**OBSERVACIONES DEL LA PRACTICA**

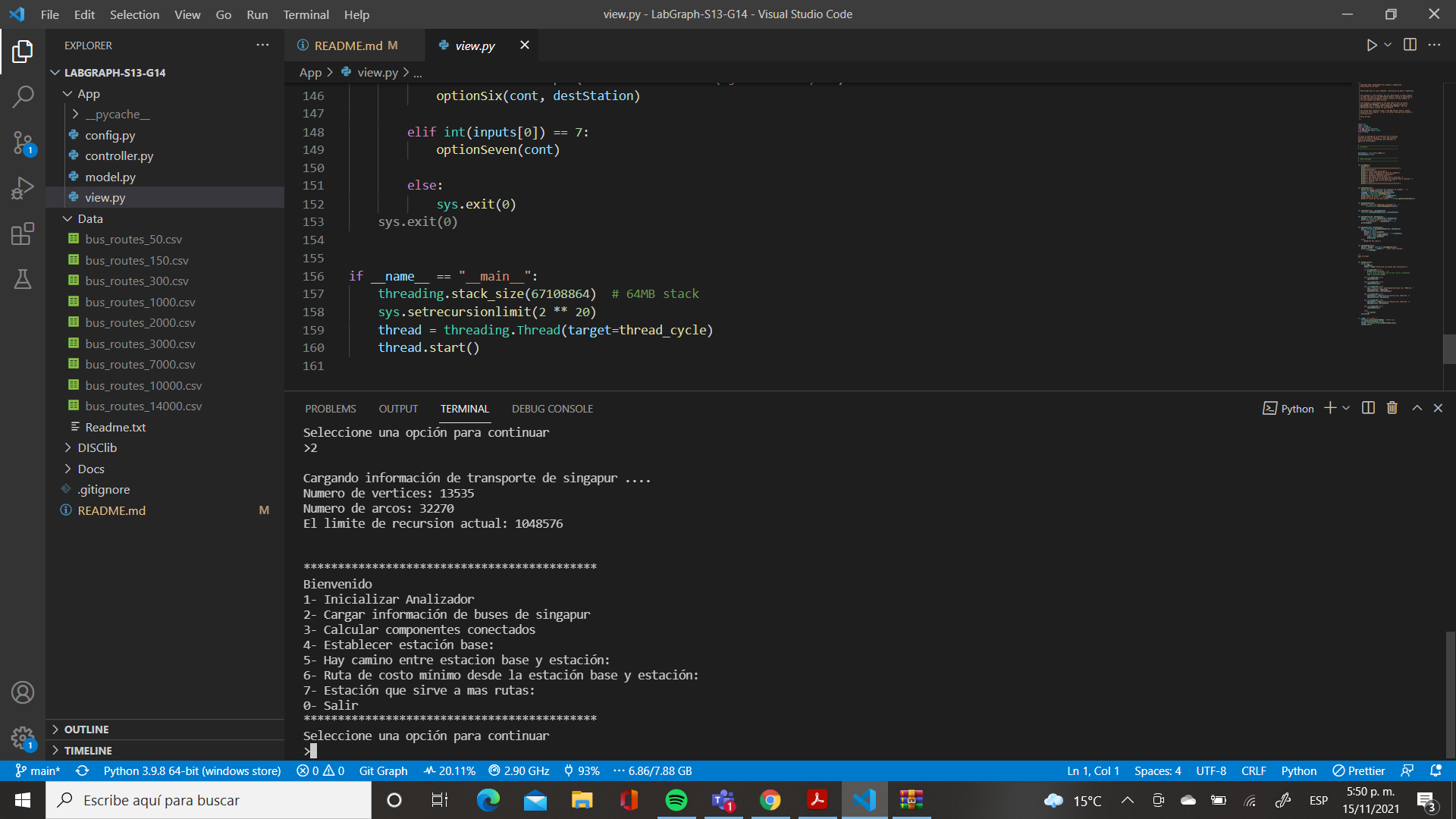
Diego Alejandro González Vargas Cod 202110240

Sebastián Guerrero Ríos Cod 202021249

# **Preguntas de análisis**

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

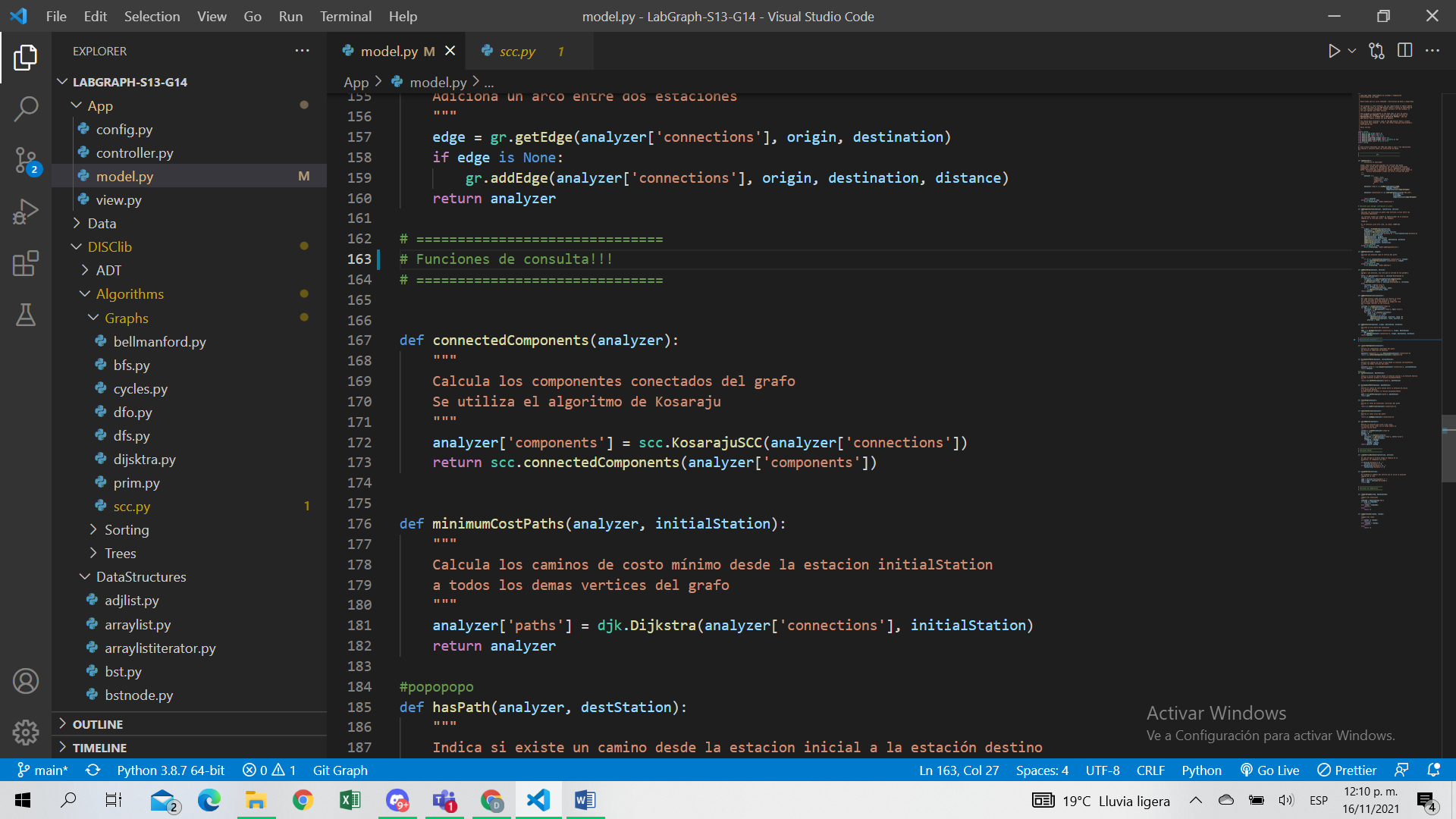
RTA/ Para poder dar respuesta a esta pregunta, se debe tener en cuenta lo enunciado en el archivo view.py, ya sobre las últimas lineas del mismo, donde se encuentra lo siguiente:



