MEJORÍA EN LA MOVILIDAD DEL TRÁFICO EN MEDELLÍN: ALGORITMO AIGNADOR DE RUTAS A EMPLEADOS DE UNA EMPRESA PARA LA OPTIMIZACIÓN DE LAS VÍAS.

Juan Sebastián Sanín V.
Universidad EAFIT
Colombia
jssaninv@eafit.edu.co

Juan Pablo Peña F.
Universidad EAFIT
Colombia
jppenaf@eafit.edu.co

Mauricio Toro
Universidad EAFIT
Colombia
mtorobe@eafit.edu.co

RESUMEN

El problema nos propone un déficit en la movilidad en Medellín y en su medio ambiente el cual ya afecta a sus habitantes, para este problema implementaremos un algoritmo el cual va a hacer un orden en los vehículos que llegan únicamente a el conductor a su lugar de trabajo, para que así recoja a sus compañeros en el trayecto sin verse afectado. El problema es muy importante para la mejora del manejo de las estructuras de datos, ya que necesitamos generar respuesta en el menor tiempo posible (4 segundos) y esto nos generará una mejor actitud al enfrentarnos a estos problemas ya que son situaciones que vemos diariamente y necesitan una respuesta urgente. Hay una gran cantidad de problemas con respecto a esta temática, como lo es, repartir periódicos en el recorrido de menor distancia, llegar a un lugar en específico con varias posibilidades de rutas diferentes, etc.

Palabras clave

- Matriz
- Voraz
- Tiempo
- Memoria
- Datos
- Ruta

1. INTRODUCCIÓN

Abordando el problema desde la ciudad de Medellín, se ha notado el aumento notable de la contaminación del aire, con un gran aporte debido al crecimiento de la venta de vehículos en la ciudad, lo cual también empeora el tráfico de esta, puesto que es uno de los medios de transporte más útiles para las personas y con el cual pueden llegar a su trabajo o estudio en su propio vehículo. Estos factores perjudican enormemente la calidad de vida, no solo de los habitantes de Medellín, sino del área metropolitana, debido a que afecta la salud de las personas que respiramos el aire tan contaminado y además al alterar el tráfico las personas tendrían que madrugar más a cumplir sus obligaciones, lo cual genera más caos y estrés.

2. PROBLEMA

La problemática a resolver consiste en: En Antioquia, más exactamente en Medellín, se presenta una alta congestión de automóviles y otros transportes, donde podemos visualizar que muchas personas utilizar autos ocupando únicamente un espacio, lo que se quiere mejorar con este proyecto es directamente a dicho problema de movilidad, en el cual las empresas propongan a sus empleados que a medida que vayan en el trayecto al trabajo, recojan a sus compañeros para así evitar el uso de más autos sin ocupar completamente todos los espacios y puedan hacerlo sin tener una alta variación en el tiempo normal en sus trayectos a la empresa.

3. TRABAJOS RELACIONADOS

3.1 El problema de Santiago

Algoritmo para encontrar el camino más corto de un punto de inicio a un punto final implementando backtracking.

El problema consiste en donde dos personas quieren hacer el recorrido de Sevilla hasta el santuario de Santiago el apóstol en España, una de las dos personas marca el camino más corto recorriendo cada ciudad como él lo presenta, mientras que la otra persona quiere hacer un recorrido más alterno. Para este problema se hace una solución mediante el recorrido de grafos (cada nodo sería cada ciudad) usando backtracking, haciendo una comparativa de cada recorrido, de una ciudad a otra, pero a medida que encuentre el recorrido con menor distancia no siga haciendo una comparativa innecesaria, si no que nos informe del camino ya registrado más corto.

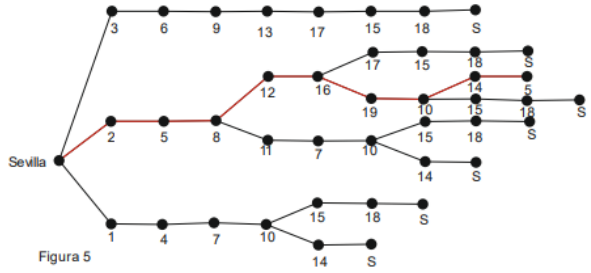

Figura 1. Representación del problema uno con sus respectivos valores de distancia entre cada ciudad(nodo) hasta el santuario (nodo final).

