ANÁLISIS DEL RETO

Sara Gómez Flórez, 202513246, s.gomezf234@uniandes.edu.co

Yara Isabella Gutierrez Diaz, 202511181, y.gutierrezd@uniandes.edu.co

Requerimiento <<1>>

Este requerimiento identifica los vuelos de una aerolínea específica que presentan un retraso en la hora de salida dentro de un rango dado de minutos Para ello, se utiliza un Árbol rojo negro (RBT) que indexa los vuelos por minutos de retraso. Se filtran las llaves del árbol dentro del rango, y luego se recorren las listas asociadas a cada llave para seleccionar solo los vuelos de la aerolínea indicada. Finalmente, los resultados se ordenan por retraso ascendente y, en caso de empate, por fecha y hora real de salida.

Descripción

Breve descripción de como abordaron la implementación del requerimiento

Entrada	• Catalog: Catalogo con la información de los vuelos • Código de aerolinea • Minutos mínimo de retraso • Minutos máximo de retraso
Salidas	• Tiempo de la ejecución en milisegundos • Total de vuelos que cumplen con el filtro • Los primeros 5 y los últimos 5 vuelos ordenados que contienen la siguiente información - ID del vuelo - Código del vuelo - Fecha - Nombre de la aerolinea - Código de la aerolinea - Aeropuerto de origen -Aeroupuerto de destino - Minutos de retraso en la salida
Implementado (Sí/No)	Si, por Yara Isabella Gutierrez Diaz

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Obtener las llaves del RBT en el rango	O(log n +k)
Recorrer las llaves y el valor asociado	O(k*m)
Filtrar por código de aerolínea	O(K*m)
Ordenar la lista con merge sort	O (r log r)

Sublistas de los primeros y últimos	0(1)
TOTAL	***O(log n + k * m + r log r)***

Análisis

La implementación usa un RBT para guardar los vuelos según los minutos de retraso. Eso hace que encontrar solo los vuelos dentro del rango que nos interesa sea mucho más rápido, sin tener que revisar todos los vuelos. Luego, de esos vuelos, se filtran solo los que son de la aerolínea que pedimos. Como ya estamos trabajando con un subconjunto más pequeño, permite que las demás operaciones se realicen con una complejidad mucho menor.

Requerimiento <<2>>

El requerimiento 2 según un aeropuerto destino y un rango de minutos de anticipo de llega a filtrar obtiene los vuelos dentro de el rango determinado de el aeropuerto. Para esto, se tiene en cuenta que un valor de minutos negativo indica una llegada anticipada para el requerimiento.

Descripción

Breve descripción de como abordaron la implementación del requerimiento

Entrada	Código del aeropuerto de destino a analizar (ej LAX) Rango de minutos de anticipo en la llegada a filtrar
Salidas	- Tiempo de ejecución del requerimiento. - Número total de vuelos que cumplen con el filtro del aeropuerto y del rango de anticipo. - Teniendo en cuenta los vuelos que cumplieron los parámetros ingresados por el usuario, se otorga la siguiente información: ID Código del vuelo Fecha Nombre de la aerolínea Código de la aerolínea Aeropuerto de origen Aeropuerto de destino Minutos de anticipo en la salida
Implementado (Sí/No)	Sí, por Sara Gomez Flórez.

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Se ingresa al árbol rojo negro con la siguiente estructura: (llave: minutos de anticipo, valor: mapa	

con llave como el código del aeropuerto destino y valor un arreglo)	
Se saca los valores de los nodos que cumplan con los minutos de anticipo (con values)	O(logn + m) m siendo los elementos a agregar dentro del rango.
Si hay vuelos en este rango, se recorre cada mapa.	O(z) z siendo los mapas en los nodos.
Entonces, se consigue el array con el código del aeropuerto dado por el usuario.	O(1)
Después, para cada vuelo dentro de el rango y el aeropuerto, se mete a un array de resultados.	O(r) r siendo la cantidad de elementos en la lista.
Se hace quick_sort sobre el resultado. Esto con un sort_crit: De forma ascendente por minuto de anticipo. Si tienen el mismo, se ordena por orden cronológico por fecha y horas real de llegada.	O(nlogn)
TOTAL	***O(logn + m + z + r + nlogn)***

Análisis

En este requerimiento 2 el usuario ingresa el código de el aeropuerto destino y un rango de minutos de anticipo. En el ordenamiento se utilizó un array list, a pesar de el costo de memoria, para poder tener la mejor complejidad posible en quick_sort (debido a la necesidad constante de utilizar las posiciones). Se utilizó un árbol rojo negro para facilitar la búsqueda del rango y de evitar un peor caso de complejidad al dejarlo necesariamente balanceado. Por otro lado, en cada nodo se tenía un map linear probing para ingresar rápidamente al código del aeropuerto destino con facilidad y sus vuelos o(1).