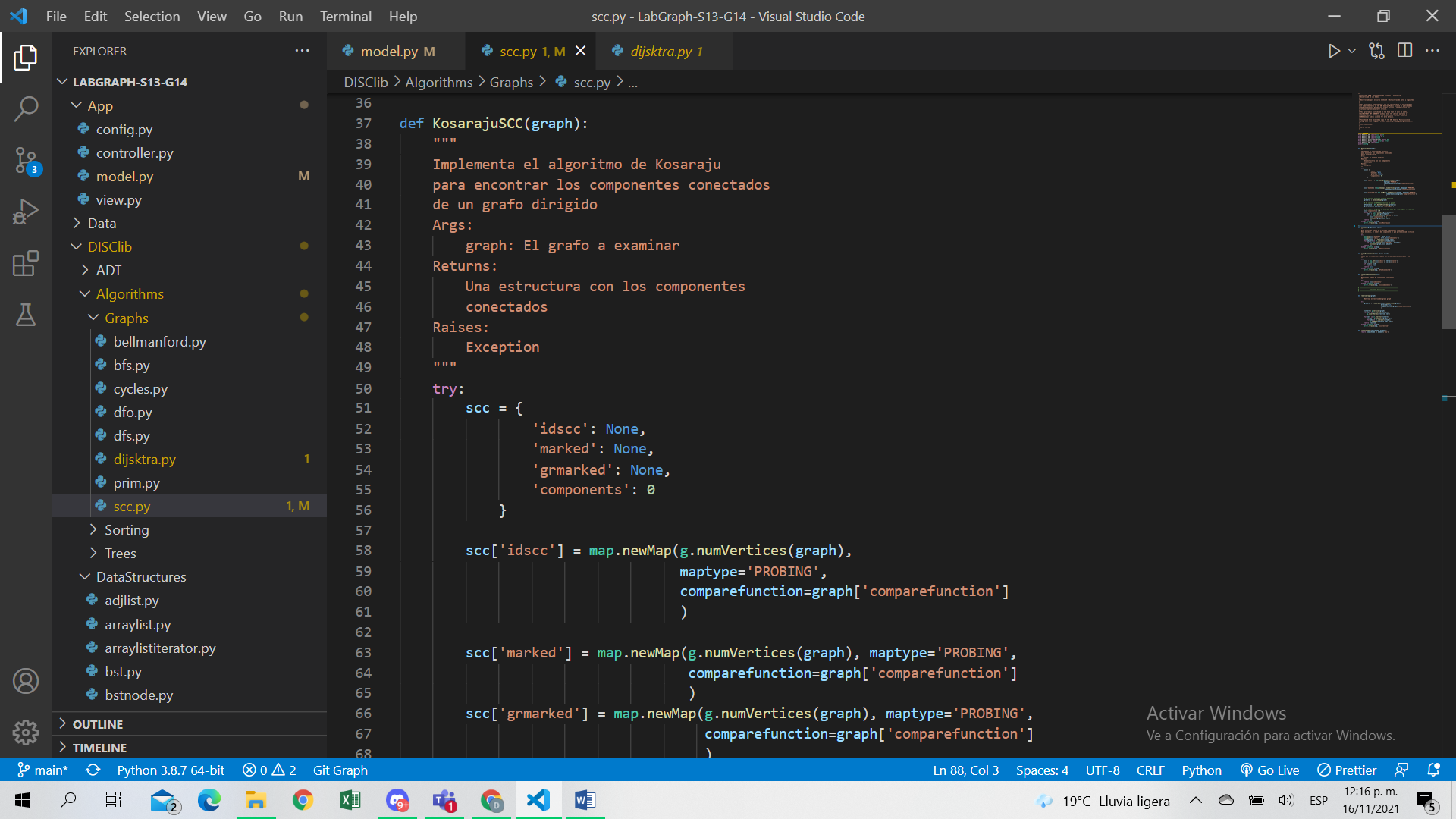
Esta función se encarga explicitamente de cambiar el limite de recursión establecido por python. En este sentido, se puede afirmar entonces que la linea 158 del código o la segunda de la función, es la encargada de definir el límite de recursión en 1048576. Es importante además resaltar que la cifra de recursión escogida es completamente arbitraria y no va asociada a la cantidad de datos cargados, ni a la cantidad de vertices o arcos que tenga el grafo construido. Así las cosas, vale la pena finalmente concluir resaltando el carácter entero de las recursiones, necesario por la misma definición del concepto

