MÉTODO DE DISEÑO DE LA INGENIERÍA

COMPUTACIÓN Y ESTRUCTURAS DISCRETAS I

TAREA INTEGRADORA III

By:

Juan David Bahamon Rodriguez

A00375826

Samuel Hernández Espitia

A00375392

David Esteban Peñaranda Scarpetta

A00375827

FASE 1: IDENTIFICACIÓN DEL PROBLEMA:

El gobierno y los alcaldes han designado a los equipos de la materia de Estructuras y Computación Discretas para que desarrollen un prototipo que permita tener una solución a los colombianos frente a la problemática de los medios masivos de transporte público.

El MIO de Cali vive una crisis que golpea en igual o mayor medida a los sistemas masivos de transporte público del país. Atraviesan fuertes dificultades por la falta de planeación de las rutas más eficientes. Los ciudadanos que hacen uso del transporte público el MIO, se ven obligados a esperar un largo tiempo para dirigirse a su destino, pues los colectivos y las puertas de las paradas se encuentran llenas de personas.

Además, lo que en promedio podría tardar 35 minutos, el sistema masivo de transporte lo hace en 65 minutos, realizando un cálculo, si lo sumamos con dos recorridos mínimos que deben hacer las personas en el día, se está hablando de una hora más en un bus o en un paradero esperando a que llegue la ruta correspondiente.

Un punto importante de esto es que los ciudadanos al observar este fracaso prefieren la compra de autos y motos, haciendo un crecimiento de la malla vial, incrementando la problemática puesto que actualmente los trancones en horas pico pueden hacer que un recorrido de 5 a 10 minutos se transforme en un viaje de 40 a 50 minutos.

Así que, se requiere crear una solución que permita recorrer todas las estaciones del mío en el menor tiempo posible, y brindarle al usuario cual es la ruta más rápida para llegar a su destino.

Condiciones:

- Determinar el camino más corto de una estación a otra.
- Recorrer todas las estaciones por el camino más corto entre una estación y otra.

FASE 2: RECOPILACIÓN DE INFORMACIÓN NECESARIA.

El Masivo Integrado de Occidente (MIO) es el sistema integrado de transporte masivo (SITM) de la ciudad colombiana de Cali. El sistema es operado por buses articulados, padrones y complementarios, los cuales se desplazan por medio de corredores troncales, pretroncales y complementarios cubriendo rutas troncales, pretroncales y alimentadoras.

Estación: es una infraestructura de transporte de pasajeros en la que se hace transferencia de un sistema a otro directamente.

El algoritmo de 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. Es decir, encuentra un subconjunto de aristas que forman un árbol con todos los vértices,

donde el peso total de todas las aristas en el árbol es el mínimo posible. Si el grafo no es conexo, entonces el algoritmo encontrará el árbol recubridor mínimo para uno de los componentes conexos que forman dicho grafo no conexo. Recuperado de: https://es.wikipedia.org/wiki/Algoritmo_de_Prim

El algoritmo de 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. Consiste en ir explorando todos los caminos más cortos que parten del vértice origen y que llevan a todos los demás vértices; cuando se obtiene el camino más corto desde el vértice origen hasta el resto de los vértices que componen el grafo, el algoritmo se detiene. Recuperado de: https://es.wikipedia.org/wiki/Algoritmo_de_Dijkstra

FASE 3: BÚSQUEDA DE SOLUCIONES CREATIVAS:

Las alternativas que se exponen a continuación están pensadas para buscar solución a la problemática de encontrar la ruta más rápida, la implementación de grafos y si se brinda una interfaz gráfica, los cuales son unas funciones muy importantes para el prototipo a desarrollar. Las alternativas son:

Alternativa 1:

Se desarrolla la solución al problema con la implementación de dos versiones de grafos, el cual al iniciar el programa se le brinda al usuario un menú por medio de la consola, pidiendo que digite un número correspondiente a la implementación que desea ejecutar.

Alternativa 2:

Se desarrolla la solución al problema con la implementación de dos versiones de grafos, el cual al iniciar el programa se le brinda al usuario un menú por medio de una interfaz, pidiendo que seleccione el botón correspondiente a la implementación que desea ejecutar.

Alternativa 3:

Se desarrolla la solución al problema con la implementación de los algoritmos de Dijkstra y Prim, los cuales deben de ser utilizados con un mínimo de 50 vértices y 50 aristas, y se le brinda al usuario interactuar con el sistema por medio de una interfaz gráfica.