Requerimiento <<4>>

En el requerimiento 4 se pide que, para un rango de fechas y una franja horaria de salida programada otorgarle al usuario las N (ingresado por el usuario) aerolíneas con mayor número de vuelos y, para cada una, obtener el vuelo con la menor duración.

Descripción

Se intentó realizar una estructura compleja, sin embargo, se decidió ir por una implementación simple que, de igual forma, tenga los requisitos pedidos y una buena complejidad temporal.

Entrada	Rango de fechas a analizar. Franja horaria programada de vuelos de salida Cantidad de N aerolíneas a visualizar
Salidas	- Tiempo de ejecución del algoritmo - Número total de aerolíneas consideradas (N ingresado por el usuario) - La duración promedio de los vuelos realizados en el rango de fechas y horas.

- Distancia promedio recorrida por los vuelos realizados en el rango de fechas y horas.
- Información del vuelo con la menor duración (de los vuelos del filtro) se da la siguiente información:
  ID
  Código
  Fecha-hora programada de salida
  Aeropuerto de origen
  Aeropuerto de destino
  Duración

Implementado (Sí/No) Si se implementó.

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Se consigue el árbol rojo negro que contiene los rangos de fechas.	O(...)
Luego, a partir de values se consigue las listas de cada fecha. (Array list)	O(logn + m) m siendo los elementos a agregar dentro del rango.
Se recorre cada lista de cada fecha.	O(r) siendo r el número de listas en cada fecha dentro del rango.
Y su vez, se recorre los vuelos en las listas.	O(r * s) siendo s el número de vuelos en cada lista.
Al vuelo se cambia su fecha de salida programada a datetime. Y luego se ve si esta en el rango (teniendo en cuenta aquellos rangos que pidan viajes en medianoche)	O(1)
Si esta en el rango, se verifica que exista en un mapa generado. Si no, se agrega al mapa con un diccionario con la información esperada de cada aerolínea.	0(1)
Para el vuelo de menor duración se compara cada vuelo para ver si supera el de menor duración establecido en el diccionario. Si es así, se reemplaza el valor. Si tienen la misma duración, se elige el más antiguo por fecha y hora programada de salida.	0(1)
Luego de este proceso, se saca los valores (diccionarios) de el mapa.	O(z) siendo z el numero de aerolíneas en el mapa.
Se organiza los diccionarios con merge_sort con un sort_crit: se organiza de mayor a menor dependiendo el número de vuelos. Si se tiene un mismo número de vuelos dentro de esas fechas, se ordena alfabéticamente por su código.	(N logn)
TOTAL	***O(logn + m + r + z + rs + nlogn)***

Análisis

En este requerimiento se pide que, al ingresar un rango de fechas, una franja horaria y cierta cantidad de aerolíneas por ver, se devuelva la información pedida de las aerolíneas con mayor numero de vuelos dentro de los parámetros dados. En este requerimiento se utilizo un árbol rojo negro para evitar un peor caso de complejidad (single linked list, complejidad o(n)) y acceder a la información en el rango dado. Luego, debido se eligió filtrar por el rango de manera individual y crear un mapa que ingresara aquellas aerolíneas dentro del rango. Este mapa es un separate chaining para que, cuando se saque todos sus valores (diccionario con información de cada aerolínea) no tenga que recorrer toda una tabla con elementos vacíos como el separate chaining y reducir la complejidad. Luego, se organiza la lista array por medio de Merge sort (nlogn).

Requerimiento <<5>>

Este requerimiento busca encontrar las N aerolíneas más puntuales en sus vuelos hacia un aeropuerto destino dentro de un rango de fechas. Se calcula la puntualidad como la diferencia absoluta entre la hora real y la hora programada de llegada. Se usa un RBT para indexar los vuelos por fecha, y dentro de cada fecha se agrupan por aerolínea usando mapas. Cada aerolínea tiene una cola de prioridad que guarda los vuelos ordenados por distancia. Luego se agrupan los vuelos por aerolínea y se calculan métricas como puntualidad promedio, duración promedio, distancia promedio y el vuelo con mayor distancia. Finalmente, se ordenan las aerolíneas por puntualidad y se seleccionan las N mejores.

