INFORME

1. Contexto Problemático

Colombia es un país que se distribuye en departamentos y cada uno de estos tiene una capital, en total son 32 las ciudades principales colombianas. Estas a su vez se encuentran conectadas por medio de carreteras principales y secundarias.

La problemática de cualquier viajero siempre será el tiempo que tomará el recorrido, pues existen carreteras con más trayecto que otras. Por tal motivo, existe la necesidad de crear un programa que ayude a la problemática por la que eventualmente cualquier ciudadano enfrentará. Brindando el servicio a cualquier usuario para que este conozca el camino más corto entre una capital A y una capital B. Entregando también las posibles vías alternas que el usuario puede tomar.

2. Desarrollo de la solución

Para resolver el caso anterior se eligió el Método de la Ingeniería del libro “Introduction to Engineering” de Paul Wright. Cada uno de los pasos del Método de la Ingeniería se presentan a continuación especificando la descripción y el procedimiento de cada uno de estos pasos.

2.1 Identificación Del Problema

De acuerdo al enunciado se pudieron detectar los siguientes problemas:

1. Permitir al usuario el ingreso de las ciudades en las que desea conocer la información, es decir la ciudad A (inicio) y la ciudad B (final)
2. El programa deberá entregar al usuario la carretera con la distancia más corta posible entre las dos ciudades que ingresó.
3. El programa deberá entregar al usuario las posibles vías alternas entre las ciudades A y B, para que este también tenga otras posibilidades.

2.2 Recopilación De La Información

Para una adecuada solución del problema propuesto, se requiere la recopilación de información, que nos ayude a decidir cual es la manera más eficiente de llegar a la solución del problema, generando diferentes ideas, y finalmente tomar el camino más conveniente. Para eso debemos esclarecer todo el funcionamiento de las estructuras de datos Grafos y además conocer un poco sobre la problemática de carreteras en Colombia.

2.2.1 Información acerca de las estructuras de datos

Grafos.

Los grafos son estructuras matemáticas que representan relaciones de pares entre objetos. Una grafo es una estructura de flujo que representa la relación entre varios objetos. Se puede visualizar utilizando los siguientes dos componentes básicos:

Nodos: Estos son los componentes más importantes en cualquier gráfico. Los nodos son entidades cuyas relaciones se expresan utilizando aristas. Si un gráfico comprende 2 nodos y B y un borde no dirigido entre ellos, expresa una relación bidireccional entre los nodos y el borde.

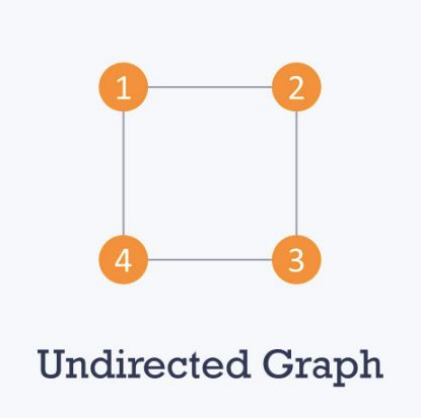
Bordes(aristas): los bordes son los componentes que se utilizan para representar las relaciones entre varios nodos en un gráfico. Un borde entre dos nodos expresa una relación unidireccional o bidireccional entre los nodos.

Tipos de nodos:

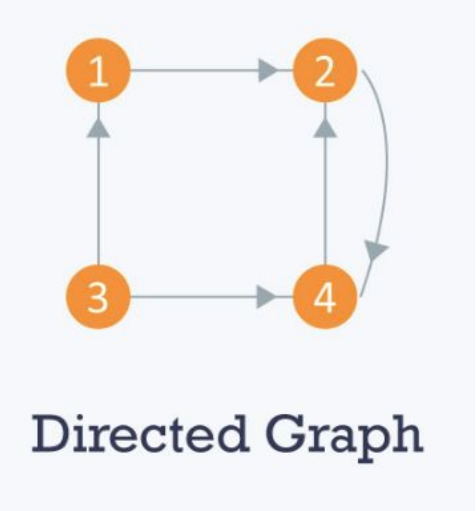
- Nodo raíz: el nodo raíz es el antecesor de todos los demás nodos en un grafo. No tiene ningún antepasado. Cada grafo consta de exactamente un nodo raíz. En general, debe comenzar a atravesar un grafo desde el nodo raíz.
- Nodos de hoja: en un grafo, los nodos de hoja representan los nodos que no tienen ningún sucesor. Estos nodos solo tienen nodos ancestros. Pueden tener cualquier cantidad de bordes entrantes pero no tendrán bordes salientes.