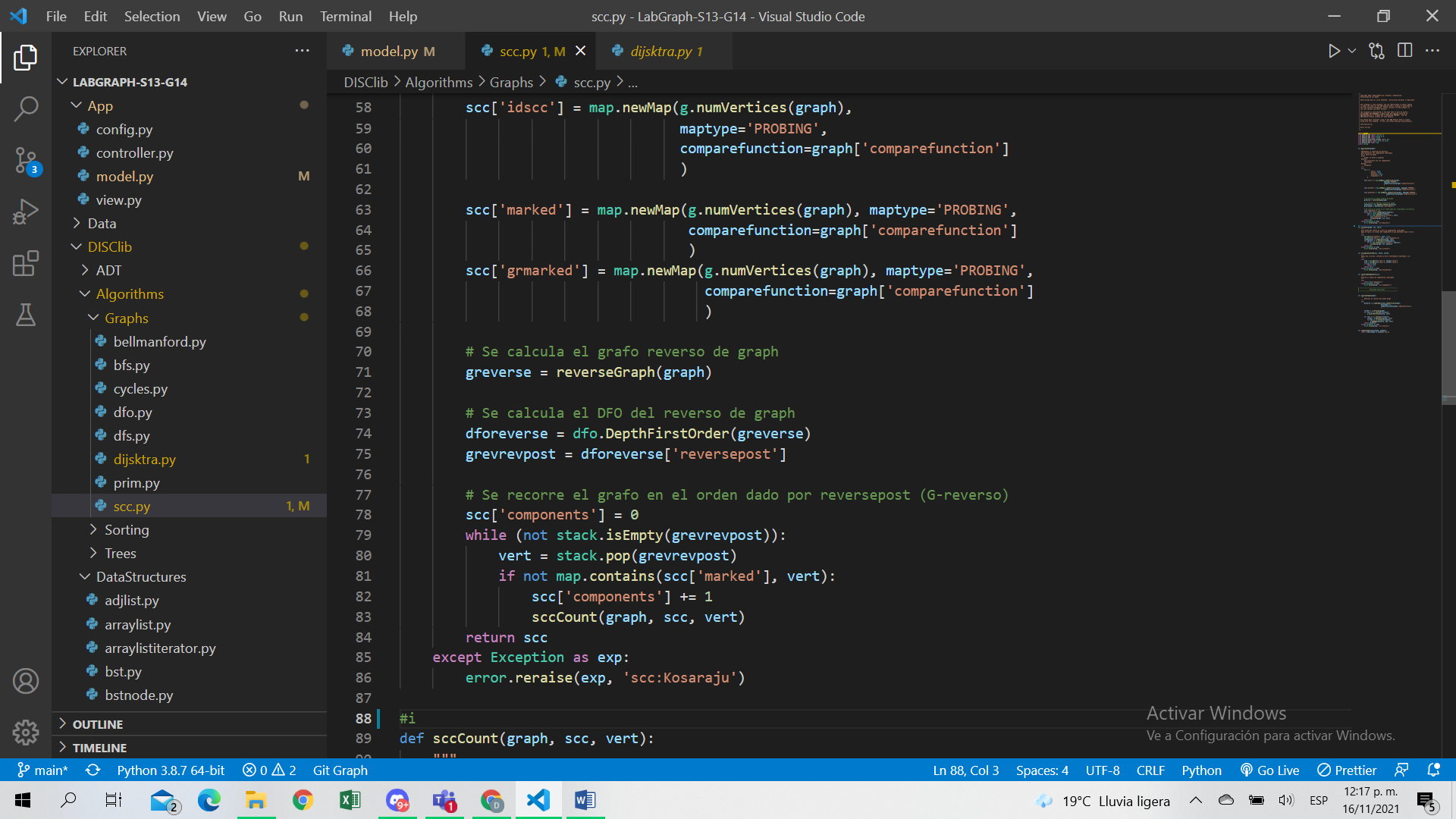
1. ¿Por qué considera que se debe hacer este cambio?

RTA/ Para poder dar repuesta a esta pregunta se hace necesario el analisis teórico de cada uno de los requerimientos que tiene la pestaña de view. En este sentido, se pueden encontrar algunas funciones de consulta de datos específicos dentro del grafo. Estas consultas, teóricamente hablando, en un gráfo dirigido, puede requerir del uso de funciones recursivas para recorrer el archivo. Así las cosas, se pueden llegar a ejecutar funciones como DFS(Depth First Search), para hacer recorridos dentro del grafo de manera recursiva, y para volumenes de datos elevados como el trabajado en este caso, los limites de recursividad nativos de python pueden llegar a ser insufientes.

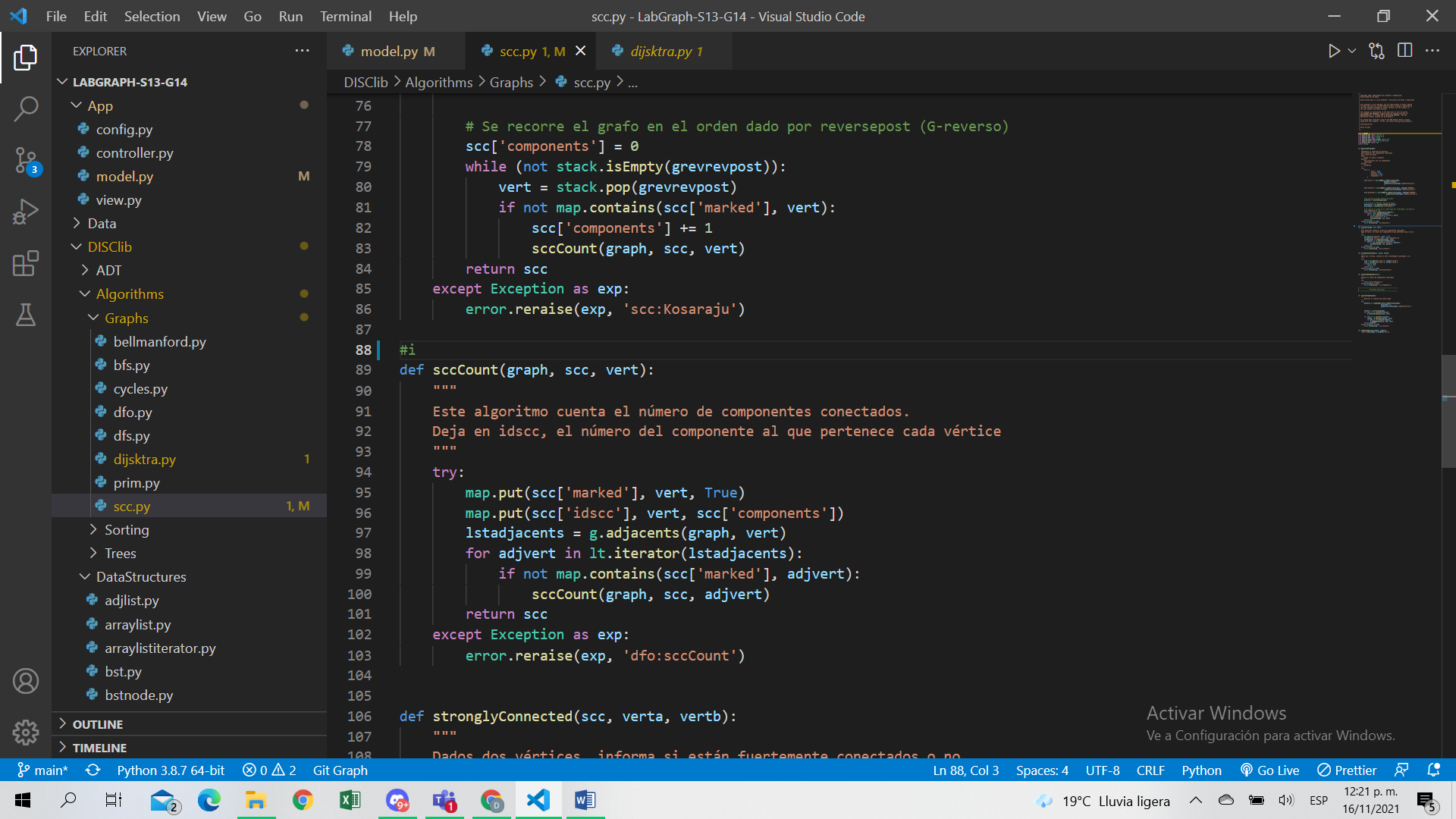
Por otra parte, para poder comprobar la existencia de estos algoritmos teóricos, se tiene que analizar un ejemplo dentro del código trabajado. En consecuencia, se puede tomar como ejemplo el caso de la función de consulta connectedComponents que se muestra a continuación:

Esta función utiliza la otra dentro del API DISClib para poder ejecutar su acción de halalr los componentes conectados dentro del gráfico. Así las cosas, el algoritmo Kosajaru debe ser tenido en cuenta como uno de especial mención dentro del API y hallar su recursividad:





