**ANÁLISIS DEL RETO 3**

Juliana Rodríguez Morales – 202421552 – js.rodriguezm1234

Maria Clara Quijano - 202420069 - m.quijanoa

Juan Andrés Lozada - 202510410-j.lozadab

# **Requerimiento 1**

## **Descripción**

Dada una aerolínea y un rango de minutos quiero identificar los vuelos que presentan un retraso en la hora de salida que están dentro del rango.

Texto

El contenido generado por IA puede ser incorrecto.

|  |  |
| --- | --- |
| **Entrada** | • Código de la aerolínea a analizar.  • Rango de minutos de retraso en salida a filtrar. |
| **Salidas** | Tiempo de la ejecución del requerimiento en milisegundos. Número total de vuelos que cumplen con el filtro de la aerolínea y rango de retraso. Teniendo en cuenta los vuelos que cumplen el filtro, presente los siguientes datos:   * ID del vuelo. * Código del vuelo. * Fecha. * Nombre de la aerolínea. * Código de la aerolínea. * Aeropuerto de origen. * Aeropuerto de destino. * Minutos de retraso en la salida. |
| **Implementado (Sí/No)** | Sí, por Juliana Rodríguez |

## **Análisis de complejidad**

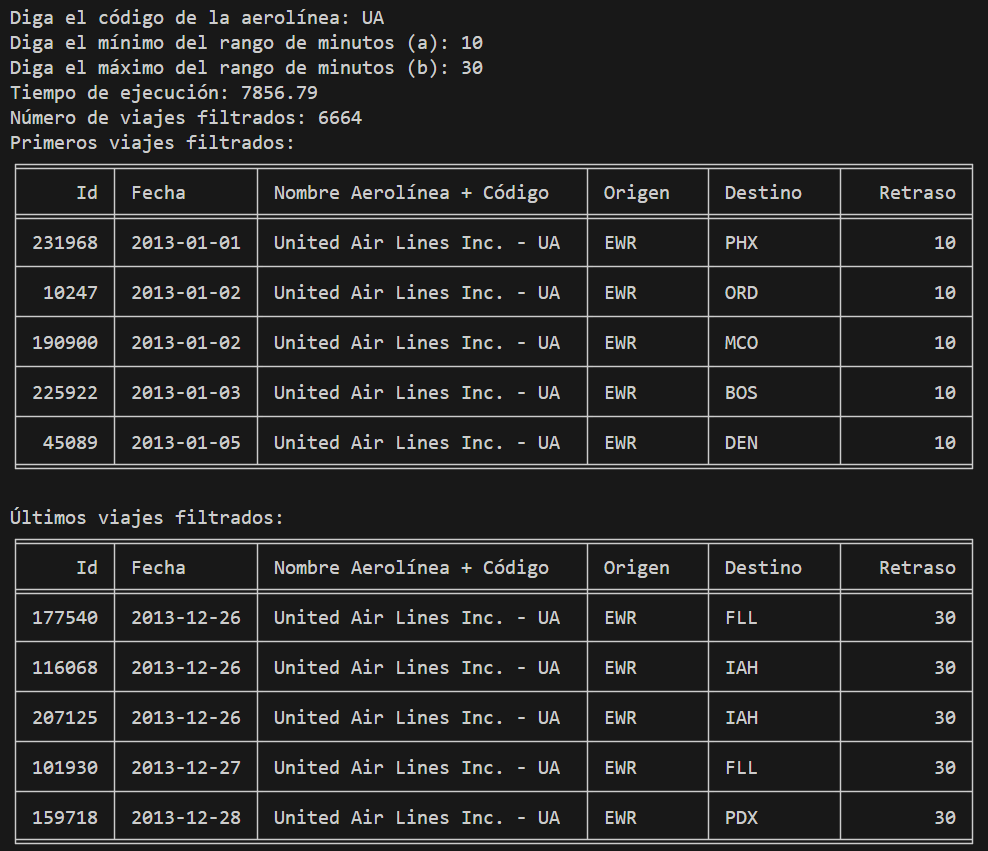
|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Obtener elemento de un mapa (submapa) | O(n/N) |
| Crear un single con las llaves del mapa | O(n) |
| Primer for (Recorre todos los submapas) | O(n) |
| Obtener elemento del submapa (lista) | O(m/M) |
| Segundo for (Recorre los trayectos de la lista) | O(m) |
| Obtener elemento de la lista (viaje) | O(m) |
| Insertar en una Cola de Prioridad | O(1+Log m) |
| ***TOTAL*** | ***O(n\*m(1+Log m))*** |

