Viajando en avión II

por
Pedro Latorre García
(CPEPA Gómez Lafuente, Zaragoza)

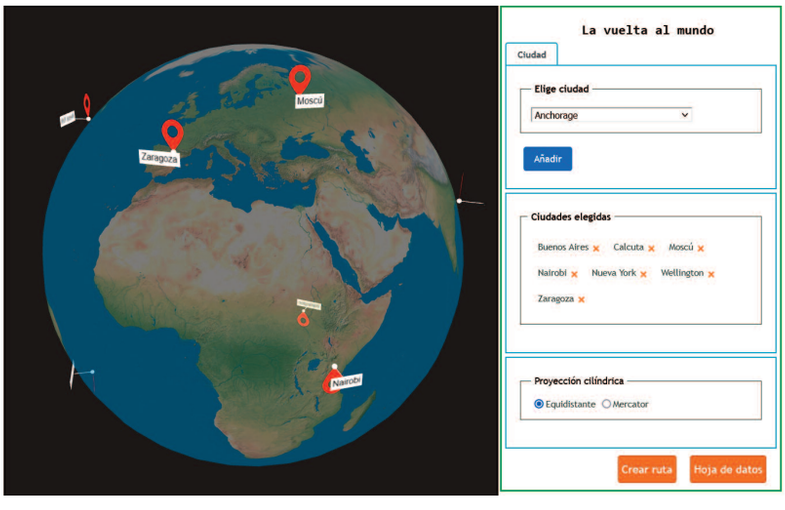
Viajando en avión es la última novedad del rincón de Petrus. Es una aplicación con una escena interactiva del globo terráqueo realizada con Three.js, una librería de código abierto escrita en el lenguaje Javascript que simplifica el uso de WebGL (Web Graphics Library). A grandes rasgos, WebGL es un conjunto de herramientas para renderizar gráficos en navegadores web. La aplicación ha sido probada con los navegadores Mozilla Firefox y Chrome en un PC y se recomienda una pantalla con una resolución mínima de 1440 × 900. Dos buenos amigos me han confirmado que no funciona en un Ipad.

Este artículo sería la continuación del escrito en marzo de 2018 en el que presentaba una versión alternativa del programa. En aquel momento los alumnos escribían las coordenadas geográficas de una ciudad en un archivo de texto e indicaban conexiones entre dos ubicaciones. Si la información estaba correctamente escrita, en una representación tridimensional de la Tierra aparecían las ciudades y caminos elegidos. El objetivo era acercar a los alumnos al maravilloso mundo del código fuente en detrimento de los omnipresentes interfaces gráficos. Más adelante desarrollaría esta idea con la aplicación Casitas en 3D.

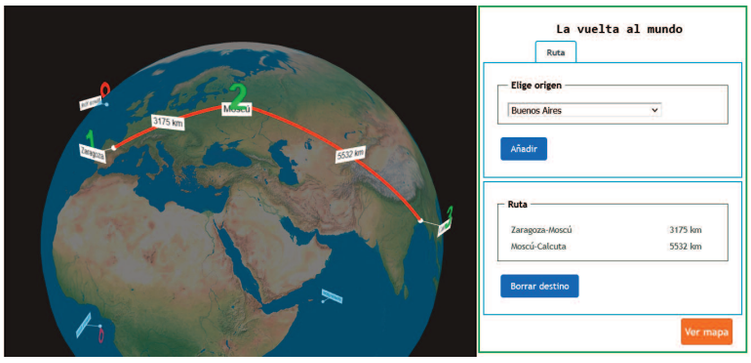
En la versión actual de la aplicación nos alejamos del modo texto. Queremos que los alumnos vean cómo son las trayectorias reales en una esfera y las comparen con su representación en un mapa. El programa está pensado también para trabajar el problema del viajante: Un viajero tiene que visitar una serie de ciudades, partiendo de un origen, visitar cada una de ellas exactamente una vez, y regresar al inicio. El objetivo es minimizar la distancia total recorrida.