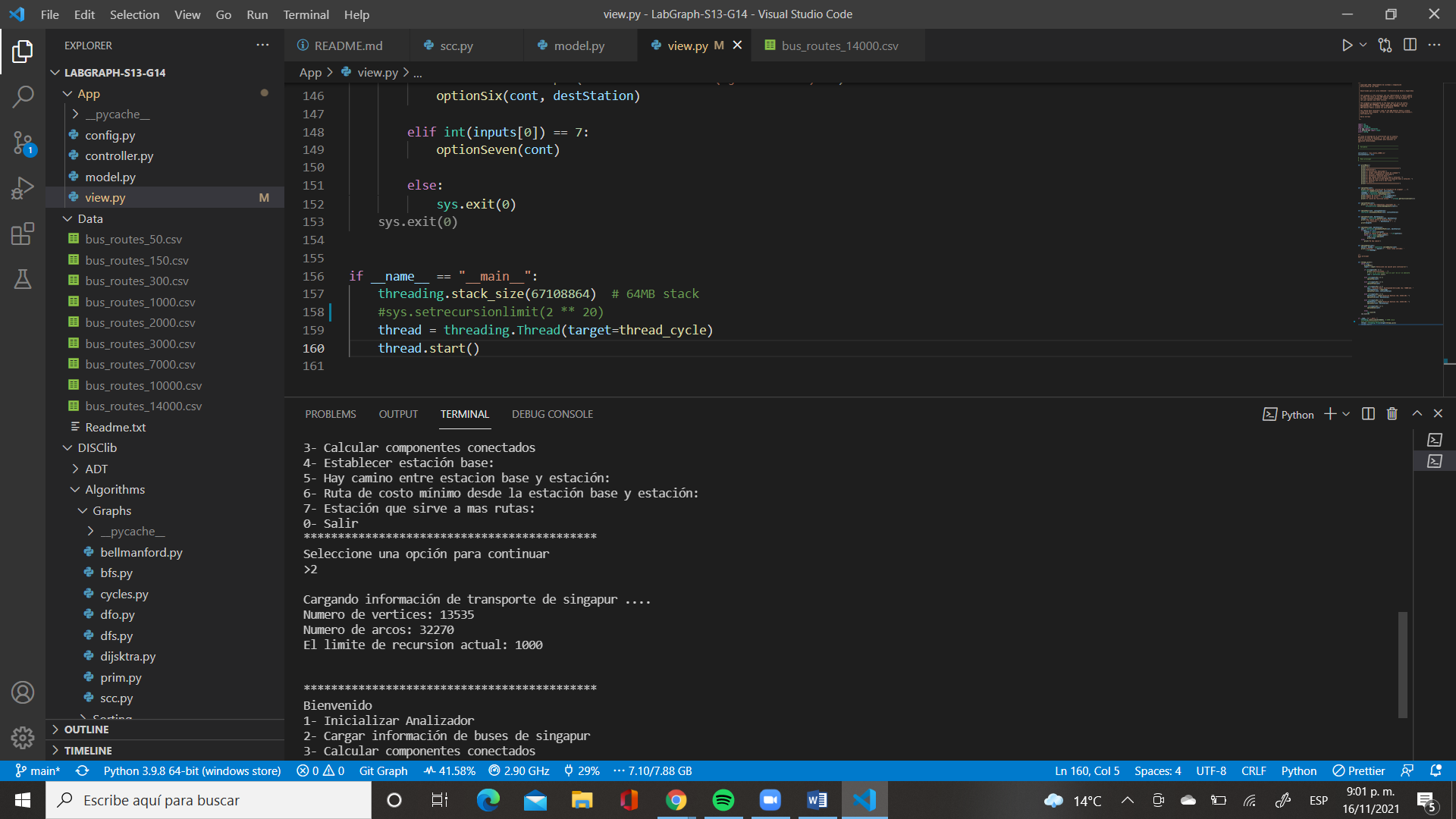
A simple vista, se puede observar cómo la función realiza un conteo simple de todos elementos conectados a partir de un sistema de marcado de elementos de conteo de las conexiones de los mismos. Sin embargo, si se ahonda en la función de conteo sccCount(), se encuentra la recursividad de la función, como sigue:



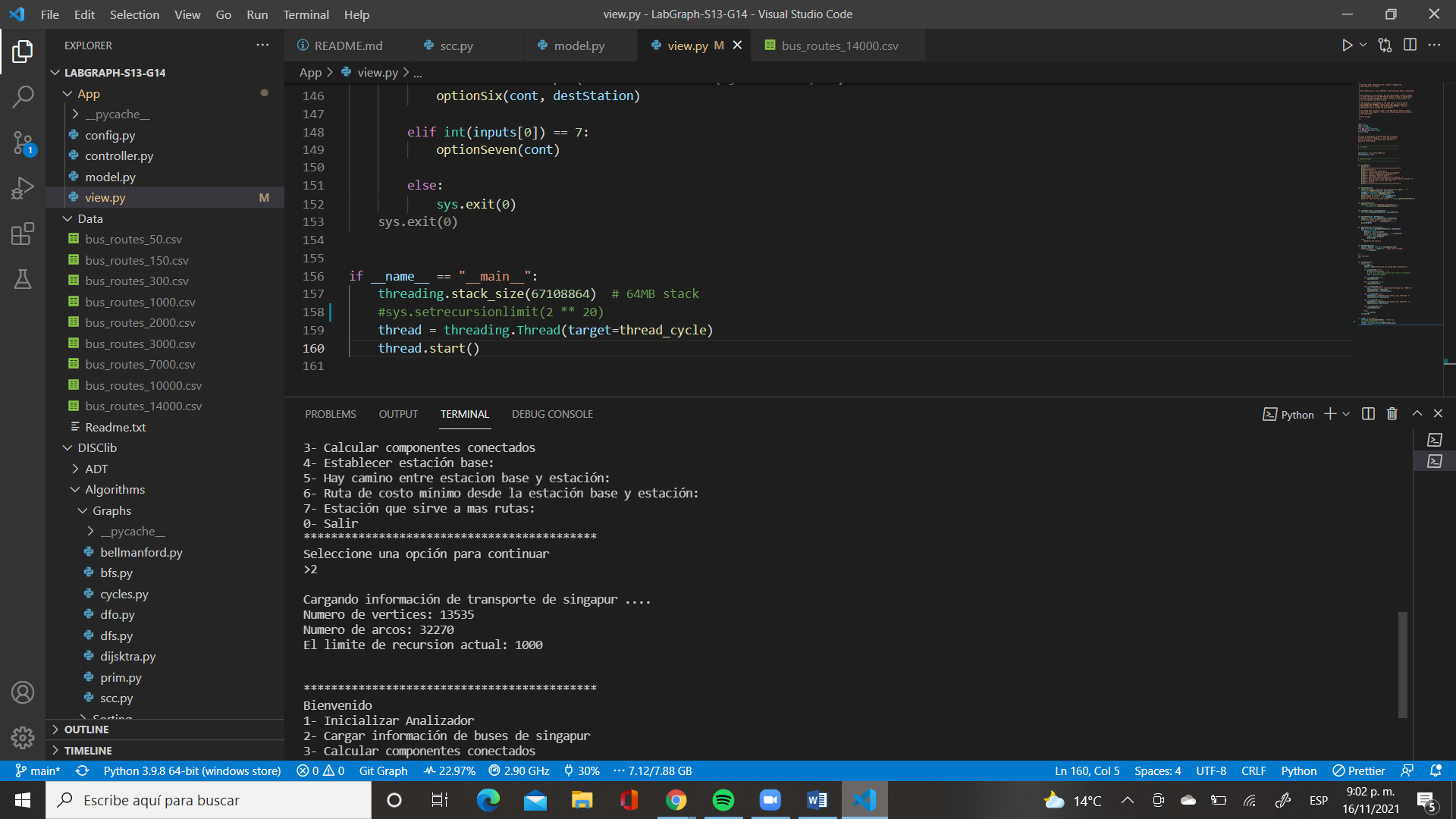
Como se observa, esta función tiene como su eje de ejecución un llamado continuo a sí misma a partir de la implementación de un ciclo for. Esta recursividad puede verse afectada por los limites nativos de python.