3.2 Problema de la gira del caballo

Algoritmo para que un caballo en un tablero de ajedrez vaya a todos los cuadros.

Este problema consiste en un caballo del ajedrez, el cual tiene que visitar todos los cuadros del tablero una única vez. Sabemos que en un tablero de 8 x 8 el número de giras posibles es de 1.305x10^35. La solución para este problema sería representar cada cuadro con los movimientos posibles del caballo en un grafo y usar un algoritmo de recorrido, el cual sería apto para este problema si se plantea un tablero

de pequeñas dimensiones, la fuerza bruta, para así ir guardando cada gira del caballo.

3.3 El recorrido del cartero

Algoritmo para que una persona recorra un barrio pasando por todas las calles y minimizando el número de calles que debe recorrer más de una vez.

Este problema consta de un cartero que tiene que salir de la oficina de correos y tiene que recorrer todas las calles entregando correos y volver a la oficina. Se tiene en cuenta que una calle va de una esquina a la otra y que la oficina de correos está ubicada en una esquina. Se busca que el recorrido minimice el número de calles que el cartero está obligado a recorrer más de una vez.

3.4 El problema de la mesa redonda

Algoritmo para para encontrar el número de permutaciones que se deben hacer para que todos los elementos de un conjunto de un tamaño de 7 puedan ser combinados todos entre sí.

Un grupo de 7 personas acuerdan cenar juntas en diferentes ocasiones. En cada ocasión se sientan alrededor de una mesa redonda de modo que cada persona tiene a sus dos lados comensales distintos en cenas diferentes. Se busca saber cuántos días deben citarse para cenar si todos quieren sentarse junto a todos los demás.

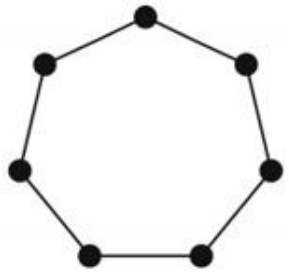

Figura 2.

Figura 3.

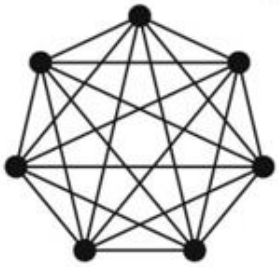

Figura 4.

4.1 Estructura de datos

La estructura de datos que se va a utilizar es la **Matriz**, ya que como todos los nodos están conectados entre sí la mejor forma de guardar el tiempo de recorrido de un nodo con otro es con esta estructura.

	1	**2**	**3**	**EAFIT**
1	0 min	4 min	8 min	12 min
2	4 min	0 min	6 min	20 min
3	8 min	6 min	0 min	5 min
EAFIT	12 min	20 min	5 min	0 min

Figura 5. Matriz. La primera fila y la primera columna representan los nodos, el resto la distancia en tiempo entre estos.

El tiempo que tarda un nodo en llegar a el mismo es 0 (cero), por tanto, la diagonal de la matriz siempre va a ser este número.

4.2 Criterios de diseño de la estructura de datos

Inicialmente, pensamos en diseñar la estructura de datos en forma de matriz debido al espacio en memoria que utiliza, ya que, en comparación con otras estructuras, en este caso la matriz es la que menos ocupa.

Además, usamos esta estructura porque se nos hace sencillo el manejo de los datos que en este caso es la distancia en tiempo de un nodo a otro, y gracias a esto se puede lograr reducir el código para un mejor entendimiento.

4.3 Análisis de Complejidad

Método	**Complejidad**
Creación de la Matriz	$O(n^2)$

Tabla 1. Complejidad de la operación para el peor de los casos.

4.4 Algoritmo

```
public static void main(String[] args) {
    int h;
    int count =0;
    try{
    int i=0;
    File f = new File("data.txt");
    Scanner s = new Scanner(f);
    while(s.hasNextLine()){
    String line = s.nextLine();
    String cortar [] = line.split(" ");
```

```
        String vx=cortar[i];
        String vy=cortar[i+1];
        String vz=cortar[i+2];
           System.out.println(vx);
//   int Vx= Integer.parseInt(cortar[i]);
//              int Vy= Integer.parseInt(cortar[i+1]);
//              int Vz= Integer.parseInt(cortar[i+2]);
            // Orden[vx][vy]=vz;
             matriz(vx,vy,vz);
            }
           // }
       // }
        }
        catch(Exception e){
       }
    }
   public static  void matriz (String vx, String vy, String vz){
      int Vx= Integer.parseInt(vx);
      int Vy= Integer.parseInt(vy);
     Orden[Vx][Vy]=vz;
   }
```

