Lab 9

ISIS 1225 (4)
| Ronald Pardo | 202111309 | r.diazp@uniandes.edu.co |
| Juan Andres Ruiz Uribe | 201914351 | ja.ruizu@uniandes.edu.co |

a) La instrucción utilizada para cambiar el límite de recursión de python es:

```
sys.setrecursionlimit(2 ** 20)
```

Este método establece la profundidad máxima de la pila de intérpretes de Python en n. Este límite evita que la recursividad infinita cause un desbordamiento de la pila de C y bloquee Python. El límite más alto posible depende de la plataforma.

b) Cambiar el límite de recursión es importante porque puede obstaculizar el funcionamiento recursivo del recorrido de busqueda DFS, el cual se encuentra implementado en este lab por medio de los archivos dfs.py y dfo.py. Si bien este está para prevenir loops infinitos y/o problemas parecidos, ciertos métodos necesitan ser llamados una cantidad de veces superior al límite por default.

c) El valor inicial que trae Python como límite de recursión es 1000, lo que significa que un método puede ser llamado un máximo de mil veces.

d) La operacion 4 se encarga de encontrar todos las rutas de costo minimo de cada vertice con el el vertice base por medio del algoritmo Dijkstra, por lo que se podria establecer que realizar esta operacion requiere que por lo menos se recorra V veces en el grafo como minimo, suponiendo que existe un solo camino para cada vertice que es directo. No obstante, en este caso cada vertice tiene multiples arcos, por lo que podemos usar la proporcion (aproximada y no precisa) de numero de arcos por cada vertice y por medio de una multiplicacion de esta con el total de vertices obtener el numero de recorridos (repito, no precisos) de cada vertice y el numero de arcos que lo conectan, adicionalmente, cabe resaltar que es posible que un mismo arco sea recorrido multiple veces, osea que seria algo similar a lo siguiente:

$$Proporcion\ = P = \frac{arcos}{vertices} = \frac{32270}{13535} = 2.38$$

$$O(\sum_{i=1}^{v}\sum_{j=1}^{c} k_i)$$

$$Donde:$$

$$k_i = numero\ de\ arcos\ de\ un\ camino\ c,$$

Lo anterior puede ser expresado de forma no precisa por medio de la proporcion P de la siguiente forma:

$$O(\sum_{i=1}^{v} 2.38 * c)$$

$c = numero\ de\ caminos\ totales,\ suponiendo\ por\ media\ aritmetica\ que\ tienen\ el\ mismo\ numero\ de\ arcos.$

Teniendo en cuenta lo anterior, no existe alguna forma de aproximar a un valor realista el numero total de caminos de cada vertice, por lo que podriamos decir que esta es la complejidad que podria estar relacionada con un tiempo de 31.46.

Asimismo, por el tipo de datos no dirigidos podriamos decir que debido a que los datos no son dirigidos y de que el grafo no tenga conexiones fuertes, entonces podemos inferir que los valores de esta sumatoria podrian ser posiblemente mayor o igual a n.

e) De acuerdo a la carga de datos se reporta que

$$V = 13535,$$
$$E = 32270,$$

adiconalmente, se aprecia que en la linea 66 del model se decide crear el graph con tipo dirigido, es decir, de arcos con un solo sentido, por lo que la formula para calcular la densidad de graphs dirigidos es la siguiente:

$$D = \frac{|E|}{|V|(|V|-1)},$$
$$D = \frac{32270}{13535(13534)} = 0.00018.$$

Por lo tanto se percibe una densidad menor a 0.3, lo cual nos indica que se trata de un grafo disperso y que no cuenta con una fuerte coneccion de arcos.

f) De acuerdo a la lineal 66 de model.py, se observa que el parametro size=14000, por lo que ese es el tamaño inicial del grafo.

g) De acuerdo a la linea 66 de model.py y a la implementacion realizada en el archivo graphstructure.py de la libreria ADT, se observa que el tipo de estructura de datos es un ADJ_LIST, la cual hace referencia a una forma de representar los grafos por medio de listas de adyacencias.

h) De acuerdo a la linea 66 de model.py, la funcion de comparacion utilizada es compareStopIds, la cual se encarga de comparar las estaciones por medio de sus paradas.