Alternativa 4:

Se desarrolla la solución al problema con la implementación de los algoritmos de Dijkstra y Prim, los cuales deben de ser utilizados con un mínimo de 50 vértices y 50 aristas, y se le brinda al usuario interactuar con el sistema por medio de una consola.

FASE 4: TRANSICIÓN DE LA FORMULACIÓN DE IDEAS A LOS DISEÑOS PRELIMINARES.

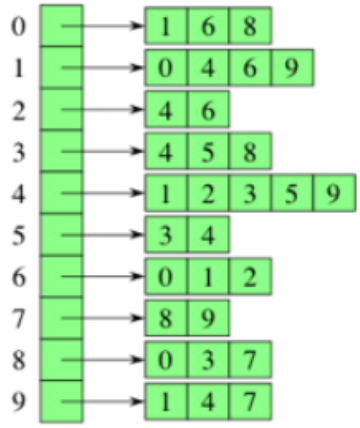
Con el fin de tener un mejor desarrollo del prototipo, hemos decidido complementar cada una de las alternativas propuestas en la fase anterior.

Alternativa 1:

- Para un grafo con V vértices, una **matriz de adyacencia** es una matriz de V × V de ceros y unos, donde la entrada en el renglón *i* y la columna *j* es 1 si y solo si la arista (i, j) está en el grafo. Si se quiere indicar un peso de la arista, se pone en la entrada del renglón *i*, columna *j* y reserva un valor especial (tal vez null) para indicar una arista ausente. Aquí está la matriz de adyacencia para el grafo de la red social:

	0	1	2	3	4	5	6	7	8	9
0	0	1	0	0	0	0	1	0	1	0
1	1	0	0	0	1	0	1	0	0	1
2	0	0	0	0	1	0	1	0	0	0
3	0	0	0	0	1	1	0	0	1	0
4	0	1	1	1	0	1	0	0	0	1
5	0	0	0	1	1	0	0	0	0	0
6	1	1	1	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	1	1
8	1	0	0	1	0	0	0	1	0	0
9	0	1	0	0	1	0	0	1	0	0

- Representar un grafo con **listas de adyacencia** combina las matrices de adyacencia con las listas de aristas. Para cada vértice *i*, almacena un arreglo de los vértices adyacentes a él. Típicamente tenemos un arreglo de V listas de adyacencia, una lista de adyacencia por vértice. Aquí está una representación de una lista de adyacencia del grafo de la red social:



- El programa brinda dos opciones por medio de la consola, el cual le permite al usuario elegir cual es la implementación que desea ejecutar.

Alternativa 2:

- El programa tendrá las dos implementaciones de los grafos de la alternativa 1, con la única diferencia que se ofrece una interfaz de usuario la cual contiene dos botones, cada uno correspondiente a la solución por medio de una matriz de adyacencia o por medio de listas de adyacencia. Al seleccionar uno de ellos, el programa dará una solución al problema haciendo uso de la implementación seleccionada, sin ningún tipo de problema.

Alternativa 3:
Algoritmo de Dijkstra:

- Sea V un conjunto de vértices de un grafo.
- Sea C una matriz de costos de las aristas del grafo, donde en C[u,v] se almacena el costo de la arista entre u y v.
- Sea S un conjunto que contendrá los vértices para los cuales ya se tiene determinado el camino mínimo.
- Sea D un arreglo unidimensional tal que D[v] es el costo del camino mínimo del vértice origen al vértice v.
- Sea P un arreglo unidimensional tal que P[v] es el vértice predecesor de v en el camino mínimo que se tiene construido.
- Sea V inicial el vértice origen. Hay que recordar que el Algoritmo Dijkstra determina los caminos mínimos que existen partiendo de un vértice origen al resto de los vértices.

Algoritmo de Prim:

- Se marca un vértice cualquiera. Será el vértice de partida.
- Se selecciona la arista de menor peso incidente en el vértice seleccionado anteriormente y se selecciona el otro vértice en el que incide dicha arista.
- Repetir el paso 2 siempre que la arista elegida enlace un vértice seleccionado y otro que no lo esté. Es decir, siempre que la arista elegida no cree ningún ciclo.
- El árbol de expansión mínima será encontrado cuando hayan sido seleccionados todos los vértices del grafo.

