RETO 4: RETOMANDO LOS AIRES

Objetivo

El objetivo de este reto es poner en práctica los conceptos aprendidos en clase acerca del TAD Grafo. Específicamente se desea:

1) Practicar los conceptos sobre grafos, en conjunto con las demás estructuras de datos (listas, Maps y Arboles) del curso para solucionar los requerimientos del reto.
2) Utilizar adecuadamente el patrón MVC (Modelo-Vista-Controlador).
3) Aprender a cargar y procesar en memoria datos en formato CSV.
4) Utilizar adecuadamente el administrador de versiones GIT y GitHub.
5) Aprender a trabajar en equipo

Fecha Límite de Entrega

Miércoles 8 de diciembre, 11:59 p.m.

Actualizaciones del Enunciado

Las actualizaciones en el enunciado del Reto son:

1. Aclaraciones en la construcción del grafo no dirigido en la **Parte 2: Carga de Datos.**
2. Notas de ayuda para solucionar las ciudades homónimas (mismo nombre) en el **Requerimiento 3.**
3. Simplificación de las variables de entrada y respuesta esperada en la implementación del **Requerimiento 5.**

Fuente de Datos

A continuación, se presenta una descripción de la fuente de datos que se utilizará en el reto. Los datos están contenidos en los archivos `airports_full.csv`, `full_routes.csv`, y `worldcities.csv` que se pueden descargar del aula unificada en Bloque Neón.

El archivo `airports_full.csv`, contiene un conjunto de datos que corresponde a la información de cada uno de los aeropuertos a los que nuestra aplicación tendrá acceso. Los datos incluyen los siguientes campos: el name (nombre), el ***city*** (ciudad en la que está situada el aeropuerto), **country** (país en el que está situado el aeropuerto), **IATA** (abreviatura usada por la **International Air Transport Association** para identificar el aeropuerto), latitud y longitud

El archivo `full_routes.csv`, contiene un conjunto de datos que corresponde a la información de las rutas de vuelos posibles entre dos aeropuertos. Los datos incluyen los siguientes campos ***Airilne*** (identificador de la aerolínea habilitada para esa ruta), ***Departure*** (código IATA del aeropuerto de origen), ***Destination*** (código IATA del aeropuerto de destino) y ***distance_km*** (distancia de la ruta dad en kilómetros).

Finalmente, el archivo *worldcities.csv,* contiene un conjunto de datos que corresponde a la información de algunas ciudades del mundo. Los datos incluyen entre otros campos: *city, city_ascii*, la posición geográfica (latitud, longitud), *país, iso2 e iso3* (abreviaturas de la ciudad), su población, y un identificador de la ciudad.

Trabajo Propuesto

Parte 1: Configuración Repositorio

Complete los siguientes pasos para configurar su repositorio de trabajo:

a) Cree en GitHub un repositorio basado en la plantilla propuesta para el reto, el cuan se encuentra en el URL: https://github.com/ISIS1225DEVS/Reto4-Template
b) Renombre el repositorio de su grupo con el esquema **Reto4-G<<Número del grupo>>** ej.: **Reto4-G01** para el grupo 1 de la sección 2.
c) Edite el **README** del repositorio e incluya los nombres completos, correo Uniandes y códigos de los miembros del equipo de **trabajo**.
d) Realice el procedimiento según lo aprendido en clase para clonar el repositorio en su máquina local y sincronizarlo con su repositorio en GitHub.
e) Descargue los datos desde la sección unificada del curso y cópielos en la carpeta **data** del repositorio local.

Parte 2: Carga de Datos

Para responder a los requerimientos presentados más adelante, usted deberá cargar la información de los archivos entregados; es importante anotar que solo es permitido leer una vez la información de cada archivo.

Para el desarrollo de este proyecto se requiere la construcción de una red de transporte aéreo conformada por los aeropuertos y los vuelos que se realizan entre los mismos (ver Ilustración 1).

Para poder experimentar con la mayoría de los algoritmos vistos en el curso se propone la creación de dos grafos para representar la red de transporte aéreo.

1) Un **dígrafo** en el cual se incluirán la totalidad de los aeropuertos (airports_full.csv) y las rutas dirigidas especificadas en el archivo full_routes.csv

2) Un **grafo no dirigido** en el cual se incluirán solamente los aeropuertos que tengan una conexión de ida y vuelta entre sí.

