Observaciones de la Práctica

Daniel Gomez, 201728920

Jenifer Arce, 202014993

Preguntas de análisis

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

La instrucción que se usa para cambiar el límite de recursión es **setrecursionlimit()** de la librería de sys. Esta función tiene como parámetro el límite de recursión al que se quiere llegar como máximo.

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

Al estar usando TADs tipo grafos, que requieren de llamados recursivos para utilizar su información, el límite necesita ser lo suficientemente grande para que las funciones puedan ejecutarse por completo cuando el número de vértices y arcos dentro de un grafo son numerosos.

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

Según la documentación[1] de Python, el valor por defecto de límite de recursión es 1000. Al modificar el view para comprobar esta información podemos confirmar que este valor es correcto.

[1] https://docs.python.org/3/library/sys.html documentación del tracebacklimit (sys.tracebacklimit) el valor por defecto es 1000

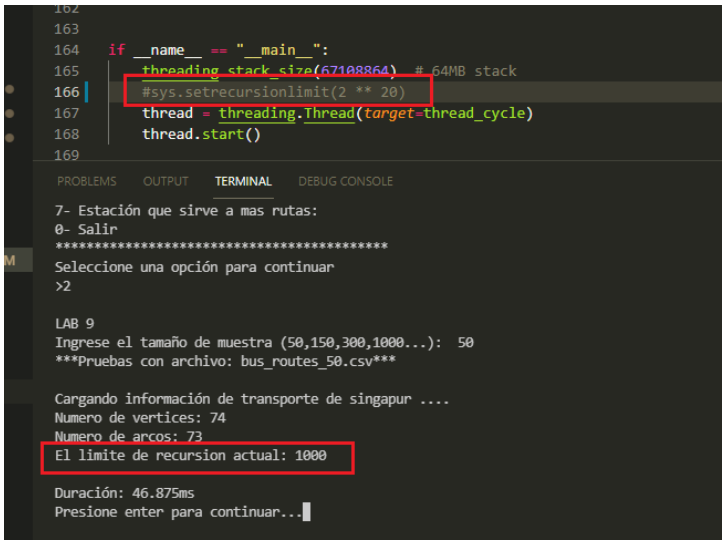

Ilustración 1. Límite de recursión original de Python

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

- Tiempos de ejecución opción 4

Archivo muestra	Carga de información (opción 1)		Ejecución opción 4
	Vértices	Arcos	[tiempo en ms]
bus_routes_50	74	73	46,875
bus_routes_150	146	146	93,75
bus_routes_300	295	382	109,375
bus_routes_1000	984	1633	531,25
bus_routes_2000	1954	3560	421,875
bus_routes_3000	2922	5773	3578,125
bus_routes_7000	6829	15334	9171,875
bus_routes_10000	9767	22758	26359,375
bus_routes_14000	13535	32270	39187,5

Tabla 1. Información ejecución opción 4

El tiempo de ejecución depende estrictamente de la cantidad de vértices y arcos que tenga que recorrer el algoritmo de Dijkstra. A mayor cantidad de conexiones y de vértices que haya en el grafo mayor será el tiempo de cálculo de los caminos entre la estación base ("75009-10") frente a las demás estaciones (o vértices).

- Tiempos de ejecución opción 6

Archivo muestra	El camino es de longitud:	Opción 6 [tiempo en ms]
bus_routes_50	59	15,625
bus_routes_150	59	15,625
bus_routes_300	65	15,625
bus_routes_1000	63	15,625
bus_routes_2000	80	15,625
bus_routes_3000	81	15,625
bus_routes_7000	95	15,625
bus_routes_10000	112	15,625
bus_routes_14000	136	15,625

Tabla 2. Información ejecución opción 6

Al analizar los tiempos de ejecución de la opción 6 se observa que estos siempre son menores o iguales a 15,625 ms. Lo anterior es debido a que en la opción 4 encontramos todos los caminos más cercanos desde la estación base ("75009-10") al resto de estaciones. Por lo cual, al buscar una estación con el costo mínimo lo que se tendrá que hacer es recorrer el grafo de respuesta de la opción 4 con la función djk.pathTo, al encontrar la estación (vértice) deseado se retorna el camino con menor costo.

e. 1. ¿El grafo definido es denso o disperso?

Para medir la densidad del grafo primero calculamos la cantidad máxima de arcos que puede tener cada vertice, para esto utilizamos la fórmula vertice*(vertice-1). Después de esto calculamos la densidad, que es la cantidad de arcos del grafo sobre la cantidad máxima de arcos (arcos/cantidad máxima arcos), dado que es una proporción este número es menor que 1. A partir de este número analizamos si es denso o disperso, para que sea denso la proporción deberá ser mayor que 0,75, de lo contrario es disperso. En el caso del grafo del laboratorio este es disperso dado que la densidad es menor que 1% en todos los tamaños de archivos.