Tipos de Grafos:

No dirigido: un grafo no dirigido es un grafo en el que todos los bordes son bidireccionales, es decir, los bordes no apuntan en una dirección específica.

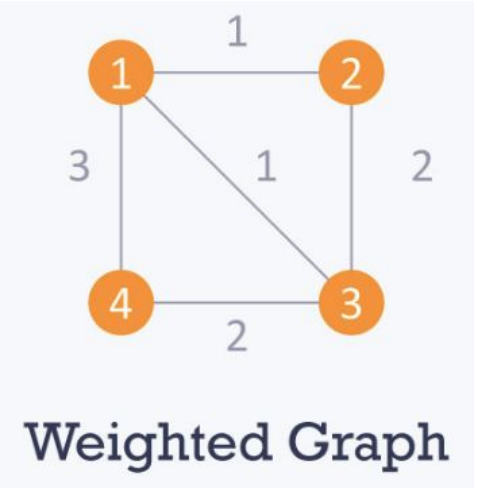


Dirigido: un grafo dirigido es un grafo en el que todos los bordes son unidireccionales, es decir, los bordes apuntan en una sola dirección.



Ponderado: en un gráfico ponderado, a cada borde se le asigna un peso o un costo. Considere una gráfica de 4 nodos como en el diagrama a continuación. Como puede ver, cada borde tiene un peso / costo asignado. Si desea pasar del vértice 1 al vértice 3, puede tomar una de las siguientes 3 rutas:

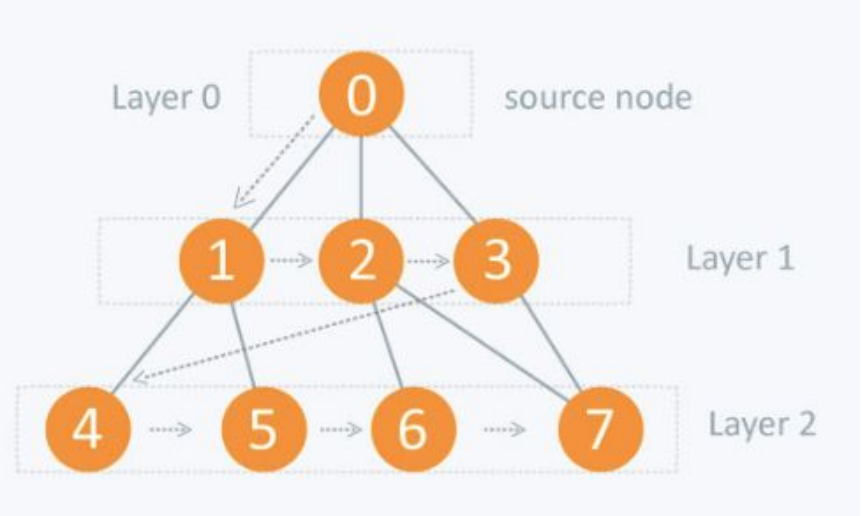
- 1 -> 2 -> 3
- 1 -> 3
- 1 -> 4 -> 3



Primera búsqueda de amplitud (BFS)

BFS es un algoritmo de desplazamiento en el que debe comenzar a atravesar desde un nodo seleccionado (origen o nodo de inicio) y recorrer el gráfico a nivel de la capa, explorando los nodos vecinos (nodos que están conectados directamente al nodo de origen). Luego debes moverte hacia los nodos vecinos del siguiente nivel.
Como sugiere el nombre BFS, debe cruzar el gráfico a lo largo de la siguiente manera:

