OBSERVACIONES DEL LA PRACTICA

Daniel Rodriguez - ds.rodriguezf1@uniandes.edu.co – 202014760

Santiago Forero - s.forerog2@uniandes.edu.co – 202111446

Preguntas de análisis

a) ¿Qué instrucción se usa para cambiar el límite de recursión de Python?
Se usa la instrucción sys.setrecursionlimit, como se ve abajo:

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

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

Debido a que por defecto el límite de recursión es 1000, y ya que en los grafos se usa mucho la recursión, sobretodo en DFS, es mejor definirla como un valor más alto en este caso 1048576

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

El valor inicial de Python es 1000

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

Pruebas de tiempo:

Numero de vertices: 74 - Numero de arcos: 73 - Tiempo: 46.875

Numero de vertices: 146 - Numero de arcos: 146 - Tiempo: 46.875

Numero de vertices: 295 - Numero de arcos: 382 - Tiempo: 78.125

Numero de vertices: 984 - Numero de arcos: 1633 - Tiempo: 343.75

Numero de vertices: 1954 - Numero de arcos: 3560 - Tiempo: 1156.25

Numero de vertices: 2922 - Numero de arcos: 5773 - Tiempo: 1906.25

Numero de vertices: 6829 - Numero de arcos: 15334 - Tiempo: 7781.25

Numero de vertices: 9767 - Numero de arcos: 22758 - Tiempo: 17125.0

Numero de vertices: 13535 - Numero de arcos: 32270 - Tiempo: 29421.875

Tomando como base los valores obtenidos parece que el número de vértices, arcos y el tiempo que toma la operación 4 tienen una relación lineal o exponencial, lo importante es que mientras mayor sea el numero de estaciones la funcion tendrá que calcular un mayor número de rutas.

e) ¿El grafo definido es denso o disperso?, ¿El grafo es dirigido o no dirigido?, ¿El grafo está fuertemente conectado?
Tomando como base la carga de 14000 podemos ver que el número de vértices es 13535 y el número de arcos es 32270. Por lo que en promedio cada estación tiene un poco más de dos conexiones con otras estaciones, por lo que es un poco disperso. El grafo sí es dirigido, ya que las rutas tienen una dirección específica entre las estaciones. Y el grafo sí esta fuertemente conectado, pues según las pruebas parece que se puede ir de cualquier estación a cualquier otra por medio de conexiones de rutas.

f) ¿Cuál es el tamaño inicial del grafo?

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

El valor inicial es 14000

g) ¿Cuál es la Estructura de datos utilizada?

```
analyzer['stops'] = m.newMap(numelements=14000,
                             maptype='PROBING',
                             comparefunction=compareStopIds)

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

Se usan dos estructuras, un mapa para las paradas y un grafo para las conecciones.

h) ¿Cuál es la función de comparación utilizada?

```
def connectedComponents(analyzer):
    """
    Calcula los componentes conectados del grafo
    Se utiliza el algoritmo de Kosaraju
    """
    analyzer['components'] = scc.KosarajuSCC(analyzer['connections'])
    return scc.connectedComponents(analyzer['components'])
```

Ya que vemos que se llama al modulo scc podemos ver que se usa la función de busqueda DFS con recursión. Y la función de comparación que se pasa al crear el grafo es:

```
def compareStopIds(stop, keyvaluestop):
    """
    Compara dos estaciones
    """
```

```
    stopcode = keyvaluestop['key']
    if (stop == stopcode):
        return 0
    elif (stop > stopcode):
        return 1
    else:
        return -1
```