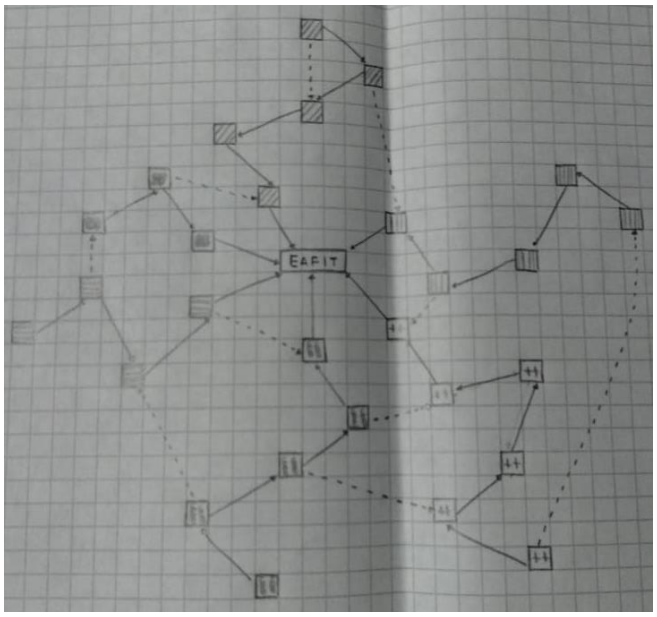

Figura 6. Gráfica del algoritmo.

5. Carpool EAFIT

Nos presentaron un problema, el cual lleva el nombre de Carpooling, que consiste en bajar el número de carros situados en los departamentos tomando un número de empleados que usan sus automóviles individualmente, en este caso se toma como punto de llegada EAFIT y cómo podemos evidenciar, se están desperdiciando 4 puestos los cuales pueden ser ocupados por otros empleados y disminuir el número de carros que están en movimiento por la ciudad.

Nos proporcionan el número de empleados, con sus distancias con respecto a EAFIT (En tiempo), la distancia con respecto a todos los otros empleados y un porcentaje de aumento del tiempo con respecto al que tenía antes yendo directamente a EAFIT.

Este problema es importante de solucionar ya que necesitamos de un medio ambiente sostenible y disminuir la pérdida de tiempo que se genera por el alto número de trancones.

La solución planteada a este problema es seleccionar a los empleados más lejanos para que estos recojan a los más cercanos, sin variar en gran cantidad su tiempo.

Como resultados para unos 205 empleados se logró disminuir en 127 carros para tener un total de 78 circulando.

En conclusión, no hay solo una forma de calcular dicho número de carros, visualizamos que nuestra solución no es la más optima ya que usamos un algoritmo voraz pero igualmente es muy funcional ya que se disminuyeron en gran cantidad el número de automóviles.

5.1 Estructura de datos

	1	2	3	4
1	0 JUAN	17	6	15
2	17 LUIS	0	19	17
3	6 CAMILO	19	0	10
4	15 MARTIN	17	10	0

Figura 7. Matriz en la cual se guarda la distancia en tiempo de un punto a otro.

5.2 Operaciones de la estructura de datos

Buscamos los mayores, los cuales eran los primeros carros.

#	0	1	2	3	4	5	6	7	8	9	10
0	0	17	6	15	15	46	33	28	24	27	60
1	17	0	19	17	15	33	17	11	11	11	47
2	6	19	0	10	20	51	32	23	29	23	65
3	15	17	10	0	18	50	31	13	28	13	63
4	15	15	20	18	0	32	31	25	15	25	45
5	46	23	51	50	32	0	20	43	23	43	17
6	33	17	32	31	31	20	0	24	17	24	33
7	28	11	23	13	25	43	24	0	21	1	57
8	24	11	29	28	15	23	17	21	0	20	36
9	27	11	23	13	25	43	24	1	20	0	46
10	60	47	65	63	45	17	33	57	36	56	0

Tabla 2. Estructura de datos con las operaciones.

Vamos al número del nodo seleccionado como primer carro y luego buscamos el más cercano.

