Engineering method

Phase one

Problem Identification:

Sometimes, getting around in the mass transit system (MIO) can be a bit confusing and even tedious, even more so if you are not clear about where you are going, and which route would be your best option. Bearing in mind that this best option should be the one that save the most time possible, in which the section traveled is of shorter distance in relation to the others and as far as possible, that there is no traffic, this last part is the most difficult to guarantee, but it can be considered when choosing one route over another.

It has been identified that, if people do not know the best route to their destination, they may face different complications such as:

Loss of time: Without knowing the best route, people may end up taking longer roads, with more traffic or even taking the wrong direction. This can result in considerable loss of time during the trip.

Traffic congestion: If many people do not know the best route, they are likely to follow the same main roads, which can lead to traffic congestion on those routes. This can lead to additional delays and frustration for drivers and passengers.

Higher fuel expenses: By taking longer or inefficient routes, people may consume more fuel than necessary. This can result in higher gasoline costs, affecting the travel budget (in case they have not been able to be located within the bus stations and have opted for other transportation alternatives).

Stress and frustration: Not knowing the best route can generate stress and frustration for people, especially if they are under time pressure or if they are in an unfamiliar area. This can negatively affect their travel experience and overall well-being.

Dependence on external help: If people do not know the route to their destination, they may rely on help from others, such as asking for directions or using navigation apps on their mobile devices. This can be helpful, but it also implies external dependence and the possibility of receiving inaccurate or confusing information.

Therefore, it is required that the public service platform provides accurate information on the most convenient transportation in terms of time and proximity to the user. The system must internally evaluate all the options and inform the user of the best one.

Requirements

Customer	Cali's integrated mass transit system (MIO)
User	Users of this system
Functional requirements	R.F 1 = Enter the user's place of departure and destination. R.F 2 =To know the distances between the different stations. R.F 3 = Providing the routes that pass-through a given station. R.F 4 = Evaluate the best route to the desired destination.
Context of the problem	Sometimes users need a more precise and clearer guide to help them find their way around the stations of the mass transit system. It is not always clear how to get to the desired place using this means of transport and although it is true that you can ask, the indications (sometimes) can be wrong and cause users to end up in totally different destinations.
Non-functional requirements	RNF1: Scalable Software

Name or identifier	Enter the user's place of departure and destination.		
Summary	The user will enter his departure station and the place where he wants to arrive, i.e., his destination.		
Inputs	**input name**	**Data type**	**Selection or repetition condition**
	departureStation	String	The departure station must be within the city of Cali.
General activities needed to obtain the results	1. Display a list of stations near the user's location. 2. The user will type in the starting point according to where he/she is. 3. Likewise, the user will type in the destination to which he/she wants to go. 4. Display the stations near the desired destination.		
Result or postcondition	Place of departure and destination determined		
Outputs	**Output name**	**Data type**	**Selection or repetition condition**

	NO OUTPUTS		

Name or identifier	To know the distances between the different stations.		
Summary	One of the steps to select the best route will be to know the distance between the stations and thus evaluate the different routes.		
Inputs	input name	Data type	Selection or repetition condition
	Not inputs		
General activities needed to obtain the results	Collect and store the distance information between the different stations in a suitable database or data structure. Given the starting point and the destination point, obtain the distance between them.		
Result or postcondition	Accurate information on the distances between the required stations.		
Outputs	Output name	Data type	Selection or repetition condition
	Distance	Double	

Name or identifier	Providing the routes that pass-through a given station.		
Summary	Each station is served by a specific bus route, so it is important to have access to this information to determine which bus routes will connect both stations, sometimes there is no direct route, so it is necessary to transfer and make use of two or more buses.		
Inputs	**input name**	**Data type**	**Selection or repetition condition**
	No inputs		
General activities needed to obtain the results	Obtain the bus routes that pass through the respective stations. Determine if there is a route that directly connects both stations. If not, the information of the other bus routes must be accessed. and select those that establish a connection between these two stations.		
Result or postcondition	We will obtain the routes that in some way connect both stations		
Outputs	**Output name**	**Data type**	**Selection or repetition condition**
	busRoute	String	If this connects one station to the other in some way

Name or identifier	Display the stations near the desired destination.		
Summary	Once the previous steps have been accomplished, we only must compare how long a route takes to the destination point, in relation to how long another route takes, and among all these, select the best one.		
Inputs	**input name**	**Data type**	**Selection or repetition condition**
	No inputs		
General activities needed to obtain the results	Obtain the routes connecting the stations. compare them with each other, until the one that takes the least time is obtained.		
Result or postcondition	The fastest route to our destination.		
Outputs	**Output name**	**Data type**	**Selection or repetition condition**
	bestRoute	String	

Phase 2

GATHERING THE NECESSARY INFORMATION:

GUÍA PARA LLEGAR A TU DESTINO

1 Ubica tu punto de partida en el mapa y determina la zona donde te encuentras, y a la que te diriges.

2 Identifica tu ruta de acuerdo a tu origen y destino en el mapa.

E27B

3 Al elegir la ruta y trayecto, ten en cuenta los trasbordos necesarios para llegar a tu destino.

E27B → 7 → A76

4 Revisa y valida las conexiones que elegiste para tu viaje con la tabla de horarios.

AVISO FRONTAL

E27B AV1N - CALLE 5 ◄ Recorrido
CAM - U. DEPORTIVA ◄ Destino

◄ Literal consecutivo
◄ Zonas de tránsito
◄ Tipo de ruta ►

TIPO DE RUTA

E	Expresa
T	Troncal
P	Pretroncal
A	Alimentadora

MAPA POR ZONAS

Norte

2 Menga Av. 3

3 Paso del Comercio Kr. 1

0 Centro Cl. 13 - Cl. 15

8 Calipso Cl. 36

4 Sanín Kr. 15

Oriente

7 Cañaveralejo Cl. 5

6 Simón Bolívar Cl. 25

5 Nuevo Latir Tr. 25 - Kr. 28D

Occidente

1 Universidades Kr. 100

Sur

SIMBOLOGÍA

ZONA

- **0** Centro
- **1** Universidades
- **2** Menga
- **3** Paso del Comercio
- **4** Andrés Sanín
- **5** Nuevo Latir
- **6** Simón Bolívar
- **7** Cañaveralejo
- **8** Calipso

Terminal
Dirección
Terminal o Estación de Cabecera

Estación
Dirección
Estación

N Norte

Estación de MIO Cable

Parada en estación y/o Terminal

iV Integración Virtual

Bici Parqueaderos en estación y/o Terminal

Museos en Estación y/o Terminal

Centros Comerciales

Zona de Restaurantes

Bares y Discotecas

Supermercados

Museos

Ampliación Zona Centro

TABLA DE RUTAS Y HORARIOS

HORARIO DE ATENCIÓN MIO Y MIO CABLE EN ESTACIONES:
- Lunes a Sábado: 5:00 a.m. a 11:00 p.m.
- Domingo y Festivo: 6:00 a.m. a 10:00 p.m.