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

RTA/ Para poder dar respuesta a esta pregunta, se considero necesaria la eliminación temporal de la funcion de reasignación del limite de recursiones, con el fin de poder establecer de manera práctica el limite que se tiene por defecto en la consola, y establecer el mismo como respuesta a esta pregunta. Así las cosas , se realiza el siguiente cambio:



Posteriormente, se ejecutó el programa con los siguientes resultados:



De este modo, se puede concluir que el límite inicial de recurisiones en python es de 1000 recursiones

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ARCHIVO | # DE VERTICES | # DE ARCOS | TIEMPO OPCIÓN 4 (seg) | TIEMPO OPCIÓN 6 (seg) |
| bus\_routes\_14000 | 13535 | 32270 | 25.59375 | 0.03125 |
| bus\_routes\_10000 | 9767 | 22758 | 14,75 | 0.015625 |
| bus\_routes\_7000 | 6829 | 15334 | 6.25 | 0.015625 |
| bus\_routes\_3000 | 2922 | 5773 | 1.859375 | 0.046875 |
| bus\_routes\_2000 | 1954 | 3560 | 0.953125 | 0.0 |
| bus\_routes\_1000 | 984 | 1633 | 0.359375 | 0.0 |
| bus\_routes\_300 | 295 | 382 | 0.0625 | 0.0 |
| bus\_routes\_150 | 146 | 146 | 0.03125 | 0.0 |
| bus\_routes\_50 | 74 | 73 | 0.03125 | 0.0 |

RTA/ Para poder dar respuesta a esta pregunta se hace necesaria la consideración de los tiempos hallados para la función Dijkstra con los diferentes volumenes de datos obtenidos, como se vió anteriormente. Adicionalmente, se considera prudente el análisis de complejidad del algoritmo para determinar su relación con la cantidad de datos y arcos. En este sentido, Dijkstra (1959) presenta que en cada uno de sus pruebas determino que, para grafos como el trabajado en este laboratorio, se puede tener, en promedio, los siguientes tiempos para su función:

*O*(|*V*|²+|A|)

En esta fórmula, es importante mencionar que la |V| hace referencia a la cantidad de vertices y |A| a la cantidad de arcos. Finalmente, se propone que se debe hacer una simplificación de la expresión teniendo en cuenta que la variable de |V| elevadad a la 2 será eminentemente superior a la |A|, por lo que se puede expresar de la siguiente manera:

*O*(|*V*|²)

Finalmente, esta fórmula debe ser contrastada con los datos obtenido, por lo que se hizo necesaria una graficación de los mismos, así:

Como se puede ver, la gráfica efectivamente describe un comportamiente cuadrático referente a la cantidad de vertices del grafo utilizado

1. ¿El grafo definido es denso o disperso?

RTA/ Para poder determinar la densidad de un grafo, se hace necesaria la conceptualización de densidad. En este sentido, Preiss(1998) propone que la densidad hace referencia a la proporción de arcos que posee un grafo en relación con los vertices del mismo. Adicionalmente, una vez determinado el marco conceptual de la densidad, se considera pertinente la inclusión de una ecuación que permita calificar de manera objetiva esta proporción mencionada. En consecuecia, se puede utilizar la ecuación de Wasserman y Faust (2013), quienes proponen la siguiente ecuacion:

A partir de la cual se dice que un grafo será denso si el resultado de esta división se acerca a la unidad. Por el contrario, para casos alejados de la unidad se podrá tener certeza de la dispersión del mismo. A continuación, se presenta el analisis de nuestro caso puntual con le archivo mas completo:

Así, se comprueba que el grafo es disperso.

1. ¿El grafo es dirigido o no dirigido?

RTA/ Para poder dar respuesta a esta pregunta se debe tener en cuenta la definición de dirección en los grafos. En este sentido, esbozando lo trabajado en clase, se peude afirmar que la dirección de un grafo hace referencia a los arcos, y a la validez de las uniones entre vertices en ambos sentidos de la unión o en uno solo de ellos. Así las cosas, aquellos grafos que cuyos arcos no sean validos para la unión bidireccional de los elementos, se considerarán dirigidos. Así mismo, aquellos cuyos arcos no tengan un sentido específico y sean validos como caminos de unión adyacente entre cada vertice, se les llamará grafos NO dirigidos.

Por otra parte, se considera pertinente la recuperación de las condiciones planteadas como contextro de los datos analizados: Se tiene recuento de las conexiones de las estaciones(vertices) a partir de las rutas de los buses(arcos). Así pues, las rutas de los buses deben ser unidireccionales, dando lugar a la interpretación del grafo como dirigido.

