Proyecto EDA

María Paula Gonzalez Escallón: m.gonzaleze@uniandes.edu.co 202012265

Jessica Alejandra Robles Moreno: j.roblesm@uniandes.edu.co 202013355

Martin Ubaque Forero: m.ubaque@uniandes.edu.co 201923281

Grupo 8

Requerimiento 1

Estructura de datos: Max PQ con llave el número de taxis y valor, el nombre de la compañía con dicha cantidad. Puesto que se busca encontrar un top y no una compañía específica, las parejas llave-valor que tengan la misma llave se almacenarán seguidas en el array del heap y tendrán un puesto seguido en el top.

Porque se eligió esa estructura de datos:

Se escogió esta estructura de datos ya que al momento de obtener los datos del MaxPQ, la complejidad seria O(1) por el número de posiciones que escoja el usuario

Orden de complejidad:

Carga por viaje: O(2n)

Obtención de datos: O(1)

Requerimiento 2:

Estructura de datos: Un mapa, donde la llave era la fecha del viaje y el valor era un dict con 2 parejas llave-valor, la primera era un mapa donde la llave era el id del taxi y el valor era un dict con los datos necesarios para calcular los puntos de ese taxi; la segunda, era una lista con los ids de los taxis para que más adelante fuese más fácil comparar si ya estaba ese taxi. Y una lista de las fechas para posteriormente hacer un árbol ordenado cuya llave es la fecha y el valor es una lista que contiene los ids de los taxis ordenados según sus puntos en la fecha, esta lista fue creada a partir de un MaxHeap.

Porque se eligió esa estructura de datos:

Elegimos la estructura de datos del mapa para poder acceder a los datos de forma más fácil, con ayuda de las listas, para posterior mente hacer un árbol ordenado de las fechas y ahí mismo decidimos hacer un MaxHeap para tener una lista ordenada por los puntos. Y así al momento de obtener los datos fuesen de la menor complejidad posible.

Orden de complejidad:

F = cantidad de fechas

N = viajes

T = taxis promedio en una fecha

F2 = cantidad de fechas en el rango

T2 = total de taxis en el rango de fechas escogido

Carga: O(F(log T) + N)

Obtención de datos de una fecha: O(1)

Obtención de datos de un rango de fechas: O(F2(T)+logT2)

Requerimiento 3:

Estructura de datos: un array list, donde cada posición representa un rango de 15 minutos, y el valor en cada posición va a ser un grafo donde cada vértice sea una “community area" y el peso de cada vértice sea la duración de cada viaje

Porque se eligió esa estructura de datos:

Elegimos esta estructura de datos para separar los viajes dependiendo del horario en el que se hicieron. No escogimos un map para esta labor ya que se utilizaría más espacio en memoria y al hacer un array list donde cada posición es un rango horario, nos hace más fácil la iteración al momento de hacer las búsquedas.

Pusimos un grafo dentro de cada posición del ArrayList para poder guardar los viajes hechos en ese rango horario y tener documentados su origen, destino y duración. También al hacer un grafo nos permite hacer el recorrido de Dijkstra para encontrar la ruta más corta entre dos vértices y su tiempo, dos elementos necesarios para solucionar el problema a resolver.

Orden de complejidad:

E= tamaño de vértices en un grafo

V= tamaño de arcos en un grafo

H = cantidad de posiciones que representan el rango de horas

Carga por viaje: O(1)

Obtención de datos: O(H(E log V))