

Mini-Router

Inteligencia Artificial

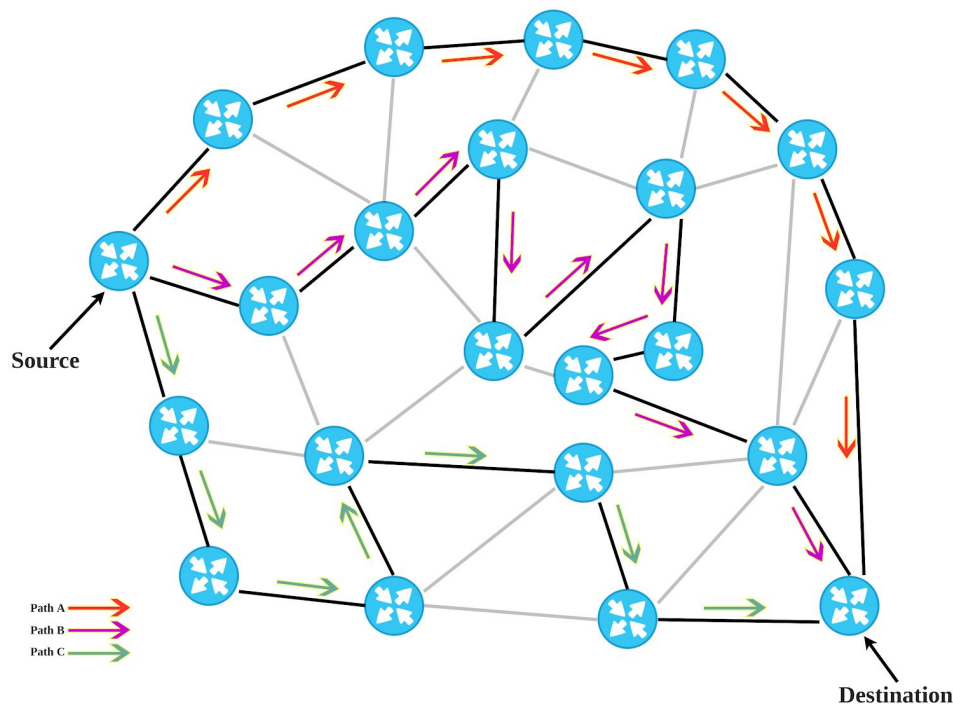
DIEGO VILLALVAZO SULZER - 155844
JOSÉ DE JESÚS SÁNCHEZ AGUILAR - 156190
SEBASTIÁN ARANDA BASSEGODA - 157465
SANTIAGO CORTIZO BARREIRO - 164596

¿Qué es un Router?

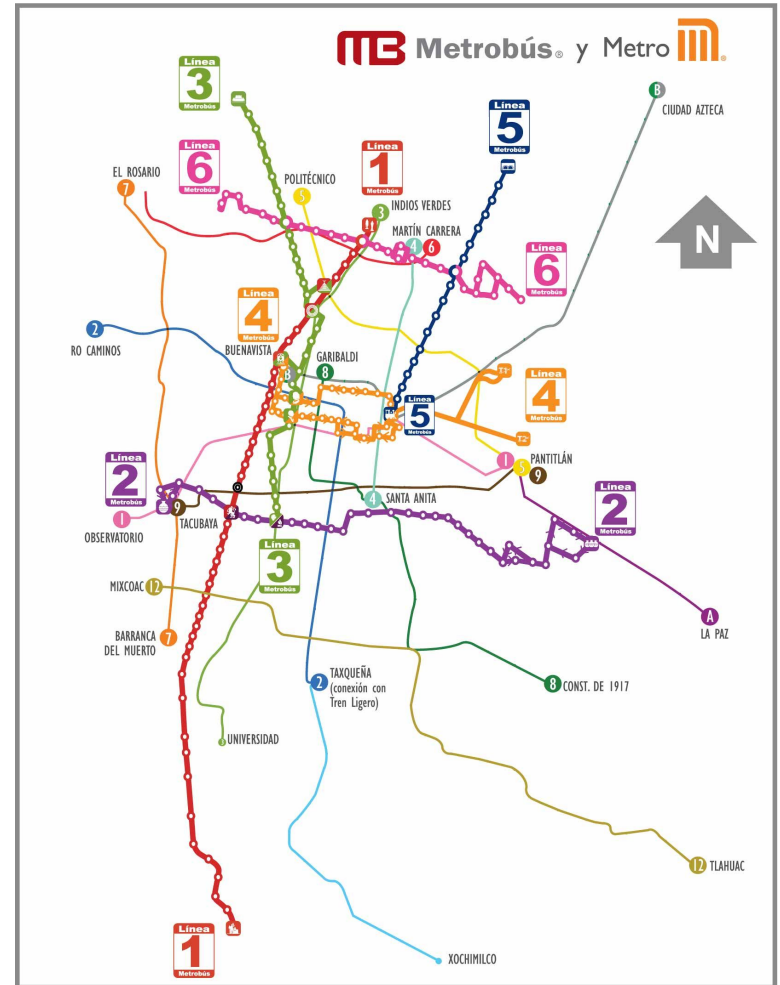
Es un sistema que permite interconectar en el ámbito de redes, computadoras.