Ilustración 1. Red de Transporte Aéreo

Por ejemplo, al revisar el archivo full_routes.csv se encuentra que entre el aeropuerto *Sochi International Airport (AER)* y *Kazan International Airport (KZN)* se tienen una ruta de *AER* a *KZN*, y otra ruta diferente de *KZN* a *AER* (Ver Tabla 1).

Tabla 1. Tabla de Rutas

Airline	Departure	Destination
2B	**AER**	**KZN**
2B	ASF	KZN
2B	ASF	MRV
2B	CEK	KZN
2B	CEK	OVB
2B	DME	KZN
2B	DME	NBC
2B	DME	TGK
2B	DME	UUA
2B	EGO	KGD
2B	EGO	KZN
2B	GYD	NBC
2B	KGD	EGO
2B	**KZN**	**AER**
2B	KZN	ASF
2B	KZN	CEK
2B	KZN	DME
2B	KZN	EGO
2B	KZN	LED
2B	KZN	SVX

Al final de la carga hay que reportar los siguientes datos:

a) El total de **aeropuertos** en cada grafo
b) El total de **rutas aéreas** en cada grafo
c) El total de **ciudades**.
d) Mostrar la información del primer **aeropuerto** cargado (nombre, ciudad, país, latitud y longitud) en cada grafo.
e) Mostrar la información de población, latitud y longitud, de la última ciudad cargada.

Parte 3: Desarrollo de los requerimientos

Requerimiento 1 (Grupal): Encontrar puntos de interconexión aérea

Como analista de vuelos **deseo** encontrar el (los) **aeropuerto(s)** que sirven como punto de interconexión a más rutas aéreas en la red en cada uno de los grafos.

Para dar respuesta a este requerimiento el equipo de desarrollo no necesita ninguna entrada, y como respuesta debe presentar en consola la siguiente información:

- Lista de aeropuertos (IATA, nombre, ciudad, país).
- Número de aeropuertos interconectados.

Requerimiento 2 (Grupal): Encontrar clústeres de tráfico aéreo

Como analista de vuelos **deseo** encontrar la cantidad de clústeres (componentes fuertemente conectados) dentro de la red de tráfico aéreo y si dos ***aeropuertos*** pertenecen o no al mismo clúster.

Las entradas de este requerimiento son:

- Código IATA del aeropuerto 1.
- Código IATA del aeropuerto 2.

Y como respuesta debe presentar en consola la siguiente información:

- Número total de clústeres presentes en la red de transporte aéreo.
- Informar si los dos ***aeropuertos*** están en el mismo clúster o no.

Requerimiento 3 (Grupal): Encontrar la ruta más corta entre ciudades

Como analista de vuelos **deseo** encontrar la ruta mínima en distancia para viajar entre dos ciudades, los puntos de origen y de destino serán los nombres de las ciudades.

Las entradas de este requerimiento son:

- Ciudad de origen.
- Ciudad de destino.

AYUDA: como paso intermedio del requerimiento al buscar las ciudades de origen y destino la aplicación debe listar las ciudades homónimas (con el mismo nombre) y permitirle al usuario elegir la ciudad especifica basada en la información geográfica del lugar como lo son el país, la subregión (departamento, estado o prefectura) y su ubicación geográfica (latitud y longitud).

Al terminar de elegir las ciudades de origen y destino de los listados podrá buscar los aeropuertos correspondientes mas cercanos a sus puntos geográficos.

Para finalizar, como respuesta debe presentar en consola la siguiente información:

- Aeropuerto de Origen.
- Aeropuerto de Destino.
- Ruta (incluir la distancia de viaje [km] de cada segmento de viaje aéreo).
- Distancia total de la ruta (incluir la distancia terrestre entre la ciudad de origen y el aeropuerto de origen y entre el aeropuerto destino y la ciudad de destino).

Nota 1: Para encontrar el aeropuerto más cercano a la latitud y longitud de la ciudad, se propone buscar los aeropuertos que se encuentran en un cuadrado de 10 km alrededor de las coordenadas de la ciudad, si no encuentra ninguno aumentar 10 km más (20 km en la segunda iteración) y así sucesivamente hasta que encuentre algún aeropuerto cercano. Dado que puede haber dos o más aeropuertos en el rango, debe seleccionar el que esté más cerca en distancia.

En la Ilustración 2 se observa dicha recomendación, En rojo estaría el primer cuadrado de búsqueda suponiendo que el punto rojo es la latitud y longitud de Bogotá y en azul el segundo cuadrado de búsqueda.