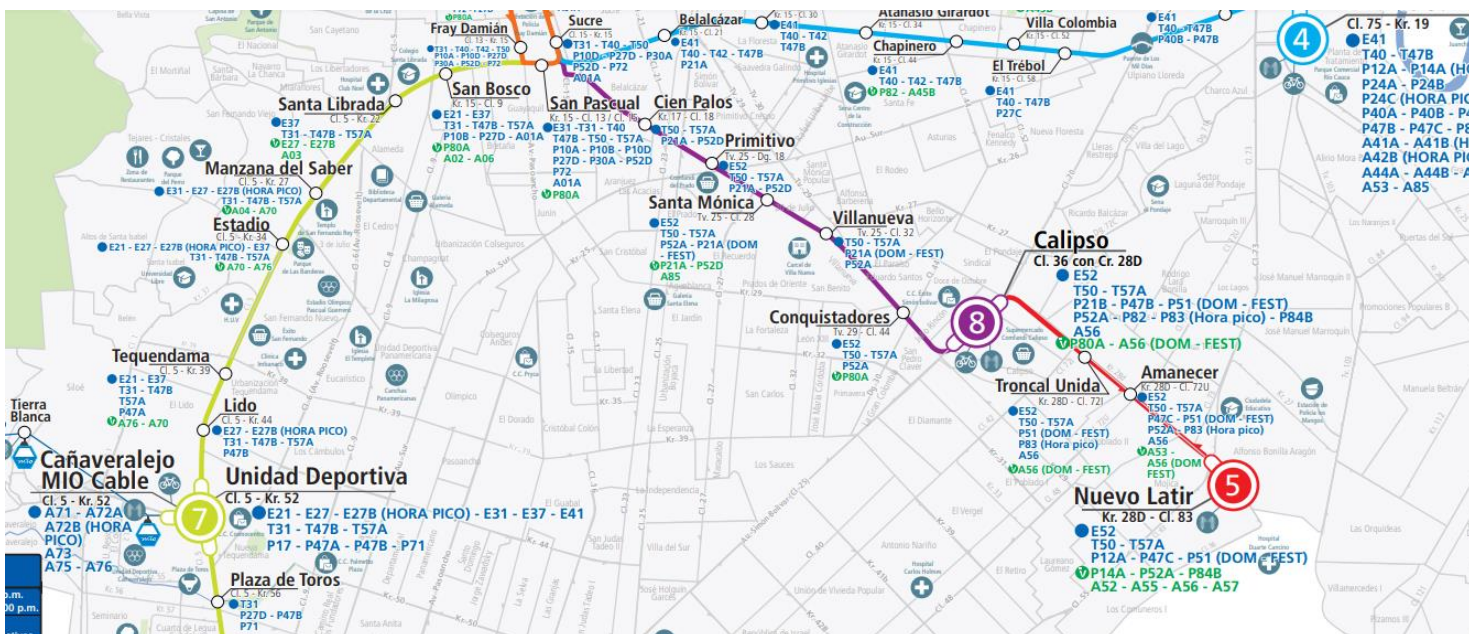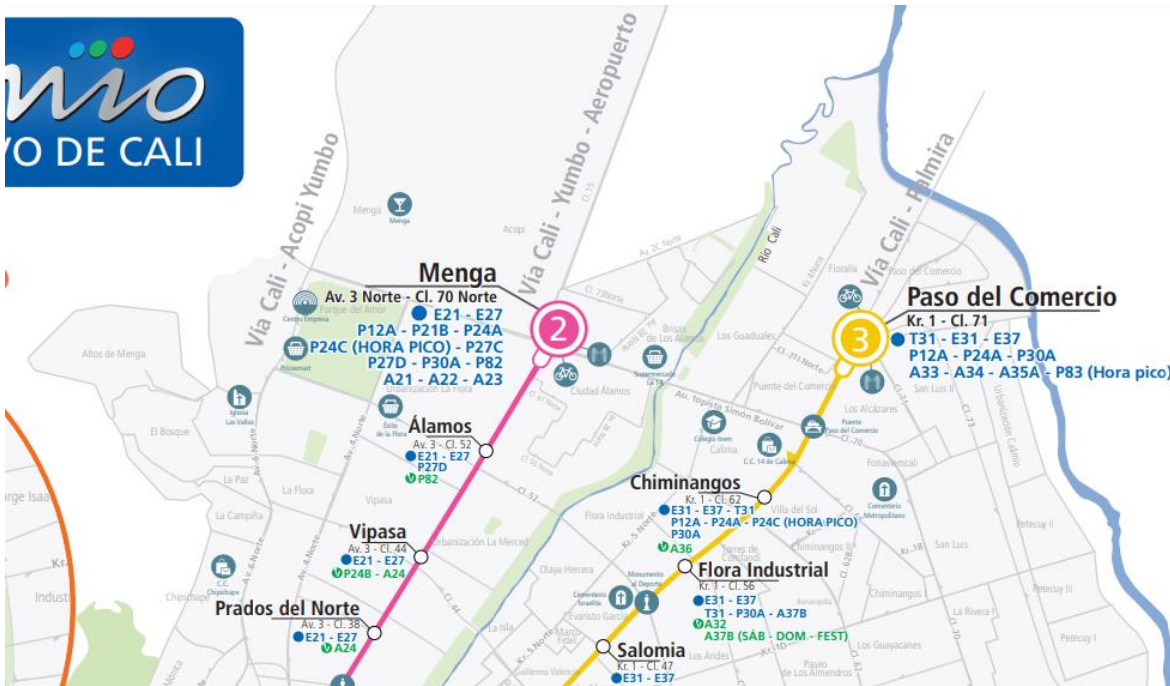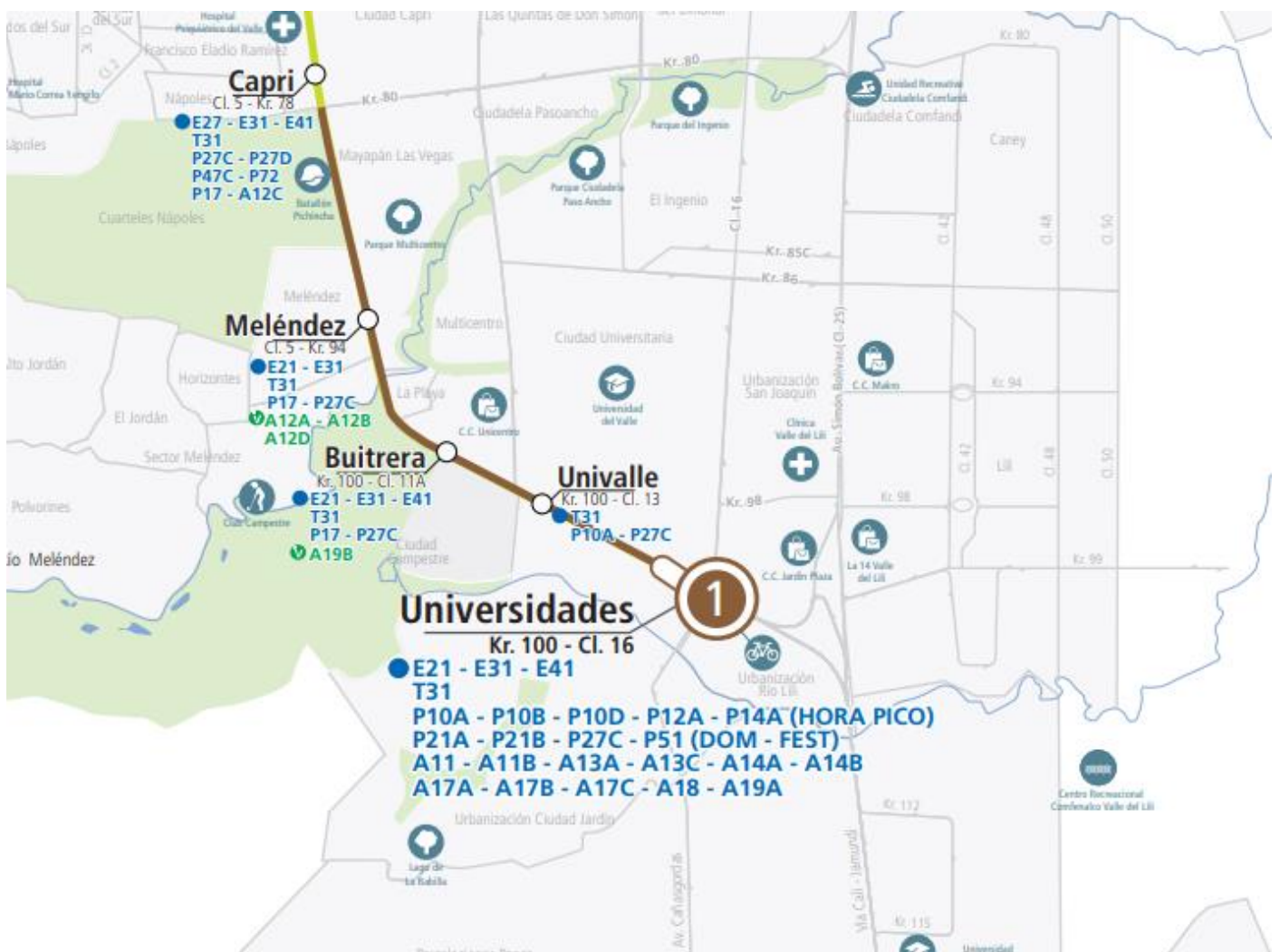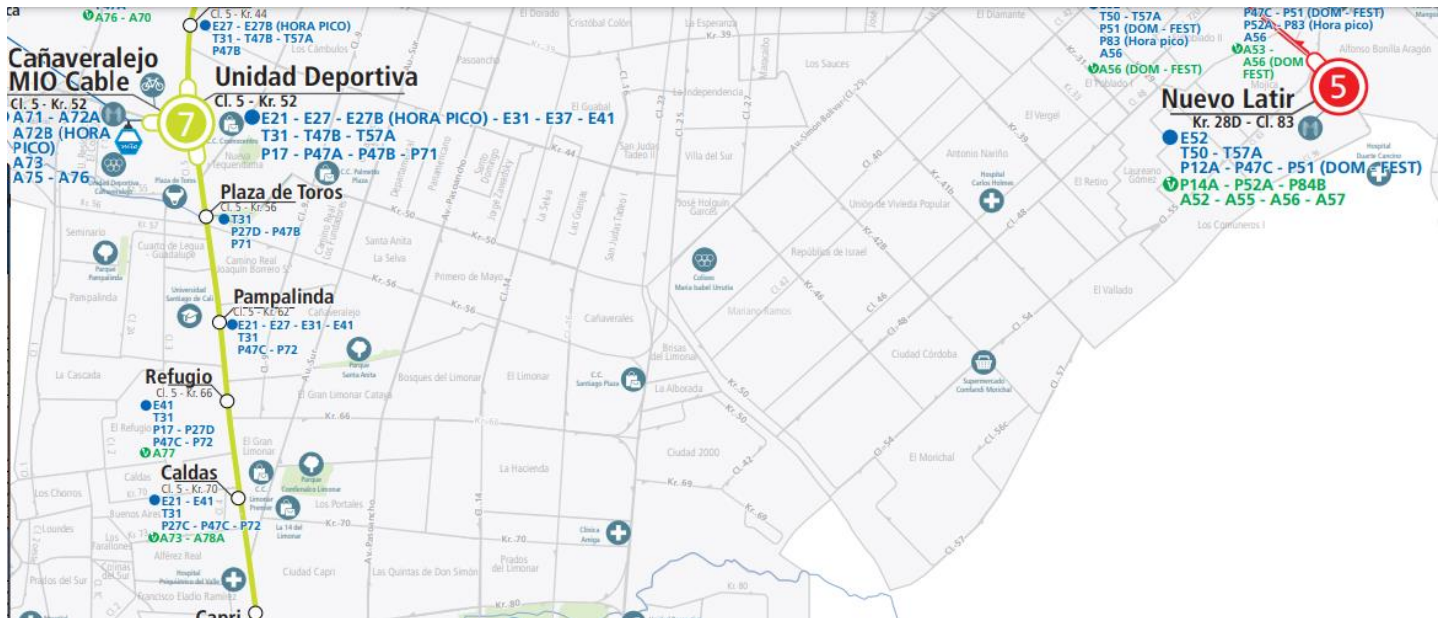| Ruta	Origen - Destino	De Lunes a Viernes	Sábados	Domingos y Festivos
E	**EXPRESAS**	Rutas que operan con buses articulados y padrones por corredores con carril exclusivo. Se detienen en algunas de las estaciones, permitiendo reducir el tiempo de viaje entre las estaciones que conecta.		
E21	Terminal Menga - Universidades	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
E27	Terminal Menga - Capri	5:00 am - 10:00 pm	5:00 am - 10:00 pm	NO OPERA
E27B	Av. Las Américas - Unidad Deportiva	5:00 - 9:00 am y 4:00 - 8:00 pm	NO OPERA	NO OPERA
E31	Terminal Paso del Comercio - Universidades	6:00 am - 9:00 pm	06:00 - 09:00 am	NO OPERA
E37	Terminal Paso de Comercio - Unidad Deportiva	05:00 am - 10:00 pm	06:00 am - 08:00 pm	NO OPERA
E41	Terminal Andrés Sanín - Universidades	05:00 am - 10:00 pm	05:00 am - 08:00 pm	08:00 am - 08:00 pm
E52	Nuevo Latir - Terminal	5:00 - 9:00 am y 4:00 - 8:00 pm	06:00 - 09:00 am	NO OPERA
T	**TRONCALES**	Rutas que operan con buses articulados y padrones por corredores con carril exclusivo, atendiendo todas las estaciones.		
T31	Terminal Paso del Comercio - Universidades	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
T40	Terminal Andrés Sanín - Centro	5:00 am - 10:00 pm	5:00 am - 10:00 pm	6:00 am - 9:00 pm
T42	Pizamos - Centro - Terminal	5:00 am - 11:00 pm	5:00 am - 11:00 pm	NO OPERA
T47B	Terminal Andrés Sanín - Unidad Deportiva	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
T50	Nuevo Latir - Centro	5:00 am - 10:00 pm	5:00 am - 10:00 pm	6:00 am - 9:00 pm
T57A	Nuevo Latir - Unidad Deportiva	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
P	**PRETRONCALES**	Rutas que operan con buses padrones por vías principales y secundarias. Conectan diferentes zonas de la ciudad, integrándose con el sistema troncal en algunas estaciones.		
P10A	Universidades - Centro	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
P10B	Universidades - El Ingenio - San Bosco	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
P10D	Universidades - Centro	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
P12A	Universidades - Nuevo Latir - T. A. Sanín - Centro Empresa	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
P14A	Universidades - Terminal Andrés Sanín	5:00 - 9:00 am y 4:00 - 8:00 pm	05:00 - 09:00 am	NO OPERA
P17	Icesi - Unidad Deportiva	5:00 - 9:00 am y 4:00 - 8:00 pm	NO OPERA	NO OPERA
P21A	Universidades - Centro - Av. Las Américas	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
P21B	Universidades - Terminal Menga	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
P24A	Terminal Menga - Terminal Andrés Sanín	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
P24B	Terminal Andrés Sanín - CAM	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
P24C	Terminal Menga - Andrés Sanín - Desepaz	5:00 - 8:00 am y 4:00 - 8:00 pm	NO OPERA	NO OPERA
P27C	Terminal Menga - Universidades	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
P27D	Terminal Menga - Capri	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
P30A	Terminal Paso del Comercio - Terminal Menga - Centro	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
P40A	Terminal Andrés Sanín - Centro	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
P40B	Terminal Andrés Sanín - Sena - Centro	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
P47A	Terminal Andrés Sanín - Unidad Deportiva	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
P47B	Terminal Andrés Sanín - Unidad Deportiva	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
P47C	Terminal Andrés Sanín - Capri	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
P51	Pízamos - T. Calipso - Universidades	NO OPERA	NO OPERA	7:00 am - 8:00 pm
P52A	Nuevo Latir - Av. Las Américas	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
P52D	Ciudad Córdoba - Centro - Terminal	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
P71	Caney - Unidad Deportiva	5:00 am - 11:00 pm	5:00 am - 11:00 pm	NO OPERA
P72	Capri - Centro Empresa	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
P80A	Terminal Calipso - Santa Elena - Centro	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
P82	Terminal Calipso - Terminal Menga	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
P83	T. Paso del Comercio - Av. Ciudad de Cali - T. Calipso	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
P84B	Terminal Andrés Sanín - Tr. 103 - Calipso	5:00 - 9:00 am y 4:00 - 8:00 pm	05:00 - 09:00 am	NO OPERA

