Práctica IA: Metro Atenas

Grupo
Julia Serrano Arrese
Carolina Garza Bravo
José Ignacio Conejero Novelo
Pablo Miralles Retuerto(Coordinador(5S2M))

ÍNDICE

1. Introducción
2. Desarrollo del Algoritmo de búsqueda A*
3. Desarrollo de la interfaz gráfica
4. Pruebas
5. Conclusiones
6. Bibliografía

1. Introducción

Este proyecto consiste en obtener el camino mínimo entre dos puntos de la red metro. El punto de partida es la estación origen y el punto final la estación destino

Para la realización del proyecto se ha contado con los datos de la red de metro de Atenas, obtenidos a través de distintas fuentes y se ha utilizado el algoritmo de búsqueda en la optimización de caminos de coste mínimo en grafos de decisión A*

La red que vamos a utilizar tiene un solo modo: metro, 3 líneas 1, 2 y 3, identificadas con los colores verde, rojo y azul, y 54 estaciones, 4 de las cuales permiten transbordo entre 2 líneas

2. Desarrollo del algoritmo de búsqueda A*

El algoritmo de búsqueda A*, a diferencia de otros algoritmos de búsqueda tiene en cuenta tanto el valor heurístico (h(n)) de los nodos como el coste real del recorrido (g(n)).

Comenzamos el proyecto con la obtención de datos, para ello creamos un json con todas las estaciones de metro en la que almacenamos: nombre de la estación, línea de metro, coordenadas (obtenidas gracias a google maps) y estaciones con las que tiene conexión.

Para desarrollar el algoritmo, necesitamos obtener g(n) y h(n) para poder así obtener f(n) y poder calcular el camino mínimo:

- Para el cálculo de g(n) implementamos la función get_distance_g, que se le pasan 2 estaciones como parámetros y nos devuelve la distancia real entre ambas estaciones (medido a través de google maps y añadido al json del que posteriormente obtenemos la información).

- Para el cálculo de h(n) utilizamos la librería geopy, que nos permite calcular la ortodrómica (arco de círculo máximo que corresponde a la distancia más corta entre dos puntos del globo).

Y obtenemos así f(n), distancia aproximada para llegar a una estación partiendo desde el origen:

$$f(n) = g(n) + h'(n)$$

Para la implementación del algoritmo hemos utilizado 2 listas: open_list y closed_list. En cada iteración, el algoritmo con el menor valor de f(n) es retirado de la open_list y se obtienen sus nodos vecinos (estaciones conectadas a dicho nodo) a partir de la función auxiliar get_neighbours. Los nodos vecinos se añaden a la open_list en caso de no estar y el algoritmo continúa con la siguiente iteración. El algoritmo finaliza cuando el nodo retirado es igual al nodo destino y obtenemos el camino mínimo a través de la función auxiliar return_path.
Hemos añadido una penalización de 5 minutos a la realización de los transbordos para que lo tenga en cuenta a la hora de escoger el camino mínimo.

3.Desarrollo de la interfaz gráfica

El desarrollo de la interfaz gráfica se ha realizado en Python 3.11, con la librería Tkinter.
En cuanto a los datos, se han utilizado, organizados en listas, los siguientes:

· descripción de las estaciones por línea
· tramos horarios
· días de la semana

y el fichero en formato png:

· Mapa de la red de metro de Atenas

Para ell diseño de esta interfaz se ha perseguido que sea sencillo, intuitivo y atractivo para el usuario final y que requiera un mínimo de pasos para su ejecución.

Dado que es una red muy pequeña con solo 54 estaciones; se ha optado por no utilizar el teclado para la entrada de datos y basarnos en el sistema de selección por listas desplegables, que apoyado con el plano de la red, que siempre aparece como fondo, nos va a permitir, por una parte, facilitar la entrada de datos al usuario al no tener que escribir, y por otra, que el chequeo de los datos de entrada sea mínimo para el desarrollador.
Se han implementado dos botones:
Calcular Ruta: chequea y valida los datos seleccionados para dar paso al algoritmo y
Salir: permite salir de la aplicación en cualquier momento.
La información que se muestra en las etiquetas sirve de guía para que el usuario sepa qué datos seleccionar en cada momento.
Sobre el fondo con el mapa de la red, lo primero que se le indica al usuario es "Selecciona origen" y " Selecciona Destino", es el momento de seleccionar las paradas donde quiere iniciar y finalizar el desplazamiento con los desplegables respectivos. Aparecen en blanco porque es un dato obligatorio que debe introducir para que se ejecute el cálculo de ruta, en caso contrario le daría un aviso para recordarle lo que debe hacer.
Las estaciones aparecen en los desplegables ordenadas por líneas, desde cabecera a terminal. Al tener presente el mapa de fondo y ser pocos datos se ha optado por no introducir otro paso intermedio de solicitar si quiere mostrar las estaciones por líneas o por orden alfabético.
Una vez introducidos los datos de las estaciones de origen y destino, se solicita seleccionar el día de la semana y el tramo horario. Estos desplegables están inicializados a "Lunes" y "De 6:10 a 20:30", la introducción de estos datos es opcional, si no se introduce nada, coje los datos mostrados y siempre vamos a obtener una respuesta.
Chequeos que realiza el interfaz de entrada:
Si la estación de origen y destino es la misma , esto incluye también que las dos estén en blanco, se muestra el mensaje "Origen y Destino no pueden ser la misma estación"
Si la estación de origen y/o destino no es una estación (es blanco o una descripción) se muestra el mensaje "Debe seleccionar el nombre de una estación origen y una estación destino"
Si la estación de origen y/o destino es el aeropuerto o una de las 4 anteriores al aeropuerto de la línea 3 y se ha seleccionado una franja horaria posterior a las 23:30 se muestra el mensaje "No hay servicio en la estación origen o destino a partir de las 23:30
Si el día de la semana es Lunes o Martes o Miércoles o Jueves o Domingo y se ha seleccionado una franja horaria posterior a las 24:00 se muestra

el mensaje “Solo hay servicio nocturno los viernes y sábados, los días laborables termina el servicio a las 24:00"
Después de mostrar el mensaje de aviso el usuario confirma que ha leído el mensaje y puede volver a introducir el dato o cerrar la aplicación.
Una vez introducidos los datos correctamente, los resultados obtenidos se pueden observar inmediatamente al pulsar el botón Calcular Ruta.

4. Pruebas

Se han chequeado los datos de entrada con todas las variaciones posibles al realizar el desarrollo. El resultado se ha comparado con la aplicación de metro implementada por la empresa de metro de Atenas y el resultado ha sido satisfactorio.

Posteriormente se ha probado la aplicación con personas ajenas al proyecto sin explicarles su funcionamiento y el resultado también ha sido satisfactorio.

5. Conclusiones

Ha sido un gran acierto el utilizar Python, pues nos ha permitido realizar modificaciones y
añadir nuevos requisitos de manera ágil y sencilla.
La aplicación al utilizar una estructura de datos, “estandarizada”, es exportable a otra red del mismo modo de transporte público.

6. Ejecución del programa

Dentro de la carpeta del programa hay un main.exe en el que haciendo doble clic lo ejecutaremos

7. Bibliografía

Portal de datos abiertos:

https://data-crtm.opendata.arcgis.com/search?collection=Dataset

https://data.europa.eu/data/datasets?catalog=open-data-greece&showcatalogdetails=true&locale=en&categories=TRAN&page=1

https://data.europa.eu/data/datasets/gis-athens-metro-line_-2_ext_piraeus?locale=en

https://docs.python.org/3.11/

https://www.atenas.net/metro

App Atenas