## **Análisis**

Dado que la estructura con la información necesaria para este requerimiento era un mapas (aeropuertos) dentro de un mapa (de aerolíneas) se utilizaron dos recorridos for para poder obtener la información de todos viajes filtrados. Asimismo, la complejidad es afectada al meter los datos dentro de una cola de prioridad aquellos vuelos que tenían un retraso dentro del rango. Aunque son dos recorridos estos tienen complejidad de variables diferentes ya que el primero recorre todos los mapas dentro de la aerolínea filtrada, y el segundo recorre todos los viajes dentro del mapa previamente recorrido, lo que encadena la complejidad.

## **Pruebas (Al menos 5)**

Hechos con large:

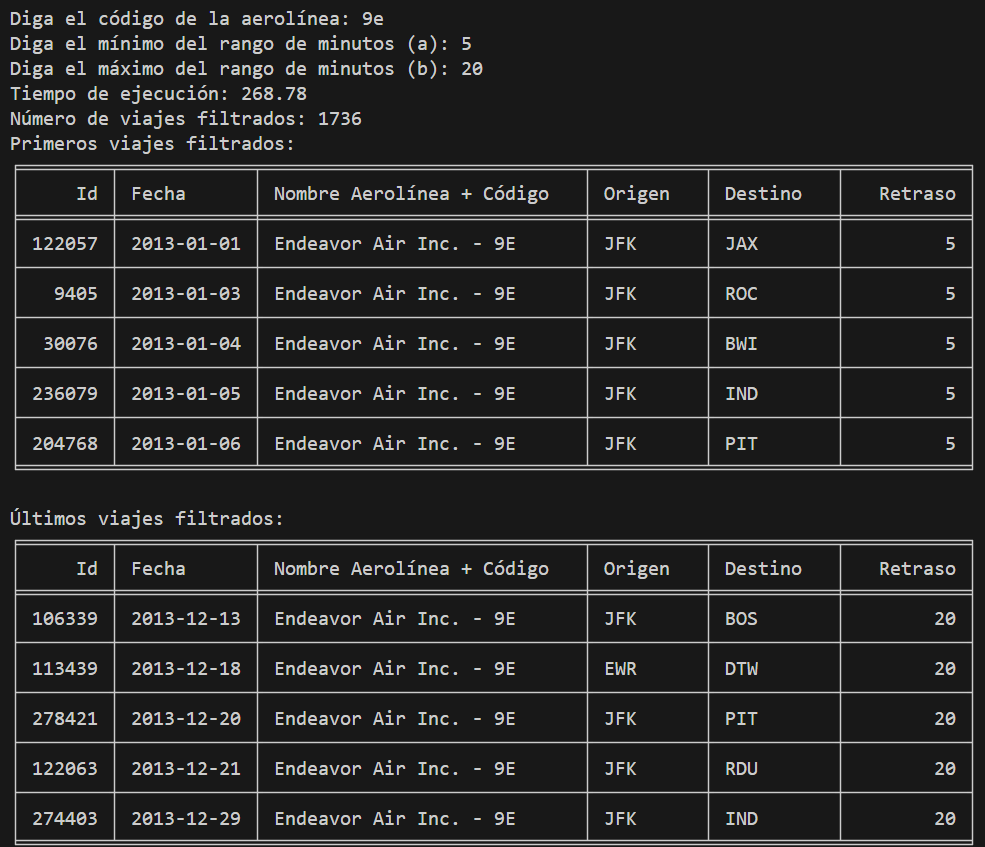


Tabla

El contenido generado por IA puede ser incorrecto.Imagen que contiene Tabla

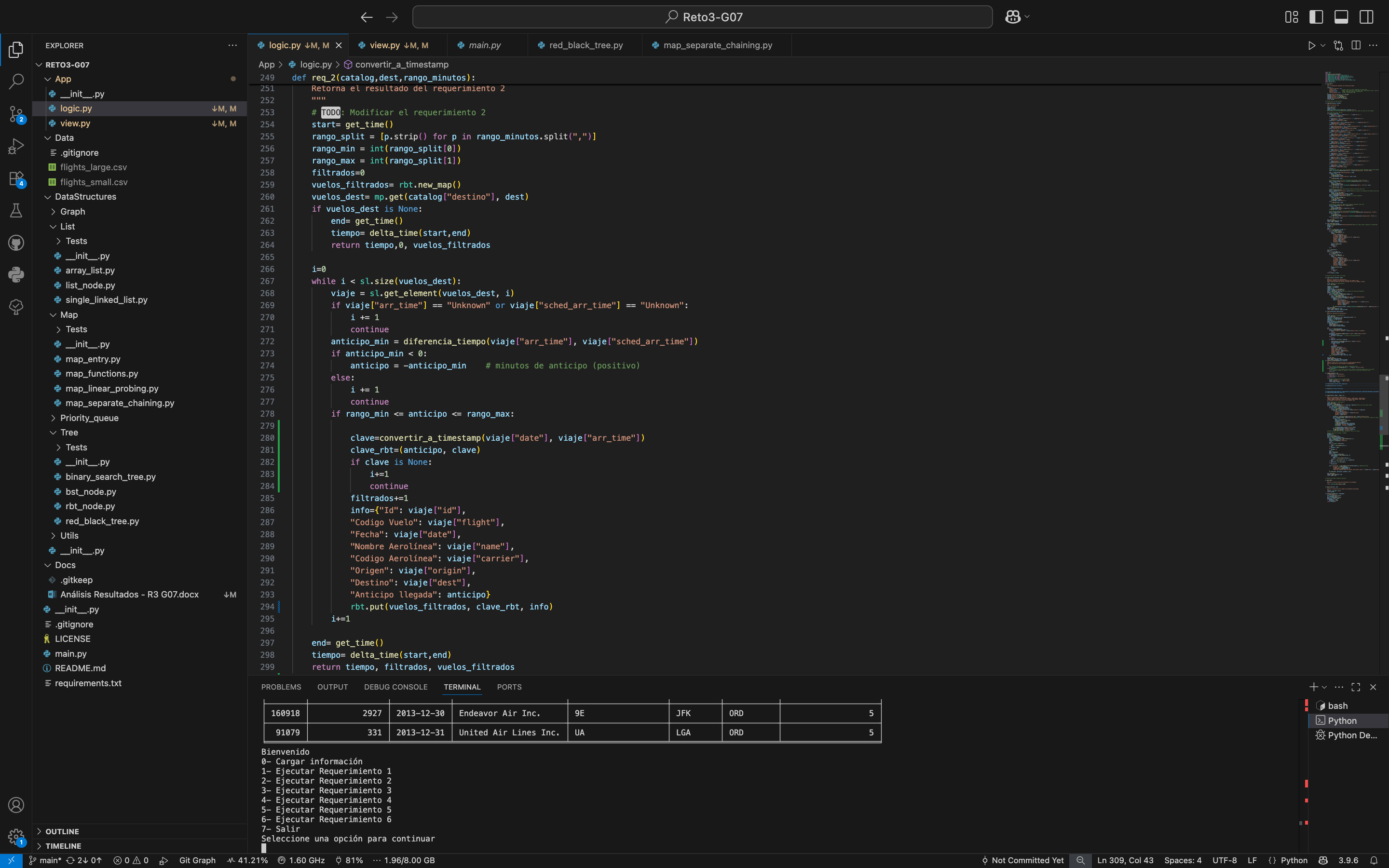
El contenido generado por IA puede ser incorrecto.Pantalla de computadora con letras

El contenido generado por IA puede ser incorrecto.



