**OBSERVACIONES DEL LA PRACTICA**

# Camilo Ortiz Cruz Cod: 201821615

# Kevin Fernando Gomez Camargo Cod: 202015120

# **Preguntas de análisis**

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

La instrucción utilizada para cambiar el límite de recursión de Python se encuentra el final del documento **view.py** y es la siguiente

Texto

Descripción generada automáticamente

Especificamente la función **setrecursionlimit** del módulo **sys** es la que permite realizar el cambio en el límite de recursión.

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

Este cambio en el límite de recursión de Python se debe hacer porque comunmente es necesario hacer uso de funciones muy recursivas para solucionar determinados problemas. Algunos ejemplos de estas funciones son los algoritmos recursivos de ordenamiento como MergeSort cuando hay una gran cantidad de elementos y en el caso de los grafos el algoritmo Depth-First Search hace una busqueda recursiva para encontrar rutas entre dos puntos al igual que otros algoritmos de grafos ya que esta estructura tiende a recorrese de forma recursiva.

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

El valor inicial que tiene Python como límite de recursión, es decir, el número de veces que una función recursiva puede llamarse a sí misma, generalmente está fijado en **1000 llamadas recursivas.**

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

|  |  |  |  |
| --- | --- | --- | --- |
| **DOCUMENTO CSV** | **VERTICES** | **ARCOS** | **TIEMPO 4 (ms)** |
| **50** | 74 | 73 | 57 |
| **150** | 146 | 146 | 62 |
| **300** | 295 | 382 | 136 |
| **1000** | 984 | 1633 | 500 |
| **2000** | 1954 | 3560 | 1955 |
| **3000** | 2922 | 5773 | 3276 |
| **7000** | 6829 | 15334 | 12464 |
| **10000** | 9767 | 22758 | 27672 |
| **14000** | 13535 | 32270 | 48772 |

De acuerdo a los valores reportados en la tabla se realizan las gráficas

La relación entre el número de vértices y el número de arcos es casi lineal.

La relación entre el número de vértices y el tiempo de la operación 4 tiene un compartamiento cercano a uno cuadratico.

Debido a que la relación entre el número de vértices y el número de arcos es casi lineal, la relación entre el número de arcos y el tiempo de la operación 4 tambien debería tener un compartamiento cercano a uno cuadratico. Esto se comprueba a continuación:

|  |  |  |  |
| --- | --- | --- | --- |
| **DOCUMENTO CSV** | **VERTICES** | **ARCOS** | **TIEMPO 6 (ms)** |
| **50** | 74 | 73 | 2,43 |
| **150** | 146 | 146 | 2,38 |
| **300** | 295 | 382 | 2,55 |
| **1000** | 984 | 1633 | 2,37 |
| **2000** | 1954 | 3560 | 3,40 |
| **3000** | 2922 | 5773 | 2,84 |
| **7000** | 6829 | 15334 | 2,96 |
| **10000** | 9767 | 22758 | 3,99 |
| **14000** | 13535 | 32270 | 7,31 |

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

El grafo es dirigido porque las rutas que siguen los buses tienen un sentido especifico entre estaciones. Por ende, tomando el documento más grande y sabiendo que el grafo es dirigido, la densidad del grafo es

1

Debido a que y cercano a 0 entonces se puede decir que el grafo es disperso.

Por otro lado, el grafo esta fuertemente conectado ya que por definición un grafo fuertemente conectado es aquel que para todo para de vertices existe un camino desde y de 1 y dado que el grafo representa las estaciones de una ruta de buses en teoria desde cualquier estación puedo llegar a cualquier otra pues este es el objetivo de una ruta de buses, no obstante puede que en el caso de algunos ejemplos del documento esto no suceda dado que tenemos una muestra de la ruta por lo que puede pasar que no se pueda llegar a cualquier estación, sin embargo, en terminos del problema el grafo de debería ser fuertemente conectado.

1. ¿Cuál es el tamaño inicial del grafo?

El tamaño inicial del grafo es definido como parametro según se muestra a continuación en el archivo **model.py** en la función **newAnalyzer()**

Escala de tiempo

Descripción generada automáticamente con confianza media

El parametro **size**  de la función **newGraph** del modulo **DISClib.ADT.graph** **(importado como gr)** es el tamaño inicial del grafo, que en este caso es de 14000.

1. ¿Cuál es la Estructura de datos utilizada?

La estructura de datos utilizada para la implementación del grafo según se muestra a continuación en el archivo **model.py** en la función **newAnalyzer()** es **ADJ\_LIST** que corresponde a un arreglo de **listas de adyacencia.** La estructura de datos se debe pasar como argumento para la generación del grafo con la función **newGraph**

Escala de tiempo

Descripción generada automáticamente con confianza media

1. ¿Cuál es la función de comparación utilizada?

La función de comparación utilizada en la implementación del grafo según la siguiente sección de codigo donde se crea el grafo en el archivo **model.py** en la función **newAnalyzer()**,

Escala de tiempo

Descripción generada automáticamente con confianza media

corresponde a la función llamada **compareStopIds,** que compara dos estaciones, definida al final del archivo **model.py** como

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

**Referencias:**

1. Hevia G. Sara (2019) (Pg. 9-10). Búsqueda de comunidades en grafos ponderados. Detección de tramas de blanqueo de capitales. Universidad Politécnica de Madrid. Recuperado de:

<https://oa.upm.es/62942/1/TFM_SARA_GARCIA_HEVIA.pdf>