**Grupo 12 Sección 07**

**Juan Andres Bernal – 202110848**

**Juan Esteban López – 202021417**

**Análisis del RETO 4 EDA Sección 7**

**Carga de datos:**

Para el Reto 4, empleamos distintas funciones para la carga de datos, iniciando con la creación de cuatro mapas, los cuales van a tener por índice, ciudades, aeropuertos, diferenciados como por nombre y código IATA, adicionalmente los dos grafos solicitados por el enunciado del reto.

 catalogo['Map'] = mp.newMap(numelements=3971,

                                     maptype='PROBING',

                                     comparefunction=None)

    catalogo['MapAirports'] = mp.newMap(numelements=3971,

                                     maptype='PROBING',

                                     comparefunction=None)

    catalogo['City'] = mp.newMap(numelements=3971,

                                     maptype='PROBING',

                                     comparefunction=None)

    catalogo['MapAirportsIATA'] = mp.newMap(numelements=3971,

                                     maptype='PROBING',

                                     comparefunction=None)

    catalogo["RouteGraphD"] = gr.newGraph(datastructure='ADJ\_LIST',

                                              directed=True,

                                              size=3971,

                                              comparefunction=None

                                              )

    catalogo["RouteGraphNoD"] = gr.newGraph(datastructure='ADJ\_LIST',

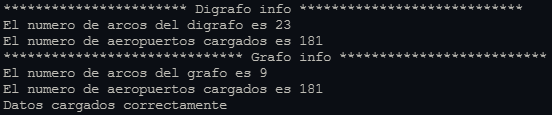
                                              directed=False,

                                              size=5,

                                              comparefunction=None

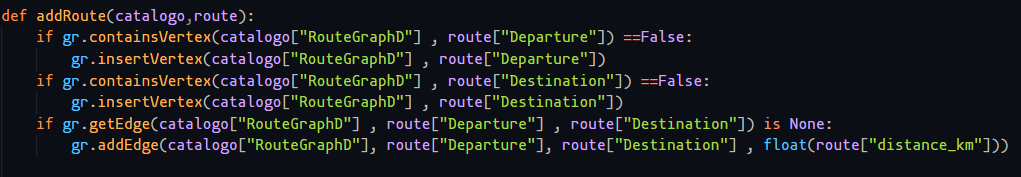
                                              )

**Output tras ejecutar la carga de datos con los archivos -small:**

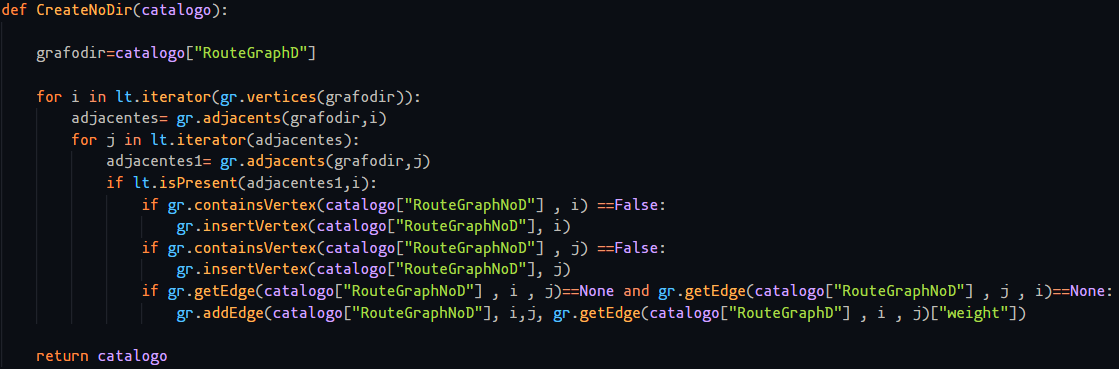
****

La carga de datos y creación de datos la realizamos teniendo en cuenta si un vértice pertenecía o no ya al grafo creado y partiendo de la condición de que el arco entre vértice de salida y llegada no existía, lo creábamos, realizamos el grafo no dirigido a partir del grafo dirigido obteniendo un total de 9 arcos.

**Creación dígrafo**

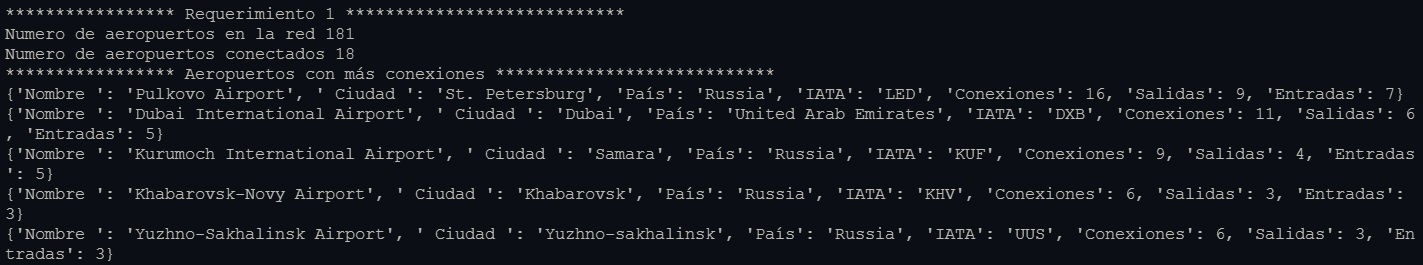
****

**Creación grafo:**

****

**Requerimiento 1 (O(N)):**

El requerimiento 1 lo hicimos a partir un mapa creado en el mismo requerimiento, donde se va a contar la cantidad de entradas y salidas que posee cada aeropuerto y realizar la sumatoria de la misma, y finalmente filtrar los datos.



**Requerimiento 2(O(N)):**

Usamos el algoritmo de Kosajaru para hallar el número de componentes conectados con su método connectedComponents y seguidamente, gracias al método StronglyConnected, encontramos si los dos aeropuertos con código IATA ingresados, se encuentran fuertemente conectados.

**Requerimiento 3(O(N)):**

Inicialmente se le solicita al usuario el nombre de la ciudad, el país y la región la cual pertenece para evitar todo tipo de problemas con las ciudades con nombres homónimos, teniendo en cuenta esto, se evita cualquier error con el tema de la carga de datos de parte de los nombres. Primero se calcula mediante una función auxiliar llamada CalcularDistancia que ayuda a encontrar el aeropuerto más cercano a las ciudades insertadas por medio de la formula Haversine. Finalmente retorna los códigos IATA de cada uno de los aeropuertos encontrados.

Gracias al algoritmo de Dijkstra encontramos el camino más óptimo para llegar de un aeropuerto a otro.

**Requerimiento 4(O(N)):**

El requerimiento 4 solicita al usuario una ciudad y la cantidad de millas a su disposición, de acuerdo a esto, busca el aeropuerto más cercano a la ciudad ingresada, a partir de ahí gracias al algoritmo de Dijkstra, buscamos la cantidad de caminos posibles desde el aeropuerto inicial buscando donde se recorra la menor distancia en el mayor número de aeropuertos posibles.

**Requerimiento 5(O(N)):**

El requerimiento 5 solicita al usuario el código IATA de un aeropuerto afectado, a partir de este código IATA, restamos el valor de los ingresos y salidas del aeropuerto afectado a la cantidad de rutas creadas, por lo que nos daría el total de nuevas rutas. Adicionalmente obtenemos los aeropuertos afectados con la función gr.adjacents y buscamos la información de ellos para el retorno