Conclusiones:

Un problema que todo programador debe resolver con la mayor cobertura de casos posible, finalmente ha sido completado satisfactoriamente pues llega a ser de una calidad comercial y realmente funcional, si bien no pudo demostrarse el día de su presentación, fueron por problemas externos al proyecto y una particularidad de una máquina desactualizada o no prevista con la mas básica indumentaria.

Pero el algoritmo es perfectamente funcional, capaz de cargar y guardar escenarios así como mostrar la ruta optima, siendo casos imposibles donde exista un sector que no tenga una ciudad asociada ó si esta no está enlazada de alguna manera con el inicio.

Se adquirió una gran cantidad de experiencia con respecto a las bases de datos de C++ al aprender a usar correctamente las señaladas evitando punteros vacíos o removiendo de forma incorrecta datos.

Comentarios al momento de hacer el código:

Ya me había enfrentado antes a este problema del viajero tradicional, abordado con conocimientos y recursos básicos sobre la introducción a programación. A pensar de que el programa era un relativo éxito para las demandas que cumplía, se quedó una cuestión sin resolver y es el concepto de las rutas falsas.

Un sencillo ejemplo, la opción A y B cuya ponderación es 1 y 4 respectivamente, a primeras luces se escogería el camino A, sin embargo esta desemboca a una ponderación mínima de 8, en contraste B tiene la opción de una ponderación de 1. Por lo tanto, la ruta marcada por el inicio A no será la correcta, pues cargará mucho más que el recorrido iniciado por B.

A esta ambigüedad de situaciones que pueden elevarse exponencialmente según las conexiones que tenga el mapa las acuñamos como enfrentamientos locales. Si se tiene una efectiva forma de llegar a dicho punto que reúne varios orígenes. Si siempre se tiene en cuenta la forma más rápida desde el origen otorgado por el usuario, no debería considerar todos los demás caminos que originarían las combinaciones de rutas.

Esto puede compararse con el método de “chop branch” es decir, cortar las ramas de un árbol descartando para su mayor eficiencia, siendo el peor de los casos una ruta óptima que termine recorriendo todo el mapa de la manera más económica.

Opté por el lenguaje de programación C++, puesto que es el mejor en cuanto a tiempos de ejecución y su inclusión del POO nos permitirá crear estructuras con métodos propios que facilitarán la implementación del algoritmo.

Se partió desde la intuición humana para resolver estos problemas, con ayuda de los grafos ponderados siguiendo y anotando los pasos usados hasta llegar a la condición propuesta, visitar cada sector. Allí se propuso el ciclo principal el cual debían seguir todas las rutas en curso y como estas debían competir por generar descendencia, si llegaban a ser superadas por otra con menor ponderación total, entonces debíamos descartar cada ruta heredera de este descarte.

Después de poder explicar la idea con pocas palabras, era turno de construir el pseudo-código y proponer las estructuras de datos. Así fue como surgieron las ciudades, aquellas que almacenarían su nombre y sección, así como los enlaces de su correspondiente precio y las rutas como el recorrido que sigue hasta cumplir su condición. Y también una clase administradora de datos nombrada Mapa, quien aportaría una referencia universal y la que plantearía los temas administrativos de las estructuras de datos.

Se pensó en listas doblemente ligadas y mapas para almacenar y acceder a la información, si bien ya se tenía mucha experiencia con estas estructuras, no venían con tantas facilidades como en Python o Java lo cual ocasionaría a futuro errores y retrasos, principalmente relacionados a punteros vacíos o accesos ilegales de memoria.

Se tuvo que modificar la forma de acceso y de eliminación de datos para evitar los mayores errores posibles pero el planteamiento raíz del algoritmo no se modificó en ningún momento.