Metro de Buenos Aires Pathfinder

Samuel Caraballo Chichiraldi (Coordinador)
Andres Chen Luo
Iván Clemente Álvarez Del Río
Felix Garcia Taboada

Grupo 2M

Introducción:

Este trabajo tiene como objetivo desarrollar una aplicación capaz de determinar la ruta más corta en cuanto a tiempo entre un conjunto de estaciones del Metro de Buenos Aires. Para lograrlo, se ha utilizado el algoritmo A*.

Esta aplicación se ha desarrollado con una interfaz gráfica en un entorno web para permitir al usuario acceder con la mayor facilidad posible.

Descripción algoritmo utilizado:

Para encontrar el camino más óptimo entre dos estaciones cualesquiera se ha modelado el mapa del metro como un grafo ponderado en donde las estaciones son los vértices y los distintos trayectos, las aristas. Los pesos corresponden al tiempo estimado en ir de un nodo al otro según la velocidad promedio de la línea. En el caso de los transbordos se ha tomado la velocidad promedio de una persona andando (5 km/h) para calcular este tiempo. Sobre este grafo, se ha aplicado el algoritmo de A* para encontrar de forma eficiente el camino más corto en términos de tiempo.

El algoritmo A* es una extensión del algoritmo de Dijkstra que incorpora una función heurística para mejorar la eficiencia en la búsqueda de rutas. Su función de coste se define como:

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

Donde $g(n)$ es el tiempo acumulado en minutos desde el origen al nodo n y $h(n)$ la heurística.

En el desarrollo del proyecto, la implementación del algoritmo es muy parecida a la del algoritmo de Dijkstra de forma que $g(n)$ representa el tiempo acumulado desde el inicio hasta n. Por otro lado, la heurística mide el tiempo en minutos que se tardaría en ir desde la estación actual (n) al objetivo en línea recta según la velocidad máxima de todas las líneas en promedio. El objetivo de esto es minimizar el tiempo del trayecto sin importar la distancia.

Decisiones de diseño:

Se ha decidido implementar la solución en una página web ya que consideramos que era la forma más fácil de poder realizar un diseño elegante y a su vez conseguir la funcionalidad esperada.

Herramientas

Los lenguajes de programación utilizados fueron HTML, CSS y JavaScript. Para tener un diseño más moderno optamos por incorporar también TailwindCSS.

Decidimos también colocar un mapa interactivo (Leaflet.js) al cual le añadimos las estaciones del metro mediante marcadores y líneas. De esta forma el usuario puede comprender mejor cómo se encuentra el subte realmente distribuido además de ver qué puede encontrar en las inmediaciones de las distintas estaciones.

Para implementar el grafo que describe las estaciones de metro conseguimos un fichero con los datos de las estaciones y lo pasamos a formato JSON para procesarlo mejor.

Para implementar el algoritmo de A* utilizamos una cola de prioridad que almacena el menor valor de $f(n)$ encontrado hasta el momento. Los pesos del grafo y la heurística son calculados a la vez que se ejecuta el algoritmo de búsqueda. Para ello, con las coordenadas geográficas de cada estación obtuvimos la distancia con la fórmula de Haversine. Esta fórmula da con mas precisión la distancia entre dos puntos del globo al tener en cuenta la curvatura del planeta. Esta fórmula no proporciona una ganancia significativa con los posibles destinos a seleccionar pero esto mejorará la precisión de la aplicación si en un futuro se decidiera ampliar o escalar.

Estructura del proyecto

El archivo principal del proyecto es index.html. En él se encuentran los componentes de la interfaz de usuario. A partir de éste, se puede solicitar el cálculo de rutas, introducir estaciones de inicio y fin, navegar por el mapa, inspeccionar estaciones, etc…
A partir de este surgen distintos scripts:

route-description.js: Contiene una descripción debajo del botón del cálculo de ruta para indicar el resumen del trayecto e información adicional como distancia y tiempo entre estaciones contiguas y la acción a ejecutar.

path-finder.js: Ejecuta el algoritmo A* para buscar el camino óptimo en tiempo entre 2 estaciones.

script.js: Contiene lógica para controlar botones, llamar funciones, determinar estaciones de entrada, construir el grafo, cargar el mapa y dibujar los marcadores y líneas del metro.

suggestions.js: Contiene el desplegable de estaciones sugeridas al introducirlas de entrada o salida. Enseña las estaciones que empiezan por lo tecleado en el input.

styles.css: Contiene animaciones y algunos estilos para los componentes

Gestión de grupo:

Durante el proyecto consideramos que hemos estado muy motivados con el mismo porque hemos podido emplear nuestros conocimientos adquiridos durante el grado para poder hacer una aplicación con una funcionalidad real a diferencia de otros trabajos previos.

Aunque consideramos que tardamos en empezar con el trabajo creemos que aún así hemos alcanzado un muy buen resultado. También hubo algo de confusión al principio pues no teníamos del todo claro qué herramientas utilizar para implementar el front end y tuvimos que empezar de nuevo el proyecto cada vez que probábamos una nueva herramienta. Al principio estuvimos evaluando la opción de hacerlo en Python pero el diseño no quedaba bien.

Algo similar ocurrió hacia el final del proyecto pues no anticipamos que ciertas tareas llevarían mucho más tiempo. Esto provocó cuellos de botellas por lo que algunos integrantes del grupo no podían avanzar hasta que otro terminase.

Algo que ha contribuido al desarrollo del proyecto era que ya nos conocíamos todos previamente por lo que teníamos confianza suficiente entre nosotros para poner pequeños detalles divertidos para darle un toque personal al proyecto y nos ayudaba a motivarnos entre nosotros.

Conclusión:

La aplicación desarrollada cumple con los objetivos propuestos, permitiendo a los usuarios encontrar las rutas más óptimas en el Metro de Buenos Aires gracias a A*. Este proyecto fue una experiencia desafiante, combinando conocimientos técnicos y creatividad para resolver un problema real.

También consideramos que hemos aprendido mucho debido a que los requisitos dejaban margen de interpretación que nos invitaban a indagar y explorar soluciones propias a diferencia de otros trabajos en el grado; cuyas directrices son más estrictas y dejan poco margen para que los alumnos desarrollen enfoques distintos.

Consideramos que el producto final ha acabado siendo de calidad comparable a otras aplicaciones similares y resulta estéticamente atractivo al usuario. Tiene una interfaz muy intuitiva y un cálculo del camino rápido y escalable en caso de una posterior ampliación del proyecto.

Referencias:

1. JavaScript Priority queue: https://cdn.jsdelivr.net/npm/js-priority-queue
2. Imágenes y logotipos: https://fontawesome.com/
3. Mapa Interactivo Leaflet: https://leafletjs.com/
4. Datos del subte: https://data.buenosaires.gob.ar/dataset/?tags=subte
5. Chris Veness, www.movable-type.co.uk. (s. f.). Calculate distance and bearing between two Latitude/Longitude points using haversine formula in JavaScript. https://www.movable-type.co.uk/scripts/latlong.html

Anexo

Longitudes y velocidades medias de las líneas

Linea A: 10,8 km
Linea B: 11,8 km
Linea C: 4,4 km
Linea D: 10,6 km
Linea E: 11,9 km

Linea	Distancia (km)	Tiempo (min)	V (media) km/min	V (media) km/h
A	10.8	23	0.4695652174	28.17391304
B	11.8	23	0.5130434783	30.7826087
C	4.4	13	0.3384615385	20.30769231
D	10.6	26	0.4076923077	24.46153846
E	11.9	24	0.4958333333	29.75