**LABORATORIO NO. 11: GRAFOS**

# Objetivos

Utilizar el Tipo Abstracto de Datos Grafo para solucionar problemas computacionales que requieran modelar relaciones matriciales

1. Analizar los órdenes de crecimiento y el desempeño de las estructuras de datos que implementan un grafo
2. Integrar los *grafos* con las otras estructuras de datos vistas en el curso

# Desarrollo

## Pregunta 1: ¿Qué características tiene el grafo definido?, ¿Tamaño inicial, es dirigido?, ¿Estructura de datos utilizada?

El analizador, se definió con la siguiente estructura:

analyzer = {

                    'stops': None,

                    'connections': None,

                    'components': None,

                    'paths': None

                    }

Ahora se explica la definición de cada una de las variables del Analyzer:

* **Stops:** Tabla de hash para guardar los vértices del grafo, recordar que el vértice está definido con la siguiente estructura “*<BusStopCode>-<ServiceNo>*”. El método de mapa para la resolución de colisiones es “Probing” y un tamaño de elementos de 14000, tamaño que responderá a la exigencia de los archivos .CSVs dispuestos para este taller.

        analyzer['stops'] = m.newMap(numelements=14000,

                                     maptype='PROBING',

                                     comparefunction=compareStopIds

* **Connections**: Grafo para representar las rutas entre estaciones.

1. La estructura de datos (TAD) Graph que se usó para representar el grafo es una lista de adyacencias (ADJ\_LIST). Esta permitirá almacenar únicamente la información relevante para del grafo (vértices y conexiones) para las diferentes rutas definidas entra estaciones.
2. Es un grafo dirigido, si es un grafo dirigido, aquí tenemos información adicional como la dirección “*<BusStopCode>-<ServiceNo>*”, que representa el código de la parada de bus y lo números de servicio que pueden parar en esa estación.
3. El tamaño (size) es proporcional a la exigencia requerida por los archivos CSVs dispuestos para este taller.

analyzer['connections'] = gr.newGraph(datastructure='ADJ\_LIST',

                                              directed=True,

                                              size=14000,

                                              comparefunction=compareStopIds)

* **Components**: Almacena la información de los componentes conectados
* **Paths**: Estructura que almacena los caminos de costo mínimo desde un vértice determinado a todos los otros vértices del grafo

## Pregunta 2: ¿Qué instrucción se usa para cambiar el límite de recursión de Python? ¿Por qué considera que se debe hacer este cambio?, ¿Cuál es el valor inicial que tiene Python cómo límite de recursión?

El valor por defecto que tiene Python como límite de recursión es 1000, Esto se puede cambiar fácilmente mediante el comando sys.setrecursionlimit(x) siendo x el nuevo límite de recursiones (antes se tiene que importar la librería sys). Este cambio es importante porque el número de recursiones necesarias para obtener cierta información dentro de los archivos puede ser muy elevado, y si en dado caso este límite se excede, el programa se detendría causando un error. Por eso es que debemos determinar el nuevo límite de recursión antes de toda acción.

## Pregunta 3: ¿Qué relación creen que existe entre el número de vértices, arcos y el tiempo que toma la operación 4? (Ayuda: ¿es un crecimiento lineal?)

|  |  |  |  |
| --- | --- | --- | --- |
| **Bus\_Routes** | **No. Vertices** | **No. de Arcos** | **Tiempo de Ejecución e Opción 4** |
| bus\_routes\_50.csv | 74 | 73 | 0,057 |
| bus\_routes\_150.csv | 146 | 146 | 0,061 |
| bus\_routes\_300.csv | 295 | 382 | 0,155 |
| bus\_routes\_1000.csv | 984 | 1633 | 0,538 |
| bus\_routes\_2000.csv | 1954 | 3560 | 1,705 |
| bus\_routes\_10000.csv | 9767 | 22768 | 21,386 |
| bus\_routes\_14000.csv | 13535 | 32301 | 33,059 |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Recuerden que cualquier documento solicitado durante las actividades debe incluirse en el repositorio GIT y que solo se calificará hasta el último **COMMIT** realizado dentro de la fecha límite del miércoles 4 de noviembre de 2020, antes de la media noche (11:59 pm).