OBSERVACIONES DE LA PRÁCTICA

LabGraph-S13-G01

Isai Daniel Chacón Silva Cod 201912015
Nicolás Aparicio Claros Cod 201911357

Preguntas de análisis

a) ¿Qué instrucción se usa para cambiar el límite de recursión de Python?

La instrucción utilizada para cambiar el límite de recursión de Python se encuentra dentro del bloque *main* del código en el archivo *view.py*. Esta es, más específicamente, la función *setrecursionlimit* del módulo *sys* con un valor de 2^{20} como parámetro.

b) ¿Por qué considera que se debe hacer este cambio?

Es necesario realizar este cambio, ya que al trabajar con este tipo de datos y en especial con estructuras de datos tales como los grafos, es posible alcanzar el tope de recursión por defecto en Python. Es por esto que, el módulo *sys*, permite acceder y modificar valores preestablecidos para el intérprete que se esté utilizando. Redefinir el límite de recursión previene al programa de entrar en un loop infinito, lo cual podría llevar al desbordamiento de la pila C, con la cual trabaja Python. Es por esto que se busca establecer un límite más alto cuando se tiene un programa que requiere una recursividad profunda y una plataforma que admita un límite más alto.

c) ¿Cuál es el valor inicial que tiene Python cómo límite de recursión?

Para obtener el valor inicial del límite de recursión de Python se utilizó la función *sys.getrecursionlimit()* antes de mapear este valor a su nuevo límite más alto. El valor obtenido fue: 1000.

d) ¿Qué relación creen que existe entre el número de vértices, arcos y el tiempo que toma la operación 4?

Al correr la operación para cada uno de los volumenes de los datos presentados, se obtuvo la siguiente grafica:

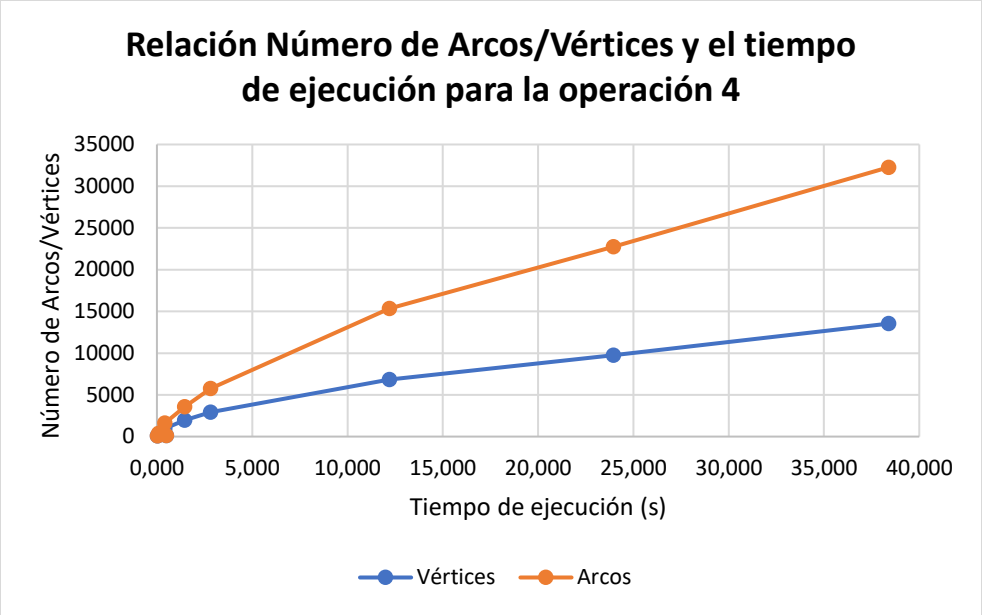

Figura 1. Comportamiento del número de arcos/vértices frente al tiempo de ejecución para la operación 4, de acuerdo a diferentes volumenes de datos dados.

Teniendo en cuenta los resultados expuestos en la Figura. 1, es posible mencionar que el comportamiento del número de nodos y/o vertices frente al tiempo de ejecucicón corresponde a una relación aproximadamente logarítmica.

La Figura 1. fue construida a partir de la toma de datos realizada. Los valores obtenidos se muestran a continuación:

Tabla 1. Valores obtenidos para el tiempo de ejecución al implementar la opción 4, variando el tamaño del archivo.

Requerimiento 4				
Archivo	**Vértices**	**Arcos**	**Tiempo**	**Datos**
bus_routes_50	74	73	0,031	50
bus_routes_150	146	146	0,501	150
bus_routes_300	295	382	0,110	300
bus_routes_1000	984	1633	0,421	1000
bus_routes_2000	1954	3560	1,468	2000
bus_routes_3000	2922	5773	2,821	3000
bus_routes_7000	6829	15334	12,208	7000
bus_routes_10000	9767	22758	23,962	10000
bus_routes_14000	13535	32270	38,396	14000

Para la instrucción también se tomaron los tiempos de ejecución dependiendo del tamaño del archivo, se registró a su vez el número de vértices, el número de arcos y la longitud del camino. Se pueden observar estos valores en la Tabla. 2.

