Metro CDMX: Encontrando la ruta más corta

Jorge García (202945)
Aline Perez (203145)
Sandra España (203200)
Marco Ramos (142244)

Introducción

El metro de la Ciudad de México, fundado en 1969 cuenta con una extensión aproximada de 226 km que abarcan incluso una parte del oriente del Estado de México. Es sin duda uno de los sistemas de transporte más concurridos, llegando a superar por momentos la alfuencia de otros metros importantes como lo es el de Moscú o el de Londres. Actualmente y de acuerdo a cifras oficiales cuenta con un total de 12 líneas y 195 estaciones dentro de la red. Se tienen 48 estaciones con correspondencia y 123 estaciones de paso. La siguiente figura tomada de la cuenta oficial de twiter de metro de la CDMX muestra la red y cómo se ve actualmente:

Figura 1: Metro de la CDMX

Como usuarios del metro, sabemos que actualmente existe un problema de saturación, pues sin lugar a dudas tal vez sigue siendo el sistema de transporte más barato para la pobación en general ya que con 5 pesos es posible cruzar la ciudad de un extremo a otro. Si bien, el presente trabajo no pretende dar una solución a dicho problema, si tiene por objetivo hacer tres tipos de recomendaciones para cualquier usuqrio:

1. La primera recomendación se base en sugerir la ruta más corta en distancia en metros

2. La segunda recomendación se basa en sugerir al usuario la ruta con menos afluencia de personas. Esto con la idea de que el usuario viaje un tanto más cómodo considerando que hay menos cantidad de personas en el viaje sugerido.
3. La última recomendación se base en una combinación de las dos anteriores y consiste en ponderar la distancia por el nivel de afluencia de personas con el objetivo de sacar una ruta más "óptima" pensando en los dos factores.

Si bien, esta solución no representa necesariamente una solución universal al los problemas que representa viajar en metro, si es una gran iniciativa en el afán de querer mejorar la movilidad de los usuarios.

Solución teórica del problema

Tal y como se describe en la introducción de este documento, la idea de poder resolver este problema consiste en encontrar la ruta más corta de una estación de metro a otra. En ese orden de ideas podemos pensar a la red de metro como una red en donde cada estacion respresenta un noda y la conexión entre cada una de ellas representa un vertice.

El peso de cada uno de los vertices puede estar representado ya sea por la distancia en metros que separa una estación de la otra o bien, por la afluencia de personas que puede haber de una estación a otra. Si pensamos cada estación como los renglones y columnas de una matriz cuadrada, es fácil ver que tratandose de las distancias la matriz resultante será simétrica, por lo que que la ruta más corta de la estación "A" a "B" será la misma que ir de "B" a "A."

Contrario a lo que sucede con las distancias, cuando hablamos de la afluencia de personas la matriz no resulta ser simétrica: esto porque la afluencia de una estación no es la misma que la de la estación vecina, y no será lo mismo ir de "A" a "B" que de "B" a "A." Un ejemplo más concreto sería que la afluencia promedio de personas de la estación "A" es 10,000 personas en un día mientras que en el caso de B sólo es de 1000 personas, si yo, como usuario del metro viajo de "A" a "B" me encontraré más personas al momento de abordar y será menos probable que alcance un lugar para sentarme o siquiera que pueda subirme al tren, mientras que si viajo de "B" a "A" es mucho más factible que suceda lo contrario, es decir, que alcance algún lugar se vuelve más probable y más aún, que realmente pueda abordar el tren.

En la literatura existen varios algoritimos que pueden ayudarnos a resolver este tipo de problemas de encontrar la ruta más corta, tal vez el uno de los más conocidos sea el método simplex, sin embargo en una red de nodos tan grande como lo es el metro, puede resultar ser poco factible, pues la función de optimización a plantear junto con todas las restricciones puede resultar ser poco manejable.

Existe otro algoritmo del cual se puede encontrar mucha literatura, publicado en 1959 lleva el nombre de su autor: el algoritmo de Dijkstra. Este algoritmo fue diseñado para encontrar la ruta más corta en una red extensa de nodos y a diferencia del método simplex no requiere definir una función de optimización ni mucho menos.