A	**ALIMENTADORAS**	Rutas que operan con buses complementarios y padrones. Conectan directamente los barrios de la ciudad con algunas de las estaciones y/o terminales del Sistema.		
A01A	San Bosco - CAM - Centro	5:00 am - 10:00 pm	5:00 am - 10:00 pm	6:00 am - 10:00 pm
A02	San Bosco - Zoológico - Atenas	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
A03	Santa Librada - Nacional	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
A04	Manzana del Saber - Bellavista	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
A05	Las Américas - CAM - La Portada	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
A06	San Bosco - Aguacatal	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
A11	ICESI - PUJ - Universidades	5:00 am - 11:00 pm	5:00 am - 11:00 pm	NO OPERA
A11B	Universidad Libre - Alférez Real - Universidades	6:00 am - 7:30 pm	7:00 am - 7:00 pm	NO OPERA
A12A	Altos de la Luisa - Estación Meléndez	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
A12B	Meléndez - Las Palmas	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
A12C	Meléndez - Capri	5:00 am - 11:00 pm	5:00 am - 11:00 pm	NO OPERA
A12D	Meléndez - Altos de Santa Elena	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
A13A	Universidades - Caney	5:00 am - 11:00 pm	5:00 am - 11:00 pm	5:00 am - 10:00 pm
A13B	Las Vegas de Comfandi - Caney	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
A13C	Universidades - Lili	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
A14A	Universidades - U. San Buenaventura	5:00 am - 11:00 pm	5:00 am - 7:00 pm	NO OPERA
A14B	Universidades - La Vorágine	NO OPERA	5:00 am - 11:00 pm	6:00 am - 10:00 pm
A17A	Universidades - U. San Martín	5:00 am - 11:00 pm	5:00 am - 8:00 pm	NO OPERA
A17B	Universidades - Hormiguero	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
A17C	Universidades - Comfenalco Cañasgordas	NO OPERA	5:00 am - 11:00 pm	6:00 am - 10:00 pm
A18	Universidades - Hacienda El Castillo	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 9:00 pm
A19A	Universidades - Comfandi Pance	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
A19B	Estación Buitrera - La Buitrera	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm

