EJERCICIO: GRAFO DEL METRO DE MADRID

Este programa en Python está diseñado para trabajar con un grafo que representa la red de metro de Madrid. A continuación, proporcionaré una explicación detallada del código por secciones:

Definición de Variables:

- **metro_madrid:** Es un diccionario que actúa como lista adyacente, creada mediante una tabla hash, para almacenar las conexiones entre las estaciones del metro de Madrid.
- **estaciones_lineas:** Es una lista que contendrá sublistas representando líneas de metro. Cada sublista contiene estaciones de una línea de manera ordenada.
- **lineas_metro:** Es un diccionario que relaciona las sublistas de **estaciones_lineas** con su respectiva línea de metro.

Definición de Funciones:

1. **create_node(node):**
   - **Parámetro:** **node** (nombre de la estación).
   - **Acción:** Convierte el nombre de la estación a mayúsculas y agrega la estación como un vértice al diccionario **metro_madrid** si aún no existe.
2. **add_connection(station1, station2):**
   - **Parámetros:** **station1** y **station2** (nombres de estaciones).
   - **Acción:** Agrega una conexión bidireccional entre dos estaciones al diccionario **metro_madrid**, creando las estaciones si aún no existen.
3. **erase_node(node):**
   - **Parámetro:** **node** (nombre de la estación).
   - **Acción:** Elimina un vértice y todas sus aristas del diccionario **metro_madrid**.
4. **erase_edge(node1, node2):**
   - **Parámetros:** **node1** y **node2** (nombres de estaciones).
   - **Acción:** Elimina la conexión bidireccional entre dos estaciones.
5. **find_adjacent_nodes(node):**
   - **Parámetro:** **node** (nombre de la estación).
   - **Acción:** Retorna una lista de estaciones adyacentes a la estación dada.
6. **show_adjacent_nodes():**
   - **Acción:** Imprime las conexiones entre estaciones.

7. **station_line(station):**

   - **Parámetro:** **station** (nombre de la estación).
   - **Acción:** Retorna la(s) línea(s) a la(s) que pertenece la estación.

8. **calc_nodes_number():**

   - **Acción:** Retorna el número de vértices en el grafo.

9. **find_path(station1, station2):**

   - **Parámetros:** **station1** y **station2** (nombres de estaciones).
   - **Acción:** Utiliza el algoritmo de búsqueda en profundidad depth first search (DFS) para encontrar un camino de la estación 1 a la estación 2. Hace uso de recursividad, al llamarse a la función **dfs** dentro de sí misma a la hora de hacer la búsqueda.

10. **find_path_improved(station1, station2):**

    - **Parámetros:** **station1** y **station2** (nombres de estaciones).
    - **Acción:** Mejora la función anterior para tener en cuenta si las dos estaciones pertenecen a la misma línea. Sin embargo, esta función aún no está del todo terminada, pues en algunos casos particulares puede presentar errores.

Lectura del CSV:

- Lee un archivo CSV llamado 'lineas_metroMadrid.csv' y almacena los datos en **datos_lineasMetroMadrid**. Incluye un control de excepciones en caso de que no se encontrase el archivo al intentar abrirlo para su lectura.

Procesamiento del CSV:

- Crea sublistas en **estaciones_lineas** para representar las estaciones de cada línea de metro.
- Elimina elementos vacíos y números de línea en las sublistas.

Creación de Conexiones:

- Utiliza la información procesada del CSV para agregar conexiones entre estaciones en el grafo **metro_madrid**.

Ejecución en Pantalla:

- Llama a la función **find_path** para encontrar un camino de ejemplo de "Puerta de Arganda" a "Tetuan".

Este código implementa operaciones básicas en grafos, como agregar y eliminar nodos, conexiones, encontrar rutas, y visualizar conexiones. Además, se han considerado mejoras en el algoritmo de búsqueda de rutas que tienen en cuenta las líneas de metro.