Explicación del algoritmo

La siguiente explicación se basa en [1] y [2]. Básicamente el algoritmo de Dijkstra comienza en el nodo que el usuario elija (nodo origen) y analiza el grafo para encontrar la ruta más corta entre dicho nodo y un nodo destino que también será elejido por el usuario. En términos generales el funcionamiento es el siguiente [2]:

- Se comienza en el nodo fuente y se va analizando cada distancia al nodo adyacente.
- Se almacena la distancia más corta encontrada hasta el momento
- Se continua con el siguiente nodo más adjacente, se evalúa la distancia y se actualizan las distancias mínimas en caso de econtrarse una distancia más corta.
- Se repite hasta visitar el nodo más lejano.

El algoritmo únicamente puede trabajar con valores positivos en los pesos, esto porque durante el desarrollo del algoritmo dichos pesos se irán sumando, si hubiera un caso negativo entronces dicho algoritmo no funcionaría

correctamente pues decir que se ha encontrado la ruta más corta sería engañoso e incluso podrías caracer de interpretabilidad.[1]

A contunuación se muestra un ejemplo de como funciona el algoritmo. Considere el siguiente grafo:

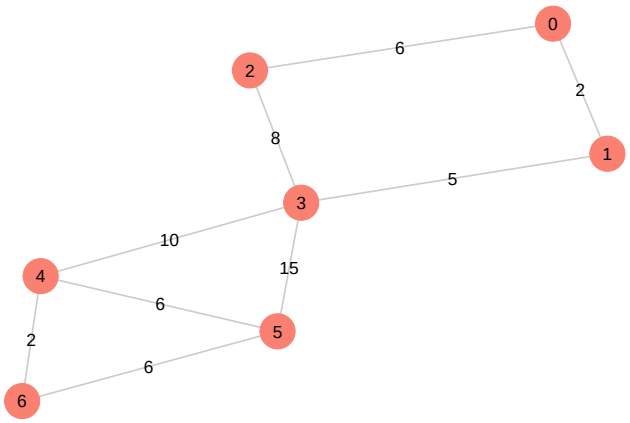

Figura 2: Ejemplo de red con 6 nodos

La figura 2 representa una conexión entre 6 nodos, la distancia de cada uno de ellos esta dado por el número que se puede observa entre sus vertices, la distancia de ir del nodo 0 al nodo 1 es 2 mientras que ir del nodo 0 al nodo 2 la distancia es de 6. Dijksta generará la distancia más corta del nodo 0 a cualquier otro nodo dentro de la red.

Inicializamos el algoritmo, la siguiente tabla muestra las distancias iniciales para cada nodo:

Nodo	Distancia
0	0
1	∞
2	∞
3	∞
4	∞
5	∞
6	∞

Note que la distancia inicial en el nodo 0 consigo mismo será 0, mientras que para el resto las distancias iniciales, dado que estas no se conocen se inicializan con el simbolo ∞. Por otro lado definimos la lista de los nodos no visitados: $\{0, 1, 2, 3, 4, 5, 6\}$. Dado que el nodo origen es el nodo 0, lo marcamos como visitado, para ello usarmos el signo "-" dentro de nuestra lista para dentar que hemos visitado dicho nodo: $\{-0, 1, 2, 3, 4, 5, 6\}$.

Ahora, revisamos la distancia que existe entre el nodo 0 y sus nodos adyacentes, que en este caso son 1 y 2. Una ves que hemos revisado las distancias, actualizamos nuestra tabla tal que:

Nodo	Distancia
0	0
1	2
2	6
3	∞

[1] Aunque podemos solucionar esto al hacer una transformación a los datos como obtener el valor absoluto de los pesos dependiendo el contexto

Nodo	Distancia
4	∞
5	∞
6	∞

Y posterior a ello, seleccionamos el nodo cuya distancia es menor al nodo origen, una vez que lo hemos seleccionado marcamos ese nodo como visitado. Adicional, agregmos un elemento más a la lista, que será la ruta, en este caso esta irá del nodo 0 al 1 tal que: $\{0 \rightarrow 1\}$ y $\{-0, -1, 2, 3, 4, 5, 6\}$

Ahora necesitamos analizar los nuevos nodos adyacentes para encontrar el camino más corto para llegar a ellos. Solo analizaremos los nodos que son adyacentes a los nodos que ya forman parte del camino más corto. El nodo 3 y el nodo 2 son adyacentes a los nodos que ya están en la ruta porque están conectados directamente al nodo 1 y al nodo 0, respectivamente. Estos son los nodos que analizaremos en el siguiente paso[2].

