Juan José Restrepo Bonilla.
Josué Rodríguez.
Manuel Castaño.

Statement:

The integrated mass transportation system for trains (SITMT) from the Colombian city of Santiago de Cali, It has been bankrupt for the last two years due to the exponential drop in users who have decided to switch to other platforms of better quality and shorter delay time given the routes they establish, then the municipal mayor's office worried about the current situation of the system, has desperately sought telematics and systems engineers to solve their problems. Surprisingly, they received a call from a professor at the Icesi University where they offered the best engineering group that studied algorithms and data structures to provide solutions to their problems. So the mayor of Cali, communicates with you to ask you to solve your problems, through software, where they need to find the best route given a specific station, know which is the last station to be able to tell your users how far you can go and also search for a specific station to be able to show you their information that you have so far. So your group has decided to set out on this great problem, where it all depends if the mayor goes bankrupt or seeks the light of hope.

Engineering method:

1. **Problem specification:**

   A train transport system needs to implement a transport software to find the best transport route.

2. **Collection of information:**

   A graph is the symbolic representation of the elements made up of a system or set, using graphic diagrams. we can also say, that a graph consists of a set of nodes (also called vertex) and a set of paths (edges) that establish relationships between nodes.

   Edges: They are the lines with which the edges of a graph are joined and with which roads are also built. If the edge lacks direction it is denoted interchangeably {a, b} or {b, a}, where a and b are the vertices it joins.

   Vertex: They are the points or nodes with which a graph is formed. We will call degree of a vertex the number of edges of which it is extreme. A vertex is said to be “pair” or “odd” depending on your grade.

   Roads: Path is called a sequence of vertices of a graph such that there is an edge, each vertex and the next.

   Graph properties:

Adjacency:
Two edges are adjacent if they have a vertex in common and two vertices are adjacent if an edge joins them.

Incidence:
An edge is incident to a vertex if it joins another.

Weighing:
Corresponds to a function that associates a value with each edge to increase the expressiveness of the model.

3. Functional requirements:

Name	1. Find minimum path by edges.
Summary	The program needs to find the minimum path to go since one station to another one according to the number of differents paths to take.
Inputs	Two stations.
Outputs	the minimum path.

Name	2. Find minimum path by distance.
Summary	The program needs to find the minimum path to go since one station to another one according to the distance.
Inputs	Two stations.
Outputs	the minimum path.

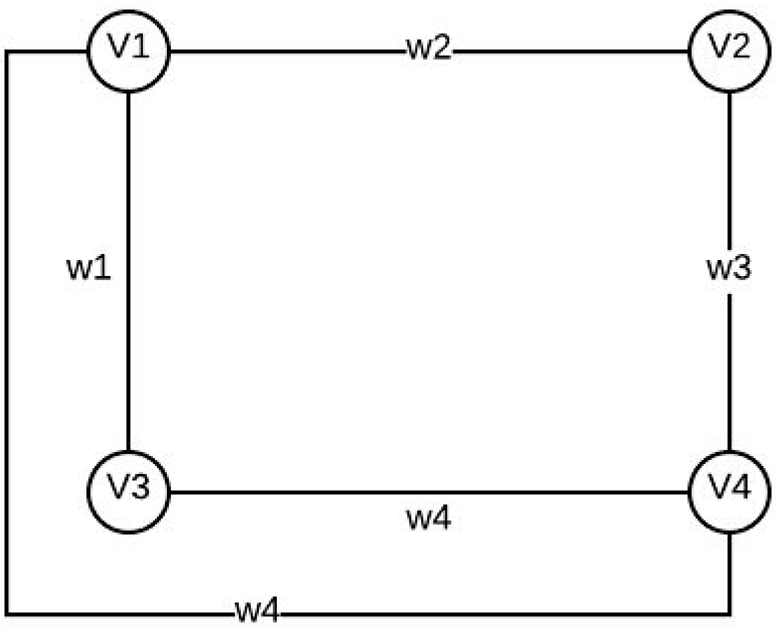
4. Creative solutions:

- Implement a simple graph where the vertex is the train station and the edges connects two stations and his weight is the time to travel between them.
- Implement a directed graph where a edge can connect two stations in only one direction.
- Implement a direct multigraph where two train stations can have more than one edge connecting them in one direction.

5. Selection of the solutions:

We are going to choose the simple graph to model the problem, because according to the Cali's major the path between two stations are made to function in both directions, so a direct graph is less efficient because we should connect two stations with two edges in both directions and there is only one path between two stations.

TAD

Graph

inv ● any two vertex only can have one edge connecting then as maximum ● the weight of each edge must be greater than zero
● AddVertex: Vertex → Boolean ● AddEdge: Vertex X Vertex X weight → Boolean ● WeightedMatrixGraph: → Graph ● SearchVertex: String→ Vertex ● DepthFirstSearch: Graph X Vertex → List ● BreadthFirstSearch: Graph X Vertex → List ● Djikstra: Graph X Vertex → List ● FloydWarshall: Graph → List ● RemoveVertex Vertex → Boolean