CARPOOLING

(Disminuir la congestión y la contaminación en la ciudad, transportando a varias personas en un auto que recorra la distancia más corta desde el lugar de inicio hasta la universidad)

Santiago Moreno Rave
Universidad Eafit
Colombia
smorenor@eafit.edu.co

Valeria Suarez Mejia
Universidad Eafit
Colombia
vsuarezm@eafit.edu.co

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

RESUMEN

Tanto como en la universidad Eafit como en la ciudad de Medellín se tiene un problema de contaminación por el aire y se busca que las personas compartan su vehículo particular haciendo una ruta en la que pueda llenar el cupo de su vehículo con las demás personas, ahora en la parte de algoritmos el problema que se tiene es que se pueda recorrer un grafo (la ciudad de Medellín) en el tiempo mínimo desde un nodo cualquiera de la ciudad hasta la universidad Eafit, algunos problemas relacionados son aquellos como la forma de la ciudad, sus vías y el tráfico.

Palabras clave

Carpooling, vehículo con cupo máximo, análisis de recorrido, rutas más cortas, recorrido mínimo por la ciudad.

Palabras clave de la clasificación de la ACM

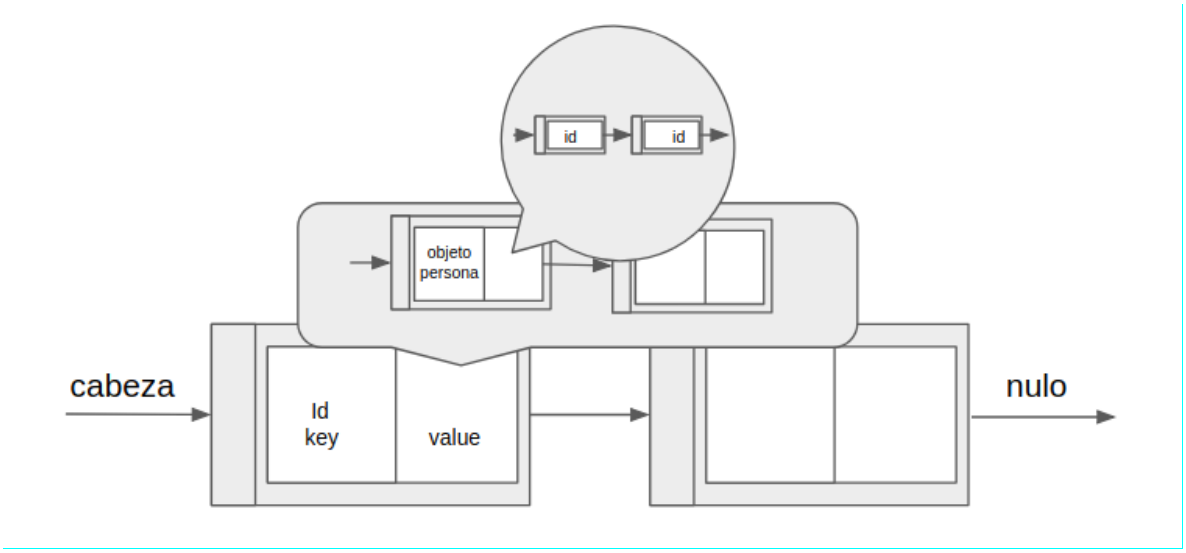
Data Structures → Graph and tree search strategies → Graph algorithms análisis → Hash-table representations [Data Storage Representations] → Linked representations [Data Storage Representations] → Shortest paths

4. TÍTULO DE LA PRIMERA SOLUCIÖN DISEÑADA

"Carpooling recoge y lleva"

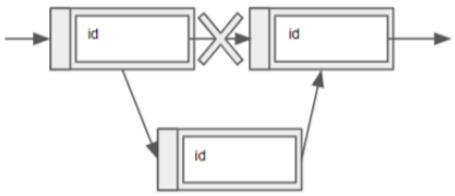
4.1 Estructura de datos

La primera versión de estructuras de datos se dividirá primero en almacenamiento de la información para ello se usará un Hashmap, la key de éste será el nodo, en el value de cada posición del HashMap, irá una LinkedList de pares, la cual contendrá en la primera posición del "par" la información personal del usuario en forma de objeto "persona", y en la otra posición del "par" estará una lista de los sucesores (el sucesor tiene como método propio obtener el peso del arco). Para la segunda parte usaremos el algoritmo o la estructura de datos de Dijkstra para el recorrido completo del grafo y encontrar el recorrido más óptimo para que una persona (el conductor) pueda llegar a x cantidad de personas, teniendo en cuenta el cupo máximo del vehículo.

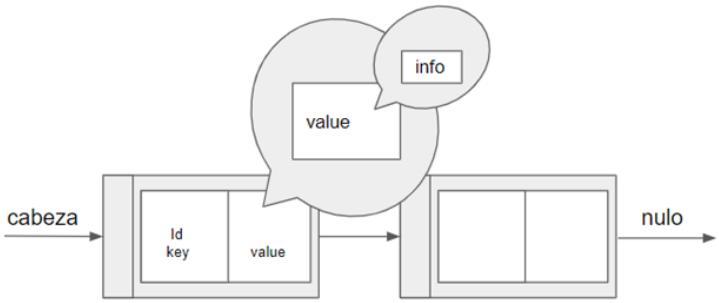


4.2 Operaciones de la estructura de datos

Add en la LinkedList:



getValue de HashMap:



4.3 Criterios de diseño de la estructura de datos

Diseñamos esta manera porque es una manera muy gráfica de representar la solución, ya que se puede interpretar como un tipo de jerarquía al momento de enlazar todas estas estructuras (se refiere a los LinkedList, HashMap, etc.). También lo diseñamos así, porque nos permite colocar o insertar elementos sin que tengan que tener algún orden es específico los nodos insertados ya que el HashMap utiliza "keys" que me permiten acceder a ellos sin sin ningún tipo de orden.