Como ya tenemos la distancia desde el nodo de origen hasta el nodo 2 anotada en nuestra lista, no necesitamos actualizar la distancia esta vez. Solo necesitamos actualizar la distancia desde el nodo de origen hasta el nuevo nodo adyacente (nodo 3). Por lo que la tabla resultante será [2]:

Nodo	Distancia
0	0
1	2
2	6
3	7
4	∞
5	∞
6	∞

Note que en este caso, la distancia actualizada será 7 pues es el resultado de la suma de 2 y 5 que es la distancia que nos toma llegar del nodo 0 al nodo 3. Ahora, de los nodos aún no visitados y basados en la tabla anterior debemos elegir el que represente la distancia más corta desde el nodo origen, en este caso la distancia más corta será 6 y marcamos este como visitado tal que $\{-0, -1, -2, 3, 4, 5, 6\}$. Ahora tenemos dos rutas que en este caso son: $\{0 \rightarrow 1\} y \{0 \rightarrow 2\}$, luego entonces debemos elegir la ruta más corta para llegar al nodo 3.

En nuestra tabla anterior, note que el nodo 3 ya tiene una distancia derivado de los pasos anteriores, pero ahora tenemos una alternativa que será considerar la distancia entre el nodo 2 y 3. Por la figura 2, la suma de estas distancias es de 14, mientras que la distancia ya registrada anteriormete es de 7 como se observa en la tabal, de forma que conservamos la ruta que originalmente habìamos tomado y agregamos el nuevo nodo tal que: $\{0 \rightarrow 1 \rightarrow 3\}$. Hemos llegado al nodo 3, así que ahora podemos marcarlo como visitado: $\{-0, -1, -2, -3, 4, 5, 6\}$

Hasta debemos de revisar las distancias que se derivan de ir a los nodos no visitados, en este caso para ir al nodo 4 la ruta definida sería $\{0 \rightarrow 1 \rightarrow 3 \rightarrow 4\}$ cuya distancia es 17, mientras que para ir al nodo 5 la ruta es $\{0 \rightarrow 1 \rightarrow 3 \rightarrow 5\}$ y cuya distancia es 22. Actualizamos de nueva cuenta nuestra tabla de distancias:

Nodo	Distancia
0	0
1	2
2	6
3	7
4	17

Nodo	Distancia
5	22
6	∞

Ahora, debemos marcar como visitado aquel nodo que nos haya dado la distancia más corta, que en este caso es el nodo 4: $\{-0, -1, -2, -3, -4, 5, 6\}$ y la ruta actualizada será $\{0 \to 1 \to 3 \to 4\}$. Una vez más repetimos los mismos pasos para el nodo 5 y 6.

Para el nodo 5, la primera opción es seguir la ruta $\{0 \to 1 \to 3 \to 5\}$ que tiene una distancia de 22. La segunda opción será $\{0 \to 1 \to 3 \to 4 \to 5\}$ el cual la distancia será 23 desde el nodo origen, claramente la primera ruta es la más corta.

Para el nodo 6 la ruta disponible es $\{0 \to 1 \to 3 \to 4 \to 6\}$ con una distancia de 19. Ahora marcamos como visitado el nodo con la ruta más corta, que en este caso será el 6: $\{-0, -1, -2, -3, -4, 5, -6\}$. Actualizamos la tabla de distancias:

Nodo	Distancia
0	0
1	2
2	6
3	7
4	17
5	22
6	19

Finalmente, el único nodo no visitado es el nodo 5, veamos como podemos incluirlo en la ruta. Existen 3 rutas posibles para llegar del nodo origen al nodo 5 que son:

