LABORATORIO NO. 11: GRAFOS

1 OBJETIVOS

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

a) Analizar los órdenes de crecimiento y el desempeño de las estructuras de datos que implementan un grafo
b) Integrar los *grafos* con las otras estructuras de datos vistas en el curso

2 DESARROLLO

En los grupos previamente definidos sigan los siguientes pasos para el laboratorio de hoy.

2.1 CARGAR EL EJEMPLO ISIS1225-SAMPLEGRAPH

Asegúrense que todos los miembros del equipo tienen acceso al ejemplo SampleGraph para ello utilicen el URL https://github.com/ISIS1225DEVS/ISIS1225-SampleGraph y sigan el proceso acostumbrado: Fork del proyecto a su espacio en GitHub y clone a su máquina local.

En la sección unificada del curso, encontrarán la sección "Rutas de buses de Singapur". Al descargar el archivo singapure_bus_routes.zip, deben descomprimirlo y copiar todos los archivos *.*CSV* al directorio Data del proyecto SampleGraph.

Estos datos están tomados del siguiente proyecto de Kaggle:

https://www.kaggle.com/gowthamvarma/singapore-bus-data-land-transport-authority

El conjunto de datos contiene información de rutas de buses, paraderos y distancias entre los paraderos del sistema de buses de la ciudad de singapur.

Los archivos de rutas tienen la siguiente información:

ServiceNo, Operator, Direction, StopSequence, BusStopCode, Distance, WD_FirstBus, WD_LastBus, SAT_FirstBus, SAT_LastBus, SUN_FirstBus, SUN_LastBus.

2.2 Entender la estructura de datos

En el ejemplo, se utilizará la información de las rutas, paraderos y distancias entre paraderos para modelar, mediante un grafo, el sistema de transporte de buses de Singapur.

Para crear el grafo se utilizan los archivos con las rutas y la secuencia de paraderos, así como la distancia entre paradas. Una parada puede servir a más de una ruta, por lo que los vértices del grafo tendrán la siguiente estructura:

<BusStopCode>-<ServiceNo>

Por ejemplo: '75009-10' para indicar que esa parada es para la ruta 10 y

'75009-101' para indicar que esa misma parada también sirve a la ruta 101.

Los arcos, representan segmentos de ruta que comunican dos paradas. Como peso de los arcos, se tiene la distancia entre las dos estaciones.

El grafo es dirigido, dado que las rutas tienen una dirección específica entre las estaciones.

Con esta explicación, revisen el archivo *view.py* e identifiquen las operaciones que puede hacer el usuario. No importa por ahora si algunas de ellas no son claras, en los siguientes laboratorios las exploraremos en más detalle.

Revisen ahora el archivo *model.py* y entiendan cómo está planteado el analizador de rutas de buses.

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

RTA\: El grafo usa un mapa auxiliar para poder realizar las conexiones entre dos vértices, los cuales son la parada y el número del recorrido, además de eso también vincula dos cambios de ruta recorriendo el mapa auxiliar para poder establecer entre cuales rutas se pueden hacer intercambios. Su tamaño inicial es igual a la sumatoria de las rutas por la cantidad de paradas que tienen, es un grafo dirigido pues las rutas tienen un sentido, es decir una ruta parte de un punto a y llega a un punto b. Se utiliza una tabla se hash y se utilizan listas, para el grafo se utiliza un grafo representado como un tabal adyacente.

2.3 Ejecutar el Ejemplo

Ahora desde el archivo view.py ejecuten el ejemplo. El archivo view.py tiene el nombre del archivo a leer. En el directorio Data, encuentran archivos de diferentes tamaños: 50, 150, 300, 1000, etc. El nombre del archivo indica aproximadamente el número de líneas del archivo CSV.

Cambien en el archivo view.py, de archivo, comenzando con el más pequeño y luego con el siguiente en tamaño hasta probar con todos.

Al ejecutar el ejemplo con cada archivo, encontrarán las siguientes opciones:

```
****************************************
Bienvenido
1- Inicializar Analizador
2- Cargar información de buses de singapur
3- Calcular componentes conectados
4- Establecer estación base:
5- Hay camino entre estación base y estación:
6- Ruta de costo mínimo desde la estación base y estación:
7- Estación que sirve a más rutas:
0- Salir
****************************************
```

Ejecuten la opción 1 para crear las estructuras.