Su función consiste en determinar la ruta que debe seguir cada paquete que pasa por el dispositivo.

En nuestro caso determinará la ruta “más eficiente” para seguir por un usuario que desee utilizar ya sea metro o metrobus.



Sistema de Metro y Metrobús de la CDMX



Datos

Información de las estaciones de la red de Metro y Metrobús obtenidos del portal de datos de la CDMX,

datos.cdmx.gob.mx

Los datos se ‘limpiaron’ con recursos de Javascript, se obtuvieron:

- 195 estaciones de metro repartidas en 12 líneas.
- 208 estaciones de metrobús repartidas en 6 líneas.



Representación de los datos en PROLOG

Una vez limpiados los datos se construyó para cada estación un hecho con el siguiente formato:

```
station(system, name, latitud, longitud, line, index).
```

Para cada estación se tiene:

- *system*: sistema que pertenece (metro o metrobús).
- *name*: nombre de la estación.
- *latitud* y *longitud*: coordenadas.
- *line*: línea del sistema a la que pertenece.
- *index*: índice que ordena un conjunto de estaciones de la misma línea. De Norte a Sur y de Oeste a Este.

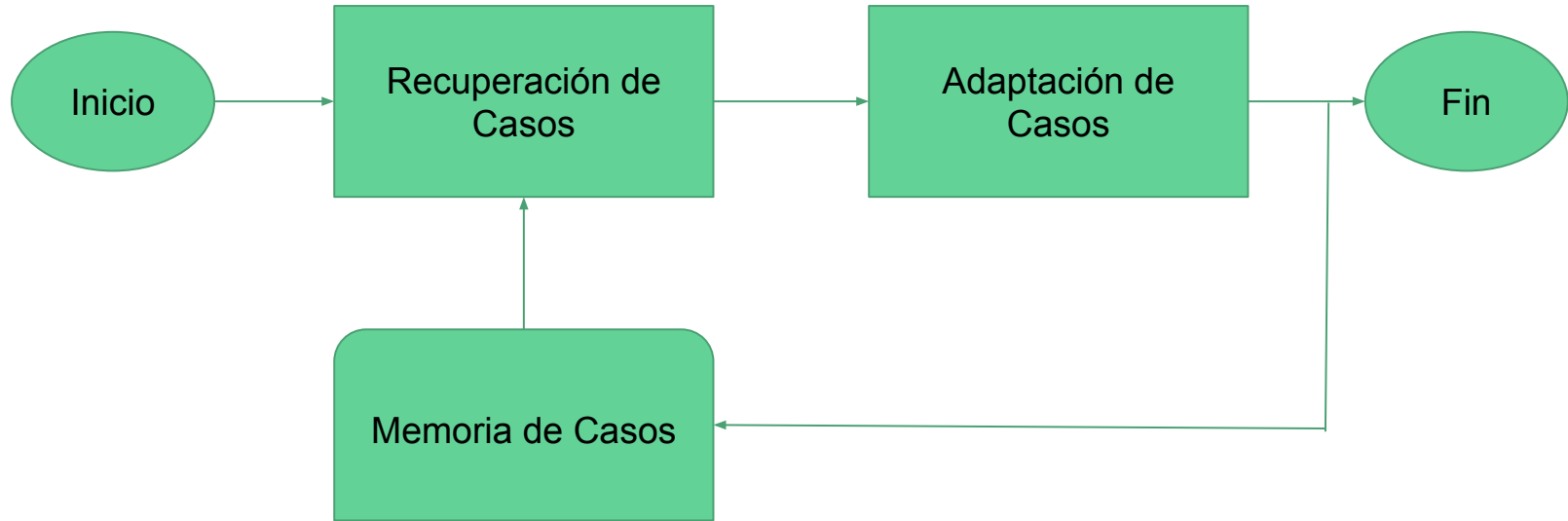
Representación de los datos en PROLOG

La representación de los datos en este formato permite encontrar las estaciones de transbordo y adyacentes de forma rápida y en ambos sistemas.

```
?- getSameStations(tacubaya, Transbordos).  
Transbordos = [[metro, tacubaya, 1], [metro, tacubaya, 7], [metro, tacubaya, 9], [mb, tacubaya, 2]].  
  
?- getSameStations(el_rosario, Transbordos).  
Transbordos = [[metro, el_rosario, 6], [metro, el_rosario, 7], [mb, el_rosario, 6]].  
  
?- getSameStations(pantitlan, Transbordos).  
Transbordos = [[metro, pantitlan, 1], [metro, pantitlan, 5], [metro, pantitlan, 9], [metro, pantitlan, 10]].
```

Además, para el cálculo de las distancias se utilizó la norma euclidiana tomando como puntos la latitud y longitud de cada estación.

Razonamiento Basado en Casos

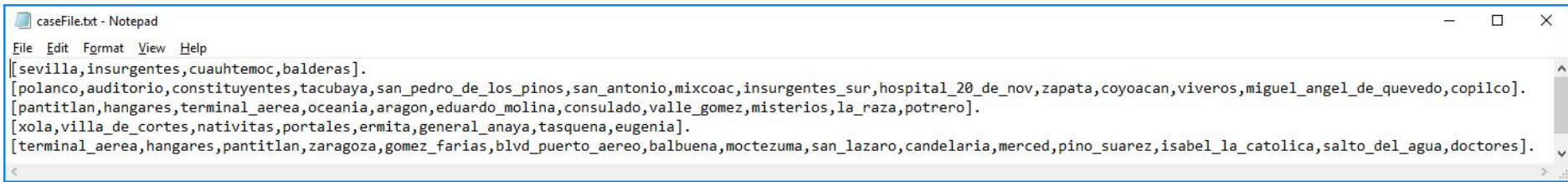


Caso

Un caso es una ruta que nuestro sistema ya haya calculado. Cada ruta contiene todas las estaciones por orden de origen a destino.

