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

| Juan Felipe Ortiz Salgado  Universidad EAFIT  Colombia  jfortizs@eafit.edu.co | Samuel Meneses Diaz  Universidad EAFIT  Colombia  smenesesd@eafit.edu.co | Mauricio Toro  Universidad Eafit  Colombia  mtorobe@eafit.edu.co |
| --- | --- | --- |

**NOTA DEL DOCENTE: Para ampliar información sobre los requerimientos aquí descritos, consulten la “*Guía para la realización del Proyecto Final de Estructura de Datos 2”* que se entrega. Al final: 1. Borrar este texto escrito en rojo, 2. Adecuar los espacios de los textos, 3. Cambiar el color de los textos a negro. Consideren además que:**

**Textos en negro** = Es todo lo que deben hacer en la entrega 1

**Textos en Verde** = Es todo lo que deben hacer en la Entrega 2

Textos en violeta = Es todo lo que deben hacer en la entrega 3

# **RESUMEN**

## Considerando nuevas estrategias para encontrar la ruta más corta para una flota de vehículos que necesita entregar un inventario a un destino específico, traería beneficios en la movilización de vehículos. Sin embargo, es difícil lograr ese objetivo porque hay diferentes variantes de este problema. Cada uno de ellos tiene una restricción específica. Como capacidad de carga limitada, horas restringidas, número insuficiente de coches y algunas otras restricciones, que hacen que muchos algoritmos se hagan para satisfacer esas necesidades. En este caso, el retroceso puede ser una buena solución al problema de buscar la mejor ruta y eliminar lo menos factible.

## **Palabras clave**

|  |
| --- |
| Modelado, Algoritmos, Clark-Wright, Método de Ahorros |

# **INTRODUCCIÓN**

Los vehículos eléctricos han sido una nueva alternativa para reducir la contaminación atmosférica de los coches de gasolina o diésel. La electricidad desempeña un papel importante y representa cero emisiones que mejoran al medio ambiente. Sin embargo, el uso de vehículos eléctricos para la carga y para el transporte de pasajeros tiene una limitación: el alcance de conducción es limitado y el tiempo de carga de la batería es relativamente largo.

# **PROBLEMA**

El problema por resolver consiste en diseñar un algoritmo para encontrar rutas óptimas para un grupo de vehículos eléctricos para entregar mercancía a un conjunto de clientes. De esta manera, el tiempo que se tarda en visitar a cada cliente, recargar la batería y terminar toda la ruta se minimizará.

## **3. TRABAJOS RELACIONADOS**

**3.1** **The Travelling Salesman (TSP)**

El vendedor viajero consiste en encontrar la ruta más corta para que una persona complete una tarea dado un grupo específico de destinos. La dificultad de este problema se produce al trabajar con muchos lugares para visitar porque el algoritmo debe estar a cargo para verificar la menor distancia o costo para cada ruta. El famoso problema tiene diferentes soluciones. Uno de ellos es usando la Fuerza Bruta. Este método encuentra la mejor ruta comparando todas las permutaciones posibles de rutas para elegir la solución única más corta. En otras palabras, calcula el tiempo que se tarda en visitar cada distancia y, finalmente, elige el tiempo más corto.

## **3.2 Random walk algorithm**

El problema en este algoritmo es encontrar, después de un tiempo fijo, la función de distribución de probabilidad de la distancia del punto desde el origen. Así que la solución es el algoritmo de caminata aleatoria, que es un proceso para determinar la ubicación probable de un punto sujeto a movimientos aleatorios, dadas las probabilidades de mover alguna distancia en alguna dirección.

