ANÁLISIS DEL RETO

Maria Gabriela Herrera Rojas, 202510147, mg.herrera@unaindes.edu.co

Estudiante 2, código 2, email 2

Estudiante 3, código 3, email 3

Requerimiento <<2>>

Plantilla para el documentar y analizar cada uno de los requerimientos.

Descripción

La función del requerimiento 2, era básicamente calcular los valores promedios de distancia, costos, propinas, peajes, etc.… y calcular la frecuencia de pasajeros y viajes.

Entrada	Catalog, payment_method
Salidas	Se espera que se retorne, una tabla en la cual estan los valores promedios de distancia, costos, propinas, peajes, etc. y la frecuencia de pasajeros y viajes.
Implementado (Sí/No)	Si se implementó, y fue realizado por María Gabriela Herrera Rojas

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Paso 1: Obtener el tamaño de la lista	O(1)
Paso 2: El recorrido de los viajes de catalogo	O(n)
Paso 3: El append, la comparacion y las operaciones internas	O(1)
Paso 4: El max sombre un diccionario	***O(m)***
TOTAL	***O(1 + n + m) = O(n)***

Análisis

La implementación recibe como entrada el catálogo de viajes y un método de pago, y genera como salida una tabla con los valores promedio de distancia, costos, propinas, pesos, pasajeros y frecuencia de viajes. En las pruebas, los resultados obtenidos fueron consistentes con las expectativas, lo que demuestra que la función cumple adecuadamente con los objetivos planteados. El desarrollo fue realizado por María Gabriela Herrera Rojas, y el análisis de complejidad muestra que el algoritmo tiene

un costo total de O(n), ya que la operación más costosa corresponde a recorrer todos los viajes del catalogo.

Requerimiento <<3>>

Descripción

Este requerimiento calcula la información promedio de los trayectos de taxis dado un rango de pagos. Para hacerlo, se filtran los viajes que cumplen con el rango ingresado por el usuario y, con esos datos, se calculan métricas como: duración promedio, precio promedio, distancia promedio, peajes, propinas, numero de trayectos, pasajero más frecuente y la fecha de finalización más común. Al final los resultados se tabulan.

Función CLAVE:

```
filtrados = []
for viaje in catalog:
    try:
        total = float(viaje["total_amount"])
        if precio_min <= total <= precio_max:
            filtrados.append(viaje)
```

Entrada	Catalog, precio min, precio max
Salidas	Todos los valores pedidos en el requerimiento, si no existen, retorna None
Implementado (Sí/No)	(SI) El requerimiento fue implementado por Juan Pablo Peñaranda

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Paso 1 Cargar el archivo y volverlo una lista de dic	O(n)
Paso 2 Filtrar los viajes que cumplen el rango de total amount	O(n)
Paso 3 Recorrer los viajes filtrados y guardar datos como distancia, precios, propinas, etc...	O(n)
Paso 4 Calcular promedios a partir de sumas acumuladas	O (1)

Paso 5 Obtener el pasajero más frecuente con max en diccionario	O(n)
Paso 6 Obtener la fecha más fecuente con max en un diccionario	O(...)
TOTAL	***O(n)***

Análisis

La función presenta una complejidad O(n) porque depende del número total de viajes cargadas desde el archivo. Las operaciones más robustas son las de recorrer el catálogo teniendo en cuenta el filtro, porque el cálculo de promedios y la búsqueda de máximos en diccionarios no afecta tanto la complejidad.

En las pruebas con el archivo “taxis-large.csv” realizadas en un computador con 16gb de RAM, un procesador 13th Gen Intel(R) Core (TM) i5-13450HX (2.40 GHz) y un sistema operativo de 64 bits, el tiempo de ejecución fue de aproximadamente 4.4 segundos para procesar casi medio millón de registros lo cual teniendo en cuenta las capacidades de la maquina es un resultado esperado. los resultados arrojaron resultados coherentes como un precio promedio cercano a $12, propinas alrededor de $1.22, y que el número de pasajeros común sea 1. esto confirma que el código es fiable incluso con semejante cantidad de datos.

Si los datos se almacenaran en un ArrayList, acceder a cada viaje por índice es O (1), lo que permite que el recorrido completo sea O(n). En cambio, si los datos estuvieran en una Singlelinked list, acceder a un elemento en particular sería O(n), pero como aquí se hace un recorrido secuencial de principio a fin, no hay diferencia en el costo total: también sería O(n).

Requerimiento <<4>>

Plantilla para el documentar y analizar cada uno de los requerimientos.

Descripción

Esta función implementa el **requisito 4**, cuyo propósito es identificar la combinación de barrios de origen y destino con el coste promedio más alto o más bajo dentro de un rango de fechas determinado. Para ello, itera todos los viajes del catálogo, filtra los que cumplen el rango de fechas, determina los barrios de origen y destino más cercanos a las coordenadas de partida y destino mediante la distancia de Haversine y acumula sus costes, duraciones y distancias. Finalmente, calcula los promedios de la combinación seleccionada según el filtro ("MÁS ALTO" o "MÁS BAJO") y devuelve un diccionario con el tiempo de ejecución, el número de viajes válidos y los promedios obtenidos.

