Requerimiento 1

¿Qué TAD utilizaron en la solución del requerimiento?

Para este requerimiento se usó una tabla de hash cuya llave eran las compañías y los valores eran una lista y una tabla de hash, en la lista se guardaba todos los viajes de cada empresa, y en la tabla de hash se guardaban los taxis de cada empresa.

Para poder dar la solución se utilizaron dos indexmaxpq, una que guardaba los servicios totales de cada compañía, y otra para los taxis afiliados totales.

¿Por qué eligieron esa estructura de datos?

Se utilizaron estas estructuras de datos ya que lo que pedían era lo más usado, pero este valor cambia dependiendo de cada iteración , por lo que se usó una indexmaxpq. Por otro lado, se utilizaron tablas de hash para maximizar la velocidad de respuesta

¿Cuál es la complejidad estimada del algoritmo implementado?

En la carga de datos se va actualizando la maxpq, cada llave es una compañía y el valor en el caso de servicios era el size de la lista en la que se guardaban todos los servicios y en el caso de cantidad de taxis afiliados era el size de la tabla de hash.

Al ejecutar el código de extraer un elemento de la maxpq era o(log(n)) , ya que la maxpq al sacar un elemento tiene que reorganizar toda la estructura.

Requerimiento 2

¿Qué TAD utilizaron en la solución del requerimiento?

Para este requerimiento se utilizó un árbol, cuya llave era una fecha y su valor era una indexmaxpq, esta tenía como valor el id del taxi y su valor era los puntos dados por la función alfa.

¿Por qué eligieron esa estructura de datos?

Se escogió esta estructura ya que se pedían rangos de tiempo por lo que usar arboles es lo más adecuado. Por otro lado, se escogió una indexmaxpq ya que los puntos de cada carro con cada iteración podían cambiar, además de que como se pedía el mayor, una cola de prioridad orientada era mayor era lo idóneo

¿Cuál es la complejidad estimada del algoritmo implementado?

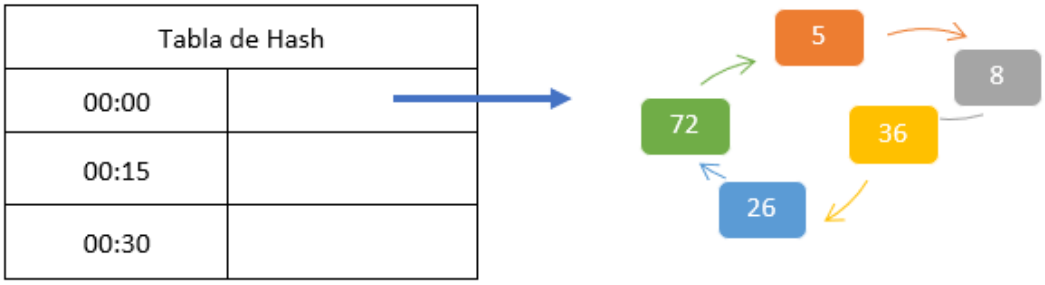
La complejidad para una fecha específica es 2log(n), ya que llegar a la llave es de orden log(n) y sacar un elemento y organizar la maxpq es de orden log(n).

La complejidad para un rango de fechas es nlog(n)+n+log(n), ya que en el peor caso sacar todos los valores del árbol es de orden log(n), eso retornaba una lista y lo que se hacía era encontrar el mayor entre todas las max de las maxpq ,por lo que en el peor caso se recorría toda la lista, eso es de orden (n), finalmente remover ese max de una max pq era de orden log(n) ya que toca reorganizar esta estructura. Finalmente, al sumar estas complejidades queda nlog(n)+n+log(n)

Requerimiento 3

¿Qué TAD utilizaron en la solución del requerimiento?

Para la implementación de este requerimiento se hizo uso de principalmente dos ADT, siendo estos, Grafos y Tablas de Hash. Cada llave en la Tabla de Hash representa los horarios de inicio en los viajes divididos cada 15 min; Asimismo el valor correspondiente a estas llaves son grafos que representan los viajes que los Taxis inician a esa hora especifica.



¿Por qué eligieron esa estructura de datos?

A pesar de que a priori un Árbol tenga mas sentido a la hora de hablar de rangos de horas, una tabla de Hash es mas eficiente a la hora de construir y cargar los datos. Además, estos rangos de tiempo solo son ventanas de 15 minutos que vienen previamente dispuestas. En lo que respecta a relacionar las horas de viaje y la conexión entre área, los Grafos nos permiten representar relaciones binarias entre elementos de un conjunto. Siendo estas representaciones binarias, el área de inicio, de llegada y la duración del trayecto.

Haciendo el uso de grafos también es posible utilizar herramientas como el algoritmo de Dijkstra, para la determinación del camino más corto, dado un vértice origen, hacia el resto de los vértices en un grafo que tiene pesos en cada arista.

¿Cuál es la complejidad estimada del algoritmo implementado?

Como fue mencionado anteriormente para obtener los viajes a diferentes horas se hace uso de una Tabla de Hash, obtener el un valor de esta tiene e ser constante, en general O(2.5).

En los Grafos, la implementación del algoritmo de Dijkstra tiene una complejidad temporal de O(Elog(V)) donde e representa el numero de vértices o Áreas y E el numero de arcos que los conectan.