*

Sistema eficiente de entrega domiciliara

Mateo Ramírez Hernández
Universidad Eafit
Medellín, Colombia
marami26@eafit.edu.co

Juan Camilo Echeverri S.
Universidad Eafit
Medellín, Colombia
jechev60@eafit.edu.co

†

RESUMEN

Este informe se encarga de proporcionar una solución a un popular problema de domicilios, para lograr esto, se buscarán diferentes soluciones e identificaremos cual será la más conveniente para cumplir el objetivo.

.

1. Introducción

El problema del domiciliario consiste en encontrar el camino más eficiente para completar un circuito entre un conjunto dado de punto sin pasar por el mismo punto dos veces, La importancia de este problema radica en su utilidad en muchos áreas, desde planificación y logística, hasta fabricación de microchips y Secuencia ADN.

En nuestro caso podemos pasar por el mismo lugar más de una vez, por lo tanto, la importancia principal del problema discutido en este informe es que es un paso adelante para lograr una solución eficiente para el domiciliario.

2. Problema

Trabajaremos en una variante del problema del domiciliario , es decir se encontrara el camino mas corto para visitar todos los vértices

3. Problemas similares

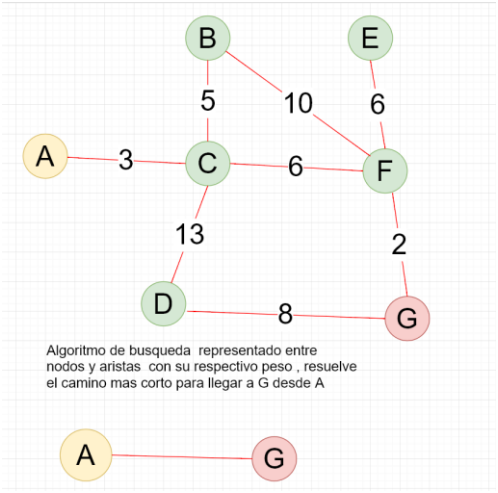
3.1 Una formulación equivalente en términos de Teoría de grafos es: dado una grafo ponderado completo (donde los vértices representan las ciudades, las aristas representan los caminos y los pesos son el costo o las distancias de estoscaminos), encontrar un ciclo de Hamilton con menor peso.

3.2 Las restricciones de retorno a la ciudad de comienzo no cambia la complejidad computacional del problema, vea Problema del camino de Hamiltoniano.

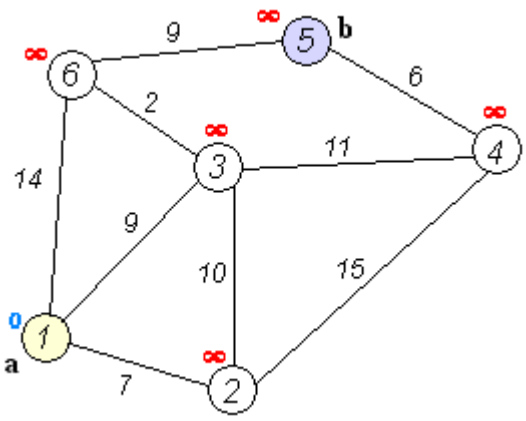
3.3 El problema de ordenamiento secuencial trata con el problema de visitar un conjunto de ciudades donde se tiene en cuenta las relaciones de precedencias entre las ciudades.

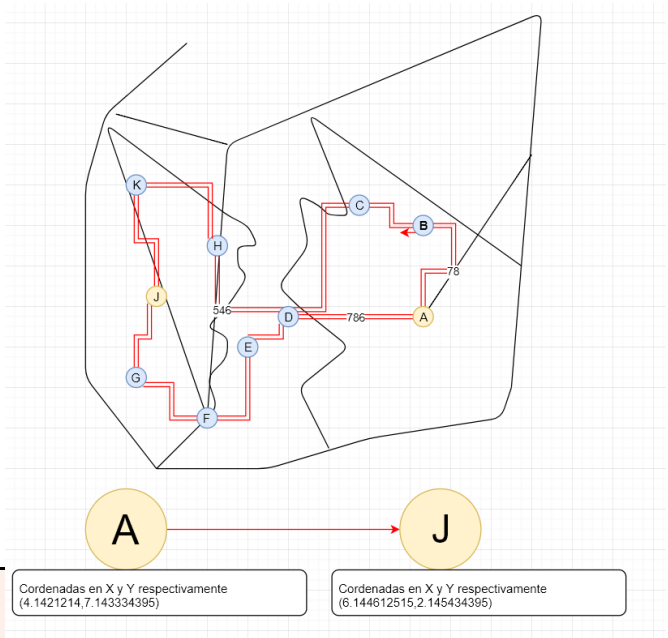
3.4 El problema del viajante comprador trata con un comprador que está cargado con un conjunto de productos. Él puede comprar estos productos en varias ciudades, pero tienen diferentes precios y no todas las ciudades ofrecen los mismos productos. El objetivo es encontrar una ruta entre un subconjunto de ciudades, los cuales minimicen el costo total (costo de viaje + costo de la compra) y habilite la compra de todos los productos requeridos.

