Reto 4: Documento de análisis

Integrantes del grupo

Grupo 11: **Federico Melo Barrero.**
202021525.
f.melo@uniandes.edu.co

Juan Camilo Prieto Avella.
201814815.
jc.prietoa@uniandes.edu.co

Análisis de complejidad de los requerimientos

Requerimiento 1

El requrimiento 1 utiliza básicamente dos operaciones: el indegree y el outdegree. El outdegree lo obtiene directamente de la lista de adyacencias y básicamente consiste en hacer un get de la tabla de hash, a causa de lo cual presenta una complejidad temporal de aproximadamente O(1).

Mientras tanto para el grafo no dirigido se saca solamente el tamaño de adyacentes que tiene un vertice desde la tabla de hash con un costo temporal de O(1) y se sabe cual es su importancia para la interconexion del grafo, luego se hace una busqueda O(N) sobre los vertices para encontrar los 5 vertices mas importantes para la interconexion.

Requerimiento 2

El requerimiento 2 hace uso del algoritmo de Kosaraju, cuya complejidad es de O(V+E), para hallar los componentes fuertemente conectados (SCC).

Requerimiento 3

El requerimiento 3 primeramente se ocupa del problema de las ciudades homónimas. Para ello usa el métoco contains sobre una tabla de hash de ciudades que tiene una complejidad de O(n), pues revisa si la ciudad existe entre todas las llaves de la tabla de hash. Si sí, devuelve una lista de ciudades homónimas de la cual el usuario puede selccionar la ciudad adecuada.

Posteriormente, el requerimiento 3 utiliza el algoritmo de Dijkstra para solucionar el problema de las cercanías. Recuérdese que el algoritmo de Dijkstra implementado con una cola de prioridad orientada al menor presenta una complejidad temporal de O(V + E).

El requerimiento busca los aeropuertos en cuadrados de 0.01 grado de latitud o longitud, aproximadamente 10km, usando la fórmula del semiverseno. Los busca en un árbol de ciudades por latitud cuyos valores son árboles de ciudades por longitud. Eso hace que sacar los rangos de ciudades en ciertas latitudes y longitudes siempre tenga complejidades logarítmicas, O(log(n)).

Requerimiento 4

El requerimiento 4 encuentra un mínimo árbol de recubrimiento del grafo no dirigido haciendo uso del algoritmo Prim en su versión eager. Dicho algoritmo presenta una complejidad temporal de O(E log(V)).

Requerimiento 5

Primero el requerimiento 5 llama el tamaño de vertices y de arcos que tiene el grafo dirigido y el no dirigido que se encuentra guardado en las estructuras de datos, por lo tanto tiene un costo de O(1). Despues para el grafo no dirigido se saca los vertices adyacentes del aeropuerto que se escriba para saber cuales seran los aeropuertos afectados y el tamaño de estos, para hacer este paso se tiene un costo temporal de O(1) por lo que los vertices adyacentes se tienen guardados en una tabla de hash con la que se realiza un get en O(1).

Por el otro lado, para el grafo dirigido se realiza dos operaciones: el indegree y el outdegree. El outdegree lo obtiene directamente de la lista de adyacencias y básicamente consiste en hacer un get de la tabla de hash, a causa de lo cual presenta una complejidad temporal de aproximadamente O(1) y el outdegree se consigue de una manera similar al al indegree. Con estos dos valores ya conocemos cuantos aeropuertos se ven afectados por el cierre de dicho aeropuerto.