A19B	Estación Buitrera - La Buitrera	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
A21	Terminal Menga - Ciudadela Floralia	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
A22	Terminal Menga - Brisas de los Álamos	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
A23	Terminal Menga - Ciudadela los Álamos	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
A24	San Miguel - El Bosque - Vipasa	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
A32	Chiminangos - Flora Industrial	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
A33	Ciudadela Floralia - Terminal Paso del Comercio	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
A34	Alcázares - Terminal Paso del Comercio	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
A35A	Calimio Norte - Terminal Paso del Comercio	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
A36	Guayacanes - Chiminangos	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
A37A	SENA - Salomia	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
A37B	SENA - Flora Industrial	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
A41A	Pízamos - Terminal Andrés Sanín	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
A41B	Decepaz - Terminal Andrés Sanín	5:00 - 9:00 am y 4:00 - 8:00 pm	5:00 - 9:00 am y 4:00 - 8:00 pm	NO OPERA
A42B	Terminal Andrés Sanín - La Casona	5:00 - 9:00 am y 4:00 - 8:00 pm	NO OPERA	NO OPERA
A44A	Terminal Andrés Sanín - Manuela Beltrán	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
A44B	Terminal Andrés Sanín - Puertas del Sol	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
A45B	San Marino - Villa Colombia	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
A47	Terminal Andrés Sanín - Pízamos	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
A52	Nuevo Latir - H. Isaías Duarte Cancino	6:00 am - 6:00 pm	NO OPERA	NO OPERA
A53	Terminal Andrés Sanín - Los Lagos - Amanecer	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
A55	Compartir - Manuela Beltrán - Nuevo Latir	5:00 am - 11:00 pm	5:00 am - 11:00 pm	NO OPERA
A56	Pízamos - Nuevo Latir	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
A57	Villa Luz - Nuevo Latir	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
A70	Mortiñal - Tequendama	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 9:00 pm
A71	Los Chorros - Terminal Cañaveralejo	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm
A72A	Jardines de la Aurora - Terminal Cañaveralejo	5:00 am - 11:00 pm	5:00 am - 11:00 pm	6:00 am - 10:00 pm

mío
YO DE CALI

N

Via Cali - Acopi Yumbo · Via Cali - Yumbo - Aeropuerto · Via Cali - Palmira

Menga
Av. 3 Norte - Cl. 70 Norte
- E21 - E27
- P12A - P21B - P24A
- P24C (HORA PICO) - P27C
- P27D - P30A - P82
- A21 - A22 - A23

2

Paso del Comercio
Kr. 1 - Cl. 71
- T31 - E31 - E37
- P12A - P24A - P30A
- A33 - A34 - A35A - P83 (Hora pico)

3

Álamos
Av. 3 - Cl. 52
- E21 - E27
- P27D
- P82

Chiminangos
Kr. 1 - Cl. 62
- E31 - E37 - T31
- P12A - P24A - P24C (HORA PICO)
- P30A
- A36

Vipasa
Av. 3 - Cl. 44
- E21 - E27
- P24B - A24

Flora Industrial
Kr. 1 - Cl. 56
- E31 - E37
- T31 - P30A - A37B
- A32
- A37B (SÁB - DOM - FEST)

Prados del Norte
Av. 3 - Cl. 38
- E21 - E27
- A24

Salomia
Kr. 1 - Cl. 47
- E31 - E37

0

Petecuy
Cl. 15 - Kr. 8
- E21 - E37
- T31 - T40 - T42 - T50
- P10A - P10D - P21A - P27D
- P30A - P52D - P72
- A01A

Santa Rosa
Cl. 13 - Kr. 10
- E21 - E37
- T31 - T40 - T42 - T50
- P10A - P10D - P27D
- P30A - P52D - P72
- A01A

Fray Damián
Cl. 13 - Kr. 15
- E31 - T40 - T42 - T50
- P10A - P10D - P27D
- P30A - P52D - P72

Belal
Cl. 15 - Kr. 15

Sucre
Cl. 15 - Kr. 15
- T31 - T40 - T50
- P10D - P27D - P30A
- P52D - P72
- A01A

San Bosco
Kr. 15 - Cl. 9

San Pascual
Kr. 15

Salomia
Kr. 1 - Cl. 47
- E31 - E37
- T31
- A37A

Popular
Kr. 1 - Cl. 44
- E31 - T31

Manzanares
- E37 - T31

Las Américas
Av. 3N - Cl. 23AN
- E21 - E27 - E27B (HORA PICO)
- P52A - P72

Versalles
- E21 - E27 - E37 - E52
- T42 - P21A - P52D
- A05

Fátima
Kr. 1 - Cl. 30
- T31

Río Cali

Piloto
- T31

Torre de Cali
Av. 3N - Kr. 4N

San Nicolás
- T31

La Ermita

San Pedro

Plaza Caycedo

Centro
Cl. 13 - Kr. 8
- E21 - E37
- T31 - T40 - T42 - T50
- P10A - P10D - P21A - P27D
- P30A - P52D - P72
- A01A

Santa Rosa

Fray Damián
- T31 - T40 - T50
- P10D - P27D - P30A - P21A
- P52D - P72

San Bosco
Cl. 15 - Cl. 9
- E21 - E27
- T31 - T47B - T57A
- P10B - P27D - A01A

Santa Librada
Cl. 5 - Kr. 22
- E37
- T31 - T47B - T57A
- E27 - E27B
- A03

San Pascual
Kr. 15 - Cl. 13 / Cl. 16
- E31 - T31 - T40
- T47B - T50 - T57A
- P10A - P10B - P10D
- P27D - P30A - P52D
- P72
- A01A

Manzana del Saber

Cien Palos
Kr. 17 - Cl. 18
- T50 - T57A
- P21A - P52D

Primitivo
Tv. 25 - Dg. 18
- E52
- T50 - T57A
- P21A - P52D

Belalcázar
Cl. 15 - Cl. 21
- E41
- T40 - T42 - T47B
- P21A

Floresta
Cl. 15 - Cl. 30
- E41
- T40 - T42
- T47B

Atanasio Girardot
Cl. 15 - Cl. 34
- E41
- T40 - T42 - T47B

Chapinero
Kr. 15 - Cl. 44
- E41

Villa Colombia
Cl. 15 - Cl. 52
- E41

El Trébol
Cl. 15 - Cl. 58
- E41
- T40 - T47B
- P27C

7 de Agosto
Cl. 15 - Cr. 71A
- E41
- T40 - T47B
- P40B - P47B

4

Andrés Sanín
Cl. 75 - Kr. 19
- E41
- T40 - T47B
- P12A - P14A (HORA...)
- P24A - P24B
- P24C (HORA PICO)
- P40A - P40B - P47...
- P47B - P47C - P84...
- A41A - A41B (HO...)
- A42B (HORA PICO)
- A44A - A44B - A4...
- A53 - A85

Fray Damián

Sucre
- T31 - T40 - T50
- P10D - P27D - P30A - P21A
- P52D - P72

Belalcázar
Cl. 15 - Cl. 30

Atanasio Girardot

Villa Colombia

4
Cl. 75 - Kr. 19
- E41
- T40 - T47B
- P12A - P14A (HO...)
- P24A - P24B
- P24C (HORA PICO)
- P40A - P40B - P4...
- P47B - P47C - P84...
- A41A - A41B (HO...)
- A42B (HORA PICO)
- A44A - A44B - A4...
- A53 - A85

Chapinero
Kr. 15 - Cl. 34
- E41
- T40 - T42 - T47B

El Trébol
Cl. 15 - Cl. 58
- E41
- T40 - T47B
- P27C

San Bosco
Cl. 15 - Cl. 9
- E21 - E27
- T31 - T47B - T57A
- P10B - P27D - A01A

Santa Librada
Cl. 5 - Kr. 22
- E37
- T31 - T47B - T57A
- E27 - E27B
- A03

Manzana del Saber
Cl. 5 - Kr. 27
- E31 - E27 - E27B (HORA PICO)
- T31 - T47B - T57A
- A04 - A70

San Pascual
Kr. 15 - Cl. 13 / Cl. 16
- E31 - T31 - T40
- T47B - T50 - T57A
- P10A - P10B - P10D
- P27D - P30A - P52D
- P72
- A01A

Cien Palos
Kr. 17 - Cl. 18
- T50 - T57A
- P21A - P52D

Primitivo
Tv. 25 - Dg. 18
- E52
- T50 - T57A
- P21A - P52D

Santa Mónica
Tv. 25 - Cl. 32
- T50 - T57A
- P52A - P21A (DOM - FEST)
- P21A - P52D
- A85

Villanueva
Tv. 25 - Cl. 32
- T50 - T57A
- P21A (DOM - FEST)
- P52A

Conquistadores
Tv. 29 - Cl. 44
- T50 - T57A
- P52D
- P80A

8

Calipso
Cl. 36 con Cr. 28D
- E52
- T50 - T57A
- P21B - P47B - P51 (DOM - FEST)
- P52A - P82 - P83 (Hora pico) - P84B
- A56
- P80A - A56 (DOM - FEST)

Troncal Unida
Kr. 28D - Cl. 72I
- E52
- T50 - T57A
- P51 (DOM - FEST)
- P83 (Hora pico)
- A56
- A56 (DOM - FEST)

Amanecer
- E52
- T50 - T57A
- P47C - P51 (DOM - FEST)
- P52A - P83 (Hora pico)
- A53
- A56 (DOM - FEST)

