

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 contruidos los arcos entre los nodos o construir los mismos. Es por esto que, si no se tiene un límite de recursión alto, python 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 como 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	646.1290000000002
300	295	382	1657.4989999999977
1000	984	1633	4419.4409999999988
2000	1954	3560	14031.076000000003
3000	2922	5773	40245.863999999997
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	25.31600000000463

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
```

PASO 9: Cargar datos del Reto

a) ¿Cuántos grafos se necesitan definir para solucionar los requerimientos del reto? y ¿Por qué?

REQ 1 Encontrar puntos de interconexión aérea: Para este requerimiento solo se hace necesario trabajar con el grafo “dígrafo” propuesto en el enunciado del reto, el cual contiene los aeropuertos y las rutas relacionadas con esos aeropuertos.

REQ2 Encontrar clústeres de tráfico aéreo: Para este requerimiento se construirá un grafo al cual se le aplicará el algoritmo de Kosaraju para poder encontrar los componentes fuertemente conectados

REQ3 Encontrar la ruta más corta entre ciudades:

REQ4 Utilizar las millas de viajero:

REQ5 Cuantificar el efecto de un aeropuerto cerrado: Para este requerimiento se trabajará con el grafo “dígrafo” propuesto en el enunciado del reto, el cual contiene los aeropuertos y las rutas relacionadas con esos aeropuertos.

b) ¿Cuáles son las características específicas de cada uno de los grafos definidos? (vértices, arcos, denso o disperso, dirigido o no dirigido).

REQ1 Encontrar puntos de interconexión aérea: El grafo de este requerimiento tendrá como vértices los aeropuertos y como arcos las rutas que existen entre los aeropuertos, este grafo será denso debido a que de un aeropuerto puede salir muchas rutas hacia otros aeropuertos y será dirigido.

REQ2 Encontrar clústeres de tráfico aéreo: El grafo de este requerimiento tendrá como vértices los aeropuertos y como arcos las rutas entre estos aeropuertos, este grafo será denso debido a que de un aeropuerto pueden salir muchas rutas hacia otros aeropuertos y será no dirigido.

REQ3 Encontrar la ruta más corta entre ciudades: El grafo de este requerimiento en especial tendrá como vértices las ciudades y como arcos las rutas entre estas, este grafo será denso debido a que de una ciudad puede haber diferentes rutas hacia otras y será no dirigido.

REQ4 Utilizar las millas de viajero: El grafo de este requerimiento tendrá como vértices las ciudades y como arcos la cantidad de millas disponibles que tiene el viajero, este grafo será denso debido a que se pueden tener diferentes combinaciones de ciudades con las millas usadas y será no dirigido.

REQ5 Cuantificar el efecto de un aeropuerto cerrado: El grafo de este requerimiento tendrá como vértices los aeropuertos y como arcos las rutas que existen entre los aeropuertos, este grafo será denso debido a que de un aeropuerto puede salir muchas rutas hacia otros aeropuertos y será dirigido.

c) Además de los grafos, ¿Qué otras estructuras de datos adicionales se necesitan para resolver los requerimientos? Y ¿Por qué?

REQ 1 Encontrar puntos de interconexión aérea: Además del grafo se utilizará un RBT el cual tendrá como llave los aeropuertos y como valor, el número de rutas que salen de este aeropuerto para así poder saber cuales aeropuertos son los que unen más rutas, y adicional a

esto, se creará una lista en la cual se almacenarán los datos de los aeropuertos con más conexiones.

REQ2 Encontrar clústeres de tráfico aéreo: No se utilizará ninguna otra estructura.

REQ3 Encontrar la ruta más corta entre ciudades: Este requerimiento además puede usar un RBT que tendrá como llave las ciudades y como valor la distancia entre las ciudades para así poder encontrar la ruta más corta entre las ciudades.

REQ4 Utilizar las millas de viajero: No se usará otra estructura.

REQ5 Cuantificar el efecto de un aeropuerto cerrado: Además del grafo se utilizará tres listas en las cuales se agregarán los vuelos de salida afectados, vuelos de entrada afectados y ciudades afectadas.