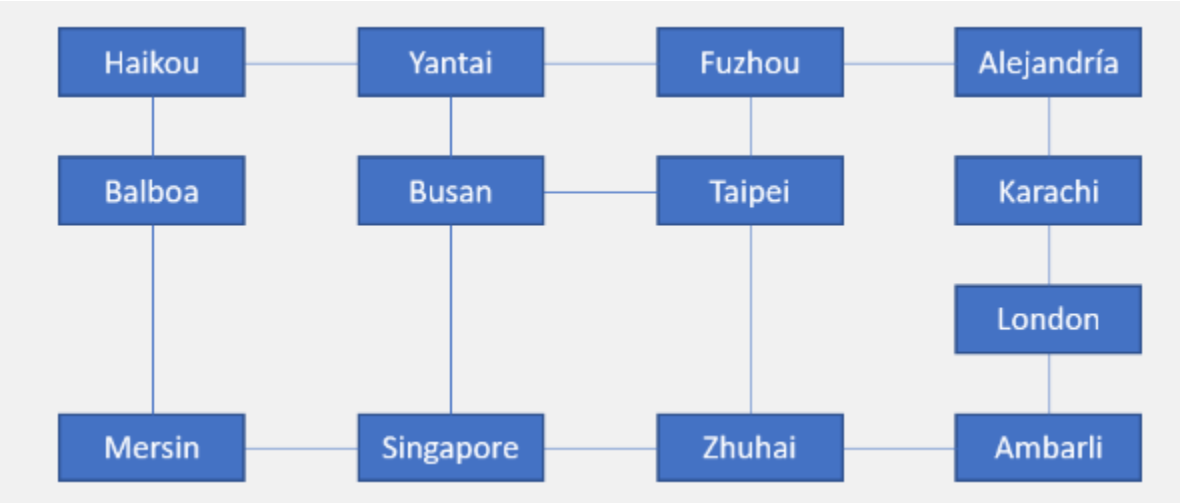
Act 4.3 - Actividad Integral de Grafos (Evidencia Competencia)

Problema

Para evitar el problema que se tuvo en el Canal de Suez, la compañía naviera "International Seas, LTD." ha determinado que cada carguero tendrá un máximo número de puertos (MNP). Este número indica la cantidad de puertos en los cuáles un carguero puede atracar antes de tener que recorrer la ruta en sentido contrario. Cada vez que la nave atraca en un puerto, disminuye el MNP en 1. Si el valor de MNP llega a cero, el carguero tendrá que "girar" y recorrer su ruta en sentido inverso. Dada de una red de puertos, un puerto inicial de un carguero y un valor MNP, deberás determinar la cantidad de puertos que no van a poder ser visitados por ese carguero.



Toma con ejemplo la red anterior. Si un carguero con MNP de 2 saliera del puerto de Busan, sólo podría llegar a los puertos Yantai, Haikou, Singopore, Mersin, Taipei, Fuzhou, Zhuhai. No podría alcanzar ningún otro puerto, ya que el MNP se habría reducido a cero al llegar a los puertos Haikou, Fuzhou, Merson y Zhuhai. Si aumentamos el valor inicial MNP a 3, saliendo del puerto Busan, se podría llegar a todos los puertos excepto a Karachi y London.

Entrada

Cada entrada tendrá el siguiente formato. Empieza con un número entero, NC, que especifica el número de conexiones entre los puertos de la red. Después, habrá NC pares de nombres de puertos. Estos pares identifican los puertos que están conectados por una ruta de navegación. No habrá más que una ruta de navegación directa entre cualquier par de puertos y ninguna red contendrá más de 30 puertos. Después de la descripción de la red, habrá un número entero, NQ, que especifica el número de consultas que se realizarán.

A continuación, habrá NQ líneas conteniendo el nombre del puerto de inicio y el MNP inicial.

Salida

Para cada consulta, muestra una sola línea que indica el número de caso de prueba (numerados secuencialmente desde uno), el número de puertos a los cuales no se puede llegar, el nombre del puerto y el valor iniciales de MNP. A continuación, encontrarás un ejemplo de entrada y salida.

Solución

En este caso, se creó un sistema que utilizara varios métodos de los grafos para poder ejecutar el programa, se creó un programa que pudiera leer un archivo de entrada que contenía un entero que representaba la cantidad de conexiones, después, la cantidad de conexiones, para terminar un numero de pruebas y las pruebas realizadas, se utilizaron las siguientes funciones:

- **addEdgeAdjList**: agrega nodos y los adyacentes a la lista de nodos.
- **findNodo**: regresa el lugar donde está ubicado el nodo, esta función sirve como auxiliar para otras funciones y métodos.
- **findListPos**: encuentra un valor en la lista, servirá de auxiliar para mas funciones.
- **puertosRestantes**: muestra el residuo de puertos, la respuesta del problema.
- **testFile**: el esqueleto del programa, este realiza el trabajo de lectura del archivo de texto y llama a las funciones para almacenarlo, así como lee los casos de prueba.

Test cases:

Se realizaron cuatro pruebas, estos son con los siguientes puertos y conexiones:

```
16
Alexandria Algeciras
Algeciras Ambarli
Ambarli Antwerp
Alexandria Balboa
Balboa Bandar
Bandar Barcelona
Antwerp Bremen
Bremen Busan
Algeciras Cai_Mep
Cai_Mep Callao
Ambarli Cartagena
Barcelona Callao
Cai_Mep Cartagena
Callao Charleston
Cartagena Charleston
Charleston Busan
4
Cai_Mep 2
Cai_Mep 3
Callao 10
Barcelona 0
```

Resultado:

```
Case 1: 5 ports not reachable from port Cai_Mep with MNP = 2.
Case 2: 1 ports not reachable from port Cai_Mep with MNP = 3.
Case 3: 0 ports not reachable from port Callao with MNP = 10.
Case 4: 12 ports not reachable from port Barcelona with MNP = 0.

Press any key to continue . . .
```

Se aprecia que todos los casos de prueba fueron exitosos, muestran cuantos puertos no pudieron ser visitados, así como la cantidad de iteraciones se le dio a cada puerto.

Complejidad:

- **addEdgeList:** Este algoritmo presenta una complejidad de O(n), siendo n la cantidad de nodos.
- **findNodo:** Este algoritmo presenta una complejidad de O(n), siendo n la cantidad de nodos presentes en el vector de nodos.
- **findPosList:** Este algoritmo tiene una complejidad de O(n), siendo n el tamaño del vector de nodos.

- **puertosRestantes:** Este algoritmo presenta una complejidad de O(n + m), siendo n los nodos y m sus vecinos.

Reflexión

Es muy normal que en problemas cotidianos se puedan encontrar implícitas estructuras de datos y algoritmos, en este caso, se buscaba una manera de relacionar varios puertos, viendo que cada puerto puede tener mas de dos vecinos, no se ve factible el uso de un árbol binario, por lo que se necesita una estructura que sea de *muchos a muchos,* aquí es donde se presentan los grafos.

Los grafos son perfectos para demostrar una estructura que tenga relacionados muchos nodos entre sí, por esto son conocidas como estructuras de tipo red.

En este problema se necesitaba saber si con cierto número de iteraciones, se podía llegar a recorrer todos los puertos partiendo de uno, sabiendo que cada puerto tiene determinado numero de conexiones y puertos vecinos, los grafos y sus funciones pueden ser fácilmente implementados.

Muchos problemas de este tipo se pueden presentar en el día a día, por ejemplo caminos a hacia una ciudad desde otras ciudades, o hasta redes de amigos y amigos en común con una persona.