**OBSERVACIONES DEL LA PRACTICA**

María José Sáenz Rodríguez Cod 202013542

Iván Camilo Ballén Méndez Cod 202011440

# **Preguntas de análisis**

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

Al final del view se usa la instrucción sys.setrecursionlimit(2\*\*20) la cual asigna como límite de recursión el valor entero que entra por parámetro.

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

Durante la consulta es evidente que se usa el algoritmo de Kosaraju el cual a su vez usa el algoritmo Depth First Order el cual a su vez usa el recorrido DFS que es un recorrido iterativo. Si el grafo es muy grande, entonces es posible que se alcance el límite de recursión por default de Python y por lo tanto se interrumpa la ejecución del requerimiento que utiliza el algoritmo de Kosaraju.

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

El valor límite de recursión de Python por default es 1000.

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

|  |  |  |  |
| --- | --- | --- | --- |
| Núm. Datos leídos | Vértices | Arcos | Tiempo [s] |
| 50 | 74 | 73 | 0.022 |
| 150 | 146 | 146 | 0.033 |
| 300 | 295 | 382 | 0.059 |
| 1000 | 984 | 1633 | 0.244 |
| 2000 | 1954 | 3560 | 0.890 |
| 3000 | 2922 | 5773 | 1.675 |
| 7000 | 6829 | 15334 | 5.846 |
| 10000 | 9767 | 22758 | 16.244 |
| 14000 | 13535 | 32270 | 20.675 |

Es claro que, con pocos datos, la cantidad de vértices y arcos es aproximadamente la misma siendo un caso muy parecido al de la estructura SINGLE\_LINKED, y precisamente al ser tan pocos datos (note que prácticamente solo hay una ruta que recorre los vértices) es muy fácil encontrar la ruta más corta (prácticamente la única ruta que hay). Sin embargo, a medida que crecen los datos leídos por el programa, vemos que hay cada vez más rutas (más arcos que vértices) y precisamente por esto es más difícil encontrar la ruta más corta entre una estación y otra. Por lo tanto, concluimos que la complejidad de búsqueda de la ruta mínima por el algoritmo de Dijkstra crece proporcionalmente a la cantidad de arcos y vértices (o lo que es lo mismo a la cantidad de rutas registradas en el programa).

A continuación, presentamos la tabla del requerimiento 6.

|  |  |  |  |
| --- | --- | --- | --- |
| Núm. Datos leídos | Vértices | Arcos | Tiempo [s] |
| 50 | 74 | 73 | 0.014 |
| 150 | 146 | 146 | 0.017 |
| 300 | 295 | 382 | 0.019 |
| 1000 | 984 | 1633 | 0.017 |
| 2000 | 1954 | 3560 | 0.023 |
| 3000 | 2922 | 5773 | 0.042 |
| 7000 | 6829 | 15334 | 0.023 |
| 10000 | 9767 | 22758 | 0.024 |
| 14000 | 13535 | 32270 | 0.030 |

1. ¿Qué características tiene el grafo definido?

Además de las características que se mencionan en los siguientes puntos, los grafos dirigidos permiten tener un orden debido a que respetan las prioridades o las precedencias para poder llegar de un vértice a otro. En el caso del grafo ‘connections’, se respeta el recorrido que se realiza de una estación a otra ya que se tienen en cuenta las rutas que se manejan en cada estación y debido a esto se define el sentido del recorrido por el grafo.

1. ¿Cuál es el tamaño inicial del grafo?

El tamaño inicial del único grafo ‘connections’ que se encuentra en el analyzer es de 14000 elementos.

1. ¿Cuál es la Estructura de datos utilizada?

La estructura de datos utilizada para el grafo es una lista de adyacencias ‘ADJ\_LIST’, ya que solo se busca guardar la información importante del archivo de rutas y tener un menor espacio al hacerlo.

1. ¿Cuál es la función de comparación utilizada?

Se utiliza compareStopIds como función de comparación, donde básicamente compara dos estaciones.