Descripción

Breve descripción de como abordaron la implementación del requerimiento

Entrada	• Fecha inicial en el formato YYYY-MM-DD • Fecha final en el formato YYYY-MM-DD • Código del aeropuerto de destino • Numero de aerolíneas más puntuales a retornar
Salidas	• Tiempo de la ejecución en milisegundos • Total de aerolíneas que cumplen con el filtro • Lista de las N aerolineas mas puntuales que continen la siguiente información - Código de aerolinea - Total de vuelos - Puntualidad promedio - Duracion promedio - Distancia promedio - Datos del vuelo con mayor distancia
Implementado (Sí/No)	Si se implementó.

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	Complejidad
Obtener las fechas en el rango dado	O(log d + f)
Agrupar vuelos por aerolinea	O(f*a*m)
Calcular metricas por aerolinea	O(a*m)
Ordenar por puntualidad utilizando merge sort	O(a log a)
Sublistas	O(1)
TOTAL	***O(log d + f *a *m+ a log a)***

Análisis

La solución usa un RBT para indexar los vuelos por fecha, lo que permite acceder rápidamente a los vuelos dentro del rango. Dentro de cada fecha, los vuelos se agrupan por aerolínea en mapas, y se almacenan en colas de prioridad ordenadas por distancia. Esto permite encontrar fácilmente el vuelo más largo de cada aerolínea.

La puntualidad se calcula como la diferencia absoluta entre la hora real y la programada de llegada, y se ajusta correctamente si el vuelo cruza la medianoche. Luego se calcula el promedio de puntualidad, duración y distancia por aerolínea. Las aerolíneas se ordenan por puntualidad (las más cercanas a cero primero), y en caso de empate se usa el código de aerolínea como criterio secundario.

Requerimiento <<6>>

Descripción

Este requerimiento busca encontrar las M aerolíneas más estables en su hora de salida, dentro de un rango de fechas y distancias. La estabilidad se mide como la desviación estándar de los retrasos (o anticipos) en la salida de sus vuelos. Se usa un RBT para indexar los vuelos por fecha, y luego se filtran por distancia. Los vuelos se agrupan por aerolínea y se calcula el promedio de retraso y su desviación estándar. Finalmente, se ordenan por estabilidad usando una cola de prioridad, y se seleccionan las M más estables. También se identifica el vuelo cuyo retraso esté más cerca al promedio.

Entrada

- Fecha inicial en el formato YYYY-MM-DD
- Fecha final en el formato YYYY-MM-DD
- Distancia mínima
- Distancia máxima
- N aerolineas mas estables a retornar

Salidas

- Tiempo de la ejecución en milisegundos
- Total de aerolíneas que cumplen con el filtro
- Lista de las N aerolineas mas puntuales que continen la siguiente información

- Código de aerolinea
- Total de vuelos analizados
- Promedio de retraso/ anticipo en minutos
- Desviación estandar
- Datos del vuelo mas cercano al promedio

Implementado (Sí/No)	Si se implementó.

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Obtener fechas en el rango	O(log d + f)
Filtrar los vuelos por distancia y agrupar por aerolínea	O(f*m)
Calcular promedio y desviación estandar por aerolinea	O(a*m)
Buscar el vuelo mas cercano al promedio	O(a*m)
Insertar en cola de prioridad	O (a log a)
Extraer los elementos de la cola	O (m log a)
TOTAL	***O(log d+ a log a)***

Análisis

La solución usa un RBT para indexar los vuelos por fecha, lo que permite acceder rápidamente a los vuelos dentro del rango. Luego se filtran por distancia y se agrupan por aerolínea. Para cada aerolínea, se calcula el promedio de retraso y la desviación estándar, que representa qué tan estables son sus horarios de salida. También se busca el vuelo más cercano al promedio, como ejemplo representativo.

Las aerolíneas se ordenan por estabilidad usando una cola de prioridad, y en caso de empate se usa el promedio de retraso como criterio secundario. El uso de estructuras como RBT y cola de prioridad permite que el algoritmo sea eficiente y escalable.