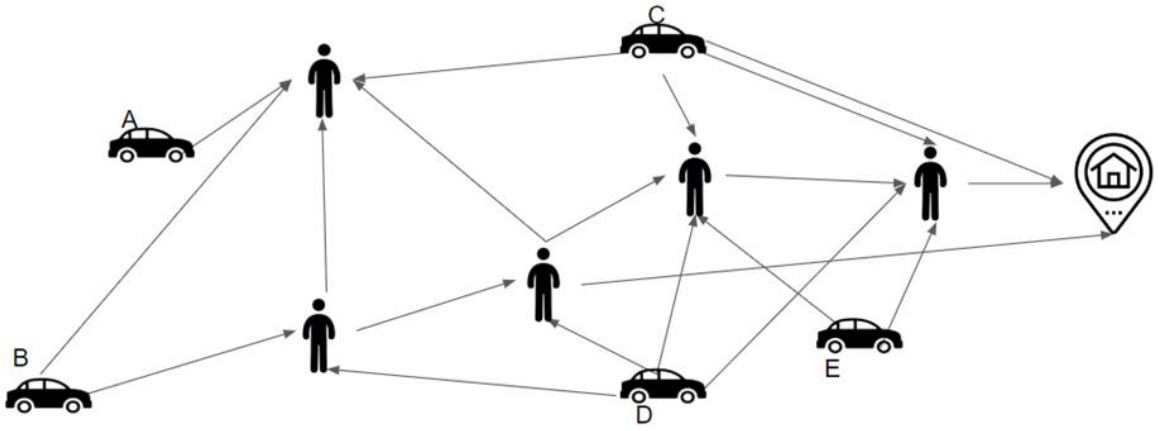
La eficiencia del recorrido de los nodos va a ser óptima, porque nos estamos asegurando de que ningún nodo se quede sin visitar, teniendo el cuenta el costo mínimo.

4.4 Análisis de Complejidad

Método	**Complejidad en el peor de los casos**	**Complejidad**
Búsqueda en HashMap	O(n)	O(1)
Búsqueda en LinkedList<Pair>	O(n)	O(n)
Remover elementos del HashMap	O(n)	O(1)
Verificar key de un HashMap	O(n)	O(1)

4.5 Algoritmo

El algoritmo se basará en hacerle un Quicksort al arreglo de los pesos de los sucesores de cada nodo, después lo que se hace es elegir la primera posición (que con el Quicksort nos aseguramos que sea el costo mínimo), luego vamos a ir al nodo sucesor elegido y se hace el mismo procedimiento hasta llegar al destino. Cada que se va a ir cambiando de nodo se va a marcar como visitado, para así tener un control de que no se revise dos veces los sucesores del mismo nodo.



4.6 Cálculo de la complejidad del algoritmo

Sub problema **Complejidad**

Crear grafo	O(n)
Recorrer	O(n^2)
Complejidad Total	O(n^2)

4.7 Criterios de diseño del algoritmo

Decidimos hacerlo de esta manera por la eficacia al recorrer el grafo y la manera en la que se manejan los datos (de las personas implicadas el carpooling).

El HashMap lo usamos porque nos disminuye la complejidad del algoritmo al momento de pasar de un nodo a otro por el hecho de que con las Key de cada posición, se va directamente al nodo.

La LinkedList Pair, la utilizamos porque nos permite almacenar dos valores, en los cuales también se pueden almacenar otro tipo de estructuras de datos, lo que nos ayuda a ser más versátiles a la hora de almacenar y relacionar los datos.

La LinkedList la utilizamos porque la complejidad de esta estructura es más baja de lo que es un arreglo normal.

4.8 Tiempos de Ejecución

	Conjunto de Datos 1	***Conjunto de Datos 2***	***.Conjunto de Datos 3***
Mejor caso	**2 ms**	**30 ms**	**62 ms**
Caso promedio	**1 ms**	**100 ms**	**10 ms**
Peor caso	**1 sg**	**2 sg**	**4 sg**

4.9 Memoria

	Conjunto de Datos 1	***Conjunto de Datos 2***	***...Conjunto de Datos n***
Consumo de memoria	123 MB	284 MB	365 MB

REFERENCIAS

1. Computing At The School, Toy Problems For The Real World, https://www.computingatschool.org.uk/data/tft/02/04Activity_Toy_Problems_Real_World.pdf

2. Chegg Study, For a given real-world problem like the one below, which, https://www.chegg.com/homework-help/questions-and-answers/problem-9-given-real-world-problem-like-one-algorithm-use-solve-would-map-input-informatio-q17272505

3. Rocío Núñez Santiago, Juan Núñez Valdés, Eduardo Paluzo Hidalgo, Elena Salguero Quirós, Jugueteando con Grafos, http://www.fisem.org/www/union/revistas/2016/46/10_21-401-1-ED.pdf