LABORATORIO No. 9: Introducción a Grafos

Estudiante 1: Sofia Torres Ramírez.
Código:202014872
Correo: s.torres21@uniandes.edu.co

Estudiante 2: Ana Margarita Florez Ruiz
Código: 201922242
Correo: a.florezr@uniandes.edu.co

PASO 4: Ejecutar y explorar el ejemplo

a. ¿Qué instrucción se usa para cambiar el límite de recursión de Python?
Para cambiar el límite de recursión se utiliza la instrucción:

```
sys.setrecursionlimit(10000000)
```

b. ¿Por qué considera que se debe hacer este cambio?
Este cambio debe hacerse debido a que al construir un grafo es necesario realizar muchos recorridos sobre los datos para poder entender cómo están construidos los arcos entre los nodos o construir los mismos. Es por esto que, si no se tiene un límite de recursión alto, phyton va a realizar el número de recursiones ya establecido previamente y al no llegar a un resultado todavía va a pensar que el programa se quedó en un ciclo infinito y lo detendrá automáticamente cuando esto no es cierto, mientras que si se tiene un límite de recursión alto se podrá dar finalidad al programa correctamente.

c. ¿Cuál es el valor inicial que tiene Python cómo límite de recursión?
El valor inicial de Python como límite de recursión en este caso es de 1048576 en nuestro caso.

Registre número de vértices, el número de arcos del grafo, y el tiempo que toma esta instrucción con cada uno de los archivos CSV en el documento de observaciones del laboratorio.

Tamaño del archivo	Número de vértices	Número de arcos	Tiempo
50	74	73	367.0410000000146
150	146	146	646.129000000002
300	295	382	1657.4989999999977
1000	984	1633	4419.440999999988
2000	1954	3560	14031.076000000003
3000	2922	5773	40245.86399999997
7000	6829	15334	95901.23100000001
10000	9767	22758	410746.60199999996

14000	13535	32270	559476.1170000001

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

Se puede ver que a medida que el número de vértices aumenta, el número de arcos se aleja al número de vértices. También se puede ver que a medida que el número de vértices aumenta, debido a que se hace necesario crear más arcos entre ellos, el tiempo de ejecución aumenta con diferencias cada vez más grandes.

Ahora, ejecuten el programa con cada uno de los archivos CSV. Por cada archivo, tomen nota del número de vértices, el número de arcos del grafo, y el tiempo de ejecución que toma la opción 6.

Tamaño del archivo	Número de vértices	Número de arcos	Tiempo
50	74	73	4.368000000027905
150	146	146	6.543000000007737
300	295	382	9.943999999997288
1000	984	1633	11.030000000005202
2000	1954	3560	11.873999999991725
3000	2922	5773	12.137999999985993
7000	6829	15334	13.32099999999059
10000	9767	22758	14.415999999997098
14000	13535	32270	25.3160000000463

PASO 5: Estudiar el ejemplo en VSCode

e. ¿El grafo definido es denso o disperso?, ¿El grafo es dirigido o no dirigido?, ¿El grafo está fuertemente conectado?

- El grafo definido es disperso debido a que, teniendo en cuenta que el archivo de tamaño 50 tiene 74 vértices y solo 73 arcos esto quiere decir que prácticamente hay solo un arco por vértice lo que hace que el grafo tenga relativamente pocos arcos por lo tanto es disperso. Al ejecutar la fórmula e/(v(v-1)), se aprecia que el resultado es 0,000503 el cual es un número mucho menor a 0,3 (la referencia para saber si un grafo es disperso).
- El grafo sí es dirigido.

- El grafo no está fuertemente conectado debido a que algunos vértices están muy separados de otros, y debido a lo anteriormente mencionado, hay muy pocos arcos para la cantidad de vértices como para que llegue a ser fuertemente conectado.

f. ¿Cuál es el tamaño inicial del grafo?

14000

g. ¿Cuál es la Estructura de datos utilizada?

ADJ_LIST o lista de adyacencia

h. ¿Cuál es la función de comparación utilizada?

La función de comparación utilizada es:

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