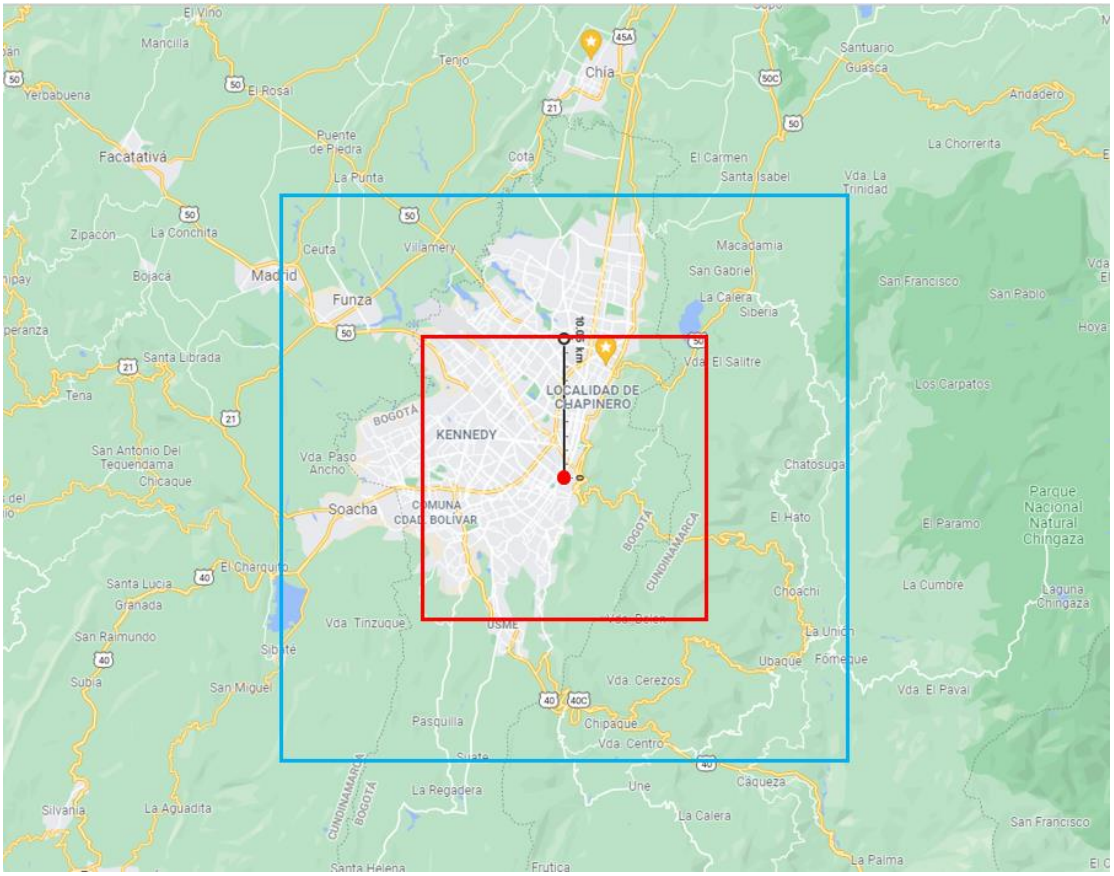

Ilustración 2. Ejemplo de búsqueda del aeropuerto más cercano a una ciudad

Nota 2: Para calcular la distancia entre dos ubicaciones geográficas que tienen latitud y longitud se sugiere utilizar la **Fórmula del semiverseno (Haversine formula)**[1,2,3,4] para calcular la distancia.

Requerimiento 4 (Grupal): Utilizar las millas de viajero

Como viajero **desea** utilizar sus millas para realizar un viaje que cubra la mayor cantidad de ciudades que él pueda visitar. Para ello se necesita identificar la red de expansión mínima en cuanto a distancia que maximice la cantidad de ciudades de la red (representadas por los aeropuertos).

Recuerde que en el plan del viajero una milla equivale a 1.60 kilómetros.

Las entradas de este requerimiento son:

- Ciudad de origen.
- Cantidad de millas disponibles del viajero.

Como respuesta debe presentar en consola la siguiente información:

- El número de nodos conectados a la red de expansión mínima.
- El costo total (distancia en [km]) de la red de expansión mínima.
- Presentar la rama más larga (mayor número de arcos entre la raíz y la hoja) que hace parte de la red de expansión mínima.
- Presentar la lista de ciudades que se recomienda visitar de acuerdo con la cantidad de millas disponibles por el usuario.

