TÍTULO (DESCRIPCIÓN CORTA DEL PROYECTO. ENTRE 8 Y 12 PALABRAS)

Daniel Felipe Gómez Martínez
Universidad Eafit
Colombia
dfgomezm@eafit.edu.co

Daniel García García
Universidad Eafit
Colombia
dgarciag@eafit.edu.co

Cesar Andrés García Posada
Universidad Eafit
Colombia
cagarciap@eafit.edu.co

RESUMEN

No es ningún secreto que el tráfico aumenta cada día en nuestra ciudad, lo que genera diversos problemas, como congestión de la carretera, deterioro de la calidad del aire, ineficiencia al pasar de un lugar a otro, disminución de la productividad, entre otros problemas.

Para controlar y / o reducir las situaciones mencionadas anteriormente, es importante e indispensable encontrar una solución para reducir el tráfico en la ciudad.

PALABRAS CLAVES

Grafo completo y dirigido → Camino más corto → Estructuras de datos → Complejidad → Eficiencia → Ordenamiento → Recorrido de grafos

1. INTRODUCCIÓN

En la ciudad de Medellín, por cada 1000 personas (población) hay aproximadamente 345 vehículos y más del 60% de los vehículos transitan por la ciudad todos los días.

Los niveles de contaminación aumentan significativamente en el Valle de Aburrá y en el país, por lo que se llevan a cabo campañas como "El Pico y Placa ambiental". Además, los numerosos embotellamientos debido a la cantidad de vehículos generan retrasos y pérdida de tiempo, entre otros aspectos.

Es esencial sacar la mayor cantidad posible de vehículos de las calles, y para esto, una estrategia de solución puede ser Vehículos compartidos para ir a la universidad, los estudiantes que viven cerca pueden compartir el vehículo.

2. PROBLEMA

Para ayudar al medio ambiente y la calidad de vida de los habitantes de la ciudad de Medellín, el problema es realizar un algoritmo de manera eficiente para reducir el tráfico a través de la estrategia de los Vehículos Compartidos. El algoritmo debe determinar la cantidad mínima de vehículos para que todas las personas puedan llegar a su destino. Se asumen varios aspectos:

1) Todas las personas van al mismo lugar.

2) En cada vehículo pueden ir 5 personas.

3) El conductor del coche toma la ruta más corta.

3. TRABAJOS RELACIONADOS

3.1 Problema de viaje canadiense

El problema del viaje canadiense, fue definido por Papadimitriou y Yannakak en 1991 [2]. Este problema es encontrar la ruta más corta entre un punto de origen y un punto a destino. Estos puntos están en un gráfico G = (V, E) (G es un gráfico, V son nodos del gráfico y E son bordes) pero, el viajero no conoce la información completa sobre el gráfico (la gráfica de entrada G puede sufrir cambios). [3]
En particular, algunos de los bordes del gráfico pueden quedar bloqueados e intransitables.
El viajero tiene dos opciones para resolver el problema:

1) Avance en el viaje y encuentre un borde bloqueado y determine un nuevo camino. El costo de la nueva ruta se agrega al último cálculo de costo.

2) Pague un recargo sobre el costo y determine previamente los bordes bloqueados, pero los bordes bloqueados pueden convertirse en un borde pasable.

El algoritmo para encontrar la ruta más corta es el avance en el gráfico que elige el nodo que tiene el borde menos costoso.

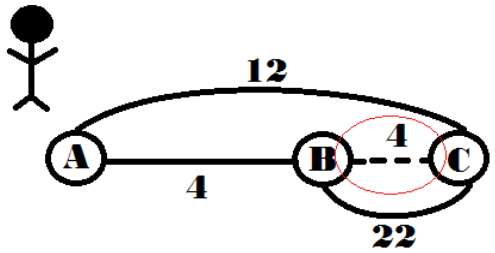


***Figure 1**. Example*

En la figura anterior, el viajero quiere ir del nodo A al nodo C, cuando llega al nodo B, se da cuenta de que el borde entre B y C está bloqueado. El viajero debe calcular la nueva ruta más corta, que sería regresar al nodo A e ir directamente al nodo C, pero el costo de esta nueva ruta se agregará al costo previamente calculado.