1. $\{0 \to 1 \to 3 \to 5\}$ con distancia de 22
2. $\{0 \to 1 \to 3 \to 4 \to 5\}$ con distancia de 23
3. $\{0 \to 1 \to 3 \to 4 \to 6 \to 5\}$ con distancia de 25

Así que la ruta seleccionada para ir del nodo 0 al nodo 5 será la primera, marcamos este nodo cómo visitado y agregamos esta nueva ruta: $\{-0, -1, -2, -3, -4, -5, -6\}$, $\{0 \to 1 \to 3 \to 5\}$. Con ello hemos encontrado la ruta más corta desde el nodo origen a cualquier otro nodo dentro de la red. Este mismo principio es el que usaremos para aplicarlo a nuestro porblema, más detalle de ello se puede ver en la sección de implementación del algoritmo.

Datos

Generación Matriz de Adyacencias

Para el ejercicio, se descargó la base de datos de la siguiente liga liga, la cual contiene la relación de las distancias en metros entre las estaciones del metro de la CDMX, detallando la línea a la que pertenecen y mostradas de forma ordenada como se muestra a continuación:

```
## # A tibble: 10 x 5
##    Linea Estacion1            Estacion2           Long_estacion Long_interestac~
##    <chr> <chr>                <chr>                       <dbl>            <dbl>
##  1 1     Pantitlan            Zaragoza                      150             1320
##  2 1     Zaragoza             GomezFarias                   150              762
##  3 1     GomezFarias          BoulevardPuertoAer~           150              611
##  4 1     BoulevardPuertoAereo Balbuena                      150              595
##  5 1     Balbuena             Moctezuma                     150              703
```

```
##  6 1     Moctezuma           SanLazaro           150              478
##  7 1     SanLazaro           Candelaria          150              866
##  8 1     Candelaria          Merced              150              698
##  9 1     Merced              PinoSuarez          150              745
## 10 1     PinoSuarez          IsabellaCatolica    150              382
```

Para poder generar la matriz de adyacencias, se consideraron como los nodos todas las estaciones (164), y los pesos para cada una el campo "Long_interestacion," que contiene la distancia entre cada estación, esta matriz será simétrica, pues en términos de esta medida es lo mismo ir de Pantitlán a Zaragoza que viceversa. Dado que la matriz es muy grande para ser visualizada, puede consultarse en la carpeta Matrices en el repositorio de GitHub.

En realidad la construcción de esta matriz fue bastante directa y con ella buscamos encontrar la ruta más corta.

Generación Matriz de Afluencias

Como complemento al modelo original y queriendo responder preguntas como: ¿Que ruta es más cómoda en términos de afluencia de personas?, es decir que pasa si como usuario del metro me interesa más que la ruta más corta la ruta más cómoda en términos del número de personas que la transitan.

Bajo este enfoque nos interesa agregar la siguiente base tomada de https://datos.cdmx.gob.mx/dataset/afluencia-diaria-del-metro-cdmx , Esta base muestra la afluencia diaria del Metro CDMX. Los datos se encuentran actualizados al 1 de diciembre de 2022 y su profundidad histórica abarca desde enero 2010 al 31 de octubre 2022.

La base de afluencia desglosada se refiere a la afluencia en el Organismo Público de Transporte STC Metro, separada por medio de acceso, los cuales son: Tarjeta Única de Movilidad Integrada, boleto y gratuidad, esta última se refiere al acceso gratuito para las personas adultos mayores, personas con discapacidad y/o niños menores de 5 años, de acuerdo a los requisitos establecidos por el Organismo.

```
## # A tibble: 10 x 6
##    fecha       anio mes   linea   estacion            afluencia
##    <chr>      <dbl> <chr> <chr>   <chr>                   <dbl>
##  1 01/01/2010  2010 Enero Linea 1 Zaragoza                20227
##  2 01/01/2010  2010 Enero Linea 1 Isabel la Católica       6487
##  3 01/01/2010  2010 Enero Linea 1 Moctezuma               10304
##  4 01/01/2010  2010 Enero Linea 1 Pino Suárez              8679
##  5 01/01/2010  2010 Enero Linea 1 Gómez Farías            19499
##  6 01/01/2010  2010 Enero Linea 6 Deptvo. 18 de Marzo       621
##  7 01/01/2010  2010 Enero Linea 6 La Villa-Basilica       24792
##  8 01/01/2010  2010 Enero Linea 9 Pantitlán               27000
##  9 01/01/2010  2010 Enero Linea 8 Aculco                  3652
## 10 01/01/2010  2010 Enero Linea 9 Velódromo               3239
```