Requerimiento 5 (Grupal): Cuantificar el efecto de un aeropuerto cerrado

Como administrador de tráfico aéreo **deseo** conocer el impacto que tendría que un aeropuerto específico saliera de funcionamiento. Para tal fin se requiere conocer la lista de aeropuertos que podrían verse afectados. Para ello las entradas de este requerimiento son:

- Código IATA del aeropuerto fuera de funcionamiento.

Y como respuesta debe presentar en consola la siguiente información:

- Número de aeropuertos afectados.
- Lista de aeropuertos afectados.

---

[1] Haversine formula, URL: https://en.wikipedia.org/wiki/Haversine_formula
[2] Calculate distance, bearing and more between Latitude/Longitude points, URL: https://www.movable-type.co.uk/scripts/latlong.html
[3] Haversine Formula in Python (Bearing and Distance between two GPS points), URL: https://stackoverflow.com/questions/4913349/haversine-formula-in-python-bearing-and-distance-between-two-gps-points
[4] haversine 2.5.1, URL: https://pypi.org/project/haversine/

Requerimiento 6 (BONO Grupal): Comparar con servicio WEB externo

Como administrador aéreo se **desea** encontrar la ruta mínima en distancia para viajar entre dos ciudades. Teniendo en cuenta que en cada ciudad se debe determinar el aeropuerto más cercano y relevante por su flujo de vuelos (ej.: El Aeropuerto Internacional el Dorado es más importante que el aeropuerto de Guaymaral), utilice el API **Airport Nearest Relevant**[5] para identificar los aeropuertos tanto de origen como de destino y compararlos con los resultados del requerimiento 3

Las entradas de este requerimiento son:

- Ciudad de origen.
- Ciudad de destino.

Y como respuesta debe presentar en consola la siguiente información:

- Aeropuerto de Origen
- Aeropuerto de Destino
- Ruta (incluir la distancia de viaje [km] de cada segmento de viaje aéreo)
- Distancia total de la ruta (incluir la distancia terrestre entre la ciudad de origen y el aeropuerto de origen y entre el aeropuerto destino y la ciudad de destino)

Nota 1: Para usar el API de **Airport Nearest Relevant** es necesario crear una cuenta y completar los pasos descritos en la la guía de autorización para peticiones[6]. **¡¡¡ATENCIÓN!!!, NO CARGUE** la información de su **usuario (CLIENT_ID)** o su **token de seguridad (SECRET)** al repositorio GitHub. De suceder esto podrá generar que personas inescrupulosas utilicen sus peticiones gratis sin que usted se entere.

Nota 2: Para calcular la distancia entre dos ubicaciones geográficas que tienen latitud y longitud se sugiere utilizar la **Fórmula del semiverseno (Haversine formula)** para calcular la distancia.

Requerimiento 7 (BONO Grupal): Visualizar gráficamente los requerimientos

Se otorgará una bonificación a los equipos de trabajo que grafiquen un mapa con los resultados de cada uno de los requerimientos obligatorios del enunciado (del primero al quinto).

Para completar este requerimiento recomendamos utilizar la librería por extensión de Python llamada "folium" que se puede instalar en su ambiente por medio del comando "pip install folium".

Para más información sobre esta librería dirigirse a los siguientes enlaces:

- Enlace oficial de PYPI, URL: https://pypi.org/project/folium/

[5] Airport Nearest Relevant, URL: https://developers.amadeus.com/self-service/category/air/api-doc/airport-nearest-relevant
[6] Authorization, URL: https://developers.amadeus.com/self-service/apis-docs/guides/authorization-262

- Enlace oficial de la librería, URL: https://github.com/python-visualization/folium
- Enlace al tutorial de la Liberia. URL: https://python-visualization.github.io/folium/quickstart.html

Parte 4: Análisis de resultados

Cree un archivo en formato **PDF** para la entrega y guárdelo en la carpeta **Docs** del repositorio, el documento debe contener las siguientes secciones:

- Nombres, código y correo Uniandes de los integrantes del grupo.
- Análisis de complejidad temporal en **Notación O** para cada uno de los requerimientos (obligatorios y bonos). Incluir una breve justificación de la complejidad temporal dada.

Entrega

Permita el acceso a su **organización y repositorio** a los monitores y profesores del curso.

Envíe el enlace (URL) del repositorio por **BrightSpace** antes de la fecha límite de entrega.

Recuerden que cualquier documento solicitado durante la práctica debe incluirse dentro del repositorio **GIT** y solo se calificarán los entregables hasta el último **COMMIT** realizado previo a la media noche **(11:59 PM) del 8 de diciembre de 2021.**