MEMORIA DE LA PRÁCTICA

Autores (Grupo 18):

Kévin Alberto López Porcheron
XXXXXXX

YYYYYYY

ZZZZZZZ

Escuela Técnica Superior de Ingenieros Informáticos
Inteligencia Artificial, Curso 2020/2021

Índice

Introducción

Esta práctica se basa en la implementación de un algoritmo de búsqueda A* y su aplicación a un escenario real de optimización: encontrar el camino más óptimo entre distintas estaciones del metro de Atenas. En nuestro caso, se van a considerar 3 parámetros:

- Tiempo
- Distancia
- Transbordos

Estos parámetros se utilizarán como pesos de las aristas. Además, todo esto estará almacenado en el fichero *aristas.json***.** Además, también almacenamos en *metro.json* la localización de las estaciones en la imagen y el radio, ya que para la interfaz lo que hacemos es dibujar círculos interactivos encima de la imagen; y por último, también tenemos el documento *coord.json* que usamos para almacenar la longitud y latitud de cada una de las estaciones.

Mas allá de realizar la implementación de este algoritmo, se pedía una interfaz gráfica; pero no se especificaba ninguna herramienta o entorno para su desarrollo. Decidímos utilizar Python por su simplicidad y su compatibilidad en distintos sistemas operativos. Además, tiene abundantes librerías, y aunque solo hicimos uso de una de ellas para la interfaz gráfica, consideramos que esto es una ventaja. Otros lenguajes que habíamos valorado eran Java, Kotlin, y C++. Es posible que C++ nos hubiera dado mayor eficiencia, pero creemos que en un proyecto tan pequeño la diferencia hubiera sido mínima. Adicionalmente utilizamos GitHub para realizar la implementación de distintas partes de forma paralela. Por último, también utilizamos los Virtual Environments de Python para permitir mayor flexibilidad.

Interfaz Gráfica

Tener una interfaz gráfica llamativa es fundamental para cualquier aplicación moderna. Por ello, decidimos desarrollar una interfaz lo más intuitiva posible, de forma que fuera tanto funcional como atractiva. Originalmente, estábamos debatiendo entre usar QT o TK, pero acabamos decidiéndonos por QT porque parecía más moderno y actualizado. Aquí se puede ver una imagen de la pantalla principal de la aplicación:

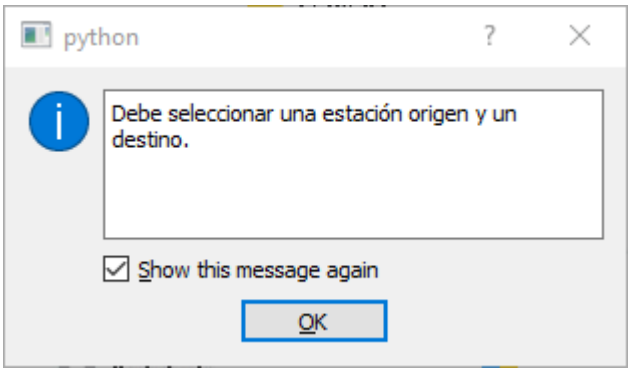
En esta imagen se pueden ver un par de detalles: por un lado, nos hemos esforzado para que los puntos y sus estaciones sean muy visibles, pero que no distraigan del resto de la aplicación; pero por otro lado también se puede ver que ofrecemos varias alternativas a la hora del funcionamiento de la aplicación.

Una persona puede seleccionar un punto en la imagen, y esto será visible y se verá reflejado en el resto de los campos de la aplicación:

Cómo se puede observar, hemos seleccionado la estación **Dafni**. Esto también se ve reflejado en el panel de la izquierda. Vamos a seleccionar el destino:

Como se puede apreciar, se ha seleccionado Doukissis Plakentias como destino. Esto también se ve reflejado en el resto de la aplicación. Si le damos al botón de buscar, el algoritmo se ocupará y nos dibujará todo el camino a seguir, y a la izquierda calculará el coste del camino en los tres parámetros. El criterio puede cambiarse en cualquier momento, y por lo tanto el algoritmo podrá encontrar distintos caminos en función de este:

Por último, también ofrecemos mensajes de error útiles cuando no se cumplen las condiciones para empezar la búsqueda:



Algorítmo A*

El algoritmo A* es uno muy usado para encontrar el camino más óptimo. Para implementarlo, utilizamos las siguientes estructuras de datos:

- Una lista donde se meten los nodos ya visitados que inicialmente contendrá el nodo origen.
- Una lista donde se meten los nodos ya evaluados.
- Un nodo temporal “actual” que será el nodo actualmente evaluado.
- Una cola donde se irá metiendo el camino óptimo. Al igual que en la primera lista, se creará con el nodo original. Se podría utilizar tanto un Stack como una Queue sin ningún problema.

Para evaluar cada nodo disponemos de dos funciones:

- $g(n)$: $Coste\ del\ Camino$
- $h(n)$: $Acumulación\ de\ g(n) + Distancia\ euclídea\ hasta\ el\ nodo\ destino$

Hemos tenido que realizar una laboriosa tarea de investigación para encontrar las coordenadas geográficas de cada estación en el mundo real para poder hacer la función de heurística$h(n)$.

Dificultades

Principalmente hemos tenido dificultades a la hora de utilizar el lenguaje de programación Python, ya que solo dos integrantes lo habían utilizado antes. Además, hemos tenido muchos problemas con PyQT5, ya que solo un miembro lo había utilizado anteriormente.

El usar Github nos ayudó bastante, ya que permitió que cada uno trabajara a su ritmo sin tener que organizar reuniones. En retrospectiva, estas reuniones hubieran sido útiles ya que en muchas ocasiones tuvimos problemas de comunicación.

También tuvimos problemas en el cálculo de la función de heurística ya que habíamos interpretado algunas cosas mal y solo nos dimos cuenta tarde.

Limitaciones

El programa tiene varias limitaciones. Por ejemplo, no es capaz de realizar la búsqueda de forma automática cuando se han seleccionado las dos estaciones. Además, solo es capaz de colorear las estaciones que conforman el camino del metro, pero no las aristas. Hacer esta labor no sería complicado, pero requeriría mucho tiempo para identificar la localización apropiada en la imagen de las aristas. Por último, la aplicación no contempla la posibilidad de que sea usada por gente daltónica, aunque dada lo relativamente común que es esta situación, no estaría de más ofrecer un modo para daltónicos. Siguiendo por esta línea de usabilidad, el metro es usado por miles de personas, incluidos turistas. Sería muy útil disponer de diversos idiomas para que cualquier persona pue