Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Monterrey

Febrero - Junio 2023

Análisis y diseño de
algoritmos avanzados
Grupo 602

Reflexión Actividad Integradora 2

Alta demanda para los Proveedores
de Servicios de Internet (ISP)

Alumno:

A01652327 Diego Esparza Hurtado

Profesor:

PhD. Oscar Márquez Calvo

Fecha:

29 de mayo de 2023

El año 2020 marcó un cambio significativo en la transición de actividades presenciales a remotas debido a la pandemia ocasionada por el COVID-19. A causa de dicho acontecimiento, prácticamente en todo el mundo se tomó la medida de mantener en casa a la mayor cantidad posible de gente, por lo que la contratación, uso y demanda de servicios de internet (Internet Service Provider-ISP) tuvo un rol muy importante.

Gracias a esto se pudo observar la importancia de tener un cableado óptimo entre localizaciones para alcanzar a llegar a distintos lugares con el mínimo cable posible, así como el mejor camino en caso de tener que visitar todas las localizaciones y también conocer la cantidad máxima de datos que puede ser intercambiada entre cada uno de los nodos principales. De igual manera, es de gran valor para los proveedores el conocer la facilidad y factibilidad de conectar a la red una nueva localidad.

Es por esto que en esta situación problema se buscan encontrar dichos datos (cableado y distancia hamiltoniana óptimos, flujo de información máximo de un nodo a otro, y factibilidad de conectar a la red nuevos puntos). Para lograr hacer esto, se recibe un archivo de entrada con la información de un grafo representado en matriz de adyacencia con grafos ponderados, donde el peso de cada arista es la distancia en kilómetros entre colonias donde puede ser implementado un cableado. También se cuenta con otra matriz que representa la capacidad máxima de transmisión de datos entre la colonia i y la colonia j.

I. Lectura del archivo:

Para realizar la lectura de cada uno de los archivos y obtener su contenido, se generó el método getFileText para recuperar cada una de las matrices que representan el grafo ponderado de distancias y el grafo dirigido de flujos. De igual manera, se recuperan los puntos que se utilizaron para generar el diagrama de Voronoi. Dicho algoritmo cuenta con una complejidad de $O(n^2)$, donde ‘n’ es igual al número de nodos, ya que está recuperando matrices.

II. Primera parte: Problema del árbol de mínima expansión (Minimum Spanning Tree MST) Cableado óptimo de colonias con fibra.

Con respecto a la obtención del MST, en dicho problema se busca conectar todos los vértices juntos, sin ciclos y con la mínima suma de pesos en los arcos. Para esto, se implementó el algoritmo de **Kruskal** cuya complejidad es **O(E log(E))**, donde E es el número de aristas con las que cuenta el grafo ponderado. Lo que hace el algoritmo es buscar la arista con menor peso para que al agregarla no forme un ciclo en el MST. A diferencia de Prim, en este algoritmo la arista a conectar no necesariamente tiene que estar conectada en dicho momento al MST.

En este caso, la implementación imprime la arista que va agregando hasta completar el MST y luego imprime la distancia total, es decir, la cantidad óptima de cable de fibra óptica para poder compartir información entre cualquiera de las colonias.

III. Segunda parte: Travelling Salesman Problem (TSP) Ruta a seguir para recorrer todos los nodos (Ciclo Hamiltoniano)

Con respecto a la ruta óptima a recorrer por parte de quien realizará la entrega de la correspondencia, es necesario pensar en el Travelling Salesman Problem (TSP), que dada una lista de colonias y sus distancias, busca determinar la ruta más corta posible donde únicamente se visita cada colonia exactamente una vez y se regresa a la colonia origen. A este camino se le conoce como *Hamiltonian cycle*.

En este caso, la implementación para encontrar el *Hamiltonian cycle* más corto fue el método de **fuerza bruta**, el cual revisa y genera todos los ciclos para poder determinar cuál de ellos es el más corto. Esto implica que la complejidad es de **O(n!)**, ya que para cada nodo en cada paso, se puede luego elegir entre el número de nodos disponibles hasta ese paso, por lo que la fórmula queda de la siguiente manera: $n! = n(n-1)(n-2)\ldots(1)$. Existen otras implementaciones como **Branch and Bound**, pero en el peor de los casos sigue contando con una complejidad de **O(n!)**.

En este caso, la implementación genera y almacena en un vector los nodos del *Hamiltonian cycle* más corto en orden en el que fueron visitados y posteriormente se imprime la distancia mínima de dicho camino.

IV. Tercera parte: Problema del flujo máximo (Flujo máximo entre un nodo origen y otro destino)

El problema del flujo máximo sucede cuando se tiene un grado dirigido y se quiere obtener el máximo flujo que puede ser trasladado de un punto de origen a un punto de destino. En este caso, dicho traslado o intercambio entre un punto de origen (colonia 0) al punto destino (colonia 9) se realizó con la implementación del algoritmo de Dinic, cuya complejidad es $\mathbf{O(E*V^2)}$.

En dicha implementación de Dinic, se hace uso de Breadth First Search (BFS) y Depth First Search (DFS) para identificar por niveles y profundidad los caminos de flujos que pueden llegar desde el nodo origen hasta el nodo destino, que en este caso son las colonias previamente mencionadas. Esto permite observar la cantidad máxima de información que puede compartirse entre la colonia 0 y la colonia 9. Finalmente, en el programa únicamente se imprime dicho valor de flujo máximo entre la colonia origen y destino.

V. Cuarta parte: Diagramas de Voronoi

Voronoi permite identificar las zonas donde un punto que caiga en ellas se encuentra más cerca de la estación a la que pertenece dicho polígono de Voronoi, por lo que es un algoritmo de gran importancia en las telecomunicaciones, ya que permite volver eficiente tanto el cableado, como la conexión de dispositivos que requieren ser cableados desde una central. En nuestra implementación se utilizó la Triangulación de Delaunay y posteriormente la creación de los polígonos de **Voronoi**. La complejidad de la implementación es $O(\mathbf{n^4})$, donde ‘n’ es la cantidad de puntos o centrales. Se imprimen en orden ascendente con respecto a su coordenada ‘x’ más pequeña y solamente los que se encuentran dentro del rectángulo delimitado por los ‘x’ y ‘y’ mínimos y máximos.

Bibliografía:

González Guerra, L. H. (s.f.). Algoritmos: análisis, diseño e implementación. Monterrey, Nuevo León, México: Editorial Digital Tecnológico de Monterrey

Márquez, O. (2023). Curso: *Análisis y diseño de algoritmos avanzados.* Presentación Power Point.