Departamento de Ingeniería de Sistemas y Computación
Estructuras de Datos y Algoritmos

ISIS-1225

ANÁLISIS DEL RETO

Luis Alejandro Rodríguez Arenas, Cód. 202321287, la.rodrigueza12@uniandes.edu.co

Santiago Rojas, Cod. 202420928, s.rojasg23@uniandes.edu.co

Requerimiento <<1>>

Descripción

Este requerimiento filtra los vuelos de una aerolínea cuyo retraso de salida (dep_delay) está dentro de un rango dado, los ordena por retraso ascendente, en empate por fecha y luego por hora de salida, y retorna los 5 primeros y los 5 últimos vuelos de la lista filtrada (si hay menos de 5, retorna solo los existentes).

Entrada	Catálogo de vuelos, código de aerolínea, retraso mínimo, retraso máximo
Salidas	Diccionario con: tiempo de ejecución, total de vuelos filtrados y listas con los 5 primeros y 5 últimos vuelos (cada vuelo incluye fecha, horas programada y real, retraso y demás datos del registro).
Implementado (Sí/No)	Si se implementó y lo hizo Luis Alejandro

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Obtener lista de vuelos de la aerolínea (rbt.get)	O(log n)
Obtener tamaño de la lista (lt.size)	O(1)
Recorrer todos los vuelos de la aerolínea	O(n)
Acceder a cada vuelo (lt.get_element)	O(1) por entrada, total O(n)
Filtro por rango de retraso	O(1) por vuelo, total O(n)

Crear lista filtrada	O(1)
Ordenamiento de la lista filtrada (lt.merge_sort)	O(n log n)
Sublistas (lt.sub_list)	O(1)
TOTAL	***O(n log n)***

Análisis

La complejidad total está dominada por el ordenamiento de los vuelos filtrados mediante merge_sort, con costo O(m log m). Las operaciones de filtrado, acceso y creación de sublistas son lineales o constantes, por lo que no cambian el orden de crecimiento global.

Requerimiento <<3>>

Descripción

Este requerimiento recorre todos los vuelos del catálogo, selecciona los que pertenecen a una aerolínea y aeropuerto destino específicos y cuya distancia está en un rango dado, inserta los vuelos válidos en un árbol RBT ordenado por (distancia, fecha-hora de llegada) y retorna la lista ordenada de vuelos dentro de dicho rango, junto con el total y el tiempo de ejecución.

Entrada	Catálogo de vuelos, código de aerolínea, aeropuerto destino, distancia mínima, distancia máxima
Salidas	Diccionario con: aerolínea, destino, total de vuelos en el rango, lista ordenada de vuelos (por distancia y fecha-hora de llegada) y tiempo de ejecución.
Implementado (Sí/No)	Si se implementó y lo hizo Santiago Rojas

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Obtener tamaño de la lista de vuelos (lt.size)	O(1)
Recorrer todos los vuelos del catálogo	O(n)
Acceder a cada vuelo (lt.get_element)	O(1) por acceso, total O(n)
Filtro por aerolínea, destino, fecha y distancia	O(1) por vuelo, total O(n)
Conversión de distancia a float	O(1) por vuelo, total O(n)

Construcción de clave y creación de registro	O(1) por vuelo válido, total O(n)
Inserción de cada vuelo válido en el RBT (rbt.put)	O(log n) por inserción, total O(n log n)
Consulta de valores en rango (rbt.values)	O(log n)
Cálculo del total de vuelos (sl.size)	O(1)
TOTAL	**O(n log n)**

Análisis

El costo dominante proviene de las inserciones y la consulta en rango sobre el árbol RBT, que aportan el término O(n log n). El recorrido del catálogo y los filtros son lineales, por lo que el requerimiento se comporta globalmente en tiempo O(n log n) en el peor caso.

Requerimiento <<4>>

Descripción

Este requerimiento filtra los vuelos cuya fecha y hora de salida programada están en un rango dado, agrupa los vuelos válidos por aerolínea en una tabla hash, acumula para cada aerolínea el número de vuelos, la suma de duraciones y distancias, y el vuelo de menor duración, y finalmente usa una priority queue para obtener las N aerolíneas con mayor número de vuelos, calculando además duración y distancia promedio por aerolínea.

Entrada	Catálogo de vuelos, fecha inicial, fecha final, hora inicial, hora final, top_n
Salidas	Diccionario con: tiempo de ejecución, total de aerolíneas retornadas y lista con las N aerolíneas con más vuelos (incluye totales, promedios y el vuelo de menor duración por aerolínea).
Implementado (Sí/No)	Si se implementó y se hizo en grupo

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Conversión de fechas y horas de entrada (datetime.strptime)	O(1)
Recorrer todos los vuelos del catálogo	O(n)

Acceder a cada vuelo (lt.get_element)	O(1) por acceso, total O(n)
Conversión de fecha y hora de cada vuelo (datetime.strptime)	O(1) por vuelo, total O(n)
Filtro por rango de fecha y hora	O(1) por vuelo, total O(n)
Operaciones de hash por aerolínea (mp.get, mp.put)	O(1) por operación, total O(n)
Actualización de agregados (conteo, suma de duración y distancia, mínimo)	O(1) por vuelo, total O(n)
Obtención del conjunto de aerolíneas (mp.key_set)	O(n)n
Inserción de cada aerolínea en la priority queue	O(log) por inserción, total O(n log n)
Extracción de las N aerolíneas con más vuelos (pq.remove)	O(log A) por extracción
Construcción de la lista resultado (lt.add_last)	O(1) por aerolínea
TOTAL	**O(n log n)**

Análisis

El recorrido del catálogo y la actualización de la tabla hash son operaciones lineales. El factor logarítmico aparece al ordenar las aerolíneas mediante la priority queue, dando lugar a una complejidad global O(n log n), adecuada para grandes volúmenes de datos.