La otra opción que tiene el viajero es pagar un costo adicional en el nodo A, donde se determinan las rutas bloqueadas.

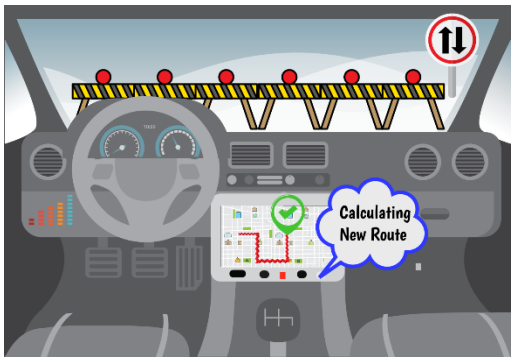


***Figure 2**. Prototype solution*

3.2 Rompiendo el Breakdow

El desglose se conoce como el momento en el que salen más autos en un área determinada, lo que aumenta considerablemente el tráfico. Un algoritmo creado en la Universidad Tecnológica de Nanyang de Singapur ha logrado reducir la congestión de tráfico, a través de la ruptura de la avería, desarrollando un algoritmo que evita las colisiones mediante la redirección de vehículos que hacen uso de rutas alternativas, haciendo así cambios en el tráfico. red. Esto es cierto a través del aprendizaje automático (IA). A la máquina se le asigna una red de tráfico y esto hace millones de cálculos en la red, estos cálculos son cambios de carreteras y su dirección en el gráfico (Red de tráfico), luego, analizando el impacto del nuevo cambio realizado en la red, elija Esos caminos que evitan las averías. [4]

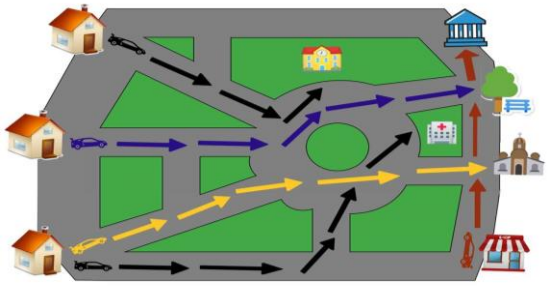


***Figure 3.** Problem*

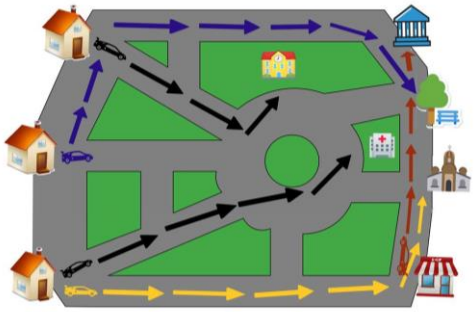


***Figure 4.** Possible solution*

3.3 Reducción de taxis

Investigadores italianos de CNR (Consiglio Nazionale Delle Ricerche), junto con expertos de MIT (Medical Innovative Technology) y la Universidad de Cornell, desarrollaron un algoritmo que puede reducir las flotas de vehículos, esto se debió al tráfico en la ciudad de Nueva York causado en gran parte por los taxis. El programa garantiza los mismos niveles de reducción del servicio de taxi, además de ofrecer turnos de trabajo más cortos para los taxistas. El método se basa en el modelo "Red de intercambio de vehículos" y lo que hace es encontrar la secuencia de viajes que puede asistir a un solo vehículo, analizando los tiempos y las coordenadas del punto de recogida y descenso del pasajero. Se organizaron taxis para que cada uno hiciera la ruta más corta sin un pasajero, reduciendo así la cantidad de taxis que ofrecen el servicio en una ciudad. [5]

