Valoración personal

La clase `LectorArchivos` no tuvo gran dificultad ya que, como no sabríamos leer un archivo, se nos proporcionó el código y sólo había que copiarlo. Sólo tuvimos un pequeño problema porque en un principio, pedíamos espacio con `new` para albergar un array de objetos de la clase `Ciudad` pero eso provocaba que se creasen en ese momento, se solucionó con un array de punteros que pedirían espacio para crearlos después.

La clase `Ciudad` no supuso ningún problema ya que habíamos usado con bastante frecuencia una clase idéntica en Fundamentos de la Programación, la única mejora fue puramente de eficiencia ya que hasta ahora no conocíamos los punteros, y copiar un objeto de una clase es más costoso que copiar un puntero a dicho objeto.

- Óscar Bermúdez Garrido

El problema del viajante de comercio supone un reto difícil de tratar, ya que es conocido que el verdadero problema no es llegar a la mejor solución, sino a una que se acerque lo máximo posible, utilizando un tiempo razonable. En este caso se nos proporcionó la heurística a utilizar, la del vecino más cercano, que reduce el problema a ir saltando de una ciudad a la más próxima no visitada, y la implementación no fue muy complicada, gracias en parte a que es una heurística que obtiene una solución y termina, frente a otras que continúan mejorando la solución sin parar.

Durante la implementación, surgieron algunos problemas debidos al uso de punteros, ya que se utilizaron para pasar información de objetos de una clase a otra. También se generaban violaciones de segmento al utilizar *arrays* en memoria dinámica mal dimensionados (para la matriz de distancias en la clase `Problema`), lo que se solucionó ajustando la dimensión correctamente. La clase con más dificultad fue `Heuristica`, ya que había que tratar de gestionar los recorridos de ciudades adecuadamente y de forma rápida, para lo que también se necesitaba que las clases pequeñas (`Ciudad`, `Recorrido`) fuesen lo más eficientes posibles, por lo que se prefirió el uso de *arrays* en memoria dinámica frente a objetos de la clase `vector`. Además, tuvimos algunos fallos al programar la clase y en un principio generaba soluciones incorrectas, lo que se resolvió más tarde, observando el comportamiento de cada método y encontrando los errores.

Finalmente, el programa terminó por solucionar correcta y eficientemente las instancias propuestas, se añadió además un pequeño script en Bash que permite la generación del gráfico asociado a cada solución de forma automática.

Personalmente, creo que hemos conseguido un programa muy trabajado, que es capaz de solucionar la instancia más grande (`pr1002`, con 1002 ciudades) en menos de 10 segundos en un buen procesador, si bien es verdad que la gestión de punteros y la recepción de los mismos en los métodos podría ser más clara, y hay partes del código que pueden resultar complejas y difíciles de entender. Probablemente podríamos haber mejorado en eficiencia en la clase `Recorrido` (que borra y genera un array entero cada vez que se añade una ciudad), y tal vez podríamos mejorar la forma en que se muestra la información en la salida estándar. Por lo demás, creo que tenemos un buen proyecto para tratar el problema del TSP. Veremos cómo se comporta cuando se le añadan más heurísticas y si es fácil de modificar para mejorar la gestión de memoria dinámica.

- Francisco David Charte Luque