OBSERVACIONES RETO 4

Integrantes del grupo

Grupo 11: **Federico Melo Barrero**

202021525

f.melo@uniandes.edu.co

Juan Camilo Prieto Avella

201814815

Jc.prietoa@uniandes.edu.co

Preguntas de análisis

a) ¿Cuántos grafos se necesitan definir para solucionar los requerimientos del reto? y ¿Por qué?

Para este reto se necesitan dos tipos de grafos para solucionar las preguntas planteadas en los requisitos, que son el dígrafo con la totalidad de los aeropuertos y un grafo no dirigido que tiene solamente los aeropuertos y las rutas que tengan tanto una ruta de ida entre los dos aeropuertos como uno de vuelta. Se habla con mayor profundidad de cómo son estos grafos en el punto siguiente. Estos son los únicos grafos necesarios porque los requerimientos nos piden realizar diferentes algoritmos de búsqueda sobre estos dos tipos de grafos. Por ejemplo, usaremos algoritmos como Dijkstra para encontrar la menor distancia entre aeropuertos pero hay requerimientos como el 4 en el que no estamos seguro que tipo de algoritmo nos pueda servir porque aún no hemos terminado de ver esto en las clases magistrales.

b) ¿Cuáles son las características específicas de cada uno de los grafos definidos? (vértices, arcos, denso o disperso, dirigido o no dirigido).

Se definen dos grafos:

El grafo 'digraph' es un dígrafo (en otras palabras, un grafo dirigido) en el cual se incluyen la totalidad de los aeropuertos (airports_full.csv) y las rutas dirigidas especificadas en el archivo full_routes.csv. Ese grafo cuenta con 9075 vértices, que son los aeropuertos con abreviación AITA única, y cuenta con 50231 arcos, que son la rutas aéreas dirigidas únicas. Nótese que no todas las rutas del archivo son únicas porque hay múltiples aerolíneas que hacen la misma ruta. Si en el dígrafo distinguiéramos las rutas por aerolínea, serían 92605. La densidad del grafo está dada por el número de arcos del grafo sobre el número total de arcos que puede tener, $v(v-1) = 9075\ (9074) = 8.235 \times 10^7$. Es decir, la densidad del grafo es

$$densidad = e / (v(v-1))$$

$$densidad = 50231 / (9075\ (9074))$$

$$densidad = 6.111 \times 10^{-4}.$$

Tomando en cuenta que si un grafo cuenta con una densidad mayor o igual a 0.75 se considera denso y si cuenta con una densidad menor a 0.3 se considera disperso, se evidencia que el grafo 'digraph' es disperso.

El grafo 'graph' es un grafo no dirigido en el cual se incluyen solamente los aeropuertos y las rutas que tengan tanto una ruta de ida entre los dos aeropuertos como una de vuelta. Ese grafo cuenta con 3473 vértices, que son los aeropuertos con abreviación AITA única que tienen ruta de ida y de regreso con algún otro, y cuenta con 22090 arcos, que son la rutas aéreas bidireccionales únicas. La densidad del grafo se calcula al igual que antes:

$$densidad = e / (v(v-1))$$

$$densidad = 1.832 \times 10^{-3}.$$

Usando el mismo criterio que antes, se observa que el grafo 'graph' es disperso.

c) Además de los grafos, ¿Qué otras estructuras de datos adicionales se necesitan para resolver los requerimientos? Y ¿Por qué?

Ademas de los grafos vamos a tener tablas de hash tipo lineal probing porque estas estructuras nos ayudan a encontrar de manera rapida (O(1)) los aeropuertos con los que se conecta un aeropuerto en especifico porque va a tener como llave los aeropuertos que existen y su valor es un array list con los aeropuertos que tiene algun vertice y su peso dependiendo del grafo. Las estructuras de datos adicionales que implementamos para facilitar la resolución de los requerimientos son las siguientes:

routes: Tabla de hash que tiene como llaves las abreviaturas AITA de cada aeropuerto y como valores arreglos con todos los otros aeropuertos que son posibles destinos de ese.

airports: Tabla de hash que tiene como llaves las abreviaturas AITA de cada aeropuerto y como valores el diccionario con la información del aeropuerto.

cities: Tabla de hash que tiene como llaves los nombres de las ciudades y como valores el diccionario con la información de la ciudad.

loaded: Estructura que almacena qué aeropuerto se cargó primero a cada grafo y qué ciudad se cargó última.

components: Almacena la información de los componentes conectados.

paths: Estructura que almancena los caminos de costo mínimo desde un vértice determinado a todos los otros vértices del grafo.

Como en la clase magistral no hemos visto aún la totalidad de los algoritmos sobre grafos, no sabemos si quizás pueda resultar conveniente hacer alguna estructura más para ayudar con la resolución de los requisitos, pero por ahora estas son las únicas estructuras adicionales que utilizaremos.