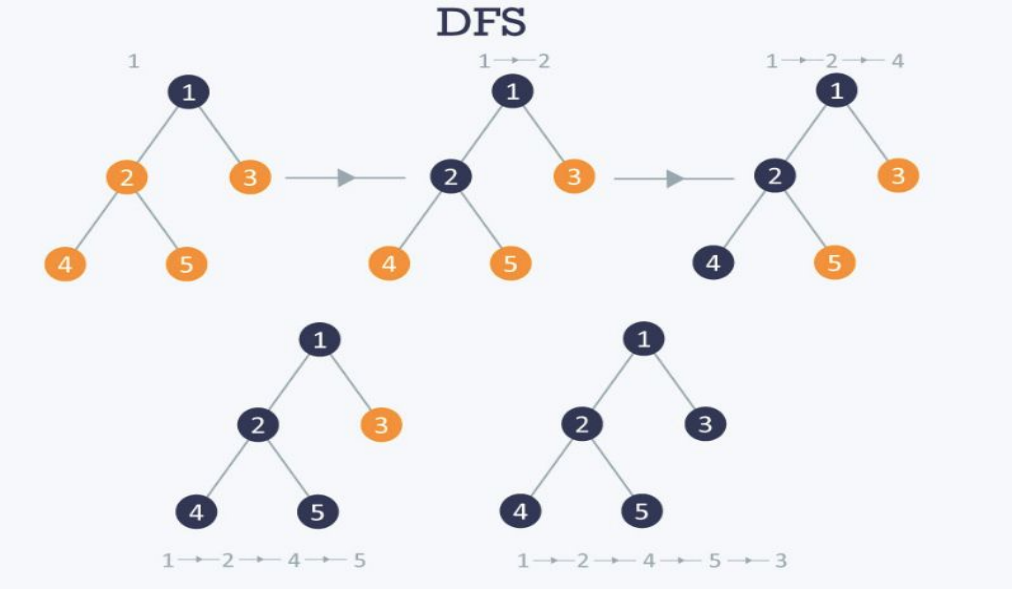
1. Primero mueva horizontalmente y visite todos los nodos de la capa actual
2. Mover a la siguiente capa



Primera búsqueda de profundidad (DFS)

El algoritmo DFS es un algoritmo recursivo que utiliza la idea de dar marcha atrás. Implica búsquedas exhaustivas de todos los nodos al avanzar, si es posible, o al retroceder.
Aquí, la palabra retroceder significa que cuando se está moviendo hacia adelante y no hay más nodos a lo largo de la ruta actual, se mueve hacia atrás en la misma ruta para encontrar nodos para atravesar. Todos los nodos serán visitados en la ruta actual hasta que todos los nodos no visitados hayan sido atravesados, después de lo cual se seleccionará la siguiente ruta.
Esta naturaleza recursiva de DFS se puede implementar utilizando pilas. La idea básica es la siguiente:
elija un nodo de inicio y empuje todos sus nodos adyacentes en una pila.
Pop un nodo de la pila para seleccionar el siguiente nodo para visitar y empujar todos sus nodos adyacentes en una pila.
Repita este proceso hasta que la pila esté vacía. Sin embargo, asegúrese de que los nodos que se visitan estén marcados. Esto evitará que visites el mismo nodo

más de una vez. Si no marca los nodos visitados y visita el mismo nodo más de una vez, puede terminar en un bucle infinito.



Algoritmo de Bellman Ford:

El algoritmo de Bellman Ford se usa para encontrar las rutas más cortas desde el vértice de origen a todos los otros vértices en un gráfico ponderado. Depende del siguiente concepto: la ruta más corta contiene como máximonorte-1 Bordes, porque el camino más corto no podría tener un ciclo.

Pasos del algoritmo:

- El bucle exterior atraviesa desde 0 : norte-1.
- Recorra todos los bordes, verifique si la siguiente distancia de nodo> distancia de nodo actual + peso de borde, en este caso, actualice la siguiente distancia de nodo a "distancia de nodo actual + peso de borde".

Este algoritmo depende del principio de relajación, donde la distancia más corta para todos los vértices se reemplaza gradualmente por valores más precisos hasta llegar a la solución óptima. Al principio, todos los vértices tienen una distancia de "Infinito", pero solo la distancia del vértice fuente =0, luego actualice todos los vértices conectados con las nuevas distancias (fuente, vértice, distancia + bordes de peso), luego aplique el mismo concepto para los nuevos vértices con nuevas distancias y así sucesivamente.

El algoritmo de Dijkstra

El algoritmo de Dijkstra tiene muchas variantes, pero la más común es encontrar las rutas más cortas desde el vértice de origen a todos los otros vértices en el gráfico.

Pasos del algoritmo:

- Establecer todas las distancias de vértices = infinito, excepto el vértice de origen, establecer la distancia de origen = 0.
- Empuje el vértice de origen en una cola de prioridad mínima en el formulario (distancia, vértice), ya que la comparación en la cola de prioridad mínima estará de acuerdo con las distancias de los vértices.
- Pop el vértice con la distancia mínima de la cola de prioridad (en primer lugar el vértice popped = fuente).
- Actualice las distancias de los vértices conectados al vértice emergente en caso de "distancia de vértice actual + peso del borde <distancia del vértice siguiente", luego empuje el vértice
- con la nueva distancia a la cola de prioridad.
- Si el vértice resaltado es visitado anteriormente, simplemente continúe sin usarlo.
- Vuelva a aplicar el mismo algoritmo hasta que la cola de prioridad esté vacía.