Por ejemplo podemos ver el top 5 de las estaciones con mayor afluencia para el día más reciente en la base '2022-10-31,' sin considerar la linea:

```
## # A tibble: 5 x 2
##   estacion             Afluencias_suma
##   <chr>                          <dbl>
## 1 Pantitlán                     159357
## 2 Indios Verdes                 102621
## 3 Constitución de 1917           87075
## 4 Tacubaya                       81492
## 5 Tasqueña                       73599
```

Ahora observemos el primer lugar de acuerdo a la linea del metro, por ejemplo Pantitlán que tiene varios cruces de lineas

```
## # A tibble: 4 x 6
##   fecha       anio mes     linea   estacion  afluencia
##   <date>     <dbl> <chr>   <chr>   <chr>         <dbl>
## 1 2022-10-31  2022 Octubre Linea 1 Pantitlán         0
## 2 2022-10-31  2022 Octubre Linea 5 Pantitlán     51160
## 3 2022-10-31  2022 Octubre Linea 9 Pantitlán     61233
## 4 2022-10-31  2022 Octubre Linea A Pantitlán     46964
```

Observemos como se comporta la afluencia de una de las estaciones más caóticas de la CDMX

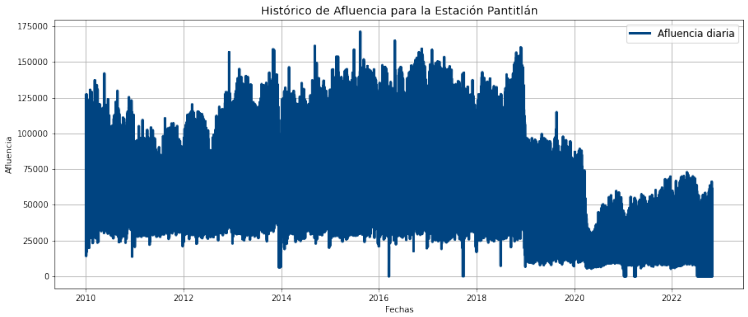

Figura 3: Pantitlán

Indios Verdes es otra de las estaciones con mayor afluencia

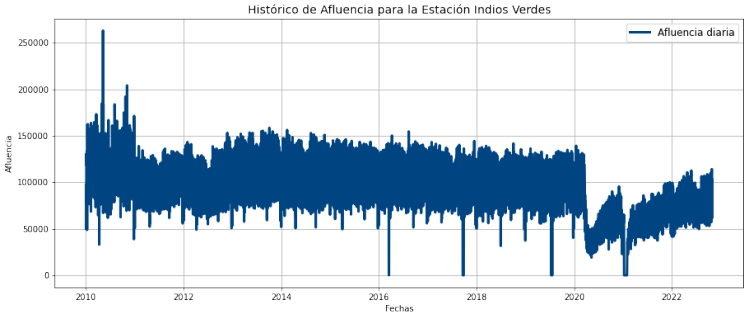

Figura 4: Indios Verdes

Observemos que Pantitlán por ejemplo es una de las estaciones que históricamente ha tenido mayor afluencia de personas, también podemos notar que antes de la pandemia, la concentración de personas era mucho mayor, que post-Covid. Esto nos lleva a preguntarnos como integrar esta información en nuestros datos dado que históricamente cada estación tiene una afluencia distinta. Lo más sencillo es cortar nuestra información a partir de mediados 2021 donde la movilidad empieza a estabilizarse debido a que la pandemia comienza su desaceleración, de esta manera podemos tomarnos como valor fijo entre nodos la mediana de los datos, para este rango de fechas, siendo este un mejor estimador que la media.