3.4 Pathfinding

Pathfinding es una base importante para los sistemas de navegación, ya que se utiliza en áreas de robótica y, especialmente, en videojuegos. El objetivo principal es encontrar la mejor manera posible desde un punto inicial hasta un punto final en representaciones del entorno llamadas mapas de malla. En el caso de los videojuegos (especialmente en los juegos de estrategia), los personajes deben moverse en la ruta más corta desde su ubicación hasta un punto final, e incluso si poseen estos dos puntos, también se debe tener en cuenta que la ruta debe tener el Mínimo de posibles obstáculos que retrasen su llegada a su destino. El algoritmo de Dijkstra (con una cola de prioridad) orientado a la geometría computacional se implementa mediante la estructura del videojuego donde se utilizan técnicas de geometría clásica, topología, teoría de grafos, teoría de conjuntos y álgebra lineal. Tan abstracto y encontrar el camino más corto y eficiente. [6]

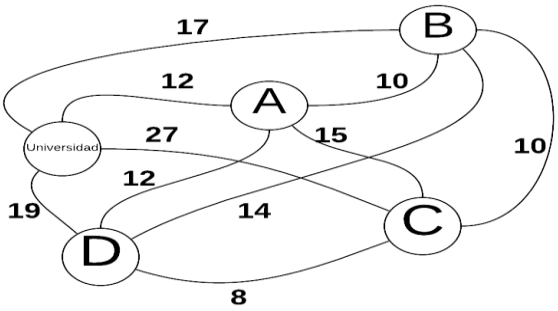
***Figure 5.** Prototype solution*

4. SOLUCIÓN PRELIMINAR PROYECTO

4.1 Estructura de datos

Para la realización de la primera solución, se usaron las siguientes estructuras de datos:

- ***Grafo:*** grafo de los puntos en la ciudad de Medellín.



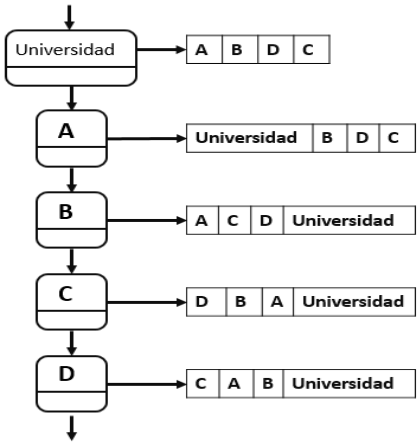
Grafica 6. *Ejemplo Grafo completo no dirigido.*

- ***Matriz:*** Para el control y maneo del grafo, utilizamos la implementación del grafo por medio de la matriz, donde se almacena el valor del peso correspondiente a cada nodo.

	Universidad	A	B	C	D
Universidad	0	7	17	27	19
A	7	0	10	15	12
B	17	10	0	10	14
C	27	15	10	0	8
D	19	12	14	8	0

Grafica 7. *Matriz correspondiente al grafo anterior*

- ***LinkedList:*** Estructura necesaria para tener de una manera organizada los nodos adyacentes a un nodo especifico

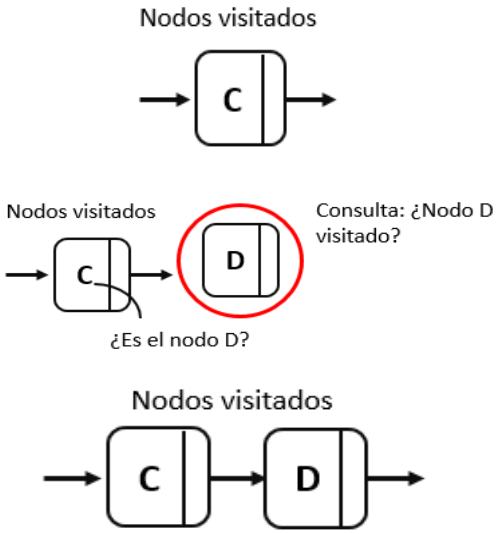


Grafica 8. *LikedList de arreglos, e la cual se ordenan los nodos adyacentes de menor a mayor costo.*

4.2 Operaciones de la estructura de datos

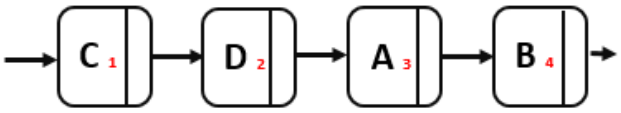
Para la solución del problema, contamos con diversas LinkedList, con el fin de tener un mayor control y una mejor funcionalidad a la hora de trabajar con nodos del grafo:

