Algoritmo para encontrar rutas óptimas para vehículos eléctricos

Tomas Atehortua Ceferino
Universidad Eafit
Colombia
tatehortuc@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
El objetivo de este proyecto es diseñar un algoritmo que encuentre rutas óptimas a los vehículos eléctricos, en este caso específico para hacer más eficiente la forma de distribuir mercancías. Esto con el fin de solucionar el problema que se presenta en las baterías de los coches por tener un rango de durabilidad limitado y un tiempo de carga muy largo.¿Cuál es la solución?, ¿cuáles los resultados? y, ¿Cuáles las conclusiones? Utilizar máximo 200 palabras.

Palabras clave
Estas son palabras claves que ustedes consideran apropiadas para indexar el informe PDF en bibliotecas o bases de datos

Palabras clave de la clasificación de la ACM

Solo las que están en esta lista de ACM, en http://bit.ly/2oVE5 2i No pueden ser inventadas por ustedes.

Ejemplo: Theory of computation → Design and analysis of algorithms → Graph algorithms analysis → Shortest paths

1. INTRODUCCIÓN

En el siguiente informe se pretende solucionar la problemática que tienen las empresas a la hora de entregar productos con camiones de carga eléctricos, para que puedan entregarlos de la manera más eficiente, teniendo en cuenta las horas de trabajo, número de autos y la vida útil de los Batería de coche. Resolver este problema permitiría mejorar las entregas y asegurar que las empresas utilizarán coches eléctricos, incluso con la necesidad de cargarlos, permitiendo una reducción de las emisiones de carbono que liberan estos camiones de carga.

2. PROBLEMA

El problema radica en la baja eficiencia que tienen las baterías de los autos eléctricos para distribuir la mercadería, pues demoran mucho en hacer los recorridos debido a que la batería se descarga constantemente. Esto tiene un alto impacto en la sociedad porque si no se soluciona, los transportistas se verían obligados a utilizar otros vehículos más dañinos para el planeta, por eso para evitar que eso suceda es necesario solucionar este problema.**3. TRABAJOS RELACIONADOS**

3.1 Algoritmo de Dijkstra

El problema de la ruta más corta es una forma de encontrar las distancias más cortas posibles entre dos vértices en un gráfico de manera que se minimice la suma de los pesos de sus aristas constituyentes, por esa razón el algoritmo de Dijkstra puede ayudarnos porque es un algoritmo que podemos usar para encontrar las distancias más cortas o los costos mínimos según lo que se represente en un gráfico. A través de los siguientes pasos:

-Comience en el vértice final marcándolo con una distancia de 0

-Identificar todos los vértices que están conectados al vértice actual con una arista

-Etiquetar el vértice actual como visitado colocando una X sobre él

-De los vértices que debes marcar, busca el que tenga la marca más pequeña y conviértelo en tu vértice actual.

-Una vez que haya etiquetado el vértice inicial como visitado, deténgase. [1]

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 al origen. Entonces, su 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 moverse a cierta distancia en alguna dirección. [2]

3.3 The traveling salesman problem

El traveling salesman problem consiste en encontrar el camino más corto que visita cada ciudad posible y regresa a la ciudad original. [3]

El problema tiene múltiples soluciones y es un problema NP-difícil ya que el número de posibilidades de solución cuando se da un gran número de ciudades es inmenso. Por eso no se recomienda solucionarlo mediante Fuerza bruta sino buscar un método de optimización. Las soluciones a este tipo de problemas comienzan a resolverse mediante algoritmos tipo enjambre inspirados en las abejas, donde la variable viajera (el vendedor en este problema) tiene cierta independencia, pero comunicación con la colmena. [4]

3.4 Floyd-Warshall algorithm

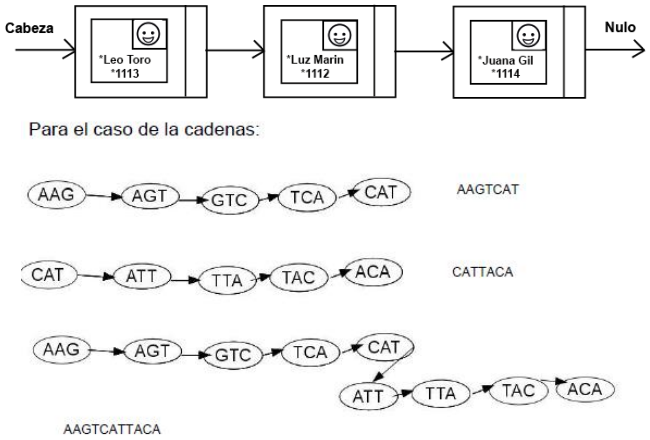
El Floyd-Warshall algorithm es un problema que busca encontrar la ruta más corta entre cada par de vértices en un gráfico dirigido ponderado por bordes. La condición para que el problema funcione es que debe ser como se mencionó antes de una gráfica ponderada, puede ser dirigida o no dirigida, pero no funciona con ciclos negativos (La suma de las aristas en el ciclo es negativa). Dicho algoritmo puede ser de ayuda, ya que para un gráfico hecho de matrices, encontraría la ruta más rápida o más barata desde cada vértice del gráfico. La forma en que funciona es que para cada vértice intenta todas las rutas posibles incluyendo los vértices por los que pasa y si es más pequeño que el especificado se reemplaza en la matriz. [5] [6]

4. TÍTULO DE LA PRIMERA SOLUCIÖN DISEÑADA

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

4.1 Estructura de datos

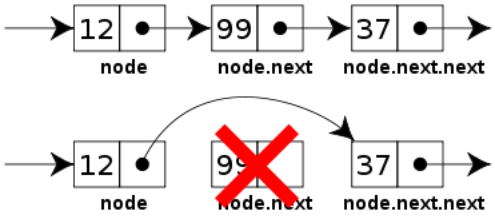
Diseñen la estructura de datos para resolver el problema y grafíquenla. No usar gráficas extraídas de internet