5.3 Criterios de diseño de la estructura de datos

La estructura de datos implementada en este problema fue la matriz de adyacencia, ya que se nos presenta un problema en el cual todos los nodos están conectados con todos entonces para ello en las celdas de la matriz guardamos los tiempos de cada nodo con el adyacente.

5.4 Análisis de Complejidad

La complejidad para las operaciones sobre la matriz será de O(n), ya que hacemos búsquedas de valores sobre una fila en específico el cual disminuye su tiempo de búsqueda ya que lo hacemos sobre la misma matriz, pero organizada.

5.5 Cálculo de la complejidad del algoritmo

Futuras mejoras del análisis de complejidades.

Llenar matriz con la lectura del archivo	O(n)
Llenar arreglo de los más lejanos	O(n)
Empezar a llenar los cupos de los carros	O(k*i*j)
Buscar el menor de los adyacentes y EAFIT	O(n)
Buscar el índice de los nodos actuales	O(n)
El mejor de los casos	O(n)
Peor de los casos	O(n^2)

Tabla 3. Complejidad de las operaciones para el peor de los casos.

5.6 Criterios de diseño del algoritmo

Se hizo un algoritmo con una estructura la cual es una matriz donde se guardan las distancias de los nodos. Se hace también un recorrido voraz, ya en algoritmo se toman estos pasos principalmente:

1. Sacar el mínimo de carros y crear un arreglo de dicho tamaño en el cual se va a guardar la distancia de mayor a menor con respecto a EAFIT.

2. Comparar que la distancia del nodo adyacente sea menor que la del nodo actual a EAFIT (para que no nos alejemos de EAFIT y el p no se vea afectado) para luego llenar cada carro con el nodo más cercano teniendo en cuenta de que la distancia del nodo actual al nodo adyacente y de este a EAFIT no supere el p.

3. Se crea otra Matriz llena de falsos, luego cuando se visite un nodo a este se le asigna un valor verdadero y si al final hay nodos que quedaron con un false estos se sumarán al número de carros ya que irán directamente a EAFIT.

El algoritmo utiliza un recorrido de forma voraz aun así no sea el más eficiente, pero si el que nos de él menor número de carros, en algunos casos como lo sería el backtraking, este nos ahorra mucho tiempo de ejecución, ya que aun así es muy optimo y eficaz y no necesitamos ver todos los caminos posibles y seleccionar el mejor, si no siempre ir seleccionando directamente el adyacente más cercano que cumpla las condiciones que le establecimos.

.

6. Conclusiones

Es importante hacer un buen análisis del algoritmo a desarrollar, puesto que cosas como la buena elección de la estructura de datos o de la forma de recorrer el grafo pueden afectar mucho a la memoria, tiempo de ejecución, resultados, es decir, en la eficaz del programa.

Durante el desarrollo del algoritmo se presentaron muchos problemas, tales como, el manejo de la estructura de datos, en este caso la matriz, la utilización de los datos, se presentaban resultados no esperados e ilógicos, pero finalmente se pudo lidiar con todos estos problemas y llegar a lo que consideramos una buena solución.

6.1 Trabajos futuros

Nos gustaría encontrar la forma de aumentar la eficacia del programa teniendo como objetivo reducir un poco más el número de carros que se tienen que utilizar en todos los recorridos.

También creemos pertinente revisar de nuevo todo el algoritmo para realizar la creación de nuevos métodos para facilitarle el entendimiento del algoritmo a otro desarrollador.

REFERENCIAS

1. Núñez S., R., Núñez V., J., Paluzo H., E. y Salguero Q., E. (2016). *Jugueteando con Grafos*. [online] Fisem.org. Disponible en: http://www.fisem.org/www/union/revistas/2016/46/10_21-401-1-ED.pdf. 4-6.

2. Miller, B., Ranum, D. and College, L. (2019). *Solución de problemas con algoritmos y estructuras de datos.* [online] Interactivepython.org. Disponible en: http://interactivepython.org/runestone/static/pythoned/Graphs/ElProblemaDeLaGiraDelCaballo.html.

3. Mate.dm.uba.ar. (2007). [online] Disponible en: http://mate.dm.uba.ar/%7Espuddu/teo_de_grafos/grafo1.doc.

4. Núñez S., R., Núñez V., J., Paluzo H., E. y Salguero Q., E. (2016). *Jugueteando con Grafos*. [online] Fisem.org. Disponible en: http://www.fisem.org/www/union/revistas/2016/46/10_21-401-1-ED.pdf. 10-11.