INTELIGENCIA ARTIFICIAL

Grupo Número 10

INTEGRANTES:

Rubén Ibáñez Redondo

Óscar Gómez Arqueros

Sergio Vega Adrián

**ALGORITMO**

Para realizar esta práctica escogimos el algoritmo de búsqueda en la optimización de caminos de coste mínimo en grafos de decisión llamado A Estrella, para ello, implementamos este algoritmo en la herramienta Eclipse de Java.

En primer lugar, creamos la clase NodeData<T> con los atributos privados ‘g’ que es coste del camino más bajo de un origen a un destino, ‘h’ que es la función heurística y ‘f’ que es el coste real del camino (f = g + h).

A continuación, hacemos una nueva clase llamada GrafoAEstrella<T> que creará un mapa con un conjunto de nodos con los métodos anadirNodo y el método anadirArista.

Después, realizaremos la última clase llamada AStar<T> en el que haremos el código en sí del algoritmo A Estrella en el que compararemos los nodos y sus respectivos valores de distancia entre ellos y creará una lista de nodos desde el origen hacia el destino que es lo que finalmente imprimiremos por pantalla.

En último lugar, creamos una función para crear nuestro grado que contenga nuestra base de datos con todo lo necesario para implementar el mapa de metro de la ciudad de Monterrey. Para ello, crearemos un mapa llamado ‘heurística’ en el que pondremos las 38 paradas de las tres líneas en la que en cada una indicaremos los nodos adyacentes a cada uno y sus respectivas distancia reales en metros en línea recta (incluido él mismo con distancia 0) para luego crear las tres líneas y añadir a cada para un nombre (un string) que es lo que se imprimirá pon pantalla.

Para acabar nos queda crear el grado y añadirle las aristas, para crear el grafo utilizaremos nuestro método anadirNodo para incluir los 38 y después con el método anadirArista incluiremos cada una de las conexiones entre nodos especificando también su distancia en línea recta en metros (Hemos incluido las distancias reales de las líneas 1 y 2, buscándolas en Google Maps, sin embargo, en la línea 3, al ser ficticia y no estar construida actualmente hemos puesto unas distancias ficticias pero coherentes respecto al resto de líneas).

Finalizaremos en el main haciendo una llamada a esta función anterior para crear el grado y creando un bucle dentro de nuestro grafo aStar con 2 atributos en el que introduciremos nuestra parada de origen y nuestra parada de destino para que imprima el camino que recorre el metro parada a parada.

**INTERFAZ GRÁFICA**

En cuanto a la interfaz gráfica, hemos utilizado la herramienta gráfica Netbeans para crear el grafo correspondiente a la red de Monterrey (38 nodos que se corresponden con cada una de las paradas y sus respectivas aristas para obtener las conexiones).

En esta interfaz, y para hacerlo más visual, hemos creado dos desplegables con la lista de las 38 paradas, dos imágenes (insertadas en una carpeta dentro del mismo src para que pertenezcan al .JAR y no estén en el disco duro del ordenador), dos cuadros de texto (en uno de ellos nos indicará la lista de paradas a seguir según el algoritmo que hemos elegido) y finalmente un botón. Para todos ellos, ajustamos su lugar dentro de la interfaz y sus dimensiones, y además, para el caso del botón asociamos su acción asociada que será calcular el camino con el algoritmo de búsqueda A estrella en nuestro grado entre las 2 paradas elegidas en los desplegables.

Así, se puede observar que existe un botón “Origen” en el que se selecciona el punto de partida, otro botón “Destino” en el que se selecciona el punto de fin de trayecto y un último botón “Calcular ruta” con el que mediante un click, obtenemos la secuencia de paradas que se debe seguir para cumplir con las especificaciones del algoritmo A\* en uno de los dos recuadros de texto, además del otro recuadro de texto que tiene un determinado texto invariable y de las dos imágenes.