5

Nuevo Latir
Kr. 28D - Cl. 83
- E52
- T50 - T57A
- P12A - P47C - P51 (DOM - FEST)
- P14A - P52A - P84B
- A52 - A55 - A56 - A57

Estadio
Cl. 5 - Cr. 34
- E21 - E27 - E27B (HORA PICO) - E37
- T31 - T47B - T57A
- A70 - A76

Tequendama
Cl. 5 - Cr. 39
- E21 - E27
- T31 - T47B
- T57A
- P47A
- A76 - A70

Lido
Cl. 5 - Kr. 44
- E27 - E27B (HORA PICO)
- T31 - T47B - T57A
- P47B

Cañaveralejo MIO Cable
Cl. 5 - Kr. 52
- A71 - A72A
- A72B (HORA PICO)
- A73
- A75 - A76

7

Unidad Deportiva
Cl. 5 - Kr. 52
- E21 - E27 - E27B (HORA PICO) - E31 - E37 - E41
- T31 - T47B - T57A
- P17 - P47A - P47B - P71

Plaza de Toros
Cl. 5 - Kr. 56
- T31
- P27D - P47B
- P71

For a better visualization, visit: https://www.habita.com.co/res/files/mio.pdf

Dijkstra's algorithm:

The algorithm maintains a set of visited vertices and a set of unvisited vertices. It starts at the source vertex and iteratively selects the unvisited vertex with the smallest tentative distance from the source. It then visits the neighbors of this vertex and updates their tentative distances if a shorter path is found. This process continues until the destination vertex is reached, or all reachable vertices have been visited.

Need for Dijkstra's Algorithm (Purpose and Use-Cases)

The need for Dijkstra's algorithm arises in many applications were finding the shortest path between two points are crucial.

For example, it can be used in the routing protocols for computer networks and used by map systems to find the shortest path between starting point and the destination.

Basic Characteristics of Dijkstra's Algorithm

Below are the basic steps of how Dijkstra's algorithm works:

So basically, Dijkstra's algorithm starts at the node source node we choose and then it analyzes the graph condition and its paths to find the optimal shortest distance between the given node and all other nodes in the graph.

Dijkstra's algorithm keeps track of the currently known shortest distance from each node to the source node and updates the value after it finds the optimal path once the algorithm finds the shortest path between the source node and destination node then the specific node is marked as visited.

Basics requirements for Implementation of Dijkstra's Algorithm

Graph: Dijkstra's Algorithm can be implemented on any graph, but it works best with a weighted Directed Graph with non-negative edge weights and the graph should be represented as a set of vertices and edges.
Source Vertex: Dijkstra's Algorithm requires a source node which is starting point for the search.
Destination vertex: Dijkstra's algorithm may be modified to terminate the search once a specific destination vertex is reached.
Non-Negative Edges: Dijkstra's algorithm works only on graphs that have positive weights this is because during the process the weights of the edge must be added to find the shortest path. If there is a negative weight in the graph, then the algorithm will not work correctly. Once a node has been marked as visited the current path to that node is marked as the shortest path to reach that node.

Can Dijkstra's algorithm work on both Directed and Undirected graphs?

Yes, Dijkstra's algorithm can work on both directed graphs and undirected graphs as this algorithm is designed to work on any type of graph if it meets the requirements of having non-negative edge weights and being connected.

In a directed graph, each edge has a direction, indicating the direction of travel between the vertices connected by the edge. In this case, the algorithm follows the direction of the edges when searching for the shortest path.

In an undirected graph, the edges have no direction, and the algorithm can traverse both forward and backward along the edges when searching for the shortest path.

Algorithm for Dijkstra's Algorithm:

Mark the source node with a current distance of 0 and the rest with infinity.

Set the non-visited node with the smallest current distance as the current node.

For each neighbor, N of the current node adds the current distance of the adjacent node with the weight of the edge connecting 0 -> 1.

If it is smaller than the current distance of Node, set it as the new current distance of N.

Mark the current node 1 as visited.

Go to step 2 if there are any nodes are unvisited.

Pseudo Code for Dijkstra's Algorithm

function Dijkstra (Graph, source):
  // Initialize distances to all nodes as infinity, except for the source node.
  distances = map infinity to all nodes
  distances = 0
  // Initialize an empty set of visited nodes and a priority queue to keep track of the nodes to visit.
  visited = empty set
  queue = new PriorityQueue ()
  queue.enqueue(source, 0)

// Loop until all nodes have been visited.

while queue is not empty:

  // Dequeue the node with the smallest distance from the priority queue.

  current = queue.dequeue()

  // If the node has already been visited, skip it.

  if current in visited:

    continue

  // Mark the node as visited.

  visited.add(current)

  // Check all neighboring nodes to see if their distances need to be updated.

  for neighbor in Graph.neighbors(current):

    // Calculate the tentative distance to the neighbor through the current node.

    tentative_distance = distances[current] + Graph.distance(current, neighbor)

    // If the tentative distance is smaller than the current distance to the neighbor, update the distance.

    if tentative_distance < distances[neighbor]:

      distances[neighbor] = tentative_distance

      // Enqueue the neighbor with its new distance to be considered for visitation in the future.

      queue.enqueue(neighbor, distances[neighbor])

// Return the calculated distances from the source to all other nodes in the graph.

return distances

Ways to Implement Dijkstra's Algorithm:

There are several ways to Implement Dijkstra's algorithm, but the most common ones are:

- Using priority queue to keep track of all vertices.
- Using an array to keep track of Distances.
- Using a set to keep track of the visited vertices.

Complexity Analysis of Dijkstra's Algorithm using Priority Queue:

Time complexity: O (E log V)

Space Complexity: O (V^2), here V is the number of Vertices.

Complexity Analysis of Dijkstra's Algorithm using Prims Algorithm:

Time Complexity: O (V^2)

Auxiliary Space: O(V)

Complexity Analysis of Dijkstra's Algorithm with minimum edges

Time Complexity: O (V^2) where V is the number of vertices and E is the number of edges.
Auxiliary Space: O (V + E)

Complexity Analysis of Dijkstra's Algorithm:
The time complexity of Dijkstra's algorithm depends on the data structure used for the priority queue. Here is a breakdown of the time complexity based on different implementations:

Using an unsorted list as the priority queue: O(V2), where V is the number of vertices in the graph. In each iteration, the algorithm searches for the vertex with the smallest distance among all unvisited vertices, which takes O(V) time. This operation is performed V times, resulting in a time complexity of O(V^2).
Using a sorted list or a binary heap as the priority queue: O (E + V log V), where E is the number of edges in the graph. In each iteration, the algorithm extracts the vertex with the smallest distance from the priority queue, which takes O (log V) time. The distance updates for the neighboring vertices take O(E) time in total. This operation is performed V times, resulting in a time complexity of O (V log V + E log V). Since E can be at most V^2, the time complexity is O (E + V log V). (itsadityash)

Bellman Ford Algorithm
We have discussed Dijkstra's algorithm for this problem. Dijkstra's algorithm is a Greedy algorithm, and the time complexity is O((V+E) Log V) (with the use of the Fibonacci heap). Dijkstra doesn't work for Graphs with negative weights, Bellman-Ford works for such graphs. Bellman-Ford is also simpler than Dijkstra and suites well for distributed systems. But time complexity of Bellman-Ford is O (V * E), which is more than Dijkstra.

How does this work?
Like other Dynamic Programming Problems, the algorithm calculates the shortest paths in a bottom-up manner. It first calculates the shortest distances which have at most one edge in the path. Then, it calculates the shortest paths with at-most 2 edges, and so on. After the i-th iteration of the outer loop, the shortest paths with at most i edges are calculated. There can be maximum |V| – 1 edges in any simple path, that is why the outer loop runs |v| – 1 times. The idea is, assuming that there is no negative weight cycle if we have calculated shortest paths with at most i edges, then an iteration over all edges guarantees to give the shortest path with at-most (i+1) edges. (Geeksforgeeks)

Floyd-Warshall Algorithm
Floyd-Warshall Algorithm is an algorithm for finding the shortest path between all the pairs of vertices in a weighted graph. This algorithm works for both the directed and undirected weighted graphs. But it does not work for the graphs with negative cycles (where the sum of the edges in a cycle is negative).

Floyd Warshall Algorithm Complexity

Time Complexity

There are three loops. Each loop has constant complexities. So, the time complexity of the Floyd-Warshall algorithm is O (n^3).

Space Complexity

The space complexity of the Floyd-Warshall algorithm is O (n^2).

Floyd Warshall Algorithm Applications

To find the shortest path is a directed graph.

To find the transitive closure of directed graphs

To find the Inversion of real matrices

For testing whether an undirected graph is bipartite (Programiz)

Algorithm A *

known as "A-Star" is an optimized variation of the Dijkstra algorithm. Is an improvement developed to the postulates of the Dijkstra algorithm that is responsible for finding shortest paths within a graph.

