OBSERVACIONES DEL LA PRACTICA

Estudiante 1 Cod 202020706
Estudiante 2 Cod 202013610

Preguntas de análisis

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

sys.setrecursionlimit(2**20)

Esta función nativa de Python permite cambiar el límite de recursiones que se realizan en el código, ya que el grafo se forma mediante funciones recursivas; usando el límite nativo de 10^4 recursiones es muy poco para que el grafo llegue a estar completo.

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

El cambio se debe realizar para que cuando se esté creando el grafo, este no se limite por el límite de recursiones que tiene Python.

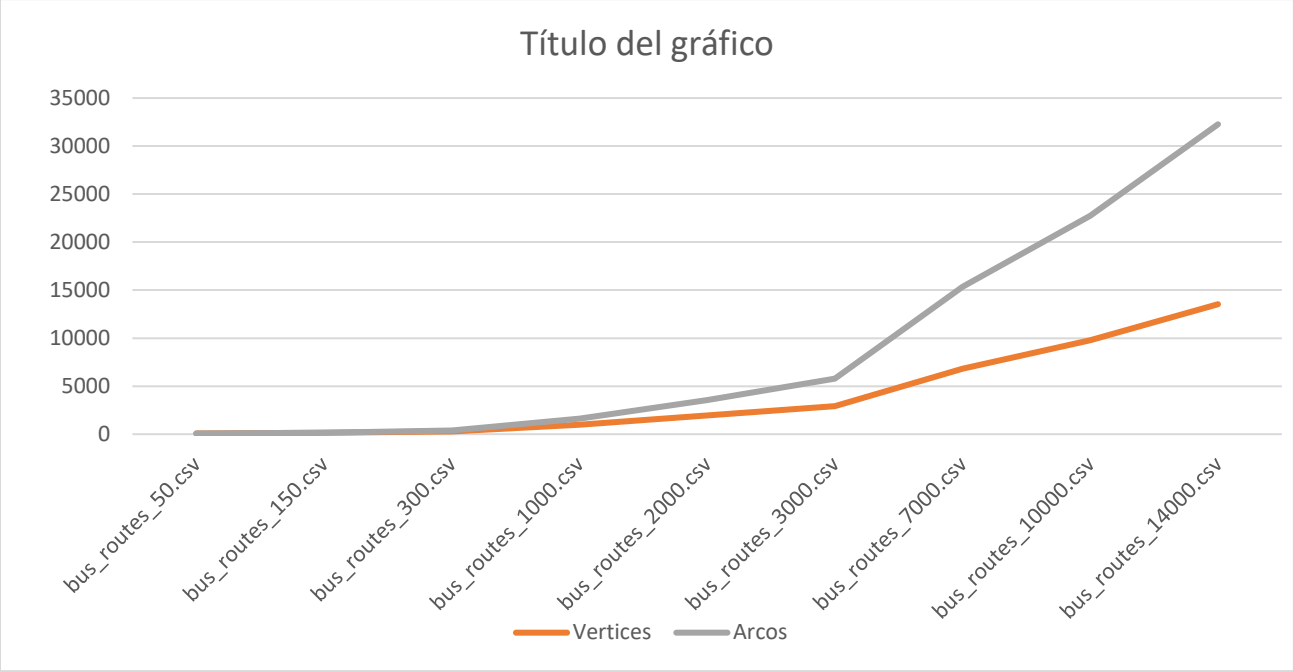
c) ¿Cuál es el valor inicial que tiene Python cómo límite de recursión?

Python tiene un límite de apróximadametne 10^4 recursiones, que, por temas de optimización, limitan en algoritmos cómo el de crear grafos o como tail recursion con valores elvedaos.

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

Opción 4

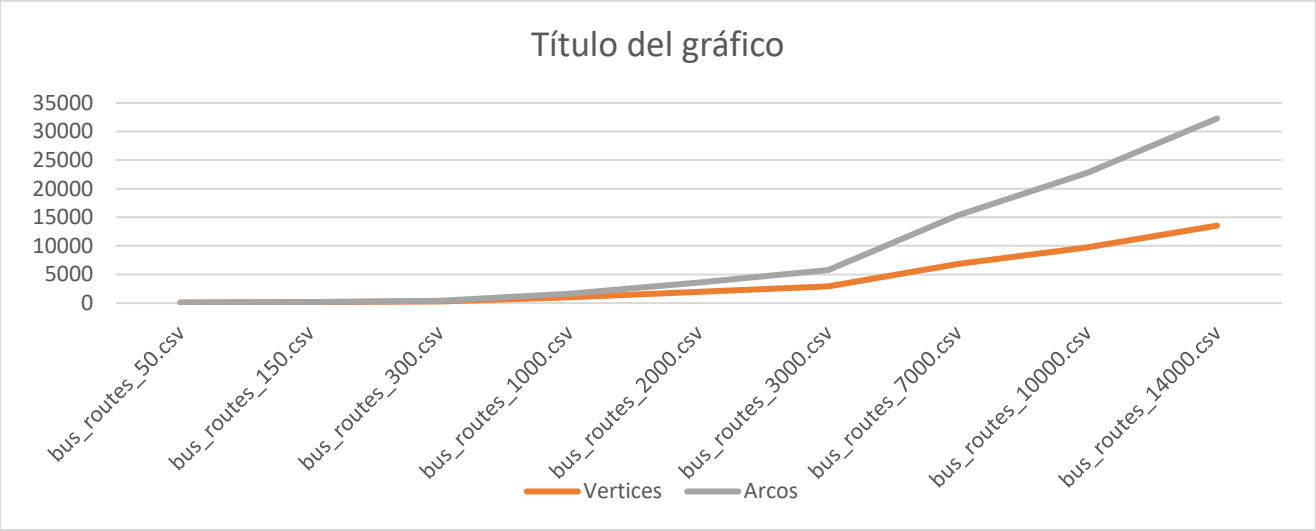
Nombre del documento	Tiempo de ejecución	#Vertices	#Arcos
bus_routes_50.csv	0.01720209999999952	74	73
bus_routes_150.csv	0.02580189999999948	146	146
bus_routes_300.csv	0.04665159999999923	295	382
bus_routes_1000.csv	0.3175662000000002	984	1633
bus_routes_2000.csv	0.8567968	1954	3560
bus_routes_3000.csv	1.7883171000000002	2922	5773
bus_routes_7000.csv	5.560262599999998	6829	15334
bus_routes_10000.csv	15.251625299999999	9767	22758
bus_routes_14000.csv	26.134499399999996	13535	32270

