OBSERVACIONES DEL LA PRACTICA

Daniel R Alonso A Cod 201419873
Nicolás Díaz Montaña Cod 202021006

Preguntas de análisis

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

La instrucción que se usa para cambiar el limite de recursión es **setrecursiónlimit()** del modulo sys, el cual establece la profundidad máxima de la pila de intérpretes de Python en limit.

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

Estableciendo un limite se evita la recursividad infinita, la cual provoca que cause un debordamiento de la Pila C y que Pyhton se quede totalmente bloqueado por la cantidad de datos ocupados en la memoria. Esto se debe a que con cada llamada recursiva se debe crear un nuevo marco de pila que va ocupando más y más memoria. Aún así, si el nuevo limite es muy bajo este tambien puede provocar errores.

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

El limite por omisión es de 1000 llamadas recursivas, para que un ciclo de recursión mal implementado no ocupe toda la memoria.

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

Archivo	Vertices	Arcos	Tiempo Opc. 4
50	74	73	0.040
150	146	146	0.048
300	295	382	0.118
1000	984	1633	0.497
2000	1954	3560	1.713
3000	2922	5773	2.970
7000	6829	15334	6.550
10000	9767	22758	24.619
14000	13535	32270	42.121

Se puede hacer evidencia en la tabla entre más vertices y arcos hallá más tiempo le tomara al programa realizar el proceso de la opción 4. Tal vez se deba a que se esta recorriendo todo el grafo por medio de BreadthFirstSearch(BFS) para buscar la estación determinada por el usuario, ya que se esta usando un proceso iterativo.

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

Si un grafo dirigido tiene v vértices, el máximo número de arcos que puede tener es v(v-1).. Ahora si el número de arcos es e, la densidad es igual a $\frac{e}{v(v-1)}$. Se puede establecer que un valor de 0.75 o superior

implica que el grafo es denso y por debajo de 0.3 que es disperso. En esta caso nuestra formula utilizando el archivo csv 14000 quedaria de la siguiente forma:

$$\frac{32270}{13535(13535-1)}$$

Esto nos da 0,000176 lo cual es menor a a 0.3 por ende se puede decir que el grafo es disperso.

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

Según la implementación del grafo este es dirigido.

Para saber si todo esta fuertemente conectado se puede utilizar la opción 3 del programa, el cual no devuelve el numero de componentes que estan fuertemente concetados. En esta caso solo hay 30 componentes que estan fuertmente conectados, comparandolos con la cantidad de vertices y arcos (13535 y 32270, respectivamente), el grafo no esta tan fuertemente conectado.

f) ¿Cuál es el tamaño inicial del grafo?

Como se puece evidenciar en la función de newAnalyzer() del model, el indice “connections” tiene un tamaño de 14000 elementos, tal como el archivo csv más grande.

g) ¿Cuál es la Estructura de datos utilizada?

La estructura de datos utilizada es “ADJ_LIST” o lista de adyaciencia las cuales solo almacenan los vertices y lo arcos de un grafo a manera en arreglos. Los indices de estas listas son los vertices y cada elemento dentro de la lista representa los otros vertices que forman arcos con el vertice principal de la lista.

h) ¿Cuál es la función de comparación utilizada?

La función se llama **compareStopsIds(),** la cual recibe como parametros un codigo de una parada y una pareja llave valor referente a las paradas. A esta pareja llave valor se le saca el codigo que identifica esa parada y se compara con la primera para determinar en que orden debe ir.