This modification focuses on the observation of informed searches within the graph that allow us to make optimal decisions about the paths to be taken to efficiently traverse the graph.

For the application of this algorithm, we must understand how to proceed to divide the cost of the route. In this case it is divided into two parts where g(n) represents the cost of the route from its origin to some node n within the graph. We also have that h(n) represents the estimated cost of the route from node n to the destination node, calculated by a smart assumption.

It balances g (n) and h (n) while iterating the graph, thus ensuring that at each iteration it chooses the node with the lowest total cost f (n) = g (n) + h (n).

Use cases of the A * algorithm

The A * algorithm, being specially designed to detect the least expensive paths within a complex graph, is normally used to find short paths between individual pairs of locations.

This is why one of its most common applications is to detect geo-locations where satellite location coordinates are known. (GraphEverywhere)
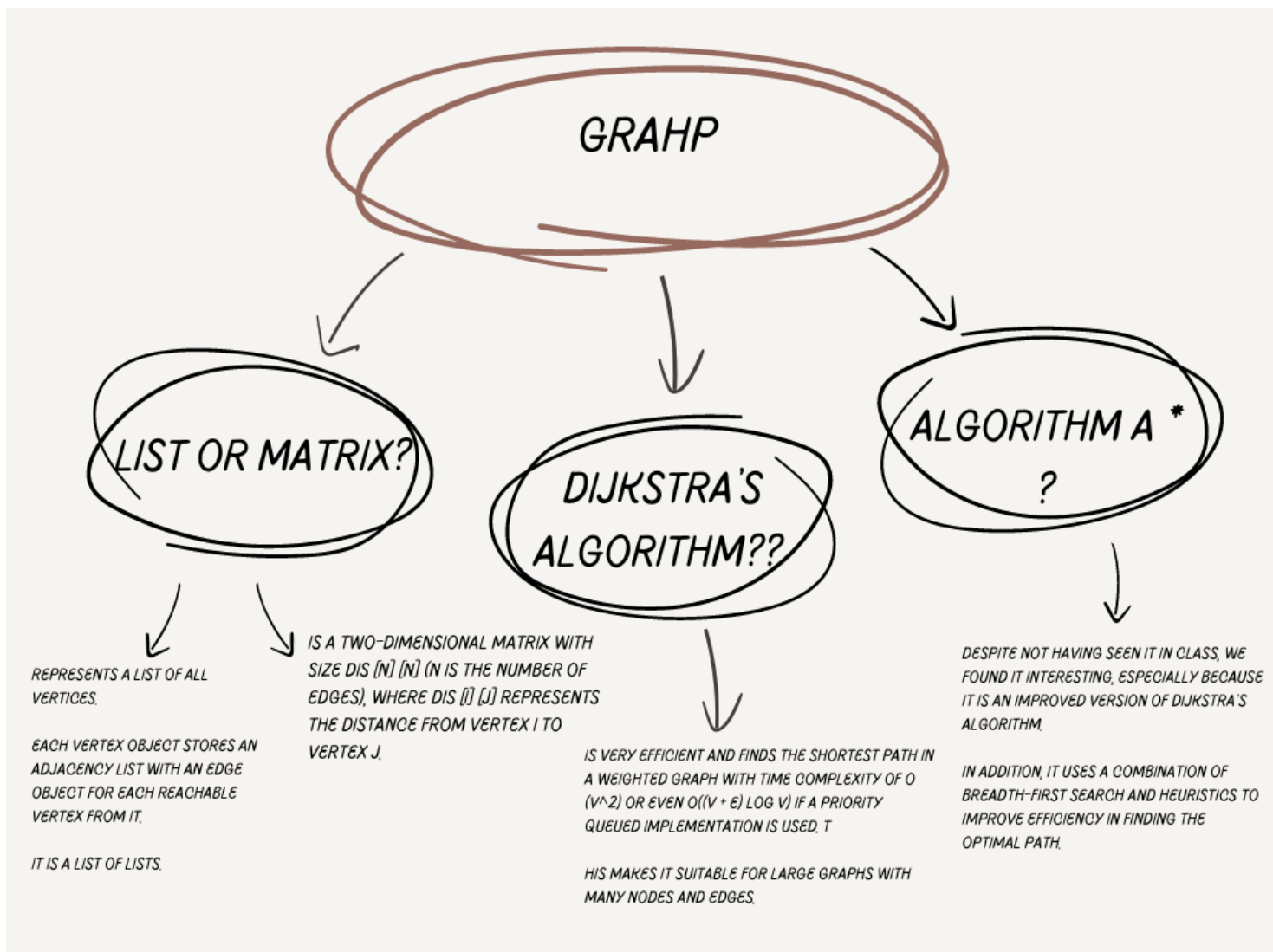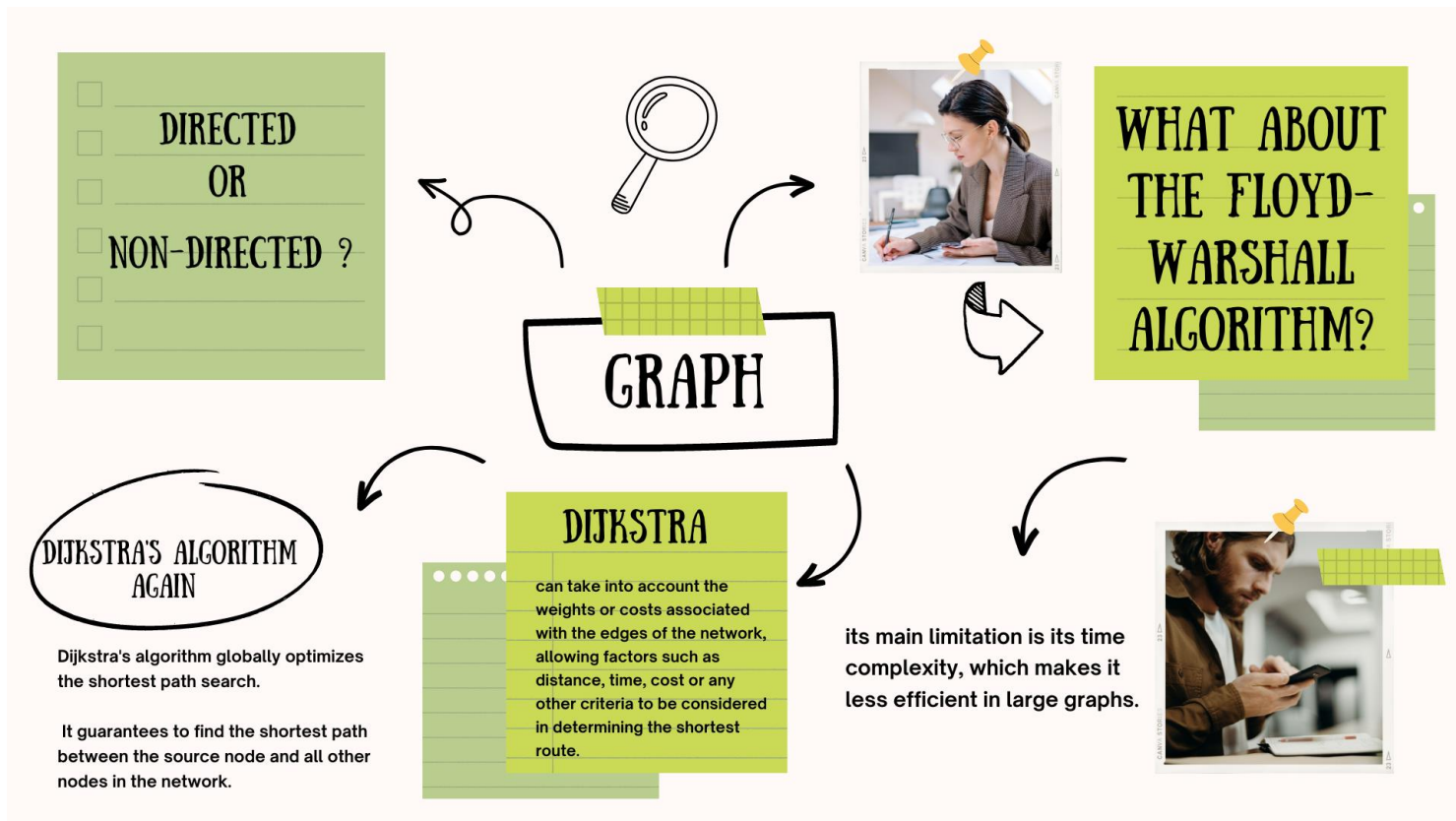
Phase 3

SEARCH FOR CREATIVE SOLUTIONS

Idea generation

Considering that the initial problem was chosen by the working group, this also enters into our search for creative solutions, before selecting this problem there were two more, one was about sudoku, create a Bot that solves simple sudokus, however, although in theory it sounded pretty good, the three members of the group are not in the same capacity so someone would have to be overloaded to carry out this idea, in addition to this, we must take into account the time, which was not going to be enough. The second idea was about currencies, compare them and choose between them, for this one, it happens almost the same as with the previous one, in addition to that we would have to investigate a little more in depth about this topic. In the end, we opted for an idea that we were all familiar with at some point. At some point we got on a MIO for the first time, and it was a bit confusing, in terms of which route to choose to go to a certain place.

