28 de junio de 2017

urtubey, Bianchini, posadas

programacion iii

informe tp nº2

camino minimo

# INFORME TP Nº2: CAMINO MINIMO

## INTRODUCCION

En este informe se pretende explicar el desarrollo realizado para resolver el problema planteado en el TP nº2 de la materia, donde se requería implementar un programa que encuentre el mejor camino entre un par de ubicaciones de un mapa, considerando las rutas disponibles y los peajes que contengan.

## IDEA

La idea fue permitir a una persona marcar puntos en un mapa e interconectarlos. Luego podría decirle al programa qué cantidad máxima de peajes está dispuesta a pagar, y finalmente el programa le mostraría el mejor camino.

## IMPLEMENTACION

El desarrollo consta de cuatro grupos principales que son una interfaz, un mapa, un grafo y una matriz. Cada uno representa un nivel de abstracción del problema y naturalmente cumple una función específica.

##### Interfaz:

Este grupo se compone de dos clases, que son *VentanaPrincipal* e *Interfaz*. La primera se dedica a interactuar con el usuario. Contiene un mapa interactivo de la clase JMap y una serie de botones, que permiten agregar puntos en el mapa, interconectarlos, crear rutas con peajes y finalmente obtener el mejor camino.

La segunda contiene los métodos que hacen a la lógica del grupo:

* **pasarDatosAlMapa:** este método instancia un objeto de la clase MapaRutas y le pasa las coordenadas de todos los puntos del mapa que se encuentran conectados. Luego agrega la relación entre ambos, es decir le dice al mapa que entre ellos existe una ruta.
* **obtenerCamino:** este método toma la coordenada de inicio y destino de la ventana principal y llama al método obtenerRutaOptima del mapa de rutas. Finalmente recibe del mapa de rutas el mejor camino, y lo retorna.
* **dibujarCaminoMinimo:** este método toma cada punto del camino mínimo y llama a dibujarLineaEntrePuntos de la ventana principal, creando en la vista una línea entre cada punto del camino y el siguiente.
* Contiene además varios métodos de agregado de puntos y eliminación, que agregan y eliminan información del JMap.

##### Mapa:

Este grupo se compone de las clases *Coordenada y MapaRutas.* Representa la implementación de un mapa lógico, que maneja las coordenadas que la interfaz le pasa. La clase principal *MapaRutas c*ontiene una instancia de *GrafoPesadoUnidireccional* y otra de *MatrizCartesiana*, ambos utilizados para manejar la información de las coordenadas, las rutas y sus peajes.

Sus métodos principales son:

* **obtenerRutaOptima**: Toma una coordenada de origen y una de destino, y un número máximo de peajes. Calcula el mejor camino existente entre ambos puntos teniendo en cuenta la máxima cantidad de peajes que se desea atravesar (en caso de ser cero llama a un método secundario que obtiene el camino mínimo sin considerar peajes). Así mismo se encarga de instanciar un grafo pesado unidireccional, que va a representar con vértices y aristas con peso a las coordenadas que contiene el mapa. Se sigue una lógica de capas para lidiar con el problema de los peajes. Luego se le pide al grafo que realice la búsqueda del camino mínimo. Este devuelve una lista de caminos.
* **obtenerMejorResultado**: toma la lista de caminos encontrados y devuelve el mejor.

##### Grafo

El grupo Grafo se compone de las clases *Grafo, GrafoUnidireccional* y *GrafoPesadoUnidireccional*. Cada uno hereda del anterior y son una forma de representar las coordenadas y las rutas del mapa como datos. La clase principal *GrafoPesadoUnidireccional* contiene una instancia de una matriz de pesos, que sirve para almacenar el peso (es decir la distancia) entre un vértice y otro. El objetivo de esta clase es realizar el cálculo del camino mínimo, usando los vértices que posee y las distancias almacenadas en la matriz.

Los métodos más importantes son:

* **obtenerCaminoMinimo**: Utilizando la técnica de A\* (“A estrella”) se obtiene la lista de vértices que conforman el camino más corto entre un vértice de origen y uno de destino. En este caso los peajes no se contemplan, porque la implementación del grafo en capas ya posee vértices que representan los peajes. Devuelve el mejor camino.
* **sePuedeLLegar**: antes de calcular el camino mínimo entre A y B, este método calcula si existe algún camino entre A y B, y lo hace iterando por cada uno, cambiando entre vértices vecinos hasta que la iteración lo lleve al destino. Si vuelve al origen, entonces no es posible llegar.

##### Matriz

Este grupo contiene un par de matrices, *MatrizCartesiana* y *MatrizRelacional* que heredan de *Matriz*. Ambas cumplen la función de almacenar datos. La primera almacena los peajes que tiene el mapa, es decir que contiene un uno si la arista v1, v2 tiene peaje, y un cero si no lo tiene. Se la instancia en la clase *MapaRutas*. La segunda almacena relaciones entre los vértices, es decir los pesos de sus aristas. De este modo, sabemos si existe la arista entre dos vértices y conociendo de inmediato el peso de la misma. Se instancia en *GrafoPesadoUnidireccional*.

Los métodos que poseen sirven para agregar relaciones, obtenerlas, y verificar vértices y aristas.

## PROBLEMAS DE IMPLEMENTACION

El principal problema de implementación que tuvimos fue la traducción desde un mapa de coordenadas y rutas a un grafo en capas. Nuestra idea principal era que el usuario pudiera elegir, desde qué punto a qué punto se calcule el camino mínimo, sin importar el sentido del camino. De este modo cada arista que se agregara al grafo iría de A a B y de B a A. Pero no se tuvo en cuenta que, al crear capas que representen los vértices con pajes, estas debían crearse también en ambos sentidos. Es decir, si del vértice 3 al 4 tenemos un peaje, se crea un vértice 8 y sabemos por la matriz de peajes que llegar desde 3 a 4 es lo mismo que llegar de 3 a 8 con un costo mayor. Pero esta relación no contempló lo que debería ocurrir si se desea ir de 4 a 3. Esto llevó a numerosos bugs que llevaron horas de testeo y cuando encontramos cuál era el problema caímos en la cuenta de que debería cambiarse toda la implementación que ya teníamos de grafo para poder solucionarlo. Se decidió que el programa realizaría entonces el cálculo de camino mínimo bien, pero con aristas que van de A a B, unidireccionalmente.