Gráfica 1: Lista simplemente encadenada de personas. Una persona es una clase que contiene nombre, cédula y foto

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



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

4.3 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 evaulación con 40%

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:

Mètodo	Complejidad
Bùsqueda Fonètica	O(1)
Imprimir bùsqueda fonètica	O(m)
Insertar palabra busqueda fonètica	O(1)
Bùsqueda autocompletado	O(s + t)
Insertar palabra en TrieHash	O(s)
Añadir bùsqueda	O(s)

Tabla 1: Tabla para reportar la complejidad

4.5 Algoritmo

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

Gráfica 3: Paso a paso cómo se ensamblan fragmentos de ADN utilizando los grafos de *Bruijn*.

4.6 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(S)
Actualizar el grafo de *Bruijn* con las secuencias	$O(A.V^2)$
Encontrar los genes	O(S)
Complejidad Total	$O(A.S^2 + V)$

Tabla 2: complejidad de cada uno de los sub problemas que componen el algoritmo. Sea A la longitud de una secuencia de ADN, S el número de secuencias de ADN, y V el número de K-meros diferentes que se obtienen de las secuencias de ADN.

NOTA: Sin complejidad total, el análisis no sirve de nada

4.7 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%.

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

Tabla 4: Consumo de memoria 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 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".

4.10 Análisis de los resultados

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

Tabla de valores durante la ejecución

Estructuras de autocompletado	LinkedList	Arrays	HashMap
Espacio en el Heap	60MB	175MB	384MB
Tiempo creación	1.16 - 1.34 s	0.82 - 1.1 s	2.23 - 2.6 s
Tiempo búsqueda ("a")	0.31 - 0.39 s	0.37 - 0.7 s	0.22 - 0.28 s
Tiempo búsqueda ("zyzzyvas")	0.088 ms	0.038 ms	0.06 ms
Búsqueda ("aerobacteriologically")	0.077 ms	0.041 ms	0.058 ms
Tiempo búsqueda todas las palabras	6.1 - 8.02 s	4.07 - 5.19 s	4.79 - 5.8 s

Tabla 5: Análisis de los resultados obtenidos con la implementación del algoritmo

5. TÍTULO DE LA SOLUCIÓN FINAL DISEÑADA

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

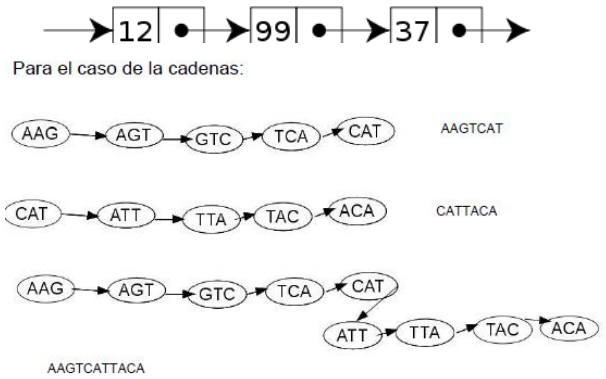
5.1 Estructura de datos

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:

Método	Complejidad
Búsqueda Fonética	O(1)
Imprimir búsqueda fonética	O(m)
Insertar palabra busqueda fonética	O(1)
Búsqueda autocompletado	O(s + t)
Insertar palabra en TrieHash	O(s)
Añadir búsqueda	O(s)

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	$O(A.N^2)$
Encontrar los genes	O(V)
Complejidad Total	$O(A.N^2 + V)$

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 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 de valores durante la ejecución

Estructuras de autocompletado	LinkedList	Arrays	HashMap
Espacio en el Heap	60MB	175MB	384MB
Tiempo creación	1.16 - 1.34 s	0.82 - 1.1 s	2.23 - 2.6 s
Tiempo búsqueda ("a")	0.31 - 0.39 s	0.37 - 0.7 s	0.22 - 0.28 s
Tiempo búsqueda ("zyzzyvas")	0.088 ms	0.038 ms	0.06 ms
Búsqueda ("aerobacteriologically")	0.077 ms	0.041 ms	0.058 ms
Tiempo búsqueda todas las palabras	6.1 - 8.02 s	4.07 - 5.19 s	4.79 - 5.8 s

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] Pennington, L., n.d. Dijkstra's Algorithm: Definition, Applications & Examples. [video] Available at: https://study.com/academy/lesson/dijkstra-s-algorithm-definition-applications-examples.html

[Accessed 14 February 2021].

[2] Encyclopedia Britannica. 2008. Random walk | mathematics and science. [online] Available at: https://www.britannica.com/science/random-walk

[Accessed 14 February 2021].

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

Available at: https://www.geeksforgeeks.org/travelling-salesman-problem-set-1/

[Accessed 15 February 2021].

[4] Nunes de Castro, L. and A. S. Masutti, T. hindawi.com, 2017. Bee-Inspired Algorithms Applied to Vehicle Routing Problems: A Survey and a Proposal. [online] Available at: https://www.hindawi.com/journals/mpe/2017/3046830/

[Accessed 15 February 2021].

[5] Floyd-Warshall Algorithm. [online] Available at: https://www.programiz.com/dsa/floyd-warshall-algorithm

[Accessed 15 February 2021].

[6] GeeksForGeeks. 2021.Floyd Warshall Algorithm | DP-16 - GeeksforGeeks. [online] Available at: https://www.geeksforgeeks.org/floyd-warshall-algorithm-dp-16/.

[Accessed 15 February 2021]