En la siguiente tabla están los cálculos que realizamos:

Archivo muestra	Carga de información (opción 1)		Densidad		
	Vértices	Arcos	máximo arcos (teoría v*(v-1))	Densidad (arcos/ máximo arcos)	¿Es denso o disperso? (denso= densidad > 0,75)
bus_routes_50	74	73	5402	0,014	Es disperso
bus_routes_150	146	146	21170	0,007	Es disperso
bus_routes_300	295	382	86730	0,004	Es disperso
bus_routes_1000	984	1633	967272	0,002	Es disperso
bus_routes_2000	1954	3560	3816162	0,001	Es disperso
bus_routes_3000	2922	5773	8535162	0,001	Es disperso
bus_routes_7000	6829	15334	46628412	0,000	Es disperso
bus_routes_10000	9767	22758	95384522	0,000	Es disperso
bus_routes_14000	13535	32270	183182690	0,000	Es disperso

Tabla 3. Análisis densidad grafo

2. **¿El grafo es dirigido o no dirigido?**

El grafo es dirigido, las rutas tienen un arco que va en un solo sentido. Adicionalmente, en la página 2 del laboratorio se específica la estructura del grafo.

3. **¿El grafo está fuertemente conectado?**

Los componentes que están conectados en el archivo large (bus_routes_14000) son 30 componentes. El número de vértices en este tamaño de archivo es 15535 y el número de componentes que están conectados es 30, la proporción entre vértices y número de componentes es equivalente a *0,002216476*, es decir que menos del 1% de los vértices están conectados entre sí. Adicionalmente, como el grafo es disperso el número de arcos es bajo, es así como no pueden existir tantas conexiones entre pares de vértices. Por ende, podemos concluir que el grafo no está fuertemente conectado.

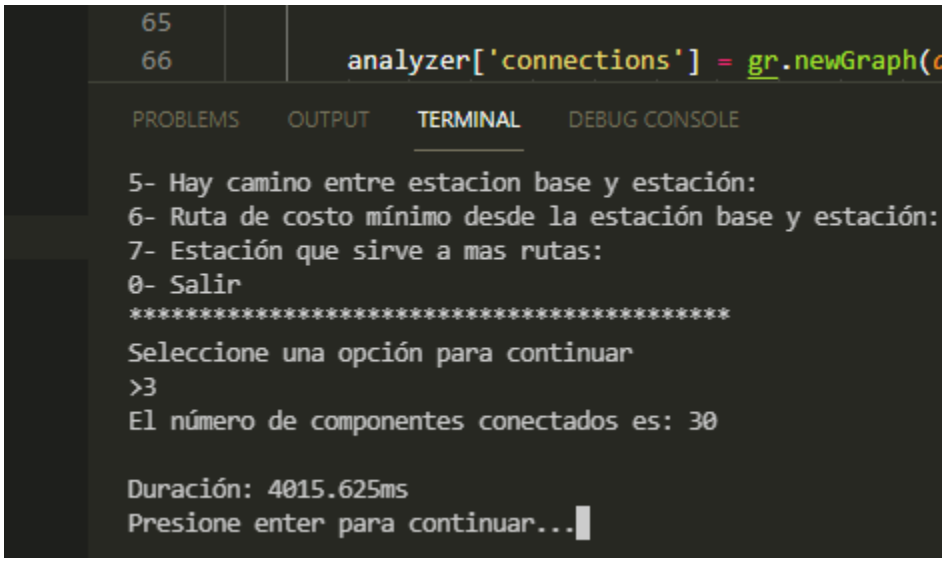

Ilustración 2. Resultado opción 3 – Archivo bus_routes_14000

f. ¿Cuál es el tamaño inicial del grafo?

El tamaño inicial del grafo es 14000 vértices.

g. ¿Cuál es la Estructura de datos utilizada?

- La estructura de datos para las estaciones (analyzer['stops']) es un mapa tipo linear probing.
- La estructura de datos utilizada para representar las rutas (analyzer['connections']) es un grafo. Para la creación interna de los vértices de un grafo se hace un mapa tipo línear probing.

h. ¿Cuál es la función de comparación utilizada?

La cmp utilizada es *compareStopIds*, la cual compara dos estaciones.

Ambiente de pruebas

	Máquina
Procesadores	Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz 1.80 GHz
Memoria RAM (GB)	8 GB
Nombre del SO	Windows 10 64-bits

Tabla 4. Ambiente de pruebas