Memoria de Casos

La memoria de casos almacena todas las nuevas rutas que se calculen.



```
caseFile.txt - Notepad
File Edit Format View Help
[sevilla,insurgentes,cuauhtemoc,balderas].
[polanco,auditorio,constituyentes,tacubaya,san_pedro_de_los_pinos,san_antonio,mixcoac,insurgentes_sur,hospital_20_de_nov,zapata,coyoacan,viveros,miguel_angel_de_quevedo,copilco].
[pantitlan,hangares,terminal_aerea,oceania,aragon,eduardo_molina,consulado,valle_gomez,misterios,la_raza,potrero].
[xola,villa_de_cortes,nativitas,portales,ermita,general_anaya,tasquena,eugenia].
[terminal_aerea,hangares,pantitlan,zaragoza,gomez_farias,blvd_puerto_aereo,balbuena,moctezuma,san_lazaro,candelaria,merced,pino_suarez,isabel_la_catolica,salto_del_agua,doctores].
```



```
/**
 * auxCompatibleCase(input,input,input,output)
 * Verifica si las estaciones están en un caso
 * input1: Estación 1
 * input2: Estación 2
 * input3: Ruta visitada
 * output: Ruta
 **/
auxCompatibleCase(_,_,[],[]).
auxCompatibleCase(Station1, Station2,[Current|Tail],Case):-
    (member(Station1,Current),member(Station2,Current) ->
    Case = Current);
    auxCompatibleCase(Station1,Station2,Tail,Case), !.

/**
```

```

/**
 * adaptCase(input,input,input,output)
 * Acota el la ruta recibida, a las dos estaciones de entrada
 * input1: Estación 1
 * input2: Estación 2
 * input3: Ruta
 * output: Ruta que empiece en una estación y termine en la otra
 */
adaptCase(Station1,Station2,Caso,Res):-
    findCase(Station1,Station2,Caso,Res1),
    reverse(Res1,Res2),
    findCase(Station1,Station2,Res2,Res3),
    reverse(Res3,Res).

/**
 * findCase(input,input,input,output).
 * Busca un caso que tenga de inicio o fin alguna de las dos estaciones
 *
 * input1: Estación 1
 * input2: Estación 2
 * input3: Ruta
 * output: Ruta que empiece en una estación y termine en la otra
 */
findCase(_,_,[],[]):-
    !.
findCase(Station1,_,[Station1|Tail],[Station1|Tail]):-
    !.
findCase(_,Station2,[Station2|Tail],[Station2|Tail]):-
    !.
findCase(Station1,Station2,[_|Tail],Res):-
    findCase(Station1,Station2,Tail,Res).

