Daniela Camacho

Ana Sofia Villa

LABORATORIO 9

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

Para cambiar el límite de recursión se usa la instrucción ***sys.setrecursionlimit()*** que se encuentra en la línea de código 158. En este caso se está asignado un límite de 2^20 que corresponde a 1048576.

Además, en la línea 157 se está teniendo en cuenta el tamaño del stack. Esto es importante ya que cada vez que se llama una función se va a agregando y si se tiene un tamaño reducido de stack, al realizar muchas recursiones puede surgir un problema.

```
155
156 if __name__ == "__main__":
157     threading.stack_size(67108864)  # 64MB stack
158     sys.setrecursionlimit(2 ** 20)
159     thread = threading.Thread(target=thread_cycle)
160     thread.start()
161
```

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

Esto se debe realizar debido a que en el archivo se están implementando grafos, los cuales necesitan de funciones recursivas para poder acceder a los diversos vértices por medio de diferentes rutas. Al cambiar el límite de recursión, python permite generar más llamados o recursiones y así los requerimientos se desarrollarán de una manera más optima

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

```
155
156 if __name__ == "__main__":
157     # threading.stack_size(67108864)  # 64MB stack
158     # sys.setrecursionlimit(2 ** 20)
159     thread = threading.Thread(target=thread_cycle)
160     thread.start()
161
```

```
>2

Cargando información de transporte de singapur ....
Numero de vertices: 13535
Numero de arcos: 32270
El limite de recursion actual: 1000
```

Al comentar las instrucciones utilizadas para cambiar el límite de recursión en las líneas 157 y 158 y volver a cargar los datos, encontramos que el límite de recursión inicial en Python es de 1000.

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

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

Cargando información de transporte de singapur ....
Numero de vertices: 13535
Numero de arcos: 32270
El limite de recursion actual: 1048576
```

Operación 4			
Archivo bus_routes	#Vértices	#Arcos	Tiempo (segundos)
50	74	73	0.0468

150	146	146	0.0625
300	295	382	0.078125
1000	984	1633	0.234375
2000	1954	3560	0.71875
3000	2922	5773	1.34375
7000	6829	15334	3.09375
10000	9767	22758	11.765625
14000	13535	32270	19.325

Creemos que en la relación como se muestra en la tabla a medida que el número de vértices y arcos crece, el tiempo de ejecución de la operación 4 crece. Esto se debe a que esta operación busca los caminos con el menor costo posible, por lo que si hay más arcos y vértices necesita realizar una búsqueda más grande.

Operación 6			
Archivo bus_routes	#Vértices	#Arcos	Tiempo (segundos)
50	74	73	0,0156
150	146	146	0.015625
300	295	382	0.015625
1000	984	1633	0.03125
2000	1954	3560	0.015625
3000	2922	5773	0.015625
7000	6829	15334	0.015625
10000	9767	22758	0.03125
14000	13535	32270	0,156

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

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

El grafo es dirigido, se puede ver en la línea 66 del código, en donde se muestra la siguiente instrucción: directed=True.

Al ser un grafo dirigido puede tener una cantidad máxima de arcos de V(V-1). En este caso, con los datos de 14000, el grafo tiene 13535 vértices por ende podría tener un número máximo arcos de 183182690 [13535*(13535-1)], sin embargo, realmente cuenta con 32270 arcos. Al calcular la densidad [32270/183182690] obtenemos 0.000176, esto indica que se trata de un grafo disperso que no está fuertemente conectado.

f) ¿Cuál es el tamaño inicial del grafo?

En la línea 67 se puede observar que el tamaño inicial cuando se crea el grafo es 14000

g) ¿Cuál es la Estructura de datos utilizada?

La estructura de datos utilizada es ADJ_LIST que se puede ver en la línea 65. ADJ LIST se refiere a la lista de adyacencia, en la cual se guarda la información más importante del grafo la cual es los adyacentes por cada vértice.

h) ¿Cuál es la función de comparación utilizada?

La función de comparación utilizada es compareStopIds. ()

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