Geeks México

BLOG DE PROGRAMACIÓN EN ESPAÑOL SOBRE JAVA, FRAMEWORKS, BASES DE DATOS, CÓMPUTO EN LA NUBE, ETC. EN ESPAÑOL Y EN INGLÉS.

Implementa un grafo de ciudades

en Java

HACE 1 SEMANA DEJA UN COMENTARIO

Una de las estructuras de datos más complejas y más útiles son los grafos (Graphs), en este post se explicará paso a paso como implementar uno en Java. Para este ejemplo se creará uno sobre algunas de las ciudades que se encuentran en México.

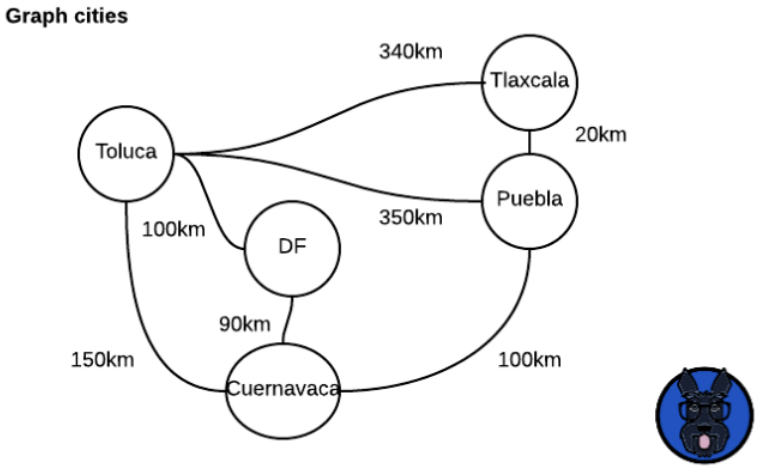
Conceptos básicos

Antes de crear el grafo debemos entender algunos conceptos básicos:

- Vértice (Vertex) : Es un punto donde varias líneas se unen
- Arista (Edge) : Es un término matemático que representa una línea que conecta 2 vértices
- Grafo (Graph) : Es el término que representa un conjunto de vértices y aristas.

Ejemplo práctico

En el ejemplo que se presentará se creará un Grafo de ciudades, cada nodo será una ciudad (DF, Toluca, Cuernavaca, Puebla y Tlaxcala) que se unirán con un conjunto de aristas con su distancia, la siguiente imagen es una representación gráfica:



Creando el modelo

Como siempre el primer paso será crear el modelo con las clases a utilizar, para esto crearemos 3 clases Node, Edge y Grap que serán mostradas a continuación:

Node.java

```
1  import java.util.ArrayList;
2  import java.util.List;
3
4  /**
5   * @author raidentrance
6   *
7   */
8  public class Node {
9      private String city;
10     private List<Edge> edges;
11
12     public Node(String city) {
13         this.city = city;
14     }
15
16     public String getCity() {
17         return city;
18     }
19
20     public void setCity(String city) {
21         this.city = city;
22     }
23
24     public List<Edge> getEdges() {
25         return edges;
26     }
27
28     public void addEdge(Edge edge) {
29         if (edges == null) {
30             edges = new ArrayList<>();
31         }
32         edges.add(edge);
```

```
33      }
34
35      @Override
36      public String toString() {
37          return "\n \tNode [city=" + city +
38      }
39  }
```

Edge.java

```
1   /**
2    * @author raidentrance
3    *
4    */
5   public class Edge {
6       private Node origin;
7       private Node destination;
8       private double distance;
9
10      public Edge(Node origin, Node destinat
11          this.origin = origin;
12          this.destination = destination;
13          this.distance = distance;
14      }
15
16      public Node getOrigin() {
17          return origin;
18      }
19
20      public void setOrigin(Node origin) {
21          this.origin = origin;
22      }
23
24      public Node getDestination() {
25          return destination;
26      }
27
28      public void setDestination(Node destir
29          this.destination = destination;
30      }
31
32      public double getDistance() {
33          return distance;
34      }
35
36      public void setDistance(double distanc
37          this.distance = distance;
38      }
39
40      @Override
41      public String toString() {
42          return "\n Edge [origin=" + origir
43                  + distance + "]";
44      }
45
46  }
```

Graph

```java
import java.util.ArrayList;
import java.util.List;

/**
 * @author raidentrance
 *
 */
public class Graph {

    private List<Node> nodes;

    public void addNode(Node node) {
        if (nodes == null) {
            nodes = new ArrayList<>();
        }
        nodes.add(node);
    }

    public List<Node> getNodes() {
        return nodes;
    }

    @Override
    public String toString() {
        return "Graph [nodes=" + nodes + '
    }

}
```

A continuación se presenta la explicación:

- *Un Graph contiene una lista de nodos*
- *Cada nodo contiene el nombre de una ciudad y una lista de aristas*
- *Cada arista contiene la unión de las ciudades así como la distancia entre ellas*

Creando el grafo de ciudades

Una vez que se creó el modelo el siguiente paso será llenarlo con la información de las ciudades.

```java
/**
 * @author raidentrance
 *
 */
public class MapRepresentation {

    public static Graph getCities() {
        Node df = new Node("DF");
        Node toluca = new Node("Toluca");
```

```
        Node cuernavaca = new Node("Cuerna
        Node puebla = new Node("Puebla");
        Node tlaxcala = new Node("Tlaxcala

        df.addEdge(new Edge(df, toluca, 10
        df.addEdge(new Edge(df, cuernavaca

        toluca.addEdge(new Edge(toluca, cu
        toluca.addEdge(new Edge(toluca, pu
        toluca.addEdge(new Edge(toluca, tl

        cuernavaca.addEdge(new Edge(cuerna

        puebla.addEdge(new Edge(puebla, tl

        Graph graph = new Graph();
        graph.addNode(df);
        graph.addNode(toluca);
        graph.addNode(cuernavaca);
        graph.addNode(puebla);
        return graph;
    }

    public static void main(String[] args)
        Graph graph = getCities();
        System.out.println(graph);
    }
}
```

Salida:

```
Graph [nodes=[
    Node [city=DF, edges=[
 Edge [origin=DF, destination=Toluca, dist
 Edge [origin=DF, destination=Cuernavaca,
    Node [city=Toluca, edges=[
 Edge [origin=Toluca, destination=Cuernava
 Edge [origin=Toluca, destination=Puebla,
 Edge [origin=Toluca, destination=Tlaxcala
    Node [city=Cuernavaca, edges=[
 Edge [origin=Cuernavaca, destination=Pueb
    Node [city=Puebla, edges=[
 Edge [origin=Puebla, destination=Tlaxcala
```

Si observamos la salida del programa podemos ver que cada ciudad cuenta con sus aristas que representan la unión de ambos puntos y cuenta con la distancia entre ellas como se mostró en la imagen.

Siguientes pasos

En próximos posts se explicará ejecutar algoritmos sobre este grafo, agregar dirección a las aristas y

muchas más operaciones, si te gusta el contenido y quieres enterarte cuando realicemos un post nuevo síguenos en nuestras redes sociales https://twitter.com/geeks_mx (https://twitter.com/geeks_mx) y https://www.facebook.com/geeksJavaMexico/ (https://www.facebook.com/geeksJavaMexico/).

Autor: Alejandro Agapito Bautista

Twitter: @raidentrance

Contacto:raidentrance@gmail.com