```

Estrategia de Búsqueda

En caso de que la memoria de casos estuviera vacía o bien, no se encontrara un caso que se adaptara a la búsqueda en cuestión se realiza la búsqueda en la red del metro y metrobús mediante el **algoritmo A***.

```
getRoute(Origen, Destino, Ruta):-  
    compatibleCase(Origen, Destino, Case),  
    (listNotEmpty(Case) ->  
        adaptCase(Origen, Destino, Case, Ruta);  
        getPath(Origen, Destino, Ruta), newCase(Ruta)).
```

Búsqueda A*

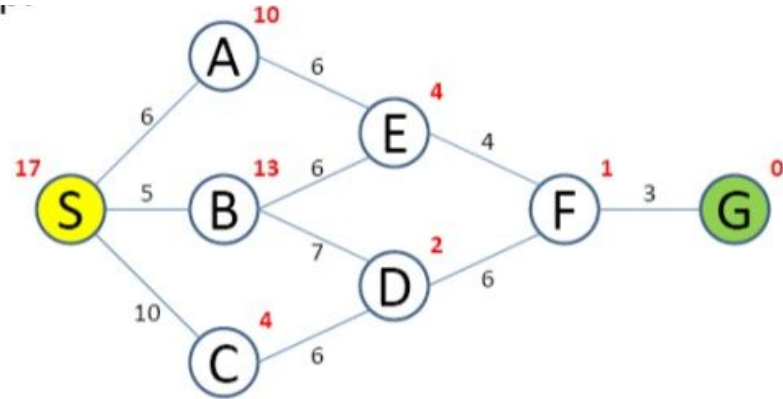
Algoritmo de búsqueda que se puede emplear en la búsqueda de caminos mínimos en una red.

Hace uso de una función de evaluación heurística que etiqueta cada nodo de la red.

$$f(n) = g(n) + h(n)$$

Donde:

- **$g(n)$** indica la distancia acumulada del nodo origen "S" al "N".
- **$h(n)$** expresa la distancia estimada desde el nodo "N" hasta el nodo destino "D".



Diseño de la búsqueda A*

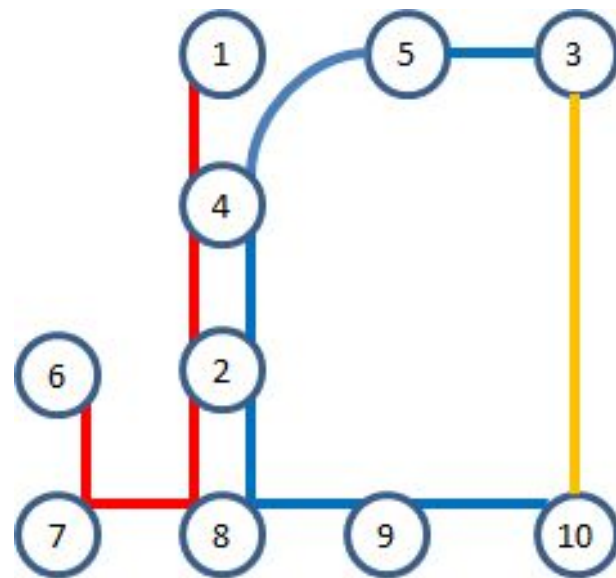
Se programaron una serie de métodos/reglas auxiliares en PROLOG.

