Isai Daniel Chacón Silva – 201912015 - Correo: id.chacon@uniandes.edu.co

Nicolás Aparicio Claros – 201911357 – Correo: n.aparicioc@uniandes.edu.co

Estructuras de datos y algoritmos

Reto 4

Grafos

En este documento se realiza el análisis de complejidad para cada uno de los algoritmos implementados para resolver cada requerimiento del reto 4 de la materia Estructuras de datos y algoritmos. En primer lugar, vale recalcar que el análisis teórico se realizó con la notación Big O, el cual es el peor caso posible para cada uno de los algoritmos realizados, sin embargo, este no ocurre siempre en la práctica y varía dependiendo en la distribución de las muestras de datos, en este caso de rutas de aeropuertos.

Para realizar los requerimientos de manera adecuada, se siguió la propuesta de generar 2 grafos, uno dirigido y otro no dirigido, de modo que se pudieran correr todos los algoritmos independientemente de si el grafo era inicializado como dirigido o no.

Análisis de complejidad

Requerimiento 1

La complejidad de este requerimiento viene dada principalmente por la función *interconexión(catalog)*. En esta se recorre la lista de vértices del grafo dirigido O(V) para formar un árbol binario de conexiones, cuyo *key* sea la suma de los términos *indegree* y *outdegree*. En caso de que una *key* se repita, se anexa al *value*, pues esta se encuentra definida como un ArrayList. Una vez se construya el mapa ordenado se itera sobre el valor obteniendo su máximo, lo cual es sencillo pues el árbol se encuentra ordenado O(logV), y se añade a una nueva lista hasta que esta posea los 5 elementos requeridos en el *view*. De modo que en total se tiene una complejidad de O(V).

Requerimiento 2

Para este requerimiento se llama a la función *clusters(catalog, iata1, iata2)*. Esta función corre el algoritmo de Kosaraju para obtener los componentes conexos del digrafo. Luego, simplemente se retornan los componentes conexos del grafo y si para 2 IATAs que entran por parámetro una variable booleana que indica si se encuentran dentro de un mismo clúster. Este requerimiento posee la misma complejidad temporal de Kosaraju, la cual es O(V+E) ya que se tiene una representación del grafo bajo listas de adyacencia.

Requerimiento 4

La complejidad temporal de este algoritmo viene dada por buscar obtener un árbol de recubrimiento MST mediante el algoritmo eager de Prim, la cual es del orden de O(ElogV). Luego, se obtienen los arcos de dicho árbol y se itera sobre sus vértices, A y B respectivamente, para añadirlos a un nuevo grafo, de modo que no se sobrescriban los datos. Esto conlleva una complejidad de O(E) en el peor caso, para el cual se tengan en cuenta todos los arcos. Una vez se construye el nuevo grafo, este es recorrido mediante el algoritmo de búsqueda en profundidad DFS, el cual incurre en una complejidad de O(V+E). Así, la complejidad total del requerimiento vendrá dada por los términos más significativos, es decir, O(V+E).

Requerimiento 5

En el requerimiento 5 obtener el número de rutas, arcos, indegree y outdegree para cierto vértice o IATA son consultas de tiempo constante O(1). Posteriormente se obtienen los adyacentes del grafo y se recorren dado que podría haber vuelos entre 2 aeropuertos para diferentes aerolíneas, las cuales se ven afectadas por el cierre del aeropuerto. Recorrer esta lista de adyacentes es O(V) en el peor caso, para el cual un aeropuerto se encuentra interconectado con todos los demás aeropuertos del mundo.