Sabemos además que las estaciones con mayores transbordos como lo es pantitlán van a presentar mayor afluencia y esta dependerá de la línea en la que viajen, para fines prácticos de este ejercicio, se asumirá pantitlán como un solo nodo sin importar la línea en la que te encuentras.

Supondremos además que la afluencia tiene dirección, es decir si vas en la línea 1 por ejemplo de Pantitlán a Zaragoza, sabemos que la mediana de afluencia en Pantitlán es n, así que la afluencia de Pantitlán a Zaragoza

será n, y si la afluencia de Zaragoza es m y su dirección es pantitlán, entonces la afluencia de esa dirección será m.

Este quizá sea uno de los supuestos más grandes al integrar esta matriz al modelo, pues estamos haciendo el supuesto de que la cantidad de personas que se reportan entran a una estación permanecen en la siguiente, cuando sabemos que en la realidad esto no necesariamente es cierto, pues si n personas suben en Pantitlán, tienen muchas opciones para moverse, ya que esa estación tiene 4 transbordos, es decir, 4 líneas contiguas, su siguiente opción puede ser Zaragoza, Agrícola Oriental, Puebla o Hangares, sin embargo, al no contar con este nivel de detalle en bases públicas, abordaremos este problema de la siguiente forma:

1. Una opción de matriz será bajo el supuesto de que las personas que entran a x estación permanecen en la siguiente.
2. Dado que cada estación tiene una afluencia diferente, se desarrollará una segunda matriz que contemple tanto las distancias previamente desarrolladas, como la proporción de afluencia de cada estación. Dicha proporción será considera de acuerdo a líneas completas, es decir, si la línea 1 tiene una afluencia total z, entonces cada una de sus estaciones tendrá un peso distinto de acuerdo a su afluencia individual, n/z.

Adicional a lo anterior no estamos considerando horarios, la afluencia es el número de personas totales al día, sin considerar que en horarios 6-9 am o 6-9 pm, son horarios de mucho mayor cocentración de personas por las horas laborales entre semana y varian bastante los fines de semana.

Antes de determinar la mediana general por estación, recordemos que el rango de fechas que estamos considerando es a partir de mediados de 2021, sin embargo, para una estación en particular: línea 12, sufrió un incidente en mayo 2021, dejándola fuera de servicio a partir de ese periodo, podemos observar en el siguiente gráfico como se ve su comportamiento de afluencia histórica. Con fines de no sacarla de nuestro análisis, pues contamos con su matriz de distancias, decidimos tomar un periodo distinto únicamente para esta línea, que contemple el mismo número de periodos (16) y que no se intersecte con la pandemia: mediados de junio 2018 a 2020.

Figura 5: Linea12

	estacion	afluencia
0	IndiosVerdes	83484.5
1	Constitucionde1917	75500.5
2	Tlahuac	58463.5
3	Tasquena	55844.5
4	CuatroCaminos	51137.0
5	InsurgentesSur	41064.5
6	Buenavista	39819.0
7	Observatorio	37927.0
8	CiudadAzteca	36665.0
9	Universidad	35232.0
10	LaPaz	33931.5
11	Pantitlan	33291.5
12	Chilpancingo	33177.0
13	PerifericoOriente	32828.5
14	Merced	31735.5

