Instituto Tecnológico y de Estudios Superiores de Monterrey
Análisis y diseño de algoritmos avanzados
TC2038
Grupo 604
Actividad Integradora 2
Reflexión de trabajo
Andrea Catalina Fernández Mena
A01197705
Catedrático:
Dr. Eduardo Arturo Rodríguez Tello

En esta actividad integradora implementamos diversos algoritmos para la comprensión sobre la transición de rutas basado en nodos como representación de conexiones, en el caso de este proyecto ejemplificando a una compañía de internet. Solicitando conocer los puntos de conexión mínima entre colonias/puntos de conexión, identificando las rutas de transición más cortas, así como determinar cuál es el máximo ancho de banda que se puede obtener entre dos colonias determinadas.

En mi caso, yo me enfoqué en el problema 1: conocer los puntos de conexión mínima entre colonias/puntos de conexión donde dada con la lectura de un grafo no dirigido, ponderada y conectada, decidí utilizar el algoritmo de Prim ya que es uno de los algoritmos más eficaces dentro de la teoría de grafos pues permite encontrar el *minimum spanning tree* dentro de un grafo no dirigido, ponderado, conexo y con aristas etiquetadas.

Dentro del proceso de desarrollo de dicho algoritmo cree dos una función denominada Prim la cuál tenía herencia de la clase grafo, así como un método llamado nextHop para representar la movilización de un grafo a otro y la función llamada missedVisits la cuál se encargaba de revisar si aún continuaban existiendo nodos dentro del árbol que permitan al algoritmo principal como función seguir iterándose.

Con la ayuda de estos dos métodos, la función realiza una visita por todos los nodos y sus posibles recorridos generando una serie de combinaciones sobre todos los caminos posibles y dictaminando así aquellos que representan los direccionamientos que impliquen el mínimo número de movimientos de expansión.

Basándome en la estructura y lógica estructural del algoritmo pude identificar que la complejidad de tiempo con la que cuenta es **O((V + E) log V)** , o bien simplificado como **O(ElogV)** siendo una complejidad eficiente, no obstante, podría ser mejorada, una opción que se me ocurre para mejorar su complejidad de tiempo y poder diseñar un modelo más rápido podría ser a través de un algoritmos como Fibonacci Heaps como una técnica de búsqueda dentro de los mismos nodos para una lectura y diagnóstico de búsqueda más rápido en recorrido dentro de los nodos y logrando convertirlo como mínimo en un algoritmo de complejidad O(n) en su tiempo, lo cuál podría ser una importante acción a considerar en futuras implementaciones.

Por otro lado, en cuánto a la complejidad de espacio de mi algoritmo de Prim diseñado, lo determiné como **O(E+V)**, la cuál sigue siendo una complejidad de eficiencia muy buena dentro de estándares posibles. Puede intentarse mejorar con el fin de que la misma se convierta en un algoritmo de orden O (1), no obstante, su estructura tendría que ser extremadamente retrabajada ya que existen muy pocos algoritmos dentro de esa estructura.

En general, fue una actividad integradora que representó un reto importante para mi aprendizaje y que me ayudó a reforzar muchos de los conocimientos vistos en clase dentro de una perspectiva mucho más práctica y útil para mi formación profesional. Fue un poco más compleja que la situación problema previa, pero esto mismo ayudó a reforzar de mejor manera los conocimientos previos junto con algunos temas y conceptos sobre métodos de búsqueda vistos en otras clases.

Fuentes bibliográficas:

Bari A. [Abdul Bari] (2018). 3.5 Prims and Kruskals Algorithms - Greedy Method [Video]. Youtube. https://youtu.be/4ZlRH0eK-qQ

Prim's Minimum Spanning Tree (MST) | Greedy Algo-5 (De GeeksforGeeks). (2022, 22 agosto). GeeksforGeeks. https://www.geeksforgeeks.org/prims-minimum-spanning-tree-mst-greedy-algo-5/