

Método de la ingeniería- Proyecto Final

1. Identificación del Problema

- a. La empresa requiere conocer la ruta más económica para la entrega de sus pedidos.
- b. El programa debe permitir agregar o eliminar ubicaciones y calcular la ruta más corta y económica para realizar las entregas.
- c. El programa debe mostrar cuales lugares ya fueron visitados (o realizado la entrega) para no volver a visitarlo o realizar la entrega.
- d. El programa debe informar sobre el consumo de gasolina realizado en la ruta de entrega.
- e. El programa debe tener información sobre la gasolina para poder realizar cálculos sobre los gastos de las entregas.

Definición del problema

Una empresa de entregas anónima de la ciudad de Cali requiere un programa economizar el consumo de combustible de sus camiones, mediante una funcionalidad que permita hallar la ruta rápida y económica entregar los pedidos. Además de eso, se requiere también que se pueda revisar desde la oficina central que pedidos se han entregado viendo en que parte de la ruta va el camión. Lo anterior con el fin de tomar datos para analizar la eficiencia de los conductores y sus ayudantes, y de esa manera tomar decisiones futuras sobre la cantidad de entregas que se deberían realizar diariamente. La empresa ha sido muy clara y específica; quiere que el programa sea modelado por uno o varios grafos, los que se necesiten. El programa debe permitir la visualización de los camiones y sus rutas, y por consiguiente el mapa de la ciudad de Cali. Adicionalmente, se debe tener información sobre la gasolina (precio y galones usados por ruta) para calcular cuánto dinero se está invirtiendo, aproximadamente, en combustible por ruta.

Requerimientos Funcionales:

Nombre	R1-Mostrar la ruta más corta para realizar las entregas
Resumen	Muestra la manera de entregar todas las entregas rápido y económicamente
Entrada	
Resultado	
Se muestra la manera más corta y económica de realizar las entregas	

Nombre	R2-Permitir agregar más ubicaciones
Resumen	Agrega nuevas ubicaciones para realizar entregas
Entrada	
Nueva ubicación	
Resultado	
Se agrega la nueva ubicación	

2. Recopilación de la información

¿Qué es un grafo?

Un grafo es un conjunto de objetos llamados nodos o vértices, que pueden estar unidos (conectados) por líneas llamada aristas.

¿Qué es un vértice?

un vértice o nodo es la unidad fundamental de la que están formados los grafos. Un grafo no dirigido está formado por un conjunto de vértices y un conjunto de aristas (pares no ordenados de vértices), mientras que un grafo dirigido está compuesto por un conjunto de vértices y un conjunto de arcos.

¿Qué es una arista?

una arista corresponde a una relación entre dos vértices de un grafo.

Para caracterizar un grafo G son suficientes únicamente el conjunto de todas sus aristas, comúnmente denotado con la letra E junto con el conjunto de sus vértices, denotado por V . Así, dicho grafo se puede representar como $G(V,E)$, o bien $G = (V,E)$.

¿Qué es BFS?

es un algoritmo de búsqueda no informada utilizado para recorrer o buscar elementos en un grafo. Intuitivamente, se comienza en la raíz (eligiendo algún nodo como elemento raíz en el caso de un grafo) y se exploran todos los vecinos de este nodo

¿Qué es DFS?

es un algoritmo de búsqueda no informada utilizado para recorrer todos los nodos de un grafo o árbol (teoría de grafos) de manera ordenada, pero no uniforme. Su funcionamiento consiste en ir expandiendo todos y cada uno de los nodos que va localizando, de forma recurrente, en un camino concreto

¿Qué es Dijkstra?

es un algoritmo para la determinación del camino más corto, dado un vértice origen, hacia el resto de los vértices en un grafo que tiene pesos en cada arista. Su nombre alude a Edsger Dijkstra, científico de la computación de los Países Bajos que lo describió por primera vez en 1959.

¿Qué es Floyd-Warshall?

es un algoritmo de análisis sobre grafos para encontrar el camino mínimo en grafos dirigidos ponderados. El algoritmo encuentra el camino entre todos los pares de vértices en una única ejecución.

¿Qué es Prim?

es un algoritmo perteneciente a la teoría de los grafos para encontrar un árbol recubridor mínimo en un grafo conexo, no dirigido y cuyas aristas están etiquetadas.

¿Qué es Kruskal?

es un algoritmo de árbol de expansión mínima que encuentra un borde del menor peso posible que conecta dos árboles del bosque.

3. Búsqueda soluciones creativas

Para buscar las rutas se pueden usar varios algoritmos o estrategias, pero es importante aclarar que hay un fragmento del enunciado que se especifica que se

debe plantear el problema como un grafo, así que para resolver el problema se usara un algoritmo de la teoría de grafos.

(Lluvia de ideas)

Alternativa 1: Algoritmo Dijkstra

Esta solución consiste en implementar el este algoritmo para retornar la lista de lugares que tiene que visitar desde el principio(La empresa) hasta el último lugar de entrega.

Alternativa 2: Algoritmo Floyd-Warshall

Esta solución consiste en hallar el valor mínimo entre el principio y final de la ruta de entregas(Grafo) y saber cuánta gasolina gasta y el costo de las entregas.

Alternativa 3: Algoritmo Prim

Esta solución consiste en ejecutar el algoritmo y mostrar un árbol con el recorrido a realizar por el trabajador para realizar todas las entregas en un camino corto y económico.

4. Transición de las ideas a los diseños preliminares.

Alternativa 1:

- Da una lista de la manera de llegar de un lugar a otro.
- Puede dar una lista sin que realice unas entregas ya que no tiene que pasar por todos los lugares de entrega.