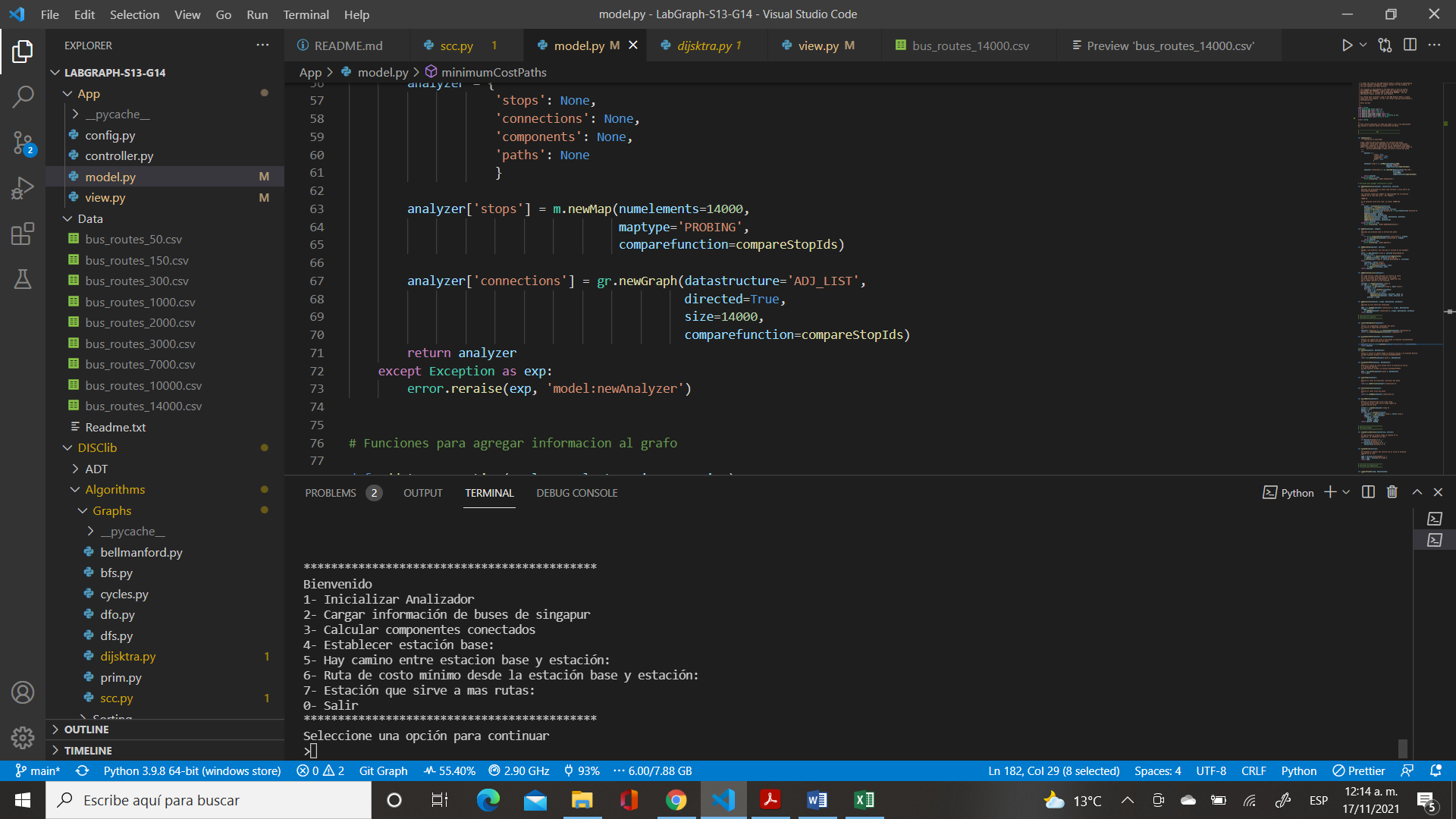
1. ¿El grafo está fuertemente conectado?

RTA/ Para poder dar respuesta a esta pregunta se hace necesario la definición del concepto de conexión dentro del contexto de grafos. En este sentido, se puede asegurar que, de acuerdo con Tarjan(1970), la conexión de un grafo hace referencia a la posible determinación de caminos entre todos los vertices del grafo. En este sentido, la definición formal indica que para todo par de vertices x,y siempre habrá un camino tanto de x a y como de y a x.

Así las cosas, para la determinación de esta particularidad dentro del caso puntual de análisis de nuestro grafo, se realizaron una serie de pruebas para las que se determinó que efectivamente, dentro de esta red de transporte acotada por nuestros datos, se pueden hallar caminos que conecten, en cualquier dirección, 2 vertices escogidos de manera completamente arbitraria y azarosa.

1. ¿Cuál es el tamaño inicial del grafo?

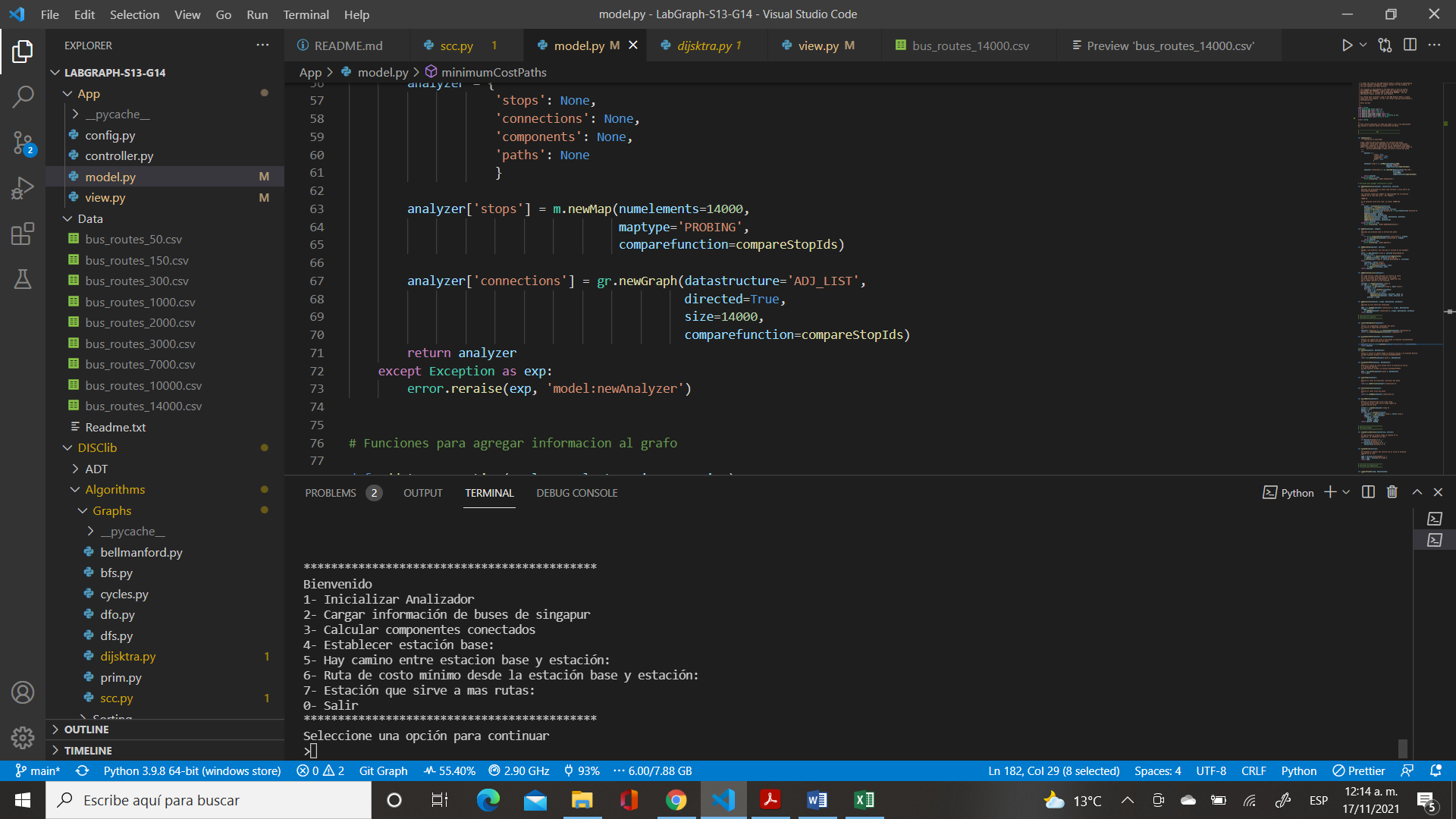
RTA/ Para poder dar respuesta a esta pregunta, se considera pertinente el analisis del codigo base dentro del archivo model.py y la construcción vacía del grafo al principio de la inicialización del catálogo. En este sentido, el fragmento de codigo de interés será el siguiente:



A partir de ella se puede ver de manera qexplícita cómo el tamaño del grafo esta determinado por el parametro size en 14 mil elementos.