Ejecuten la opción 2 para crear el grafo a partir del archivo de rutas. Por ejemplo, al ejecutar el programa con el archivo: 'bus_routes_300.csv'

Obtienen la siguiente información:

```
****************************************
Cargando información de transporte de singapur ....
Numero de vértices: 295
Numero de arcos: 382
El límite de recursión actual: 1000
El límite de recursión se ajusta a: 20000
Tiempo de ejecución: 0.05674999799999991
****************************************
```

- En este caso verán varios mensajes.
- El primero es que se actualiza el número de llamados recursivos de Python. Identifiquen en el programa, dónde se realiza el cambio.

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?

RTA\: sys.getrecursionlimit() & sys.setrecursionlimit(recursionLimit), el primer comando es usado para obtener el límite de recursión y el segundo para ajustarlo a un valor deseado, es necesario recordar que si deseamos realizar estas operaciones será necesario importar el módulo sys .Debemos realizar estos cambios en los límites de recursión pues de no hacerlo Python arrojara errores

cuando este realizando operaciones en nuestra estructura de datos, pues al ser tan grande puede que la cantidad de procesos necesarios supere este limite y entonces el programa sea detenido abruptamente. Inicialmente Python tiene un límite de recursión de 1000 iteraciones.

Cada instrucción del programa arroja el tiempo que tomó su ejecución. La opción 4, que luego trabajaremos más a fondo en el curso, es la que más tiempo toma.

Al ejecutarla verán algo como:

```
****************************************
Estación Base: BusStopCode-ServiceNo (Ej.: 75009-10): 75009-10
Tiempo de ejecución: 0.06441147900000033
****************************************
```

Utilicen el vértice propuesto en el ejemplo: 75009-10 y tomen nota del tiempo que toma esta instrucción con cada uno de los archivos CSV.

Nota: Deben ejecutar siempre las opciones 1 y 2 antes de usar la opción 4.

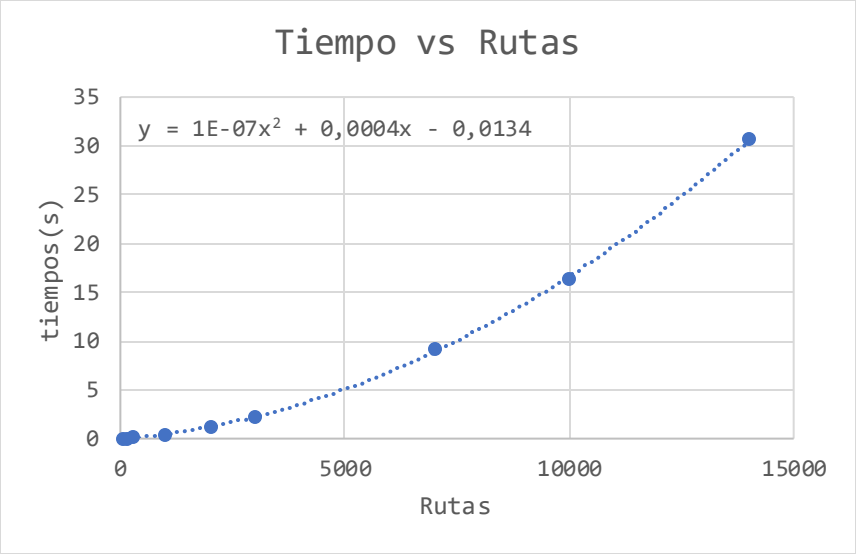
Esta operación, calcula la ruta más corta desde la estación indicada (75009-10) a todas las otras estaciones (todos los vértices del grafo).