- **Cola de prioridades:** una lista que recibe una estación con los valores $g(n)$ y $h(n)$, y los inserta en la lista calculando el valor $f(n)$.
- **Estaciones adyacentes:** para una estación en particular se obtienen las estaciones cercanas a la misma y con ayuda del nodo destino y de la norma euclidiana se decide a qué estación avanzar.
- **Estaciones de conexión:** busca las estaciones en la base de conocimiento que tengan el mismo nombre, esto nos regresa una lista con las estaciones de conexión tanto en un sistema como en otro.



Algoritmo A*

1. Tomar el nodo inicial o el más chico en la cola de prioridades.
2. Expandir el nodo tomado, obtener sus nodos hijos (siguiente estación y conexiones), obtener sus valores heurísticos y añadirlos a la cola de prioridades.
3. Repetir el paso 1 hasta que el nodo inicial sea el nodo destino.
4. Encontrado el destino, realizar “brack-tracking” con el camino encontrado.



Ejemplo de consulta.

```
2 ?- main.  
Bienvenido a Weis, el mejor sistema inteligente de navegaci3n en la CDMX.  
Por favor ingrese el origen de su viaje.  
|: polanco.  
Ahora, escriba el destino de su viaje.  
|: copilco.  
La ruta a seguir es:  
polanco -> auditorio -> constituyentes -> tacubaya -> san_pedro_de_los_pinos -> san_antonio -> mixcoac -> insurgentes_sur -> hospital_20_de_no  
v -> zapata -> coyoacan -> viveros -> miguel_angel_de_quevedo -> copilco -> Â¡Has llegado a tu destino!  
Â¿Desea hacer otro viaje?  
|: si.  
Por favor ingrese el origen de su viaje.  
|: pantitlan.  
Ahora, escriba el destino de su viaje.  
|: potrero.  
La ruta a seguir es:  
pantitlan -> hangares -> terminal_aerea -> oceania -> aragon -> eduardo_molina -> consulado -> valle_gomez -> misterios -> la_raza -> potrero  
-> Â¡Has llegado a tu destino!  
Â¿Desea hacer otro viaje?  
|: no.  
Gracias por usar Weis.
```


Mejoras y trabajo futuro

El diseño de esta solución permite integrar al dominio de búsqueda otros sistemas de transporte en la ciudad como:

- Ecobici,
- RTP,
- Transportes eléctricos,
- Cablebús,
- Tren suburbano, entre otros.



Mejoras y trabajo futuro

También es posible eficientar la recuperación y adaptación de casos para encontrar una ruta aún más óptima.

Mejorar aspectos de la heurística para minimizar factores como

- Tiempo de traslado total,
- Número de estaciones recorridas,
- Número de transbordos,
- Número de cambios de sistema,
- Costo del recorrido,
- Horas 'pico', entre otros.



Conclusiones

- Se logró afianzar los conocimientos obtenidos sobre PROLOG, mini-router y búsqueda A*.
- La búsqueda en A* resulta más eficiente que otros algoritmos de búsqueda en grafos como Dijkstra.
- Se pudo observar y comprender la eficiencia que presenta la implementación de un mini-router que combina los aspectos de razonamiento basado en casos en conjunto con la búsqueda A*.
- Existen diversas formas de implementar el router, priorizando los diferentes métodos de razonamiento y algoritmos de búsqueda.

Referencias

1. Russel, S and Norvig, P, Artificial Intelligence: A Modern Approach, Prentice Hall, 2010.
2. Agencia Digital de Inovación Pública CDMX - <https://datos.cdmx.gob.mx/pages/home/>
3. Movilidad Integrada de la Ciudad de México -
<https://semovi.cdmx.gob.mx/storage/app/media/Mapa%20MI%20CDMX%20JPG.jpg>