1. ¿Cuál es la Estructura de datos utilizada?

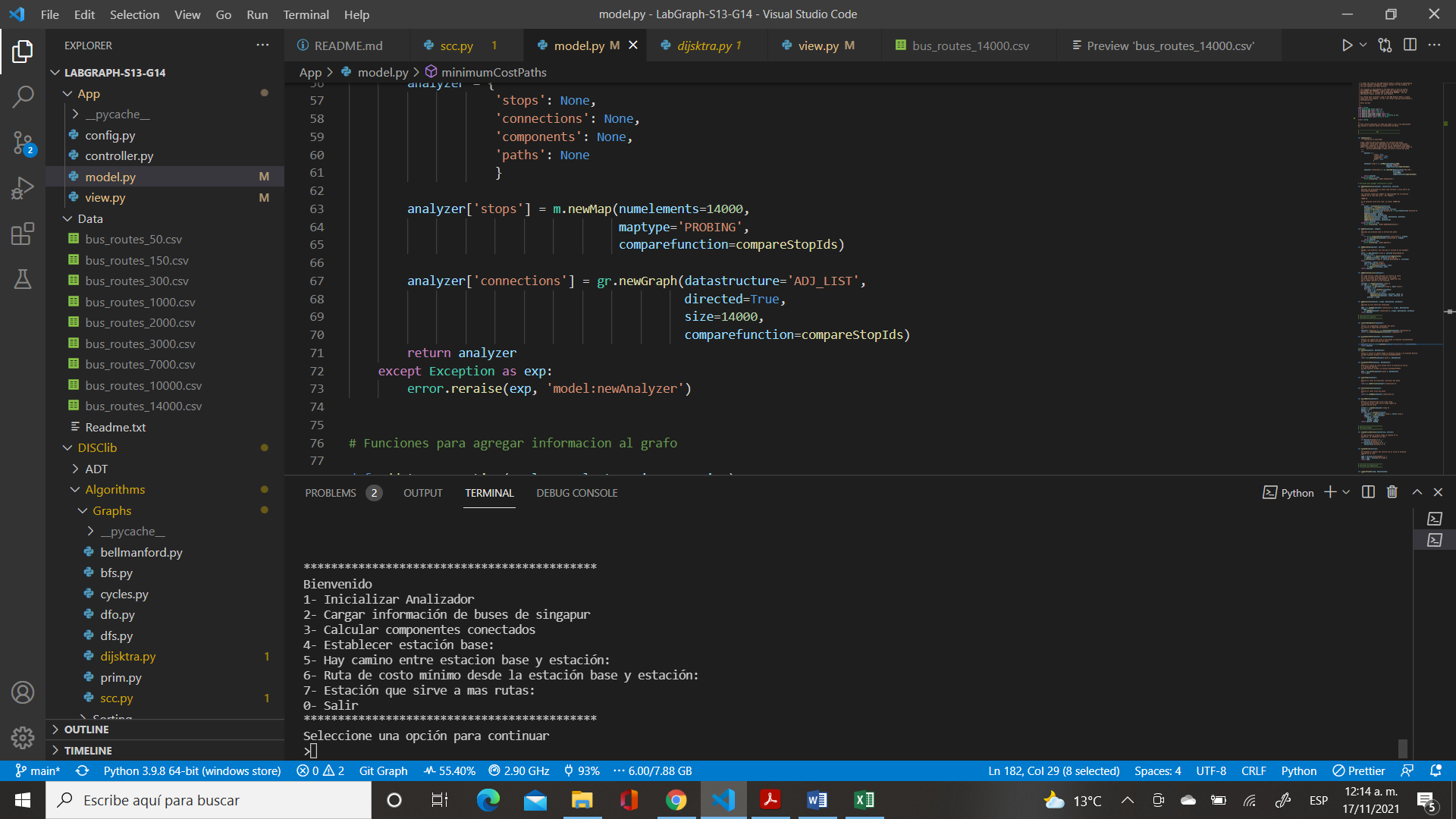
RTA/ Nuevamente, para poder dar respuesta a esta pregunta, se considera pertinente el análisis del código base dentro del archivo model.py y la construcción vacía del grafo al principio de la inicialización del catálogo. En este sentido, el fragmento de código de interés será el siguiente:



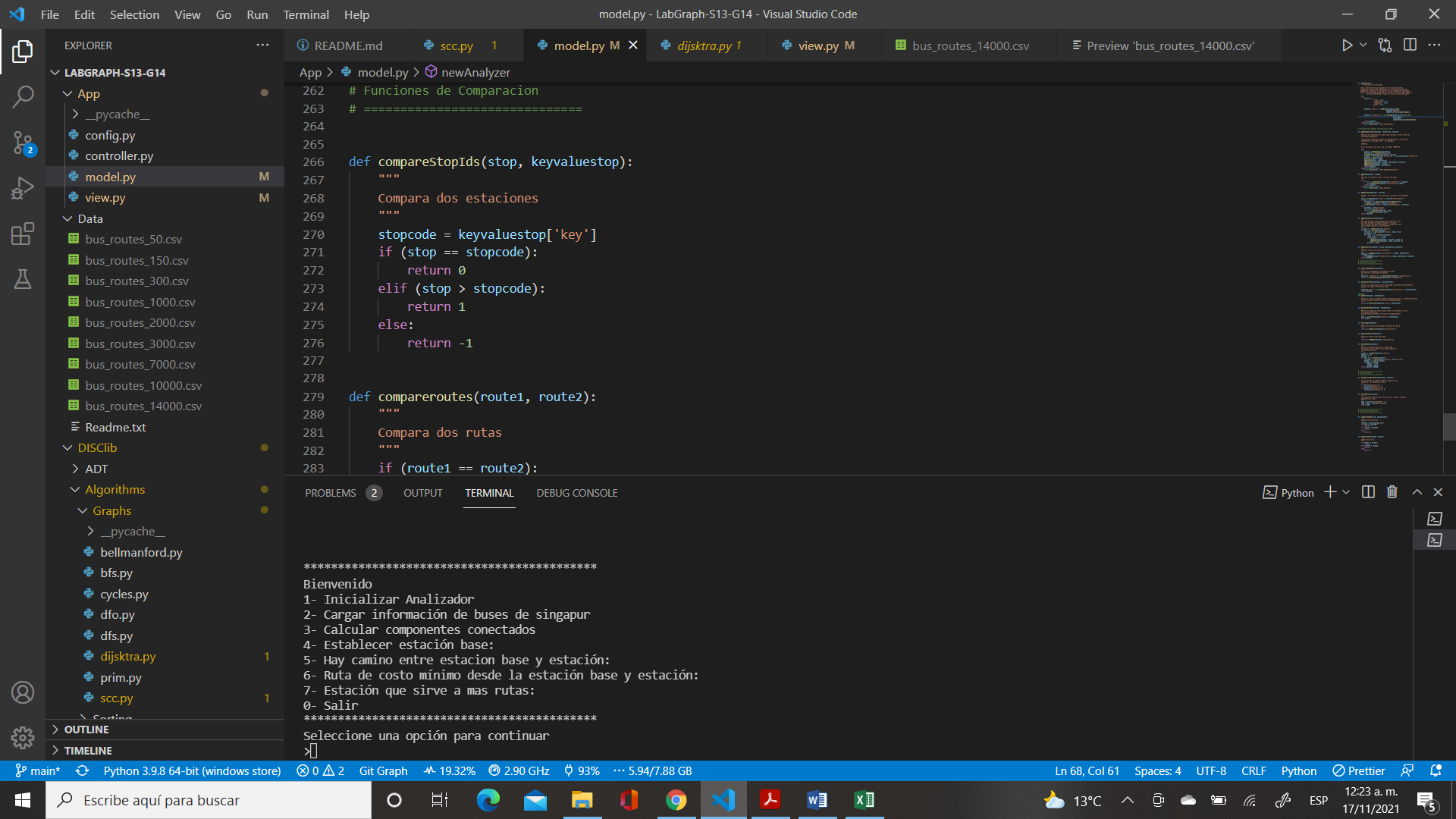
Como se puede observar, para el caso de la estructura de datos también se tiene un parámetro específico dentro de la creación del grafo en el catálogo que nos ayuda a determinar esta información. Asi las cosas, para el caso del grafo del ejercicio se tiene como base una lista de adyacencias como estructura de datos de almacenamiento detrás del grafo.