El algoritmo de 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. El algoritmo de Floyd-Warshall es un ejemplo de programación dinámica.

Los pasos a dar en la aplicación del algoritmo de Floyd son los siguientes:

- Formar las matrices iniciales C y D.
- Se toma k=1.
- Se selecciona la fila y la columna k de la matriz C y entonces, para i y j, con $i \neq k$, $j \neq k$ e $i \neq j$, hacemos:

Si (Cik + Ckj) < Cij → Dij = Dkj y Cij = Cik + CkjEn caso contrario, dejamos las matrices como están.

- Si k ≤ n, aumentamos k en una unidad y repetimos el paso anterior, en caso contrario páramos las interacciones.
- La matriz final C contiene los costes óptimos para ir de un vértice a otro, mientras que la matriz D contiene los penúltimos vértices de los caminos óptimos que unen dos vértices, lo cual permite reconstruir cualquier camino óptimo para ir de un vértice a otro.

El algoritmo de Kruskal

El algoritmo de Kruskal construye el árbol de expansión agregando bordes uno por uno en un árbol de expansión en crecimiento. El algoritmo de Kruskal sigue un enfoque codicioso, ya que en cada iteración encuentra un borde que tiene menos peso y lo agrega al creciente árbol de expansión.

Pasos del algoritmo:

- Ordena los bordes del gráfico con respecto a sus pesos.
- Comience agregando bordes al MST desde el borde con el peso más pequeño hasta el borde del peso más grande.
- Solo agregue bordes que no formen un ciclo, bordes que solo conecten componentes desconectados.

Algoritmo de Prim

El algoritmo de Prim también usa el enfoque codicioso para encontrar el árbol de expansión mínimo. En el algoritmo de Prim, crecemos el árbol de expansión desde una posición inicial. A diferencia de un borde en Kruskal, agregamos vértice al creciente árbol de expansión en Prim's.

Pasos del algoritmo:

- Mantener dos conjuntos desarticulados de vértices. Uno que contiene vértices que están en el árbol de expansión creciente y otro que no están en el árbol de expansión creciente.
- Seleccione el vértice más barato que está conectado al árbol de expansión creciente y no está en el árbol de expansión creciente y agréguelo al árbol de expansión creciente. Esto se puede hacer usando colas de prioridad. Inserte los vértices, que están conectados al creciente árbol de expansión, en la Cola de prioridad.

- Compruebe si hay ciclos. Para hacerlo, marque los nodos que ya se han seleccionado e inserte sólo los nodos en la Cola de prioridad que no están marcados.

2.2.2 Información acerca de la red vial colombiana

¿Que es?

La Red Nacional de Carreteras es la red vial de Colombia regulada por el Ministerio de Transporte colombiano mediante el Instituto Nacional de Vías (INVÍAS) y sus direcciones territoriales (Decreto 1735 de agosto de 2001[2]) y a veces delegadas a empresas privadas por concesión.

El sistema se compone por la Red Primaria (Grandes Autopistas, a cargo de la nación), Red Secundaria (a cargo de departamentos) y Red terciaria (compuesta por carreteras terciarias o caminos interveredales, a cargo de los municipios).

Cifras.

La Red de Carreteras colombiana es de 206.727 km, de los cuales 19.306 km corresponde a la Red Primaria Nacional, 45.137 km corresponde a la Red Secundaria Nacional y 142.284 km corresponden a la Red Terciaria Nacional. Asimismo, cuenta con 5.097 puentes a nivel nacional y 1.266,80 km en Doble Calzada, 10 viaductos y 40 túneles. Según un informe de la Cámara Colombiana de Infraestructura, Colombia tiene 9 km de vías por cada kilómetro cuadrado de área

Clasificación.

Carreteras primarias.
Corresponde la Red Vial Nacional de rutas nacionales primarias, se encuentran a cargo del Instituto Nacional de Vías INVIAS y pueden ser concesionadas por medio de la Agencia Nacional de Infraestructura ANI. Este tipo de carreteras puede ser calzadas divididas según las exigencias propias de cada proyecto. Deben ser pavimentadas.