## **3.3 Profitable Vehicle Routing Problem with Multiple Trips**

## El problema de ruteo de vehículos (VRP) relaciona dos variantes que son el VRP rentable y el VRP con viajes múltiples. En relación con la rentabilidad, consiste en las limitaciones para atender a un grupo de clientes debido a la escasez presupuestaria o por la insuficiencia de la oferta. Por otro lado, en relación con los múltiples viajes, significa que una flota de vehículos limitada tiene que realizar varias rutas con un horario estricto. El problema se solucionó mediante el uso de dos algoritmos basados en tres matrices: algoritmo de escalada en colina y algoritmo de descenso de vecindad variable.

**3.4 Problema de enrutamiento de vehículos capacitados (CVRP)**

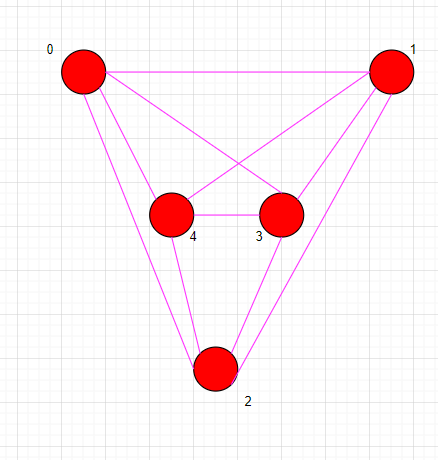
Consiste en encontrar una ruta que no requiera un consumo excesivo de tiempo y recursos. La dificultad se da porque hay una capacidad de carga limitada y debe volver al almacén principal. Este problema tiene diferentes soluciones, pero una de las más famosas son las técnicas de metaheurística, que se basa en el uso de los parámetros proporcionados por el usuario con el fin de encontrar un resultado eficiente. En este caso, se debe realizar una búsqueda en profundidad, encontrando las rutas que no requieren demasiado tiempo, es decir, el retroceso debe ser implementado.

## **4. TÍTULO DE LA PRIMERA SOLUCIÖN DISEÑADA**

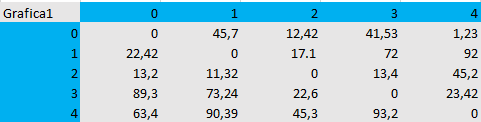
## A continuación, explicamos la estructura de datos y el algoritmo.

## **4.1 Estructura de datos**

## Esta estructura es una matriz de adyacencia en todo este grafico es un mapa de X ciudad y sus nodos distribuidos en ella, y las líneas representan la distancia entre cada nodo. Es una estructura sencilla pero se le pueden implementar estaciones de carga, parqueadero de los carros y más cosas.



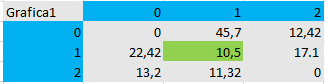
## **Gráfica 1:** lo que se puede ver la distancia entre dos nodos específicos



## **4.2 Operaciones de la estructura de datos**

## Diseñen las operaciones de la estructura de datos para solucionar el problema eficientemente. Incluyan una imagen explicando cada operación

* **addArc:** Es el método para agregar un nuevo arc, es decir una nueva conexión entre dos vértices. addArc recibe el valor del nodo de origen, el valor del nodo de destino y el peso de estos dos.



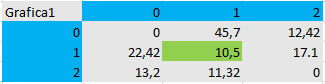
addArc(1, 1, 10.5)

1: nodoOrigen

1: nodoDestino

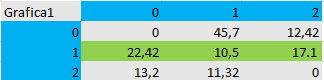
10.5: Peso

* **getWeight:** Este es el método para obtener el peso o la distancia entre el nodoOrigen y el nodoDestino



getWeight(1,1) = 10.5

* **getSuccessors:** Este método entrega la lista de los sucesores de un vertice. Y el método recibe el numero de un nodo.

****

getSuccessors(1) = 0,1,2

**Gráfica 2:** Imagen de una operación de borrado de una lista encadenada

## **4.3 Criterios de diseño de la estructura de datos**

Para este proyecto se escogió la matriz de adyacencia, ya que pensamos en lo eficaz que puede ser ya que el tiempo de ejecución de O(1).