- ***Consulta y agregado:*** LinkedList de nodos visitados: con el objetivo de no volver a visitar nodos ya visitados, para o aumentar el costo



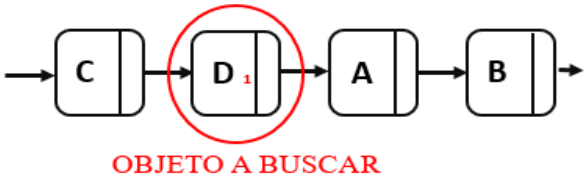
Grafica 9. *Agregando a la lista de nodos visitados, un nuevo nodo.*

- ***Size:*** *Nos devuelve el tamaño de la lista.*



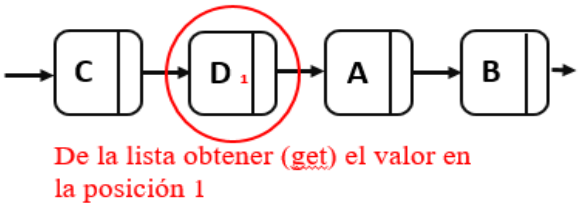
Grafica 10. *Tamaño de la LinkedList*

- ***IndexOf:*** Nos proporciona la información correspondiente a en qué lugar de la lista está el objeto que se desea buscar.



Grafica 11. *Ejemplo indexOf(B)*

- ***Get:*** *Esta operación, nos permite obtener el valor que hay en una determinada posición de la lista.*



Grafica 12. *Ejemplo get(1)*

4.3 Criterios de diseño de la estructura de datos

La estructura de datos utilizada es una LinkedList de listas. (Inicialmente se trabaja con una LinkedList con arreglos), con el fin de ordenar de menor a mayor los nodos adyacentes a cada nodo. Esto con el objetivo de realizar descartes a medida que se va calculando un camino especifico. Además, tomando en cuenta que la complejidad de los métodos de la LinkedList son bastantes eficientes.

4.4 Análisis de Complejidad

A continuación, se muestra una tabla con la complejidad de algunos métodos de la LinkedList

Método	Complejidad
get()	O(n)
add()	O(1)
remove()	O(1)
countains()	O(n)
size()	O(n)

Tabla 1. Complejidad métodos LinkedList

4.5 Algoritmo

Tomando en cuenta que se quiere determinar la cantidad mínima de vehículos que se requieran para que todas las personas puedan llegar a la universidad, usando la estrategia de vehículo compartidos, tenemos que es necesario disminuir el costo necesario de traslado del vértice más lejano de la universidad, o bien sea aprovechar el recorrido de este recogiendo otras personas.

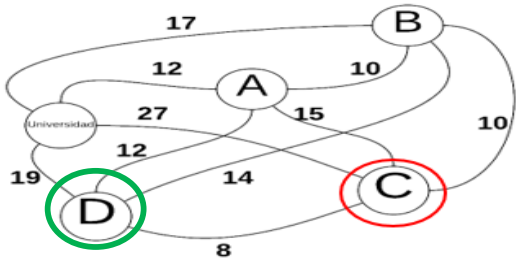
Lo dicho anteriormente se realizará por medio de la siguiente proceso o algoritmo:

1) Se crea un arreglo con el costo adicional de cada uno de los vértices, para realizar a comparación más adelante.
2) Se determina el vértice más lejano al vértice de llegada, en este caso es la universidad. Si es la segunda o la N iteración, se descartan los vértices ya visitados.
3) Se determina el menor vértice del vértice elegido en el punto anterior.
    a. Se calcula el costo hasta el momento.
    b. Se realiza una comparación entre el costo encontrado en el punto anterior y se determina si este costo es menor o igual al costo analizado en el puto 1. Es decir, se determina si es viable la realización del recorrido calculado.