Entrada	Filtro, catalogo, fecha_i, fecha_f
Salidas	Se espera que se retorne, una tabla en la cual esta el viaje con menos costo o mayor costo, y su promedio en duracion, etc..
Implementado (Sí/No)	Si se implemtno pero no funciona

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Paso	Descripción	Complejidad
1	Inicialización de variables y estructuras auxiliares (conteo_trayectos, comb_f, fechas)	**O(1)**
2	Recorrido de todos los viajes en el catálogo (for i in range(size))	**O(n)**
3	Conversión de fechas y extracción de coordenadas de cada viaje	**O(1)** (por viaje → total O(n))
4	Verificación y acumulación de datos en el diccionario comb_f	**O(1)**
5	Selección de la combinación con mayor/menor costo promedio (recorrido de k combinaciones)	**O(k)**
6	Cálculo de promedios y armado del resultado final	**O(1)**
TOTAL		**O(n)**

Análisis

La función implementada recibe como entrada el catálogo de viajes y un rango de fechas, y como salida la combinación de vecindarios de origen y destino con el mayor o menor costo promedio según el filtro especificado. También incluye información adicional como el número de viajes válidos, la distancia, la duración y el costo promedio, junto con el tiempo de ejecución. En las pruebas realizadas, los resultados fueron consistentes, identificando correctamente las combinaciones de vecindarios en diferentes escenarios. El análisis de complejidad muestra que el algoritmo tiene un costo total de **O(m)**, dado que para cada uno de los n viajes, se recorren los `m` vecindarios para determinar el origen y el destino más cercanos. A esto se suma la selección final de combinacion, `pero aún está dominada por el término O(n)`

Requerimiento <<1>>

Descripción

Este requerimiento se encarga de tomar la cantidad de pasajeros del taxi como filtro de la función, a partir de esto toma los datos resultantes de este filtro y saca las estadísticas de los promedios de esta información filtrada.

Entrada	Estructuras de datos del modelo, ID.
Salidas	Se retorna un diccionario con toda la información de los promedios de las estadísticas p
Implementado (Sí/No)	Si. Implementado por Ileanne Valderrama

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Paso 1: Se crea es_verdad para almacenar resultados	O(1)
Paso 2: Se filtran los trayectos por la cantidad de pasajeros	O(n)
Paso 3: Se sacan los cálculos requeridos (tiempo promedio, costo promedio, menor costo, mayor costo...), esto se hace sobre la matriz.	O(k)
Paso 4: Se devuelven los resultados en un diccionario con todas las estadísticas requeridas	O(1)
TOTAL	***O(n)***

Análisis

Al acceder un elemento en array, se debe hacer por indexación, lo que tiene una complejidad de O(1), la función tiene una complejidad de O(n) gracias a que recorrer el csv taxis, se hace desde diferentes partes del csv, como payment_type, passanger_count, al momento de añadir esto a las operaciones que se realizan, de filtración, comparación, suma, resta y demás, se denota la necesidad de filtrar los datos, lo que hace más complejo el recorrido.

Requerimiento <<5>>

Descripción

El requerimiento 5 filtra según el rango de fechas y el costo de los trayectos, a partir de esto, es que calcula el costo del trayecto, según el momento en el que es mayor o menor, en esta función, se sacan cosas como un costo mínimo, uno máximo, y el costo promedio para poder resolver la incógnita.

Entrada	Catalog, fecha_inicial, fecha_final, filtro_costo
Salidas	Se retorna un diccionario con toda la información de los promedios de las estadísticas requeridas en el ejercicio.
Implementado (Sí/No)	Implementado por Ileanne Valderrama

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Paso 1: Se crea es_verdad para almacenar los datos	O(n)
Paso 2: Se filtra por trayectos (en fechas) y por costo	O(n)
Paso 3: Calculo de las estadísticas	O(n)
Paso 4: Devolver resultados	O(n)
TOTAL	***O(n)***

Análisis

Acceder por array a un elemento es tiene una dificultad de O1, pero esta también recorre toda la lista de taxis que filtra por fechas y costo, acá se hacen filtros, comparaciones y muchos tipos de operaciones, lo que la convierten en una función con una dificultad de O(n).

Requerimiento <<6>>

Descripción

El requerimiento 6 filtra según él según un barrio de inicio y un rango de fechas aplicado solo a la fecha de inicio del viaje. A partir de este filtro, calcula estadística requerida para solucionar el problema.

Entrada	Catalog, fecha_inicial, fecha_final, filtro_costo
Salidas	Se retorna un diccionario con toda la información de los promedios de las estadísticas requeridas en el ejercicio.
Implementado (Sí/No)	Implementado por Ileanne Valderrama

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Paso 1: Inicialización de listas y diccionario	O(n)
Paso 2: Se filtra por trayectos (en fechas)	O(n)
Paso 3: Calculo de las estadísticas	O(n)
Paso 4: Determinación del barrio más visitado	O(m * k)
Paso 5: Retrono de resultados	O
TOTAL	***O(n)***

Análisis

Aunque acceder a elementos en un ArrayList es O(1), la función recorre toda la lista de taxis para aplicar los filtros y calcular las estadísticas por lo que la complejidad total es O(n).