- El programa hace uso del algoritmo correspondiente para dar una solución al programa, es decir, si el usuario quiere llegar de una estación a otra en el menor tiempo posible, el programa va a hacer uso del algoritmo de Dijkstra. En el otro caso de que se requiera recorrer todas las estaciones en el menor tiempo posible, se va a hacer uso del algoritmo de Prim. La interacción para encontrar dichas soluciones se realizará por medio de una interfaz gráfica.

Alternativa 4:

- El programa brinda la solución a los problemas por medio de los mismos algoritmos mencionados en la alternativa 3, con la única diferencia de que la interacción con el usuario es por medio de una consola.

Alternativa 5:

- Se desarrolla el programa haciendo una unión de la alternativa 2 y la alternativa 3, brindándole al usuario una interfaz amigable, y su implementación contiene dos versiones de grafos la cual se puede hacer la ejecución de cualquiera de las dos sin afectar el resultado de la solución al problema y permite hacer uso de ambos algoritmos.

FASE 5: EVALUACIÓN Y SELECCIÓN DE LA MEJOR SOLUCIÓN.
Teniendo en cuenta las diferentes posibles soluciones, se ha realizado un análisis de la solución más factible, con base en los siguientes criterios:

- **Criterio A. Velocidad:** La alternativa es rápida en el proceso de ejecutar determinado grafo y encontrar la solución.
- **Criterio B. Compatibilidad:** La alternativa es compatible con el diseño de la interfaz gráfica y con el entorno de desarrollo utilizado por el equipo de desarrolladores.
- **Criterio C. Funcionalidad:** La alternativa es eficiente, eficaz y es aproximada a la solución deseada.
- **Criterio D. Experiencia de usuario:** La alternativa brinda una buena experiencia de usuario.

Se evaluará cada una de las alternativas asignando una calificación del 0 al 10, siendo 10 la mejor calificación y 0 la peor.

Criterio/ Alternativa	**Criterio A**	**Criterio B**	**Criterio C**	**Criterio D**	**Total**
Alternativa 1	10	0	8	5	23
Alternativa 2	10	10	8	10	38
Alternativa 3	10	10	8	10	38
Alternativa 4	10	0	8	5	23
Alternativa 5	10	10	10	10	40

Según la evaluación de cada una de las alternativas, con un total de 40 puntos, la mejor solución es usar las dos implementaciones de los grafos, los dos algoritmos y brindarle al usuario una interfaz gráfica amigable.

FASE 6: PREPARACIÓN DE INFORMES Y ESPECIFICACIONES.

Problema: Implementación de la solución haciendo uso de un determinado grafo.

Flujo del sistema: El usuario ingresa al sistema y la interfaz le ofrece 2 opciones: implementación por medio de una matriz de adyacencia o por medio de una lista de adyacencia.

- Cuando el usuario selecciona la implementación por medio de una matriz de adyacencia, el desarrollo de la solución será implementando este grafo o viceversa.

Problema: Recorrer de una estación a otra o todas las estaciones por la ruta que tome menos tiempo.

Flujo del sistema: Una vez seleccionado la forma de implementación, la interfaz le ofrece 2 opciones: recorrer todas las estaciones en el menor tiempo posible, o recorrer de una

estación a otra en el menor tiempo posible. El usuario selecciona el problema que desea solucionar.

- Cuando selecciona recorrer todas las estaciones en el menor tiempo posible, el usuario no debe de ingresar algún tipo de dato, simplemente el programa le va a mostrar cual es la ruta más rápida.
- Cuando selecciona recorrer de una estación a otra en el menor tiempo posible, el usuario ingresa la estación de origen y la estación a la que se dirige, y el programa toma la estación de origen y calcula la mejor ruta posible para llegar a la estación de destino, y debe mostrar el resultado al usuario.

FASE 7: IMPLEMENTACIÓN DEL DISEÑO.

La implementación del diseño de desarrolla en el lenguaje de programación java y se hace uso de la interfaz gráfica javaFx. Haciendo uso de grafos con matriz de adyacencia y listas de adyacencia, como también el uso de los algoritmos de Dijkstra y Prim.

Para el prototipo a desarrollar, se debe implementar las siguientes tareas:

1. Grafo con matriz de adyacencia.
2. Grafo con lista de adyacencia.
3. Algoritmo de Dijkstra.
4. Algoritmo de Prim.
5. Realizar la implementación compatible con javaFx.
6. Realizar una interfaz de usuario amigable.