Carreteras secundarias.
Corresponde a la Red Vial Secundaria y son vías que unen las cabeceras municipales entre sí y/o que vienen desde una cabecera municipal y conectan con una Carretera primaria. Pueden ser pavimentadas o en afirmado aunque la mayor parte se encuentran en afirmado. Se encuentran a cargo de cada

departamento el cual debe llevar inventario y establecer un plan vial departamental para su mejoramiento. también se le considera generalmente como rutas o carreteras departamentales. Existen algunas carreteras secundarias que son entregadas en concesión a privados para su construcción y mejoramiento.

Carreteras terciarias.
Corresponde a la Red Vial Terciaria, son vías que unen las cabeceras municipales con sus veredas o unen veredas entre sí. mayormente están en afirmado. Si se pavimentan deben cumplir a las condiciones geométricas fijadas para las Vías secundarias. las carreteras terciarias generalmente se encuentran a cargo de los municipios, así mismo hay carreteras a cargo del departamento y carreteras terciarias a cargo del INVIAS.

Caminos Vecinales.
Corresponde a carreteables que no se encuentran clasificados en ninguna de las redes nacionales de carreteras, dichos carreteables son construidos y se encuentran a cargo de los municipios. También hay carreteables que son construidos por la comunidad para suplir una necesidad de transporte y en ocasiones por grupos al margen de la ley para la movilización de tropas o transportar productos ilícitos. la mayoría de los carreteables no cumplen con las exigencias del INVIAS y el Ministerio de Transporte, estando mayormente en afirmado, en mal estado o transitables solamente en la estación seca.

2.3 Búsqueda Soluciones Creativas

1. Para la solución del problema, podremos crear Listas enlazadas de las capitales donde su anterior y si siguiente sean las ciudades que están unidas por medio de las carreteras.
2. Para la solución del problema, podremos crear diferentes listas (ArrayList en lenguaje Java) en donde almacenaremos datos de tipo Ciudad en donde tendremos atributos como las carreteras que tiene y las ciudades adyacentes a él.
3. Los árboles son estructuras eficaces que nos ayudan a acceder a los datos que tenemos almacenados, por lo tanto una estrategia sería hacer uso de estos tipos de árboles para encontrar las carreteras adyacentes
4. Las estructuras vistas en las últimas semanas del curso llamadas Grafos nos ayudarían para este problema, usando los nodos como si fueran las capitales y las aristas como si fueran las carreteras, para la distancia de cada carretera se podría usar un grafo ponderado.

2.4 Transición De Las Ideas A Los Diseños Preliminares

- **Alternativa 1:** Al hacer uso de listas enlazadas tendríamos el problema de que no podremos manejar más de una relación, pues en esta problemática, perfectamente entre dos ciudades pueden existir 2 o más carreteras, lo cual en listas sería difícil de trabajar.
- **Alternativa 2:** Con las ArrayList su desventaja ocurre al momento de comparar con los árboles y los grados, pues en este caso es más rápido (eficaz) trabajar con árboles o grafos, ya que el acceder a un dato dentro de él es mucho más rápido que buscar un dato en ArrayList.
- **Alternativa 3 y 4**: Durante nuestra preparación en el curso, hemos aprendido que todo árbol es un grafo, pero que no todo grafo es un árbol. Por lo cual decidimos trabajar con las estructuras grafos, y para ser mas especifico con los grafos ponderados que nos servirá para trabajar mediante el costo, la distancia de la carretera, las aristas, las carreteras como tal y los nodos las ciudades capitales. concluimos entonces que haremos uso de la ALTERNATIVA 4.

2.5 Evaluación Y Selección De La Mejor Solución

De acuerdo a las observaciones que le hicimos a cada alternativa encontramos al final la aternativa que cumplira con el objetivo de manera óptima.

- Hacer uso de Las estructuras Grafos, siendo el grafo ponderado el centro de nuestra solución del problema.

2.6 Preparación De Informes Y Especificaciones

Especificación del Problema (en términos de entrada/salida)

- ***Problema***: Encontrar el recorrido más corto entre dos ciudades capitales de colombia.
- ***Entradas:*** Capital de inicio, capital de destino.
- ***Salida:*** Se encontró el camino más corto entre las dos ciudades capitales.

2.7 Implementación Del Diseño

- Los requerimientos funcionales serán entregados en un archivo adicional.
- El código del programa será entregado en lenguaje Java, implementado en Java eclipse.