OBSERVACIONES DEL LA PRACTICA

Estudiante 1: Cod Ehimar Andres Vargas Malaver – e.vargasm@uniandes.edu.co -- 202014902
Estudiante 2 Cod XXXX

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 recursión es “sys.setrecursionlimit(2 ** 20)”

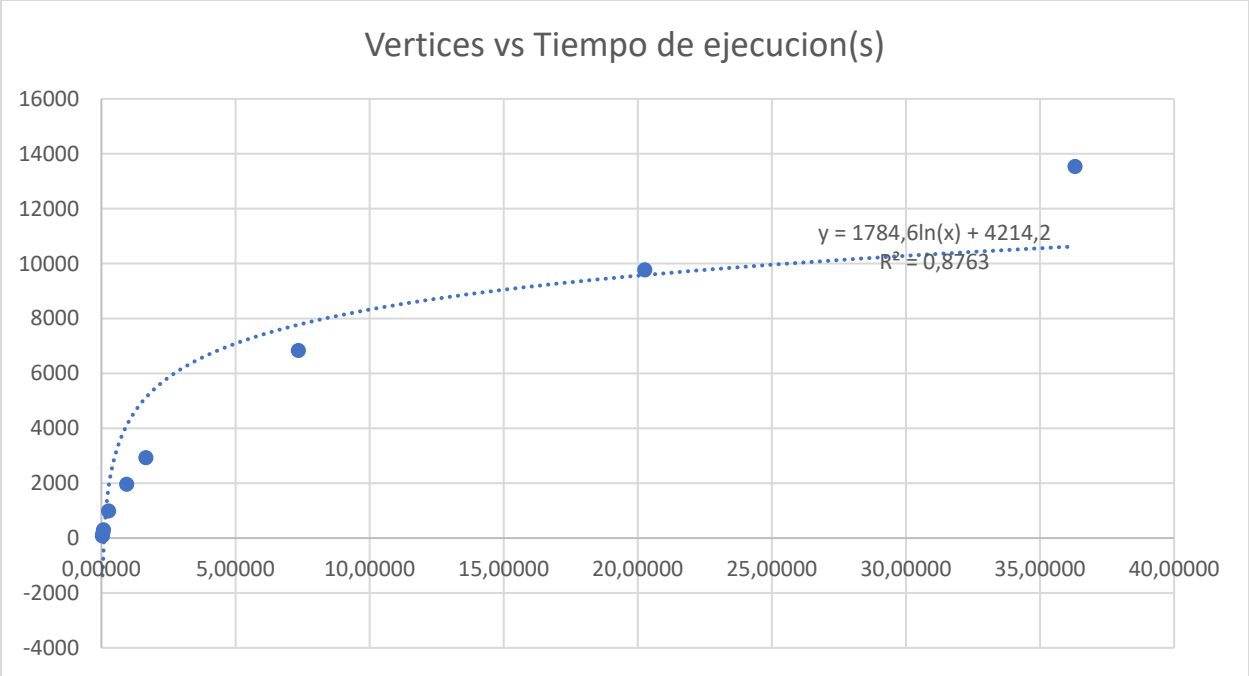
b) ¿Por qué considera que se debe hacer este cambio?

Se debe realizar el cambio del límite de recursiones debido a que las funciones usadas para el grafo son recursivas, además en algunas funciones de consulta se usan algoritmos recursivos como Kosaraju el cual usa un gran número de recursiones para llevar poder encontrar las componentes conectadas.

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

El valor inicial de recursiones de Python son 1000 recursiones, si se supera este límite, el programa retorna error

Cantidad de datos	#vertices	#arcos	Tiempo De ejecución (s)
50	74	73	0.04397
150	146	146	0.05287
300	295	382	0.08187
1000	984	1633	0.27707
2000	1954	3560	0.9498
3000	2922	5773	1.6639
7000	6829	15334	7.3505
10000	9767	22758	20.2664
14000	13535	32270	36.3108



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

La relación entre el número de vértices y el tiempo de ejecución es de tipo logarítmica, y el valor es de **1784,6ln(n) + 4214,2**

Cantidad de datos	#vertices	#arcos	Tiempo De ejecución (s)
50	74	73	0.0004889998
150	146	146	0.0013814999
300	295	382	0.0013253000
1000	984	1633	0.0015777000
2000	1954	3560	0.0014760999
3000	2922	5773	0.0010912999
7000	6829	15334	0.0013097000
10000	9767	22758	0.0013301999
14000	13535	32270	0.0021883999

Se puede observar que lso tiempos de respuesta son muy bajos dado el grafo permite acceder rapida mente a la ingormacion dentro de este.

e) ¿Qué características tiene el grafo definido?

Es un grafo dirigido, ósea, los arcos definen una relación unidireccional entre las dos componentes conectadas.

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

f) ¿Cuál es el tamaño inicial del grafo?

El tamaño inicial del grafo pretende guardar 14000 vértices.

g) ¿Cuál es la Estructura de datos utilizada?

Se usan listas de adyacencia ya que el grafo es dirigido y se recomienda usar esta estructura dado que consume menos memoria en comparación a la matriz de adyacencia.

h) ¿Cuál es la función de comparación utilizada?

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

Esta función compara las id de las paradas de los buses y dice cuál es el mayor de los dos.