OBSERVACIONES DEL LA PRACTICA

Estudiante 1: Paola Campiño Cod 202020785

Estudiante 2: Santiago Ballén Cod 202023544

Links:

Reto 4: https://github.com/EDA2021-1-SEC07-G-2Grupo/Reto4-S11-G02.git

Lab10: https://github.com/EDA2021-1-SEC07-G-2Grupo/ISIS1225-SampleGraph-S10-G02.git

Preguntas de análisis

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

Se puede modificar el tamaño máximo de recursión mediante la función `sys.setrecursionlimit(n).`

*n siendo el nuevo límite de recursión.

En el programa se presenta de la siguiente forma:

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

Información recuperada de:

https://uniwebsidad.com/libros/algoritmos-python/capitulo-18/limitaciones

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

La razón por la cual se está haciendo este cambio es por que se necesita que el programa haga más de 1000 recursiones. Usualmente si el programa llega a hacer cerca de 1000 este va a lanzar error y el programa no va a seguir, pero al más recursivo este programa es necesario hacer que el limite sea más grande.

Información recuperada de:

https://uniwebsidad.com/libros/algoritmos-python/capitulo-18/limitaciones

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

El limite de recursión por deafult en python es de 1000 llamadas recursivas.

Información recuperada de:

https://uniwebsidad.com/libros/algoritmos-python/capitulo-18/limitaciones

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

En la operación 4 se está calculando el camino más corto con el costo minimo a cada una de las paradas. La relación con los vertices y arcos está en que a mayor cantidad de vertices y arcos más caminos hay que calcular, por lo que el tiempo de respuesta de la operación 4 es mayor.

De 75009-10

Para hacer el conteo del tiempo se usó la función time.procestime()

Tamaño del Archivo	Vertices (totales)	Arcos (totales)	Tiempo de ejecución (opción 4)
14,000	13535	32270	63043ms
10,000	9767	22758	39578.12ms
7,000	6829	15334	12296.87ms
3,000	2922	5773	4031.25ms
2,000	1954	3560	2406.25ms

E. ¿Qué características tiene el grafo definido?

La caracteristica principal de el grafo definido o grafos dirigidos está en que los arcos son unidireccionales, lo que quiere decir es que los arcos solo apuntan a una dirección.

F. ¿Cuál es el tamaño inicial del grafo?

En un principio hay 13535 vertices y 32270 arcos en el grafo y en total 30 componentes conectados en el caso del archivo con 14,000 elementos. El tamaño es mayor que cuando se pide un camino especifico.

G. ¿Cuál es la Estructura de datos utilizada?

En este caso se está usando un grafo con una estructura de lista de adjacencia, lo que quiere decir que se está usando una lista en la que se representa los arcos. Y para que la busqueda sea más rapida se está utilizando una tabla de hash para poder acceder a la lista de los arcos con una complejidad de O(K).

H. ¿Cuál es la función de comparación utilizada?

Para la creación del grafo se está usando la siguiente función de comparación:

```
def compareroutes(route1, route2):
    """
    Compara dos rutas
    """
    if (route1 == route2):
        return 0
    elif (route1 > route2):
        return 1
    else:
        return -1
```

y para la creación de la tabla de hash se está usando la función a continuación:

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