OBSERVACIONES LAB 9

Nicholas Barake 202020664

Jesed Domínguez 202011992

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

```
if __name__ == "__main__":
    threading.stack_size(67108864)  # 64MB stack
    sys.setrecursionlimit(2 ** 20)
    thread = threading.Thread(target=thread_cycle)
    thread.start()
```

Específicamente, la instrucción es sys.setrecursionlimit(). Donde se debe importar el módulo de sys de Python, y el parámetro es un int.

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

Python maneja las recursiones con pilas, y entre más recursiones se realizan, más profunda es la pila. Como Python tiene un límite de recursión para evitar que, al realizar funciones recursivas, ocurra una recursión infinita y Python "estalle", es importante cambiar ese limite para poder manejar archivos de datos grandes. En especial al usar grafos, que son recursivos.

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

Por defecto, Python tiene un límite de 10^3 (1000).

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

La relación entre la cantidad de arcos y vértices, y el tiempo es directamente proporcional, por lo que cuando el número de arcos y vértices incrementan, el tiempo de ejecución también. Esto se debe al número de comparaciones que el algoritmo debe realizar para encontrar la ruta más corta. Entre más vértices (estaciones) y arcos (distancias) hay, más comparaciones se tienen que realizar para saber cuál es el vértice con el cual se conecta directamente al ingresado como parámetro, con el arco de menor peso.

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

- La densidad de los grafos cambia con el tamaño de los archivos. Como se puede ver, en el archivo mas pequeño hay 74 vértices y 73 arcos. Lo cual quiere decir que no hay tantas conexiones entre un vértice y otro, en otras palabras, un vértice no se conecta con muchos otros vértices. Mientras que, si miras el archivo más grande, hay 15535 vértices y 32270 arcos. Eso es casi el doble de arcos de lo que hay de vértices, lo cual quiere decir que de una sola estación salen y entran varias rutas (por lo general).
- Es dirigido, porque las rutas (arcos) tienen una dirección especifica entre estaciones.

- Sí se cree que está fuertemente conectado por lo que hay mas arcos de lo que hay vértices, lo cual quiere decir que hay varias rutas que conectan más de dos estaciones entre sí. Es decir, hay muchas rutas que salen y entran de una sola estación.

f. ¿Cuál es el tamaño inicial del grafo?

El tamaño de un grafo es el número de arcos que tiene. Y para el archivo "…14000", el tamaño es 32270, DESPUES DE CARGAR LOS DATOS. Pero inicialmente cuando se inicializa el analizador…

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

Se puede ver que el tamaño inicial es 14000. A medida que se van cargando los datos, se va agrandando.

g. ¿Cuál es la Estructura de datos utilizada?

La estructura utilizada (para el grafo) es "ADJ_LIST", lo cual quiere decir lista de adyacencia.

h. ¿Cuál es la función de comparación utilizada?

La función utilizada es compareStopIds. Como dice el nombre de la función, compara los ids de cada estación (parada).