Grafos (int dimension)**
- **public int getSize()→** Devuelve el tamaño del grafo
- **public boolean addNode(T node)→** Inserta un nuevo nodo que se le pasa como parámetro.
  - Si ya existe, no lo inserta y devuelve false.
  - Si recibe un nodo nulo, no lo inserta y lanza una NullPointerException.
  - Si no cabe, no lo inserta y lanza una FullStructureException.
- **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 o es null
- **public boolean existsNode(T node)→** indica si existe (true) o no (false) el nodo en el grafo
- **public boolean removeNode(T node)→** Si existe, borra el nodo deseado del vector de nodos así como las aristas en las que forma parte y devuelve true.

  Si el nodo no existe devuelve false.

  Si recibe un nodo nulo, lanza una NullPointerException

addEdge, existsEdge, getEdge, removeEdge

- **public boolean addEdge(T source, T target, double weight)→** Inserta una arista con el peso indicado (> 0) entre dos nodos, uno origen y otro destino.

  Devuelve true si la inserta

  Devuelve falso si ya existe el arco (arista)

  Lanza una excepcion ElementNotPresentException si no existe el nodo origen o destino

  Lanza una IllegalArgumentException si el peso es invalido.
- **public boolean existsEdge(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 conecta dos nodos.
  - Si los nodos source o target no existen, lanza una excepción de tipo ElementNotPresentException
  - Si tanto source como target existen, pero la arista a eliminar no, devuelve -1
- **public boolean removeEdge(T source, T target)→** Borra la arista del grafo que conecta dos nodos.
  - Si la arista existe, se borra y devuelve true.
  - Si los nodos source o target no existen, lanza una excepción de tipo ElementNotPresentException
  - Si tanto source como target existen, pero la arista a eliminar no, devuelve false.

public String toString() {

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

   }
```

Tarea a realizar

- Pruebas unitarias exhaustivas para cada método