PAQUETE

MODELO

**Clase Grafo**

**Variables de clase**

* *double[][] D*: representa la matriz de distancias interna del grafo.

**Métodos de clase**

A continuación, veremos la descripción de los métodos más importantes de esta clase:

* *void agregarArista(int i, int j, double distancia)* : se encarga de agregar una arista entre los vértices **i** y **j** colocando en esa posición de la matriz la distancia entre ambos.
* *void borrarArista(int i, int j):* se encarga de eliminar la arista entre los vértices **i** y **j** volviendo nula la distancia entre ellos.
* *boolean existeArista(int i, int j):* verifica si existe una arista entre los vértices i,j es decir si la posición de la matriz es mayor a cero.
* *set<Integer> vecinos(int i)*: retorna una colección con aquellos vértices que tengan una arista con el vértice i.

**Clase AGM**

Esta clase implementa una versión del famoso algoritmo de Prim para obtener un AGM.

**Variables de clase**

* *Grafo grafoAGM:* variable que contendrá el árbol generador mínimo del grafo pasado como parámetro en el constructor.
* *boolean marcados[]:* arreglo que usaremos para ver que vértices ya han sido agregado en el *grafoAGM* para luego buscar la arista mínima entre todos ellos.
* *ArrayList<Arista> aristas:* contendrá las aristas finales del árbol generador mínimo.

**Métodos de clase**

* AGM (Grafo grafo): constructor que recibe como parámetro un grafo con el cual se trabajara para armar su árbol generador mínimo.
* *void verificarTamanio():* comprueba que el grafo no este vacío, es decir que contenga al menos un vértice.
* *boolean todosMarcados():* chequea si todos los vértices están marcados es decir si fueron agregados al árbol generador mínimo.
* *void generarAGM (Grafo grafo):* método que construye el árbol generador mínimo. Lo que hace su algoritmo es lo siguiente:

Supongamos que tenemos este grafo, lo que hacemos en primer lugar es marcar un vértice cualquiera, nosotros marcamos el vértice 0. Luego por cada vértice marcado recorremos sus vecinos para encontrar la arista con menor peso entre todas.

Grafo

10

12

9

2

30

7

15

En este caso la arista con menos peso fue la del vértice 0 y el 1. Agregamos esa arista al *grafoAGM* y marcamos en el arreglo el vértice 1. Ahora repitiendo el mismo proceso buscara la arista con menor peso entre los vértices marcados.

Grafo

10

12

9

2

30

7

15

Se agrega la arista (1,5) al grafoAGM porque fue la menor y marcamos el vértice 5 en el arreglo.

Este proceso se repite hasta tener todos los vértices marcados.

Grafo

10

12

9

2

30

7

15

Finalmete el grafoAGM quedaría conformado por estas aristas.

Grafo

12

9

2

7

15

Aristas de las cuales se elegirá la menor para agregar al grafoAGM.

Vértices ya marcados y aristas agregadas al grafoAGM.

Vértices no marcados.

**Clase Arista**

**Variables de clase**

* *int vertice1, vertice2:* representan los vértices que se unen mediante una arista.
* *double peso:* representa el valor de la distancia entre ambos vértices.

Los métodos de esta clase no son relevantes ya que devuelven información de la Arista en sí misma.