OBSERVACIONES DE LA PRACTICA

Nicolas Perez Teran Cod 202116903
Juan Pablo Rodríguez Briceño Cod 202022764

Preguntas de análisis

a. ¿Qué instrucción se usa para cambiar el límite de recursión de Python?
RTA: En el archivo sys.pyi, en la línea 223 existe la función setrecursionlimit(__limit: int), la cual sirve para cambiar este límite. Esta función en el archivo view.py es llamada en la linea 158

b. ¿Por qué considera que se debe hacer este cambio?
RTA: Aquí hice un experimento, eliminé la línea donde se cambia el límite de recursión para obtener el inicial que tiene Python.:

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

En contraste con el anterior, vemos que al modificarse aumento considerablemente.

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

Con esto, creo que se hizo el cambio para evitar posibles errores al recorrer el grafo; recordemos que es algo muy recursivo y esto puede hacer que se alcance el límite, lo cual causaría que Python no corra el algoritmo.

c. ¿Cuál es el valor inicial que tiene Python cómo límite de recursión?
RTA: Como podemos ver en la primera imagen de arriba, vendría a ser 1000.

d. ¿Qué relación creen que existe entre el número de vértices, arcos y el tiempo que toma la operación 4?
RTA: A medida que crece el número de vértices, es necesario que existan más arcos (como mínimo, se requiere que tengan el mismo número de vértices en adelante). De la misma manera que hay más vértices, el número de arcos puede crecer a un ritmo mayor de lo que lo hacen los vértices, ya que hay muchas más posibilidades en las que se puedan conectar los datos. Sin embargo, podemos decir que el tiempo aumenta principalmente porque tiene que probar más camino, ya que existen cada vez más vértices.

Datos de la operación 4

Requerimiento 4			
Numero de datos	Vértices	Arcos	Tiempo (ms)
50	74	73	31.25
150	146	146	62.5
300	295	382	78.125

1000	984	1633	343.75
2000	1954	3650	1390.625
3000	2922	5773	2296.875
7000	6829	15334	6437.5
10000	9767	22758	18703.125
14000	13535	32270	36921.875

Datos de la operación 6

Requerimiento 6			
Numero de datos	Vértices	Arcos	Tiempo (ms)
50	74	73	15.625
150	146	146	15.625
300	295	382	15.625
1000	984	1633	15.625
2000	1954	3650	15.625
3000	2922	5773	15.625
7000	6829	15334	15.625
10000	9767	22758	15.625
14000	13535	32270	15.625

*El tiempo de ms podría ser 0.0 si no se tiene en cuenta que se debe mostrar datos al usuario.

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

RTA: El grafo este disperso, ya que a pesar de que contiene una gran cantidad de elementos, si utilizamos una función que implementamos llamada acc(graph) --La cual llama a la función KosarajuSCC(graph), para luego pasar lo que retorne a la otra función connectedComponents(scc), y terminara devolviéndonos los elementos conectados--, podemos observar que para 14000 datos, solo se crearan 30 conexiones.

f. ¿Cuál es el tamaño inicial del grafo?

RTA: El tamaño inicial del grafo es de 14000 vértices

g. ¿Cuál es la Estructura de datos utilizada?

RTA: La estructura de datos usada es “ADJ_LIST”

h. ¿Cuál es la función de comparación utilizada?

RTA: La función de comparación utilizada es compareStopIds, la cual compara dos estaciones. Puede dar 0, 1 o –1 si el código de paradero es igual, mayor o menor respectivamente.