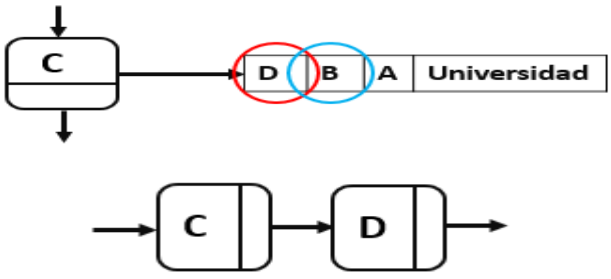
    Al ser viable el camino, en un arreglo de vértices visitados, se agrega el camino encontrado, con el fin de no volver pasar sobre este vértice. De otro modo seguimos con el siguiente vértice menor encontrado. Este proceso se realiza hasta obtener un automóvil lleno o hasta visitar todos los vértices adyacentes.

    Al tener un automóvil lleno volvemos al numeral 2. Este proceso se repite hasta tener todos los vértices visitados.

{Vértice de llegada}, {(Vértices Adecentes, Costo)}
{Universidad},{(A, 12), (B, 17), (C, 27), (D, 19)}
Mayor{Universidad} = (C, 27)



Menor{D} = (C)
CostoMinimoInicial{D} = 19
ComparacionCosto = NuevoCosto <= CostoMinimoInicial * Constante



Este proceso se repite hasta que el NuevoCosto > CostoMinimoInicial * Constante, en este caso seguiría el

siguiente menor, por ejemplo, el vértice encerrado en azul, en caso de llenar un automóvil continuaríamos con el vértice encerrado en verde.

Grafica 13. *Solución paso por paso*

4.6 Cálculo de la complejidad del algoritmo

Sub problema	Complejidad
Ordenamiento	$O(n^2 \log n)$
Llenar un carro	$O(n^2)$
Solución Preliminar	$O(n)$
Complejidad Total	$O(n^3)$

Tabla 2: complejidad de cada uno de los sub problemas que componen el algoritmo. Sea N la cantidad de vértices del grafo.

4.7 Criterios de diseño del algoritmo

Debido a que el problema consta en disminuir la cantidad de vehículos que llegan a un determinado punto, en este caso la universidad, teniendo en cuenta que por vértice hay una forma de llegar al punto, nos enfocamos en como disminuir el costo de los trayectos más largos, a través de nuevos caminos, en los cuales se pueden recoger otras personas, teniendo en cuenta que en cada vértice hay una persona con un vehículo.

4.8 Tiempos de Ejecución

	Conjunto de 5 vértices	***Conjunto de 205 vértices***
Mejor caso	0 msg	5 seg
Caso promedio	0 msg	5 seg
Peor caso	0 msg	5 seg

Tabla 3: Tiempos de ejecución del algoritmo con diferentes conjuntos de datos

4.9 Memoria

Mencionar la memoria que consume el programa para varios ejemplos

	Conjunto de 5 vértices	***Conjunto de Datos 2***
Consumo de memoria	65 MB	2665 MB

Tabla 4: Consumo de memoria del algoritmo con diferentes conjuntos de datos

REFERENCES

[1] Álvarez C Víctor Andrés. 2015 *Recuperado de:* https://www.elcolombiano.com/antioquia/movilidad/en-medellin-transita-un-carro-por-cada-tres-habitantes-EB3232363

[2] Chung-Shou. Yamming Huang 2014 *Recuperado de:* https://www.sciencedirect.com/science/article/pii/S0304397514001327

[3] Bnya Zahy. Felner Ariel. Eyal Shimony Solomon. (S, f) *Recuperado de:* https://www.aaai.org/ocs/index.php/IJCAI/IJCAI-09/paper/viewFile/487/683

[4] Sabál Antonio 2017 *Recuperado de:* https://blogthinkbig.com/el-algoritmo-que-puede-acabar-con-la-congestion-del-trafico

[5] Barbieri Alberto 2018 *Recuperado de:* https://www.nobbot.com/futuro/algoritmos-reducir-trafico-nueva-york/

[6] Cuevas Carvajal Danilo Alejandro 2013 *Recuperado de: http://opac.pucv.cl/pucv_txt/txt-5000/UCE5372_01.pdf*