También esta matriz ayudo a representar un grafo finito no dirigido, el cual fue mostrado en el punto 4.1 para mostrar los vértices y los puntos de la ciudad. Los elementos de la matriz nos dan a entender la ciudad, los puntos si están separados o no, a que distancia están. Es decir que nos basamos en lo eficaz que puede ser la ejecución de este.

**4.4 Análisis de Complejidad**

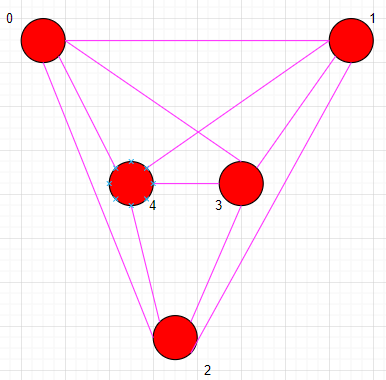
Calculen la complejidad de las operaciones de la estructura de datos para el peor de los casos. Vean un ejemplo para reportarla:

|  |  |
| --- | --- |
| **Método** | **Complejidad** |
| addArc | O(1) |
| getWeight | O(1) |
| getSuccessors | O(n) |

**Nota:** La letra n representa el número de nodos.

**Tabla 1:** Tabla para reportar la complejidad

**4.5 Algoritmo**



****

**Gráfica 3:**

**4.6** **Cálculo de la complejidad del algoritmo**

|  |  |
| --- | --- |
| **Sub problema** | **Complejidad** |
| Crear la matriz de ahorros | O(N2) |
| Añadir un camino a la lista de registro | O(1) |
| Método para validar si se puede realizar el camino  Método para eliminar los ahorros de los clientes visitados  Método calculador de ruta | O(N)  O(N2)  O(N3) |
| **Complejidad Total** | O(N3) |

**Tabla 2:** complejidad de cada uno de los subproblemas que componen el algoritmo. N número de clientes a visitar y M numero de estaciones de carga.

**4.7 Criterios de diseño del algoritmo**

El algoritmo que se implementó se basó en el algoritmo Clarke-Wright. Se conoce el algoritmo de ahorro porque funciona calculando el guardar entre dos rutas para construir caminos diferentes, con el cual se analiza la posibilidad de combinar dos trazados en una sola ruta, y este tiene altas posibilidades de ser la óptima. En este caso, se preocupa más por encontrar una ruta eficiente bajo el tiempo y las restricciones de la batería por lo que garantiza la mejor solución, es eficiente para resolver el problema sin exceder el tiempo máximo de respuesta. Además, durante las diferentes operaciones se utilizaron matrices que permitían buscar en O (1).

Teniendo en cuenta que el objetivo principal del algoritmo consistía en priorizar el tiempo en que este lograba generar el ruteo de vehículos, se decidió priorizar el tiempo.

**4.8 Tiempos de Ejecución**

Calculen, (I) el tiempo de ejecución y (II) la memoria usada del algoritmo, para el Conjunto de Datos que está en el ZIP:

## Tomen 100 veces el tiempo de ejecución y memoria de ejecución, para cada conjunto de datos

## 

|  |  |  |  |
| --- | --- | --- | --- |
|  | ***Conjunto de Datos 1*** | ***Conjunto de Datos 2*** | ***Conjunto de Datos 3*** |
| *Mejor caso* | 1 sg | 0.8 sg | 1.3 sg |
| *Caso promedio* | 3.5 sg | 3.4 sg | 3 sg |
| *Peor caso* | 6 sg | 5.5 sg | 6.3 sg |

## **Tabla 3:** Tiempos de ejecución del algoritmo con diferentes conjuntos de datos, Datos 1 con 320 clientes, datos 2 con 345 clientes, datos 3 con 359 clientes.

