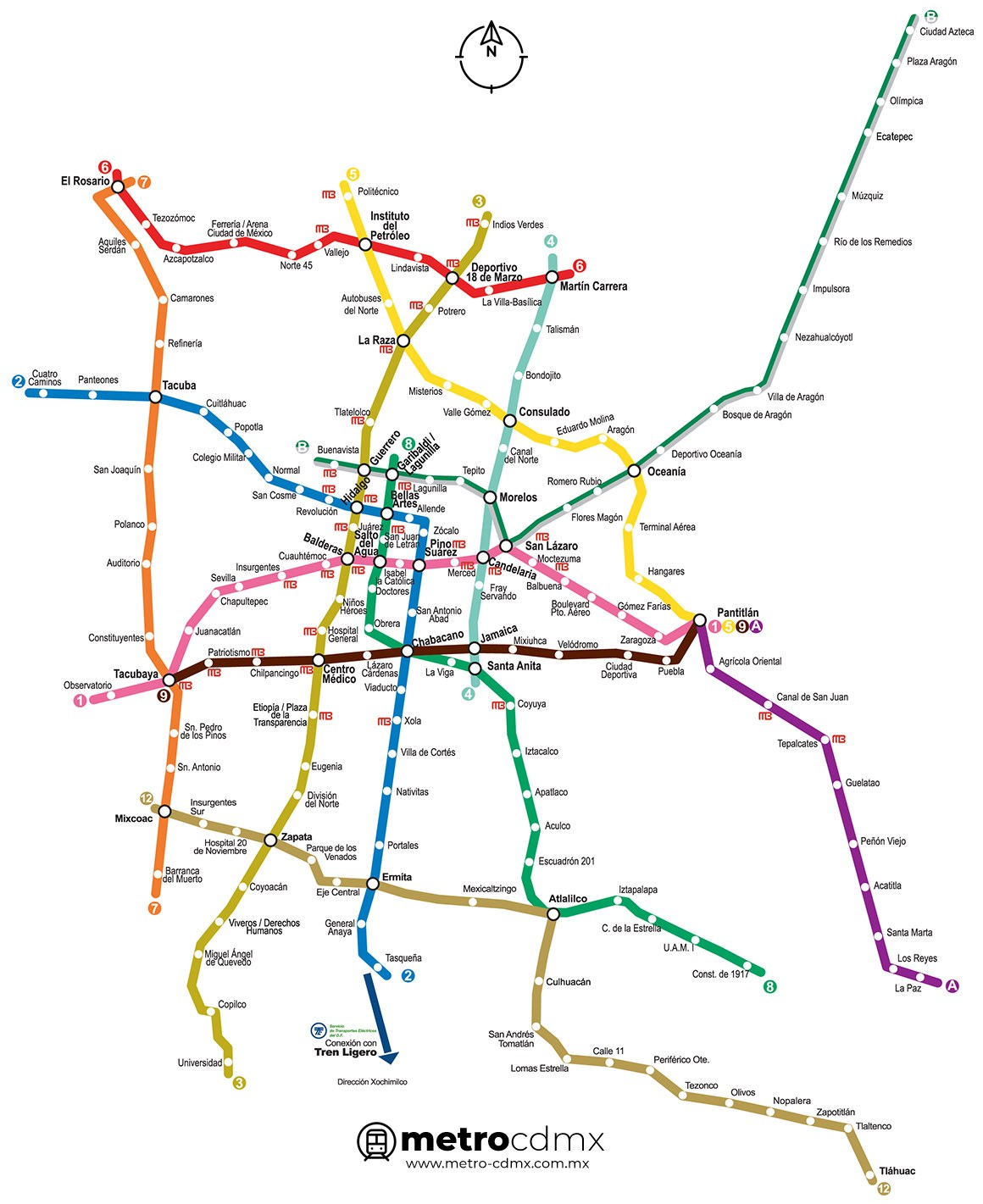
**A14 Semana 14 - Actividad 01**

**Búsqueda en profundidad (DFS) y búsqueda en anchura (BFS)**

Continuamos con el estudio de la eficiencia y complejidad de los grafos, para ello en esta semana realizaremos lo siguiente:

1. Descarga **el mapa del metro de la ciudad de México,** el cual se muestra en la siguiente Figura 1. Metro CDMX. Visualízalo como un “**grafo”**.



*Figura 1. Metro CDMX*

1. Has estado trabajando con un archivo de texto que indica la descripción de un grafo como la *Figura 1. “entrada.txt”*, por lo que tendrás que construirlo según lo que se requiera para la actividad.

| **100 nodos**  1 Juarez 10 300  2 Mexicaltzingo 20 300  3 Washington 30 300  …  9 Santa Filomena  10  **190 relaciones**  1 2  2 3  3 4  4 10  4 98  …  189 190 |
| --- |

*Figura 1. “entrada.txt”*

1. Realizado el archivo de “*Entrada.txt*” y utilizando el operador de direccionamiento en la línea de comando

p.e. → c://construyegrafo.exe **<** entrada.txt **>** salida.txt

1. Codifica un código en *C++* que al recibir el archivo:
   1. Se definan los dos nodos (origen-destino)
   2. Se realice una búsqueda en profundidad (DFS) y una búsqueda en anchura (BFS)
   3. Se calcule el tiempo de ejecución.
   4. En el archivo de salida deberá estar el recorrido hecho por cada una de las búsquedas y su tiempo de ejecución. (En cada ejecución se genera un archivo diferente)
2. Para tener un mejor entendimiento de lo realizado, realiza una serie de imágenes o animación del recorrido del algoritmo en Python.

Ayúdate de algún paquete externo de *‘Python’* como los paquetes **‘*NetworkX*’, *‘Matplotlib’* o *‘PyGame’***.

(También es válido y opcional que realices una animación en otro lenguaje como ‘*Processing*’)

1. Solo construye la estructura de las matrices, no se está requiriendo la realización de una clase, ni que se instancien los objetos.
2. Realiza un archivo fuente y ejecutable por cada caso.
3. Apóyate de hacer llamadas a funciones definidas por el programador.
4. La compilación y ejecución se tiene que realizar por la terminal.
5. Utiliza el operador de flujo y la llamada a un archivo de texto con el caso a analizar.
6. Al iniciar la iteración toma el tiempo de inicio, y al finalizar la iteración toma el tiempo final. Utiliza un método de C++.
7. Se sugiere presentar la diferencia en horas, minutos, segundos, milisegundos (00:00:00:00). Revisar la biblioteca ***<ctime>.***
8. En caso de no convertir el tiempo al formato requerido del punto 7 previo, solo respeta utilizar una misma medida de tiempo para todos los análisis posteriores.
9. Observa que no se requiere que hagas un menú o definas objetos.
10. Recuerda siempre tienes que trabajar en una terminal y generar la salida del archivo según tu necesidad.
11. Puedes experimentar haciendo lo mismo en otro lenguaje.