Focusing now on the possible solutions to our problem:

GRAHP

LIST OR MATRIX?

REPRESENTS A LIST OF ALL VERTICES.

EACH VERTEX OBJECT STORES AN ADJACENCY LIST WITH AN EDGE OBJECT FOR EACH REACHABLE VERTEX FROM IT.

IT IS A LIST OF LISTS.

IS A TWO-DIMENSIONAL MATRIX WITH SIZE DIS [N] [N] (N IS THE NUMBER OF EDGES), WHERE DIS [i] [J] REPRESENTS THE DISTANCE FROM VERTEX I TO VERTEX J.

DIJKSTRA'S ALGORITHM??

IS VERY EFFICIENT AND FINDS THE SHORTEST PATH IN A WEIGHTED GRAPH WITH TIME COMPLEXITY OF O (V^2) OR EVEN O((V + E) LOG V) IF A PRIORITY QUEUED IMPLEMENTATION IS USED. T

HIS MAKES IT SUITABLE FOR LARGE GRAPHS WITH MANY NODES AND EDGES.

ALGORITHM A * ?

DESPITE NOT HAVING SEEN IT IN CLASS, WE FOUND IT INTERESTING, ESPECIALLY BECAUSE IT IS AN IMPROVED VERSION OF DIJKSTRA'S ALGORITHM.

IN ADDITION, IT USES A COMBINATION OF BREADTH-FIRST SEARCH AND HEURISTICS TO IMPROVE EFFICIENCY IN FINDING THE OPTIMAL PATH.

Phase 4

IDEAS AND PRELIMINARY DESIGNS:

How will the station information be stored?

The information of the stations will be loaded in a Json file.

- **To represent the graph, both the matrix and the adjacency list will be implemented.**

Undirected graph	Directed graph
In an undirected graph, the edges do not have a specific direction, which implies that one can travel in both directions between bus stations. That is, there is both a relationship from A to B and from B to A.	In this case the edges have a specific direction, which will indicate where to go, i.e., if the direction goes from A → B, the direction will be that and there will be no direction from B → A.
There are efficient algorithms for finding the shortest path in undirected graphs, such as the modified Dijkstra algorithm or the breadth-first search (BFS) algorithm. These algorithms can be adapted to find the fastest route in an undirected graph, although	You can take advantage of efficient algorithms to find the shortest path, such as Dijkstra's algorithm or the A* (A-Star) algorithm. These algorithms are specifically designed for directed graphs and can compute the fastest route more efficiently.

they may require additional modifications to account for directionality or sense constraints if relevant.

In terms of computational complexity, algorithms for undirected graphs can be more efficient than algorithms for directed graphs in certain cases. For example, the modified Dijkstra's algorithm for undirected graphs has a complexity of O $((V + E) \log V)$, while Dijkstra's algorithm for directed graphs has a complexity of O $((V + E) \log V + V^2)$ in the worst case.

BFS

```cpp
void BFS(grafo &g,int s){
    int v;
    queue<int> cola;
    list<int>::iterator it;

    nodoVisitado[s]=true;
    cola.push(s);
    while(!cola.empty()){
        v=cola.front(),cola.pop();
        nodoProcesado[v]=true;
        procesarNodo(v);
        for(it=g.lados[v].begin(); it!=g.lados[v].end(); ++it){
            if(!nodoVisitado[*it]){
                cola.push(*it),nodoVisitado[*it]=true;
                nodoPadre[*it]=v;
            }
            if(!nodoProcesado[*it]) {
                procesarLado(v,*it);
            }
        }
    }
}
```

- Find the shortest route in terms of number of edges, but not necessarily in terms of distance or travel time. In public transport, it is more important to find the fastest route based on travel time, considering factors such as traffic and vehicle speeds.

- It assumes **that all edges have the same weight or cost**, which means that it does not consider differences in travel times between connections. However, in public transport, routes may have different travel times due to traffic congestion, additional stops, or even different types of vehicles.
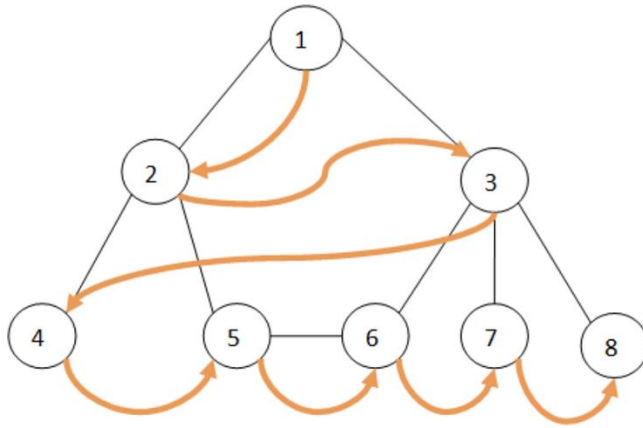
This helps us to understand that for this problem the BFS is not so indicated, not if we want to comply with what we have mentioned above.

DIJKSTRA'S ALGORITHM

Throughout the document, we have mentioned this algorithm and its advantages to solve this type of problems, therefore, it can be deduced that it will be included in our implementation.

The following is a more graphic illustration of how this algorithm works.
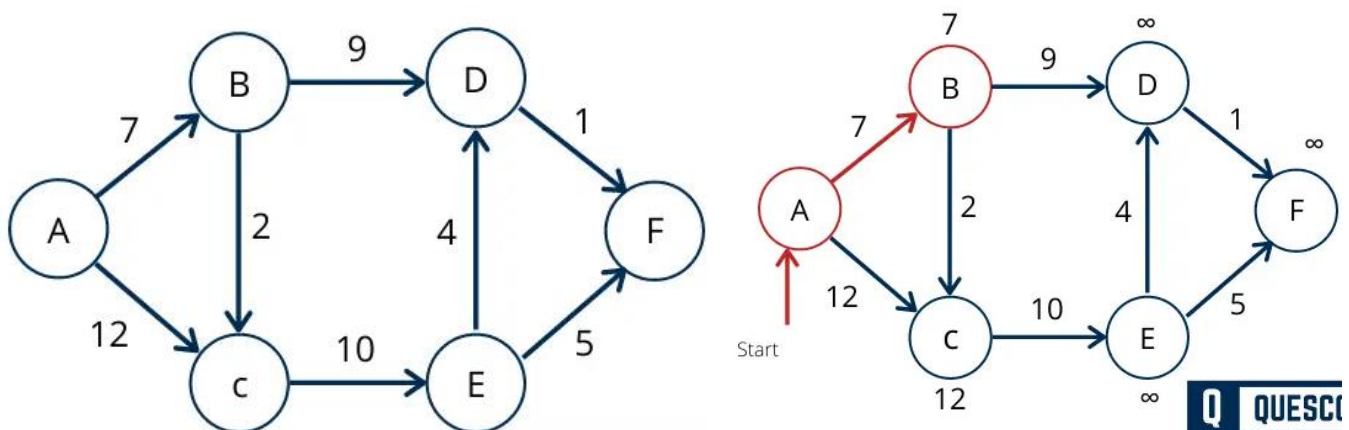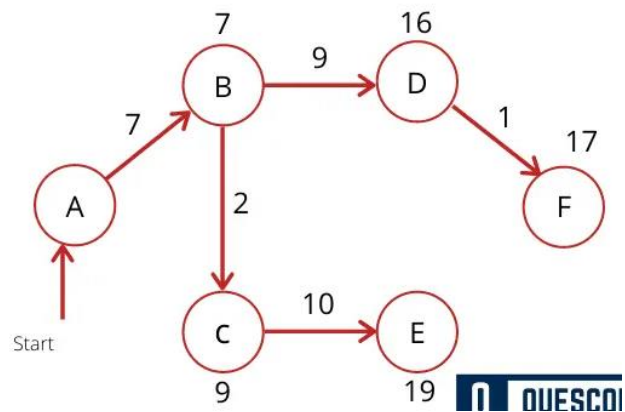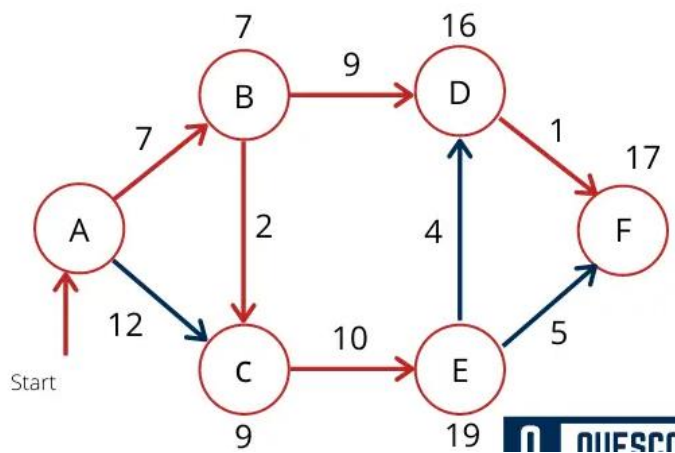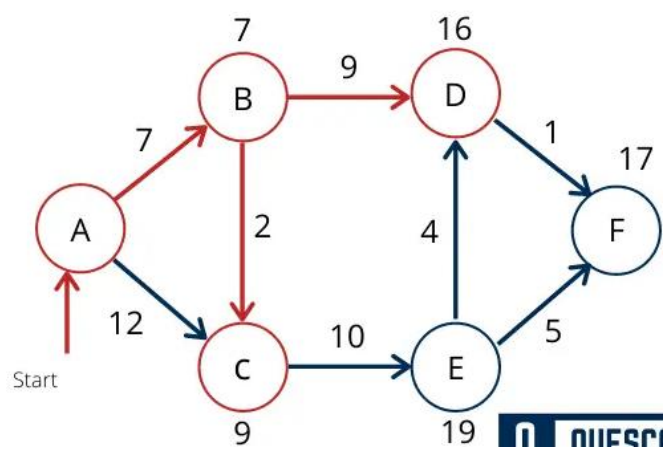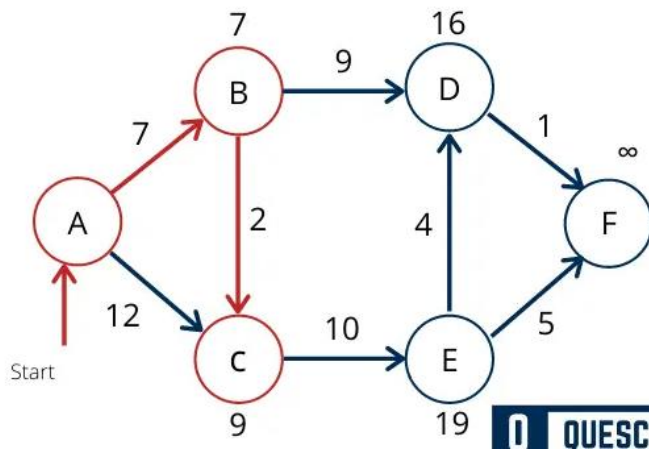
Considering that we must make use of at least two such algorithms, we will evaluate the **Floyd-Warshall algorithm.**

```
FLOYD - WARSHALL (W)
1. n ← rows [W].
2. D⁰ ← W
3. for k ← 1 to n
4. do for i ← 1 to n
5. do for j ← 1 to n
6. do dᵢⱼ⁽ᵏ⁾ ← min (dᵢⱼ⁽ᵏ⁻¹⁾,dᵢₖ⁽ᵏ⁻¹⁾+dₖⱼ⁽ᵏ⁻¹⁾ )
7. return D⁽ⁿ⁾
```