## **4.9 Memoria**

Mencionar la memoria que consume el programa para varios ejemplos

|  |  |  |  |
| --- | --- | --- | --- |
|  | ***Conjunto de Datos 1*** | ***Conjunto de Datos 2*** | ***...Conjunto de Datos n*** |
| **Consumo de memoria** | 0.8 MB | 0.8 MB | 0.8 MB |

## **Tabla 4:** Consumo de memoria del algoritmo con diferentes conjuntos de datos, Datos 1 con 320 clientes, datos 2 con 345 clientes, datos 3 con 359 clientes.

## **4.10 Análisis de los resultados**

Expliquen los resultados obtenidos. Hagan una gráfica con los datos obtenidos, como, por ejemplo:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Memoria(KB)** | **Tiempo de ejecución** | **Número de clientes** | **Número de vehículos** | **Tiempo total** |
| (1) | 755 | 14 s | 320 | 33 | 159.52 |
| (2) | 826 | 9 s | 320 | 27 | 95.97 |
| (3) | 825 | 11 s | 320 | 29 | 119.29 |
| (4) | 826 | 12 s | 320 | 33 | 159.52 |
| (5) | 825 | 16 s | 320 | 27 | 95.97 |
| (6) | 825 | 14 s | 320 | 29 | 119.29 |
| (7) | 825 | 13 s | 320 | 33 | 159.52 |
| (8) | 825 | 10 s | 320 | 27 | 95.97 |
| (9) | 825 | 11 s | 320 | 29 | 119.29 |
| (10) | 826 | 14 s | 320 | 33 | 159.52 |
| (11) | 825 | 16 s | 320 | 27 | 95.97 |
| (12) | 826 | 14 s | 320 | 29 | 119.29 |

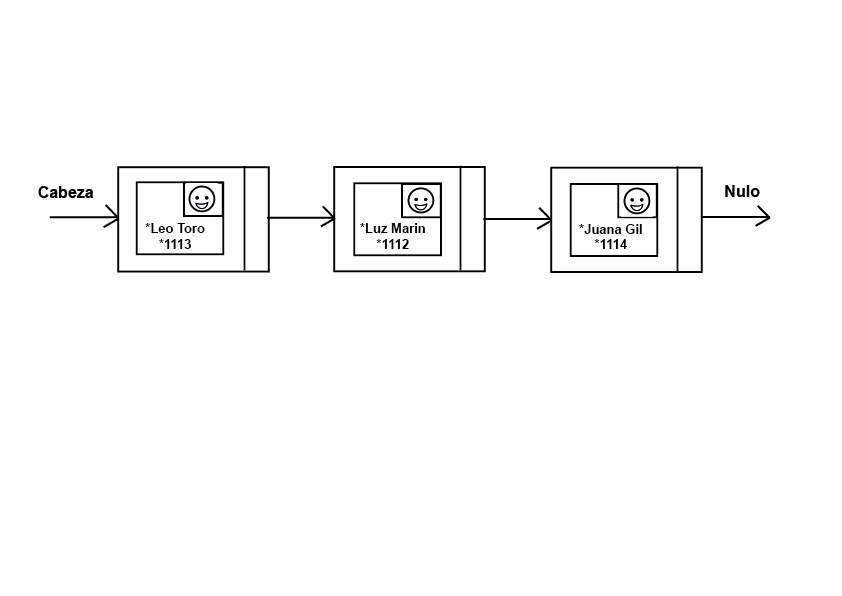
**Tabla 5:** Análisis de los resultados obtenidos de la ejecución del algoritmo.