Veamos un simple ejemplo de uso del programa con el ejercicio:

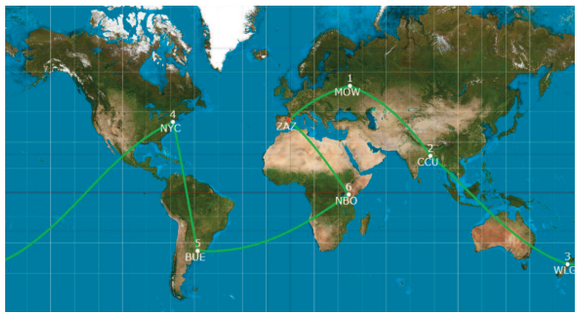
> Petrus vive en Zaragoza y quiere visitar las siguientes metrópolis: Moscú, Nairobi, Wellington, Nueva York, Calcuta y Buenos Aires. El viaje empieza y termina en Zaragoza. Le gustaría encontrar una ruta lo más corta posible que pase una vez por cada ciudad. Con ayuda del programa trata de encontrarla. Necesitarás una computadora que soporte gráficos 3D en el navegador web.



El primer paso es elegir las ciudades con el selector Elige ciudad y pulsando el botón Añadir. Según las introduces, aparecerán en el recuadro central. Si queremos quitar alguna, apretaremos la cruz roja adyacente. Seleccionaremos el tipo de proyección cilíndrica para el mapa, equidistante o Mercator. Cuando hayamos elegido todas las ciudades, con el botón Hoja de datos descargaremos una hoja Excel con las distancias entre las ubicaciones. El botón Crear ruta nos lleva al segundo paso.



Con el selector ELIGE ORIGEN vamos creando la ruta, la cual se irá mostrando en el recuadro inferior. Para eliminar el último tramo usaremos el botón BORRAR DESTINO. Además de la ayuda de la hoja de cálculo, podemos ver el mapa con su botón homónimo. Cuando hayamos completado el circuito, la aplicación muestra la longitud de la ruta mínima.



La aplicación indica que la longitud de la ruta de la imagen es 59 490 km y la de la óptima 51 484 km. Mirando en la consola de las herramientas para desarrolladores del navegador encontraremos que dicho camino es Zaragoza–Moscú–Nairobi–Calcuta–Wellington–Buenos Aires–Nueva York-Zaragoza.

El problema del viajante (TSP: *travelling salesman problem*)

Para estar seguros de encontrar la solución en un problema TSP es necesario comprobar todos los caminos posibles. En nuestro ejemplo hay 360 casos. En general, para n urbes el número de posibilidades asciende a $(n-1)!/2$, valor que crece muy rápidamente. Incluso con una modesta lista de ciudades para visitar, la cantidad de posibles recorridos se vuelve enorme. Como dice en el artículo del blog del Master of Science in Machine Learning & Data Science, Northwestern University:

> Los enfoques de fuerza bruta, que van probando todos los casos, son computacionalmente inviables y aún no se han descubierto soluciones exactas eficientes en tiempo polinomial. En un intento por abordar problemas difíciles como el TSP, los investigadores han desarrollado a lo largo de los años una amplia variedad de enfoques heurísticos. Si bien no se garantiza que alcancen un óptimo global, estas técnicas arrojarán un resultado casi óptimo con una cantidad limitada de tiempo y recursos disponibles.

En el problema expuesto en la bitácora, encontrar la ruta más corta que recorra las 30 ciudades con un estadio de la Liga de Béisbol Americana, existen $4{,}42 \times 10^{32}$ posibles caminos.

Comentemos un par de heurísticas muy sencillas. El algoritmo del *vecino más cercano* es el método más simple para crear una ruta. Es fácil de entender, pero no siempre encuentra la solución óptima: Desde la ciudad actual hay que dirigirse a la más cercana que todavía no se ha visitado. La hoja de cálculo servirá para buscar la siguiente ubicación del camino. Este algoritmo es una manera de introducir a los estudiantes en la resolución de problemas de optimización y pensamiento algorítmico. Al aplicarlo a nuestro ejemplo se consigue una ruta de 67 868 km, resultado bastante alejado del óptimo.

Un algoritmo más eficiente que me ha recomendado ChatGPT es el de *inserción más barata*. Se comienza con un ciclo pequeño (por ejemplo, tres ciudades) y para cada ciudad no visitada, hay que ir calculando d, la distancia de los dos tramos creados al colocarla en cada posición posible de la ruta actual. Se inserta la ciudad que minimice el valor de d en la posición del camino donde se obtenga dicho mínimo. Se repite el proceso hasta que todas las ciudades hayan sido colocadas en el circuito. En nuestro caso, obtenemos un camino de 60 008 km. Aplicar este método solo con la ayuda de una hoja de cálculo resulta un poco tedioso.

En algunos grupos de Educación Secundaria y Bachillerato sería interesante que los alumnos viesen cómo se implementan estos algoritmos en un lenguaje de programación como Python o Javascript.