Figura 6: Tabla medianas

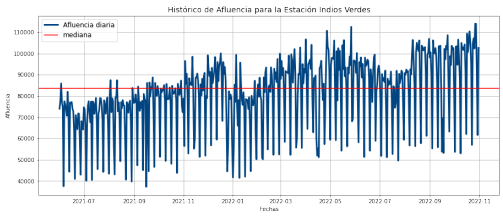

Figura 7: Mediana Indios Verdes

	Linea	estacionA	afluenciaA	estacionB	afluenciaB	AfluenciaInteraccion	PropAfluenciaA	PropAfluenciaB
0	1	Pantitlan	33291.5	Zaragoza	28184.0	340306.0	0.097828	0.082820
1	1	Zaragoza	28184.0	GomezFarias	23083.5	340306.0	0.082820	0.067832
2	1	GomezFarias	23083.5	BoulevardPuertoAereo	12290.5	340306.0	0.067832	0.036116
3	1	BoulevardPuertoAereo	12290.5	Balbuena	7115.0	340306.0	0.036116	0.020908
4	1	Balbuena	7115.0	Moctezuma	13997.5	340306.0	0.020908	0.041132
5	1	Moctezuma	13997.5	SanLazaro	12877.0	340306.0	0.041132	0.037839
6	1	SanLazaro	12877.0	Candelaria	8674.5	340306.0	0.037839	0.025490
7	1	Candelaria	8674.5	Merced	31735.5	340306.0	0.025490	0.093256
8	1	Merced	31735.5	PinoSuarez	15539.5	340306.0	0.093256	0.045663
9	1	PinoSuarez	15539.5	IsabellaCatolica	15061.5	340306.0	0.045663	0.044259

Figura 8: Tabla de Afluencias Finales

Implementación del algoritmo

Para poder plantear una solución al problema, se hizo uso de los algoritmos ya existentes definidos por la comunidad científca. En esta caso, nos basamos en la función *dijkstra_path* cuya documentación podemos encontrar en [3]. Dicha función trabaja con redes definidas en python y aplica el mismo principio que se explicó en la sección anterior.

Se plantea la solución a 3 situaciones que bajo un mismo contexto proponen soluciones diferentes y son:

1. Encontrar la ruta más corta con base en la distancia.
2. Encontrar la ruta más "cómoda" con base en aquella con menor afluencia de personas.
3. Proponer una ruta óptima considerando la combinación de los dos puntos anteriores.

Para el caso del punto 3, la idea de ruta óptima es un criterio propio como resultado de la combinación de la distancia y la afluencia de personas por cada una de las estaciones. Los datos de entrada, ya sea considerando la distancia entre estaciones o bien el número de personas se convierten a una matriz de la siguente forma:

	Pantitlan	Zaragoza	GomezFarias	BoulevardPuertoAereo
Pantitlan	0.0	1320.0	0.0	0.0
Zaragoza	1320.0	0.0	762.0	0.0
GomezFarias	0.0	762.0	0.0	611.0
BoulevardPuertoAereo	0.0	0.0	611.0	0.0
Balbuena	0.0	0.0	0.0	595.0
...	...	...	...	...

Figura 9: Matriz de adyacencias

Si existe una conexión entre una estación y otra se muestra un valor positivo que indica, según sea el caso, la distancia, la afluencia de personas o una combinación de ambas. Estás entradas deben ser positivas o bien cero pues es un requisito que debe cumplirse para que el algoritmo funcione correctamente (ver Explicación del algoritmo).

Posterior a ello, se arma un arreglo que será el input para armar los nodos y los vertices de una grafo. Se toma la matriz de adyacencias y se genera una salida como la que sigue:

```
(('Pantitlan', 'Hangares', 1644.0),
 ('Pantitlan', 'Puebla', 1380.0),
 ('Pantitlan', 'AgricolaOriental', 1409.0),
 ('Zaragoza', 'Pantitlan', 1320.0),
 ('Zaragoza', 'GomezFarias', 762.0),
 ('GomezFarias', 'Zaragoza', 762.0),
 ('GomezFarias', 'BoulevardPuertoAereo', 611.0),
 ('BoulevardPuertoAereo', 'GomezFarias', 611.0),
 ('BoulevardPuertoAereo', 'Balbuena', 595.0))
```

Figura 10: Nodos y pesos por cada vertice

La figura 5 es sólo una muestra de todos los nodos y pesos definidos, por ejemplo podemos ver que existe una conección entre el nodo “Pantitlan” y el nodo “Hangares” cuyo peso (distancia en este caso) es de 1644 metros.

La salida anterior es la base para generar el grafo que será utilizado para poder encontrar la ruta más corta en donde, como ya se mencionó cada estación representa un nodo, existe un vertice dentro de cada uno de ellos siempre que los pesos entre ambos sea positivo. La siguiente figura muestra la red de nodos completa para el metro de la CDMX.