1. ¿Cuál es la función de comparación utilizada?

RTA/ Una vez más, para poder dar respuesta a esta pregunta, se considera pertinente el análisis del código base dentro del archivo model.py y la construcción vacía del grafo al principio de la inicialización del catálogo. En este sentido, el fragmento de código de interés será el siguiente:



Dado lo anterior, se puede observar que la función de comparación es dada también de manera explicita dentro de uno de los parámetros de formación del grafo. Sin embargo, sería enriquecedor poder analizar más a fondo esta función, por lo que a continuación se presenta su código:



Ahora bien, a partir de este código enseñado, se considera que se puede tener una certeza de que los vertices/elementos del grafo se determinan a partir de los ID de las paradas de cada una de las rutas que llegan en los datos crudos.

**PREGUNTAS RETO:**

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

RTA/Para poder dar respuesta a esta pregunta, se considera pertinente la realización de 2 procedimientos: Establecimiento de los grafos necesarios y Establecimiento de satisfacción de cada uno de los requerimientos:

GRAFOS NECESARIOS: 2

* Grafo contenedor de rutas entre aeropuertos donde los vértices sean cada uno de los IATA de los aeropuertos
* Grafo contenedor de las rutas entre ciudades, donde los vértices sean las ciudades y los arcos cada una de las rutas entre ellas

CUMPLIMIENTO DE REQUERIMIENTOS:

* REQ 1: Este requerimiento se puede cumplir en su totalidad, se puede utilizar un algoritmo Dijkstra a partir de un grafo que identifique a los aeropuertos por su IATA
* REQ 2: Este requerimiento se puede cumplir a cabalidad a partir de la identificación de clústeres establecidos por los aeropuertos con mayor cantidad de conexiones entre sus códigos IATA, y satisfacer con ese mismo grafo el requerimiento de la distancia más corta entre 2 aeropuertos según su IATA
* REQ 3: Para este requerimiento se puede utilizar un grafo que tenga por vértices precisamente a las ciudades y por arcos las rutas de todos los aeropuertos que estén dentro de las áreas demarcadas por los enunciados, y a partir de allí construir la solución al requerimiento
* REQ 4: Para poder satisfacer completamente este requerimiento se considera oportuna la reutilización del grafo creado para la función anterior, de modo que se tenga una red de rutas entre ciudades sobre la cual buscar la red de expansión mínima y la cantidad de millas de su rama más larga y recomendada
* REQ 5: Para cumplir a cabalidad con lo establecido en este requerimiento se considera oportuno la reutilización del grafo creado para el primer requerimiento. A partir de este se descubrirán todas las conexiones de un aeropuerto específico según su IATA. Posteriormente, se evaluarán las ubicaciones de cada uno de los vértices involucrados.
* REQ 6: Para cumplir con este requerimiento nuevamente se considera que la utilización del grafo del requerimiento 3 será más que suficiente para el cálculo y las búsquedas necesarias para la obtención de la respuesta de este requerimiento.

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

RTA/ Para poder dar respuesta a este enunciado, se considera pertinente la tabulación de los datos de cada uno de los grafos

|  |  |  |  |
| --- | --- | --- | --- |
| GRAFO #1: AEROPUERTOS POR IATA | | | |
| VERTICES | ARCOS | DENSIDAD | DIRECCION |
| 9000 aprox  (cantidad de aeropuertos del documento es 9076) | 92000 aprox  (cantidad de rutas del documento es de 92606) | Disperso(De acuerdo con los calculos de la formula propuesta para el análisis del laboratorio) | Dirigido: Las rutas entre ciudades tienen una dirección definida |

|  |  |  |  |
| --- | --- | --- | --- |
| GRAFO #2: RUTAS POR CIUDAD | | | |
| VERTICES | ARCOS | DENSIDAD | DIRECCION |
| 41000 aprox  (la cantidad de ciudades en el documento base es de 41002) | 92000 aprox  (cantidad de rutas del documento es de 92606) | Disperso(De acuerdo con los calculos de la formula propuesta para el análisis del laboratorio) | Dirigido: Las rutas entre ciudades tienen una dirección definida |

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

RTA/Adicionalmente, en aras de poder almacenar la información de las respuestas de una mejor manera, se considera pertinente el trabajo con listas y mapas. Las listas permitirán mejores impresiones y ordenamiento de los datos ya obtenidos como respuesta. Los mapas, por otra parte, pueden servir como indices auxiliares de rápida consulta, así como también se debe mencionar su imporancia como base para las listas de adyacencia de los grafos.

**REFERENCIAS:**

Dijkstra, E. W. (1959). «A note on two problems in connexion with graphs» (http://www-m3.m a.tum.de/twiki/pub/MN0506/WebHome/dijkstra.pdf). Numerische Mathematik 1: 269-271. S2CID 123284777 (https://api.semanticscholar.org/CorpusID:123284777). doi:10.1007/BF01386390 (https://dx.do i.org/10.1007%2FBF01386390).

Preiss, Bruno (1998). Data Structures and Algorithms with Object-Oriented Design Patterns in C++. John Wiley & Sons. ISBN 0-471-24134-2.

Wasserman, Stanley; Faust, Katherine (2013) [1994]. Análisis de redes sociales: Métodos y aplicaciones. Madrid: Centro de Investigaciones Sociológicas. ISBN 978-84-7476-631-8. OCLC 871814053 (<https://www.worldcat.org/oclc/871814053>).

R.E. Tarjan, Depth-First search and linear graph algorithms, SIAM J. Comp. 1 (1972) 146-60.