Requerimiento <<5>>

Descripción

Este requerimiento recorre todos los vuelos del catálogo, filtra los que están en un rango de fechas y llegan a un aeropuerto destino dado, agrupa los vuelos válidos por aerolínea en una tabla hash, calcula para cada aerolínea promedios de retraso a la llegada, duración y distancia, determina su vuelo de mayor distancia al aeropuerto dado y finalmente usa una priority queue para obtener las **N aerolíneas más puntuales** (menor valor absoluto del retraso promedio).

Entrada	Catálogo de vuelos, fecha inicial, fecha final, aeropuerto destino, top N
Salidas	Diccionario con: tiempo de ejecución, total de aerolíneas retornadas y lista con las N aerolíneas más puntuales (incluye promedios, distancia media, puntualidad absoluta y vuelo de mayor distancia).
Implementado (Sí/No)	Si se implementó y se hizo en grupo

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Recorrer todos los vuelos del catálogo	O(n)
Acceder a cada vuelo (lt.get_element)	O(1) por acceso, total O(n)
Filtro por fecha, destino y aerolínea	O(1) por vuelo, total O(n)
Conversión de retraso, duración y distancia a float	O(1) por vuelo, total O(n)
Operaciones en la tabla hash de aerolíneas (mp.get, mp.put)	O(1) por operación, total O(n)
Actualización de acumulados y vuelo de máxima distancia	O(1) por vuelo, total O(n)
Obtención del conjunto de claves (mp.key_set)	O(n)
Cálculo de promedios y puntualidad absoluta por aerolínea	O(1) por aerolínea, total O(n)
Inserción en la priority queue según puntualidad	O(log n) por aerolínea, total O(n log n)
Extracción de las N aerolíneas más puntuales (pq.remove)	O(log n) por extracción
Construcción de la lista resultado (lt.add_last)	O(1) por aerolínea
TOTAL	**O(n log n)**

Análisis

El requerimiento combina un recorrido lineal del catálogo con operaciones de hash O(1) y el ordenamiento de aerolíneas mediante una priority queue. Por ello, la complejidad final está dominada por O(n log n), donde el término logarítmico proviene de la estructura de prioridad usada para seleccionar las aerolíneas más puntuales.

Requerimiento <<6>>

Descripción

Este requerimiento filtra los vuelos de un catálogo por rango de fechas y distancias, agrupa los vuelos válidos por aerolínea en una tabla hash, calcula para cada aerolínea la media y la desviación estándar del retraso de salida (dep_delay), realiza un segundo recorrido para encontrar el vuelo más representativo (el que tiene retraso más cercano a la media), y finalmente usa una priority queue para obtener las M aerolíneas con menor desviación estándar, es decir, las más consistentes en sus retrasos.

Entrada	Catálogo de vuelos, fecha inicial, fecha final, distancia mínima, distancia máxima, top m
Salidas	Diccionario con: tiempo de ejecución, total de aerolíneas retornadas y lista de M aerolíneas más consistentes (incluye media del retraso, desviación estándar y vuelo más representativo).
Implementado (Sí/No)	Si se hizo en grupo

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Primer recorrido del catálogo	O(n)
Acceder a cada vuelo (lt.get_element)	O(1) por acceso, total O(n)
Filtro por fecha, distancia y aerolínea	O(1) por vuelo, total O(n)
Operaciones en hash para acumulados (mp.get, mp.put)	O(1) por operación, total O(n)
Actualización de sumas y sumas de cuadrados	O(1) por vuelo, total O(n)

Obtención del conjunto de aerolíneas (mp.key_set)	O(n)
Cálculo de media y desviación estándar por aerolínea	O(1) por aerolínea, total O(n)
Segundo recorrido del catálogo	O(n)
Selección del vuelo más representativo por aerolínea	O(1) por vuelo, total O(n)
Inserción de aerolíneas en la priority queue	O(log n) por aerolínea, total O(n log n)
Extracción de las M aerolíneas con menor desviación (pq.remove)	O(log A) por extracción
Construcción de la lista resultado (lt.add_last)	O(1) por aerolínea
TOTAL	**O(n log n)**

Análisis

El requerimiento realiza recorridos lineales sobre el conjunto de vuelos y usa una tabla hash para mantener los acumulados estadísticos por aerolínea en tiempo O(1). El ordenamiento final de las aerolíneas por desviación estándar se hace mediante una priority queue, lo que añade el término O(n log n) el cual define la complejidad total del algoritmo.