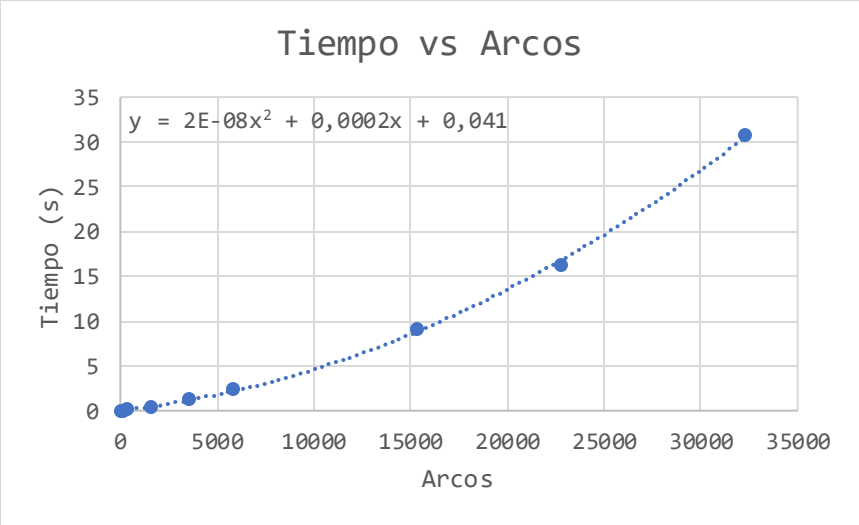
Si luego utilizan la opción 6, y ponen por ejemplo como estación destino: 15151-10, verán el camino propuesto para ir de la estación 75009-10 a la estación 15151-10.

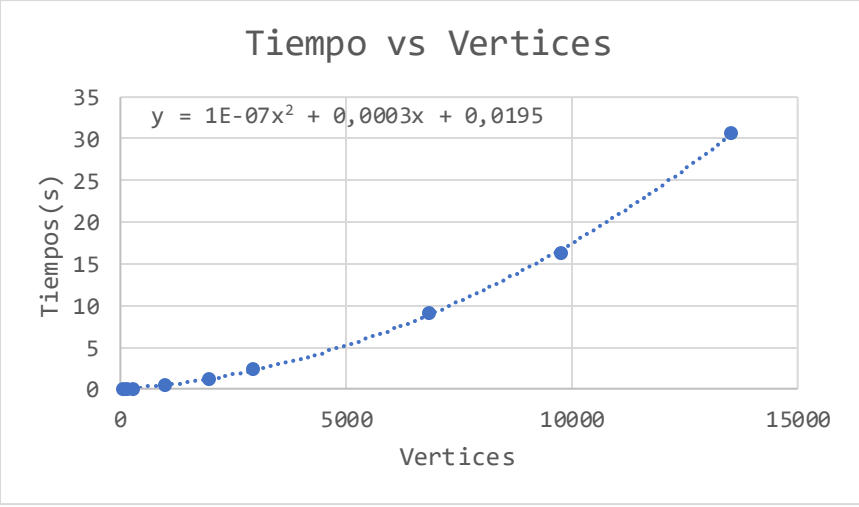
Ahora, ejecuten el programa con cada uno de los archivos CSV. Por cada archivo, tomen nota del número de vértices y el número de arcos del grafo (reportado cuando ejecutan la opción 2) y el tiempo de ejecución que toma la opción 4. (Esta prueba se debe hacer siempre en el mismo computador, idealmente con todos los programas cerrados, solo ejecutando VSCode)

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?)

Tiempo(s)	N° rutas	Arcos	Vértices
0.02	50	73	74
0.04	150	146	146
0.08	300	382	295
0.39	1000	1633	984
1.25	2000	3560	1954
2.31	3000	5773	2922
9.12	7000	15342	6829
16.31	10000	22768	9767
30.68	14000	32301	13535


Tiempo vs Rutas
y = 1E-07x² + 0,0004x - 0,0134
tiempos(s)
Rutas
0
5
10
15
20
25
30
35
0
5000
10000
15000


Tiempo vs Arcos
y = 2E-08x² + 0,0002x + 0,041
Tiempo (s)
Arcos
0
5
10
15
20
25
30
35
0
5000
10000
15000
20000
25000
30000
35000


Tiempo vs Vertices
y = 1E-07x² + 0,0003x + 0,0195
Tiempos(s)
Vertices
0
5
10
15
20
25
30
35
0
5000
10000
15000

RTA\: Vemos que en todos los casos los tiempos presentan curvas de crecimiento polinomiales de grado dos. Es decir que aproximadamente nuestro algoritmo presenta un crecimiento $O(N^2)$ en el peor de los casos considerando tanto vértices como arcos.

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).