Alternativa 2:

- Da la distancia para entregar todas las entregas.
- Se puede calcular el costo con la distancia mínima.
- Lo malo es que solo da la distancia y no el recorrido que hacer.

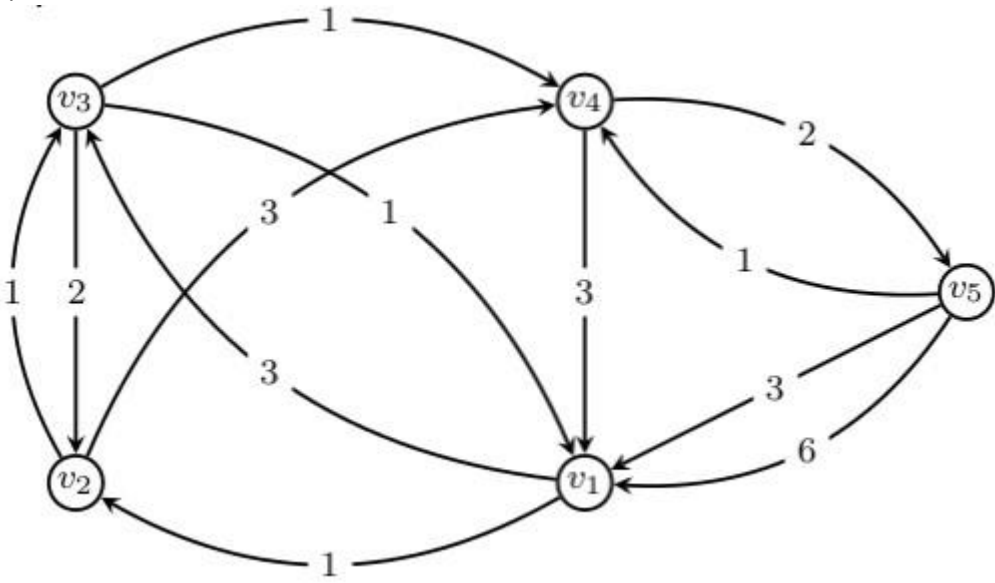
Alternativa 3:

- Muestra un árbol de camino más corto para realizar todas las entregas

5. Evaluación y selección de la mejor solución

Después de investigar y entender las posibles soluciones se llega a la conclusión que se utiliza la alternativa 3 ya que es la que mejor se relaciona con el problema y los requerimientos.

6. Preparación de informes y especificaciones

TAD Graph		
Representation:		
		
Invariant: No direccionado		
Operations:		
Grafo	→	Graph
AddVert	Vertex	
AddArist	Vertex, Vertex, Integer	
getVert	Vertex	getVerts
DeleVert	Vertex	
DeleArist	Arist	
BFS	Vertex	
DFS	Vertex	

AddVert(Vertex)
Adds a Vertex to the graph. Pre: void Post: $V = V + 1$

AddArit(Vertex,Vertex,Integer)
Adds a weighted edge from one vertex to another. Pre: void Post: $\text{Graph.weight} = \text{Graph.weight} + e.\text{weight}$

getVert (Vertex)
Gives the information of a Vertex. Pre: $\text{Vertex} \in \text{Graph} \wedge \text{Vertex} \neq \text{nil}$ Post: Vertex

getVerts ()
Gives the list of the vertexes of the graph. Pre: $\text{Vertex}(i) \in \text{Graph} \wedge \text{Vertex}(i) \neq \text{nil}$ Post: $\text{Vertex} \{V1,V2...\text{Vn}\}$

DeleVert(Vertex)
Deletes a Vertex of the graph. Pre: $\text{Vertex} \in \text{Graph} \wedge \text{Vertex} \neq \text{nil}$ Post: $V = V - 1$

DeleArist (Edge)
Deletes an edge of the graph. Pre: Edge \in Graph \wedge Edge \neq nil Post: Graph.weight = Graph.weight - e.weight

BFS(Vertex)
Explores all the neighbor nodes of a vertex selected as root node. Pre: Vertex \in Graph \wedge Vertex \neq nil Post: void

DFS(Vertex)
Explores the possible routes as far as possible of a vertex selected as root node. Pre: Vertex \in Graph \wedge Vertex \neq nil Post: void

-Diseño de pruebas unitarias

P.1	Objetivo de la prueba= comprobar si se agrega un vértice			
Clase	Método	Escenario	Valores de entrada	Resultado
Graph	AddVert(Vertex)	Escenario 1		

P.2	Objetivo de la prueba= comprobar si se agrega un arista			
-----	---	--	--	--

Clase	Método	Escenario	Valores de entrada	Resultado
Graph	AddArit(Vertex,Vertex,Integer)	Escenario 1		

P.3	Objetivo de la prueba= comprobar si retorna el vértice			
Clase	Método	Escenario	Valores de entrada	Resultado
Graph	getVert (Vertex)	Escenario 1		

P.4	Objetivo de la prueba= comprobar si retorna los vértices			
Clase	Método	Escenario	Valores de entrada	Resultado
Graph	getVerts ()	Escenario 1		

P.5	Objetivo de la prueba= comprobar si elimina un vértice			
Clase	Método	Escenario	Valores de entrada	Resultado
Graph	DeleVert(Vertex)	Escenario 1		

P.6	Objetivo de la prueba= comprobar si elimina una arista			
Clase	Método	Escenario	Valores de entrada	Resultado
Graph	DeleArist (Edge)	Escenario 1		

7. Implementación del diseño

Enlace al repositorio de GitHub:

<https://github.com/Sergiiok/FinalProjectAED>