The **Floyd-Warshall** algorithm allows finding the shortest paths between all pairs of nodes in a graph, which means that it can determine the fastest route between any pair of bus stations. This is useful if you need to calculate the optimal route for multiple destinations or if you want to have a complete knowledge of all possible

routes. The latter is very useful for our problem, considering the different routes that are handled within the city.

If the graph representing the connections between bus stations has weights on the edges representing travel times or distances, the Floyd-Warshall algorithm can take these weights into account and find the fastest route taking travel times into account. You do not need to modify the algorithm to account for the weights, unlike other algorithms such as Dijkstra or A*.

Although the Floyd-Warshall algorithm has a complexity of O (V^3), where V is the number of nodes, it can be a good choice if the graph size is not too large. If the number of bus stations is moderate, the Floyd-Warshall algorithm can be efficient and provide accurate results for finding the fastest route between all stations.

So, this algorithm **can be used to find the fastest route between bus stations**, especially if the optimal route needs to be calculated for multiple destinations or if the graph has edge weights representing travel times. However, it must be evaluated if it is suitable for the size of the graph in question, i.e., not having many stations.

PRIM'S ALGORITHM
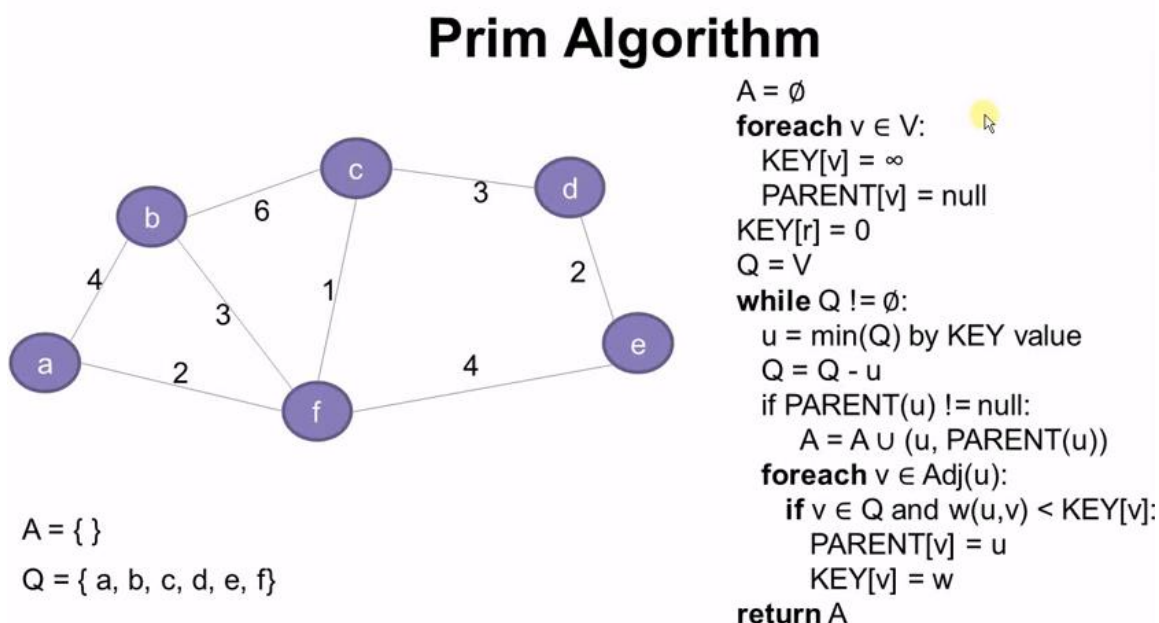
Prim's algorithm is used to find the minimum spanning tree in a weighted undirected graph, and its objective is to connect all the nodes of the graph in an optimal way by minimizing the total weight of the edges.

Prim's algorithm applies to undirected graphs since the process of constructing the minimum spanning tree does not require a sense of edges. Therefore, if the graph representing the bus stations and the connections between them is undirected, Prim's algorithm could be applicable.

If the graph has weights on the edges representing the distances between bus stations, Prim's algorithm can consider these weights and build a minimum spanning tree based on distance. However, it should be noted that this will not necessarily find the fastest route in terms of travel time, as it does not consider factors such as traffic or vehicle speeds.

The prim algorithm **can be modified** to consider the weights of the edges that will represent the distances, even so, this does not guarantee that the fastest route will be found in terms of travel time.

Phase 5

SOLUTION SELECTION:

With some alternatives previously discarded, and others that although not completely discarded, are not the most suitable or are at an intermediate point, where with some modifications / adaptations can be applicable to the solution of our problem.

Then, our final solution will be a system that allows to consult the best route from a starting point to a destination point, guiding the users of this massive system and providing them with a better experience.

To achieve this system, we defined a graph that will work with both adjacency lists and adjacency matrix, it was also decided that it would be an undirected graph, previously we evaluated which option would be better, but in this case, it is not that one is better than another, but it depends on the application you want to give, in our case, we decided to do it this way because it will be able to go from one station to another. Thus, avoiding the need to establish two relationships and simply with one, the two possibilities are already within our reach.

We also decided to consider Dijkstra's algorithm, which according to everything evaluated, seems to be the most suitable for this type of cases, although the Floyd-Warshall algorithm can also be useful, since, after studying it, it was determined that it is applicable to achieve the solution of the problem. Another algorithm that could be very useful is the A* algorithm, however this was not considered for the solution. Why? because we did not know this algorithm and its applications, however, despite not being used in our context, it is one of the most suitable, even more than the prim algorithm, which was another algorithm that was implemented to solve the problem, however, this algorithm must be modified in our context, if we want it to help with the solution to the problem, therefore, the A* algorithm would have been a better option.