**5. TÍTULO DE LA SOLUCIÓN FINAL DISEÑADA**

## A continuación, explicamos la estructura de datos y el algoritmo.

## **5.1 Estructura de datos**

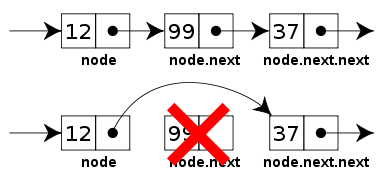
## Diseñen la estructura de datos para resolver el problema y grafíquenla. No usar gráficas extraídas de internet



## **Gráfica 4:** Lista simplemente encadenada de personas. Una persona es una clase que contiene nombre, cédula y foto

## **5.2 Operaciones de la estructura de datos**

## Diseñen las operaciones de la estructura de datos para solucionar el problema eficientemente. Incluyan una imagen explicando cada operación



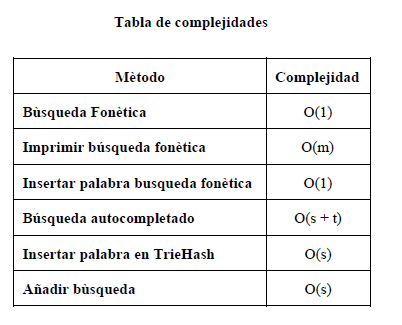
**Gráfica 5:** Imagen de una operación de borrado de una lista encadenada

## **5.3 Criterios de diseño de la estructura de datos**

Expliquen con criterios objetivos, por qué diseñaron así la estructura de datos. Criterios objetivos son, por ejemplo, la eficiencia en tiempo y memoria. Criterios no objetivos y que rebajan la nota son: “me enfermé”, “fue la primera que encontré”, “la hice el último día”, etc. Recuerden: este es el numeral que más vale en la evaluación con 40%

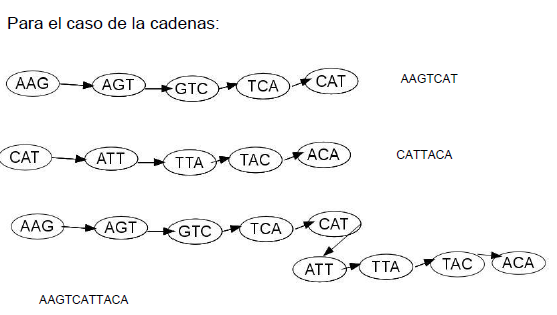
**5.4 Análisis de Complejidad**

Calculen la complejidad de las operaciones de la estructura de datos para el peor de los casos. Vean un ejemplo para reportarla:



**Tabla 6:** Tabla para reportar la complejidad

**5.5 Algoritmo**

Diseñen el algoritmo para resolver el problema y grafíquenlo. No usen gráficas extraídas de internet

**Gráfica 6:** Paso a paso cómo se ensamblan fragmentos de ADN utilizando los grafos de *Bruijn*.

**5.6** **Cálculo de la complejidad del algoritmo**

Calculen la complejidad del algoritmo para el peor de los casos, el mejor de los casos y el caso promedio

|  |  |
| --- | --- |
| **Sub problema** | **Complejidad** |
| Crear el grafo de *Bruijn* con las secuencias | O(N) |
| Actualizar el grafo de *Bruijn* con las secuencias | O(A.N2) |
| Encontrar los genes | O(V) |
| **Complejidad Total** | O(A.N2 + V) |

**Tabla 7:** complejidad de cada uno de los sub problemas que componen el algoritmo. Sea A la longitud de una secuencia de ADN, N el número de secuencias de ADN, y V el número de K-meros diferentes que se obtienen de las secuencias de ADN.

**5.7 Criterios de diseño del algoritmo**

Expliquen por qué diseñaron así el algoritmo. Usen criterios objetivos. Criterios objetivos son, por ejemplo, la eficiencia en tiempo y memoria. Criterios no objetivos y que rebajan la nota son: “me enfermé”, “fue la primera que encontré”, “la hice el último día”, etc. Recuerden: este es el numeral que más vale en la evaluación con 40%

.