4. Diseño de la estructura de datos



5. Diseño de las operaciones de la estructura de datos





6. Calculo de las operaciones de la estructura de datos

Método	Complejidad
Dijkstra	O(ml og v)
Coordenadas ingresadas	O(n + 1)
Complejidad total	O((n+1)*mlogv))

7. Criterios de diseño de la estructura de datos

Para el desarrollo del problema, al haber discutido varias opciones, se concluyo que la mejor manera era una solución basada en el algoritmo de Dijkstra, ya que tiene un orden de complejidad y presenta una solución eficiente en cuanto a camino entre dos nodos.

Se plantearon tener otro tipo de soluciones pero fueron descartadas por la ineficiencia que ellas presentan, por ejemplo es el caso de fuerza bruta que tiene una complejidad de O(n!) y de programación dinámica que tiene complejidad de Ô(n^2 2^n)

8. Diseño del algoritmo

A continuación, se muestra un ejemplo de cómo funciona el algoritmo aplicando Dijkstra de un nodo a otro

Es decir de A a B se aplicó el algoritmo Dijkstra y el camino que se muestra en rojo es el camino mas corto sea redundante para ir de A a B, este algoritmo se emplea de nodo a nodo para resolver el problema de cual seria el camino mas corto de A hasta J

9. Calculo de la complejidad del algoritmo

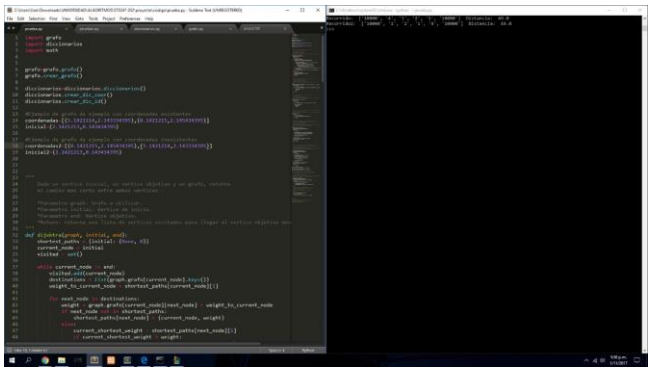
Mejor caso	O(m log v)
Caso promedio	O((n+1)*m log v)
Peor de los casos	O((n+1)*m log v)

10. Criterios de diseño del Algoritmo

Para el desarrollo del problema, al haber discutido varias opciones, se concluyo que la mejor manera era una solución basada en el algoritmo de Dijkstra, ya que tiene un orden de complejidad y presenta una solución eficiente en cuanto a camino entre dos nodos.

Se plantearon tener otro tipo de soluciones pero fueron descartadas por la ineficiencia que ellas presentan, por ejemplo es el caso de fuerza bruta que tiene una complejidad de O(n!) y de programación dinámica que tiene complejidad de Ô(n^2 2^n).

En Python se uso un diccionario dentro de un diccionario



11. Resultados obtenidos

	Conjunto de datos 1 (3 coordenadas)	Conjunto de datos 2 (4 coordenadas)	Conjunto de datos 3 (6 coordenadas)	Conjunto de datos n (mas de 8 Coordenadas)
Mejor caso	2.0713 seg	2.7588 seg	48.248 seg	87.631 seg
Caso promedio	2.07945 seg	2.7643 seg	48.517 seg	90.752 seg
Peor caso	2.08245 seg	2.7767 seg	50.973 seg	118.730 seg

	Conjunto de datos 1 (3 coordenadas)	Conjunto de datos 2 (4 coordenadas)	Conjunto de datos 3 (6 coordenadas)	Conjunto de datos n (mas de 8 Coordenadas)
Mejor caso (bytes)		183427072	184217600	184545280
Caso promedio (bytes)		183537664	186326178	187635212
Peor caso (bytes)		183541760	190021213	190127262

12. Conclusiones

- El resultado fue lo que se esperaba, tener una eficiencia a la hora de resolver el problema
- Al realizar las pruebas de ejecución respecto al tiempo pudimos observar que por mayor que sean las coordenadas, si están muy cercas entre si , el tiempo de ejecución es muy bajo respecto a coordenadas de mayor distancia
- Al realizar la practica nos encontramos con problemas manejables como por ejemplo la ineficiencia del algoritmo en algunos casos