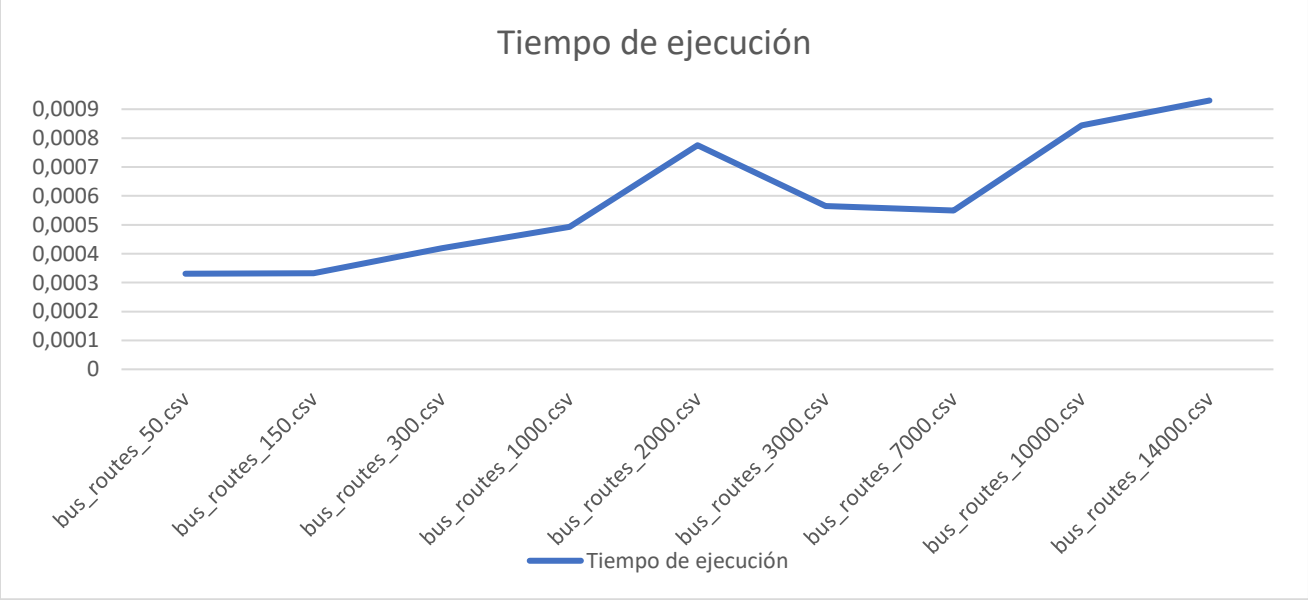


Cómo podemos observar en las gráficas, los vertices y los arcos a medida que aumenta la cantidad de datos, tanto el tiempo de ejecución como los vertices como los arcos parecen tener un creciemiento exponencial.

Opción 6

Nombre del documento	Tiempo de ejecución	#Vertices	#Arcos
bus_routes_50.csv	0.0003307999999997264	74	73
bus_routes_150.csv	0.0003323999999993248	146	146
bus_routes_300.csv	0.00041890000000057624	295	382
bus_routes_1000.csv	0.0004928999999992812	984	1633
bus_routes_2000.csv	0.0007748999999985244	1954	3560
bus_routes_3000.csv	0.0005654000000013184	2922	5773
bus_routes_7000.csv	0.0005491000000001805	6829	15334
bus_routes_10000.csv	0.0008437999999950989	9767	22758
bus_routes_14000.csv	0.0009300999999977648	13535	32270





La gráficas nos muestran un comportamiento similar en los vértices y los arcos como en el análisis realizado de la opción # 4. Sin embargo, la opción # 6 tiene un comportamiento particular cuando se ejecutan los archivos 2000, 3000 y 700 dando resultados mayores y menores respectivamente.

e) ¿Qué características tiene el grafo definido?

Es un grafo direccionado con pesos

f) ¿Cuál es el tamaño inicial del grafo?

El grafo tiene un tamaño inicial de 14000 definido cuando se crea

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

g) ¿Cuál es la Estructura de datos utilizada?

La estructura de datos utilizada en el ejemplo corresponde a un grafo dirigido, es decir que sus arcos tienen dirección a un vértice en específico.

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
```