**5.8 Tiempos de Ejecución**

Calculen, (I) el tiempo de ejecución y (II) la memoria usada del algoritmo, para el Conjunto de Datos que está en el ZIP:

## Tomen 100 veces el tiempo de ejecución y memoria de ejecución, para cada conjunto de datos

## 

|  |  |  |  |
| --- | --- | --- | --- |
|  | ***Conjunto de Datos 1*** | ***Conjunto de Datos 2*** | ***...Conjunto de Datos n*** |
| *Mejor caso* | 10 sg | 20 sg | 5 sg |
| *Caso promedio* | 12 sg | 10 sg | 35 sg |
| *Peor caso* | 15 sg | 21 sg | 35 sg |

## **Tabla 8:** Tiempos de ejecución del algoritmo con diferentes conjuntos de datos

## Para medir la memoria que consume un programa, se utilizan generadores de perfiles (en Inglés, profilers). Uno muy bueno para Java es VisualVM, desarrollado por Oracle,[http://docs.oracle.com/javase/7/docs/technotes/guides/visualvm/profiler.html](about:blank) No dejen de usarlo en sus proyectos y en la vida. Para usarlo hay que generar un .jar que es como un ejecutable de Java. En Netbeans "martillo con escoba" y en BlueJ "archivo, generar .jar".

## **5.9 Memoria**

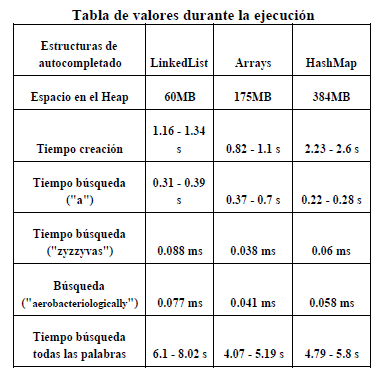
Mencionar la memoria que consume el programa para varios ejemplos

|  |  |  |  |
| --- | --- | --- | --- |
|  | ***Conjunto de Datos 1*** | ***Conjunto de Datos 2*** | ***...Conjunto de Datos n*** |
| **Consumo de memoria** | 10 MB | 20 MB | 5 MB |

## **Tabla 9:** Consumo de memoria del algoritmo con diferentes conjuntos de datos

## **5.10 Análisis de los resultados**

Expliquen los resultados obtenidos. Hagan una gráfica con los datos obtenidos, como por ejemplo:



**Tabla 10:** Análisis de los resultados obtenidos con la implementación del algoritmo

**6. CONCLUSIONES**

Para escribirlas, procedan de la siguiente forma: 1. En un párrafo escriban un resumen de lo más importante que hablaron en el reporte. 2. En otro expliquen los resultados más importantes, por ejemplo, los que se obtuvieron con la solución final. 3.Luego, comparen la primera solución que hicieron con los trabajos relacionados y la solución final. 4. Por último, expliquen los trabajos futuros para una posible continuación de este Proyecto. Aquí también pueden mencionar los problemas que tuvieron durante el desarrollo del proyecto

**6.1 Trabajos futuros**

Respondan ¿Qué les gustaría mejorar en el futuro? ¿Qué les gustaría mejor al algoritmo, estructura de datos, implementación?

# **AGRADECIMIENTOS**

Identifiquen el tipo de agradecimiento que van a escribir: para una persona o para una institución. Luego, escríbanlo de acuerdo al idioma y tengan en cuenta que: 1. El nombre del docente no va porque él es autor. 2. Tampoco sitios de internet ni autores de artículo leídos con quienes no se han contactado. 3. Los nombres que sí van son quienes ayudaron, compañeros del curso o docentes de otros cursos.

Aquí un ejemplo en inglés: This research was supported/partially supported by [Name of Foundation, Grant maker, Donor].

We thank for assistance with [particular technique, methodology] to [Name Surname, position, institution name] for comments that greatly improved the manuscript.

# **REFERENCIAS**

[1] *Travelling Salesman Problem | Set 1 (Naive and Dynamic Programming) - GeeksforGeeks*. *GeeksforGeeks.com*, 2018.

Disponible en: [https://www.geeksforgeeks.org/travelling-salesman-problem-set-1/](about:blank)

[consultado 13 February 2021].