**OBSERVACIONES DEL LA PRACTICA**

Juan Sebastián Ortega Romero | Cod 202021703

Yesid Camilo Almanza |Cod 201921773

# **Preguntas de análisis**

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

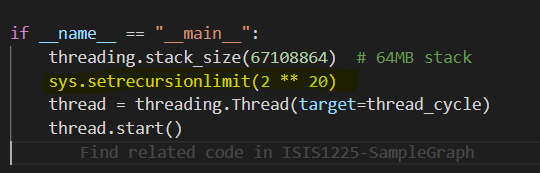


Figura 1. Cambio del límite de recursión

Como se puede observar en la Figura 1, la función **sys.setrecursionlimit()** es la encargada de modificar el límite de recursión de Python. En el caso de este laboratorio se utiliza el valor de 2\*\*30 (1048576) como nuevo límite de recursión.

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

Este cambio es necesario debido a que varios algorítmos de búsqueda dentro de un grafo se basan en la recursión vertice a vertice para lograr a travesar todas las rutas disponibles. Por lo tanto, si se tiene una gran cantidad de vertices, va a ser necesario establecer un límite de recursión mayor a esta cantidad para asegurarse que todos los vertices han sido evaluados en la búsqueda. Un ejemplo de esto se encuentra en el algortimo DFS (Depth First Search) donde a partir de un vertice raiz se analizan los vertices adyacentes y luego el algoritmo es aplicado de forma recursiva sobre estos mismos. En este laboratorio, al utilizar este cambio se obtiene cerca de 104757.6 % más de capacidad recursiva.

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

Es necesario aclarar que el límite de recursión inicial en Python es de **1000 llamados** recursivos.

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Archivo** | **Número de vertices** | **Número de arcos** | **Tiempo de la operación 4 (ms)** |
| bus\_routes\_50.csv | 73 | 74 | 30.67 |
| bus\_routes\_150.csv | 146 | 146 | 41.23 |
| bus\_routes\_300.csv | 295 | 382 | 70.48 |
| bus\_routes\_1000.csv | 984 | 1633 | 405.45 |
| bus\_routes\_2000.csv | 1954 | 3560 | 1392.62 |
| bus\_routes\_3000.csv | 2922 | 5773 | 1987.48 |
| bus\_routes\_7000.csv | 6829 | 15334 | 5844.51 |
| bus\_routes\_10000.csv | 9767 | 22758 | 19265.75 |
| bus\_routes\_14000.csv | 13535 | 32270 | 35856.872 |

**Tabla 1:** Número de vértices y arcos contra el tiempo de la operación 4

Como se puede observar en la Tabla 1, existe una evidente correlación entre el tiempo de ejecución de la operación 4 y la cantidad de vértices con sus respectivos arcos que son cargados en el programa. A medida que aumenta el número de vértices también aumenta el tiempo de la operación. Esto puede ser debido al hecho de que para construir un grafo siempre será necesario que primero se visiten todos sus nodos, por lo tanto, entre más grande el grafo, mayor durará cualquier algoritmo en recorrerlo y construirlo a partir de un nodo raíz.

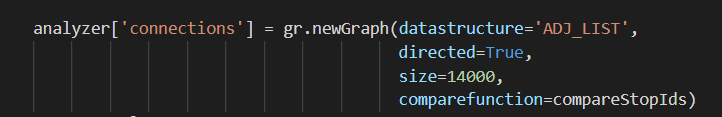
1. Tiempo de ejecución de la operación 6

|  |  |  |  |
| --- | --- | --- | --- |
| **Archivo** | **Número de vertices** | **Número de arcos** | **Tiempo de la operación 6 (ms)** |
| bus\_routes\_50.csv | 73 | 74 | 14.97 |
| bus\_routes\_150.csv | 146 | 146 | 17.51 |
| bus\_routes\_300.csv | 295 | 382 | 17.89 |
| bus\_routes\_1000.csv | 984 | 1633 | 16.45 |
| bus\_routes\_2000.csv | 1954 | 3560 | 19.29 |
| bus\_routes\_3000.csv | 2922 | 5773 | 21.14 |
| bus\_routes\_7000.csv | 6829 | 15334 | 21.91 |
| bus\_routes\_10000.csv | 9767 | 22758 | 33.80 |
| bus\_routes\_14000.csv | 13535 | 32270 | 35.37 |

**Tabla 2:** Número de vértices y arcos contra el tiempo de la operación 6

Como se puede observar en la tabla 2, el tiempo de la operación 6 también aumenta a medida que aumenta la cantidad de vertices y arcos. Sin embargo, a comparación de la operación 4 su aumento es mínimo por lo que puede que no esté tan relacionado con esta caracterísitica. Por el contrario, el tiempo de operación parece determinado por la longitud del camino entre ambos vertices, la cual no necesariamente depende de la cantidad total de vertices ya que dos vertices pueden estar alejador por la misma longitud sin importar que tan grande sea la base de datos.

1. ¿Qué características tiene el grafo definido?



**Figura 2.** Construcción de grafo en el codigo

Como se puede observar en la Figura 2, el grafo utilizado para desarrollar esta práctica de ejemplo es un **grafo dirigido**, lo cual está establecido por medio del parámetro *directed* de la función gr.newGraph. Los grafos dirigidos o definidos cumplen con las siguientes características:

* Los vértices apuntan hacia una única dirección definida.
* No existe una relación simétrica sobre el conjunto de vértices.
* Las conexiones entre vértices se agrupan en parejas de tipo (a,b)

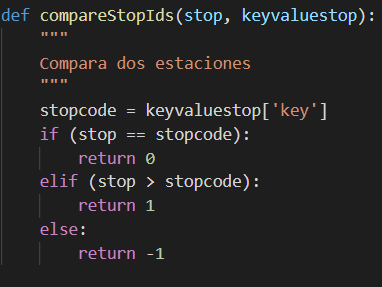
1. ¿Cuál es el tamaño inicial del grafo?

El tamaño inicial de un grafo hace referencia al número de conexiones o arcos que posee el grafo. En este caso el parámetro *size* de la función para instanciar grafos vista en la Figura 2 nos permite asumir que se está iniciando un grafo con disponibilidad para **14,000** arcos.

1. ¿Cuál es la Estructura de datos utilizada?

Como lo indica el parámetro *datastructure,* se está utilizando una **lista de adyacencia** para modelar el grafo.

1. ¿Cuál es la función de comparación utilizada?



**Figura 3:** Función de comparación

La función de comparación utilizada es **compareStopIds** la cual se encarga de comparar individualmente el valor que tiene cada parada única, es decir, el valor exclusivo de cada vértice también entendido como su llave. De esta forma, el grafo puede diferenciar a todos sus vértices en el momento que se necesite operar sobre estos.