Tabla 2. Valores obtenidos para el tiempo de ejecución al implementar la opción 6, variando el tamaño del archivo.

Requerimiento 6					
Archivo	**Vértices**	**Arcos**	**Longitud camino**	**Tiempo**	**Datos**
bus_routes_50	74	73	59	0,019	50
bus_routes_150	146	146	59	0,021	150
bus_routes_300	295	382	65	0,025	300
bus_routes_1000	984	1633	63	0,030	1000
bus_routes_2000	1954	3560	80	0,030	2000
bus_routes_3000	2922	5773	81	0,040	3000
bus_routes_7000	6829	15334	95	0,038	7000
bus_routes_10000	9767	22758	112	0,044	10000
bus_routes_14000	13535	32270	136	0,067	14000

e) ¿El grafo definido es denso o disperso?, ¿El grafo es dirigido o no dirigido?, ¿El grafo está fuertemente conectado?

Para el grafo de este laboratorio, los arcos representan segmentos de ruta que comunican dos estaciones en el sistema de rutas de Singapur; para las cuales se tiene como peso la distancia entre dos estaciones respectivamente. También, es importante resaltar que el grafo es dirigido, dado que las rutas tienen una dirección definida en particular entre las estaciones.

La densidad de un grafo es cuantificable. Esta se define como $\frac{e}{v(v-1)}$; para e número de arcos y v vértices. Así, para el caso de *bus_routes_14000*, $\delta = \frac{e}{v(v-1)} = \frac{32270}{13535(13534)} = 1.76 * 10^{-4}$; lo que indica que el grafo es disperso.

Por otro lado, se dice que un grafo es fuertemente conectado si todos sus vértices están fuertemente conectados unos con otros, i.e., existe un camino de un vértice v a uno w, y otro de w a v. Teniendo en cuenta esta definición, se podría inferir que el grafo de las rutas de buses no se encontraría fuertemente conectado en sí mismo. Sin embargo, al correr la opción 3 del menú se encontró que el número total de componentes conectados son exactamente 30 para el número de rutas en el sistema de Buses de Singapur.

f) ¿Cuál es el tamaño inicial del grafo?

Inicialmente, al instanciar el *analyzer* en el *model.py*, se observó que el número de elementos para el grado de conexiones se inicializó con *size=14000*, dado que el *.csv* cargado tiene aproximadamente esta cantidad de líneas y por ende, de datos.

Además, al utilizar el archivo *bus_routes_14000*, aquel con mayor cantidad de datos, se obtuvieron los siguientes tamaños del grafo tras cargarlo:

```
Cargando información de transporte de singapur ....
Numero de vertices: 13535
Numero de arcos: 32270
```

Figura 2. Información cargada en el grafo

g) ¿Cuál es la Estructura de datos utilizada?

Se utiliza un analizador que es un diccionario nativo de Python. Este, por su parte, posee una *keys* alusiva a las paradas de autobús y conexiones entre estas, cuyos *values* son un **mapa** y un **grafo** respectivamente, de la librería de DISCLIB.ADT. Para este caso de análisis, nos centraremos en la representación del grafo de conexiones.

```
analyzer['connections'] = gr.newGraph(datastructure='ADJ_LIST',
                                      directed=True,
                                      size=14000,
                                      comparefunction=compareStopIds)
```

Figura 3. Construcción del grafo connections

En el código se puede observar que la estructura de datos utilizado para el grafo es una lista de adyacencia, dirigida, de tamaño 14,000. Su función de comparación es *compareStopIds*, que se discute a continuación.

h) ¿Cuál es la función de comparación utilizada?

La función de comparación utilizada se denomina *compareStopIds()*, la cuak se encarga de comparar dos estaciones que entran por parámetro *stop* y *keyvaluestop*. Cuando la parada que entra por parametro es igual a la que se obtiene en *keyvaluestop[“key”]*, la función se encarga de retornar un 0. Si la parada por entrada es mayor a la de *keyvaluestop[“key”]* se retorna un 1. Finalmente, si no se cumplen ninguna de las dos condiciones anteriores, se retorna el valor de -1.

```
def compareStopIds(stop, keyvaluestop):
    """
    Compara dos estaciones
    """
    stopcode = keyvaluestop['key']
    if (stop == stopcode):
        return 0
    elif (stop > stopcode):
        return 1
    else:
        return -1
```

Figura 4. Función de comparación compareStopIds.