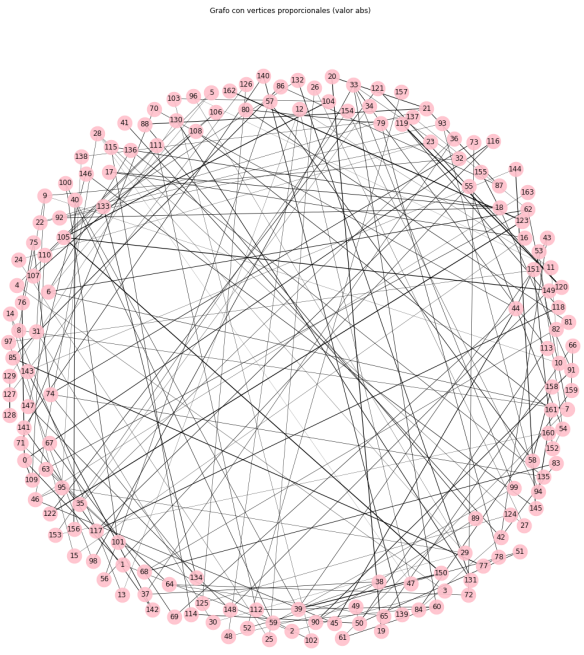

Figura 11: Representación del Metro en un Gráfo

Una vez definido red se hace uso de la función meniconada al inicio de esta sección. La función recibe como parámetro un grafo, el nombre de una estación del metro que sea el origen y una estación destino tal y como se muestra a continuación:

```
nx.dijkstra_path(G, 'ElRosario', 'Deportivo18deMarzo')
```

Figura 12: Función dikjstra

Resultados y conclusiones

Los resultados que se derivan de la implementación del modelo se muestran como en la figura 6, en donde de froma descendente cada entrada indica la siguiente estación que se deberá tomar para llegar a la estación de destino ya sea porque se ha tomado la ruta más corta, la ruta con menor afluencia o la ruta que hemos denonimado la más óptima en tiempo y confort.

```
['Hidalgo',
 'BellasArtes',
 'SanJuandeLetran',
 'SaltodelAgua',
 'IsabellaCatolica',
 'PinoSuarez',
 'Merced',
 'Candelaria']
```

Figura 13: Salida del algoritmo: rtua más corta de Hidalgo a Candelaria

El ejemplo anterior resulta interesante porque implicaría hacer transbordes entre líneas que uno como usuario podría pensar que no son necesarios o que resultan ser más tediosos. Para el ejemplo anterior, la siguiente imagen muestra dentro del mapa de la red del metro marcada en rojo, la ruta sugerida para llegar al destino planteado:

Figura 14: Ruta sugerida

Note que conseguir ir por la ruta más corta implica hacer un par de transbordes, el primero de la línea azul a la verde en la esación Bellas Artes, el segundo sería en Salto del Agua hacia la línea rosa y de ahí sobre esa misma ruta hasta llegar a Candelaria, sin embargo es probable que resulte ser más conveniente hacer sólo un transborde pues representa una situación más cómoda y se hace menos tiempo en llegar al destino que seguir la ruta más corta.

Escenarios como este plantean agregar otro tipo de datos como lo es el tiempo promedio de transborde entre

estaciones. Como éste caso, puede haber muchas otras situaciones que sin duda enriquecen la solución al problema planteado, pero que para fines de este trabajo han quedado fuera del scope.

Referencias

1. Dijkstra, E. W. (2022). A note on two problems in connexion with graphs. In *Edsger wybe dijkstra: His life, work, and legacy* (pp. 287–290).

2. Dijkstra's Shortest Path Algorithm - A Detailed and Visual Introduction. (2020). In *freeCodeCamp.org*. https://www.freecodecamp.org/news/dijkstras-shortest-path-algorithm-visual-introduction/

3. *Dijkstra_path — NetworkX 2.8.8 documentation*. (n.d.). Retrieved December 5, 2022, from https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.shortest_paths.weighted.dijkstra_path.html

4. Sanchez, G. (2012). Mapa del metro del DF con RgoogleMaps? In *Mextadísticas*. https://mextatistics.wordpress.com/2012/05/10/mapa-del-metro-del-df-en-rgooglemaps/

5. *Optimización*. (n.d.). Retrieved December 7, 2022, from https://itam-ds.github.io/analisis-numerico-computo-cientifico/README.html