Promedio de tiempos: 2599.006 ms

# **Requerimiento 2**



## **Descripción**

Toman los datos de vuelos y les saca el anticipo en la hora de llegada dentro de un rango de minutos si pasan el filtro del aeropuerto destino.

|  |  |
| --- | --- |
| **Entrada** | * Código destino * Rango minutos de anticipo en la llegada a filtrar |
| **Salidas** | * Tiempo de ejecución * Número total de vuelos que cumplen con el filtro del aeropuerto y del rango de anticipo * Vuelos que cumplen el filtro:   + Id   + Código   + Fecha   + Nombre   + Aeropuerto origen   + Aeropuerto destino   + Minutos anticipo en salida |
| **Implementado (Sí/No)** | Si, Maria Clara Quijano |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

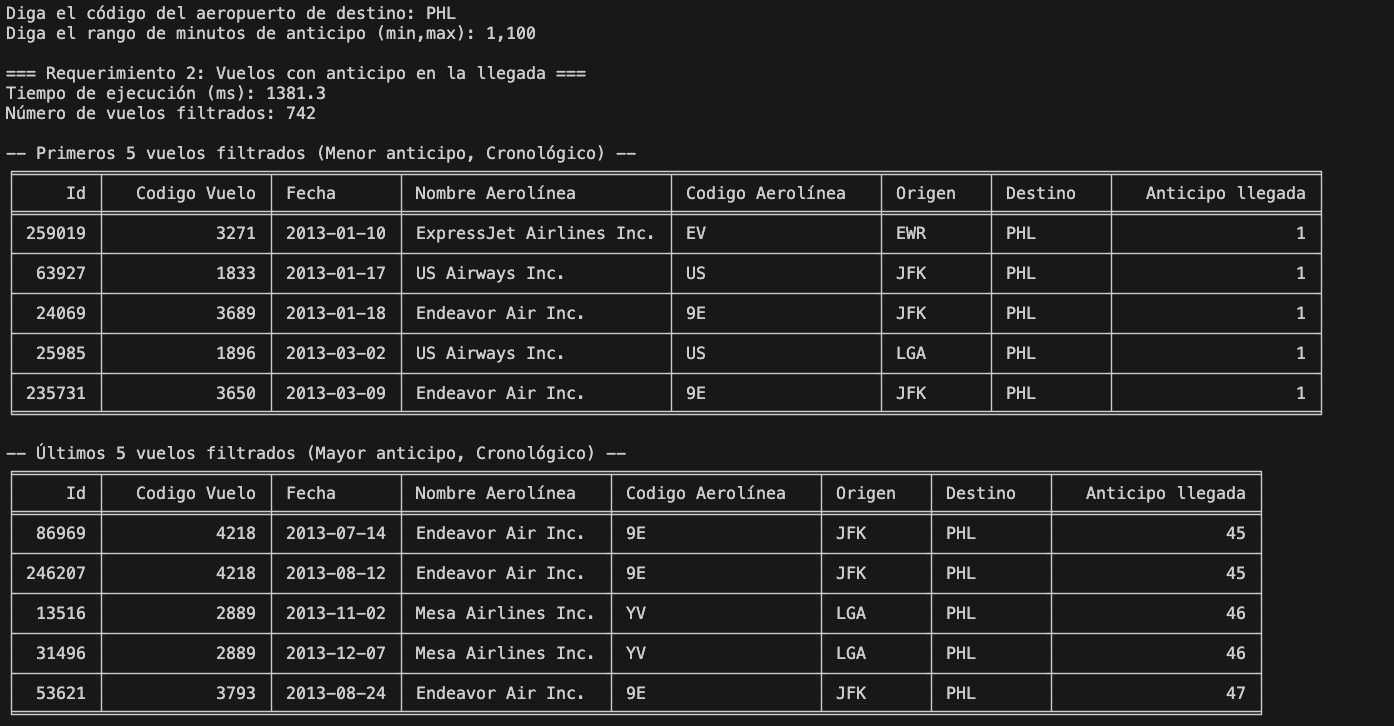
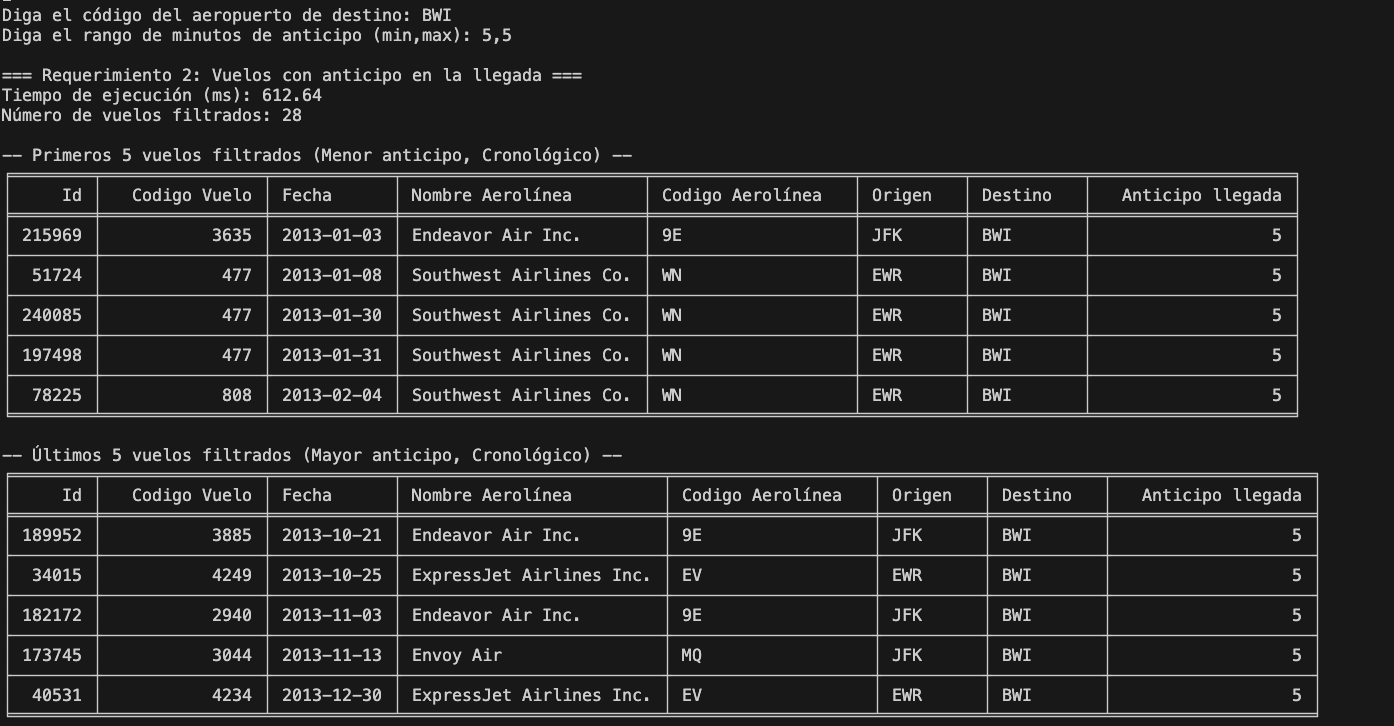
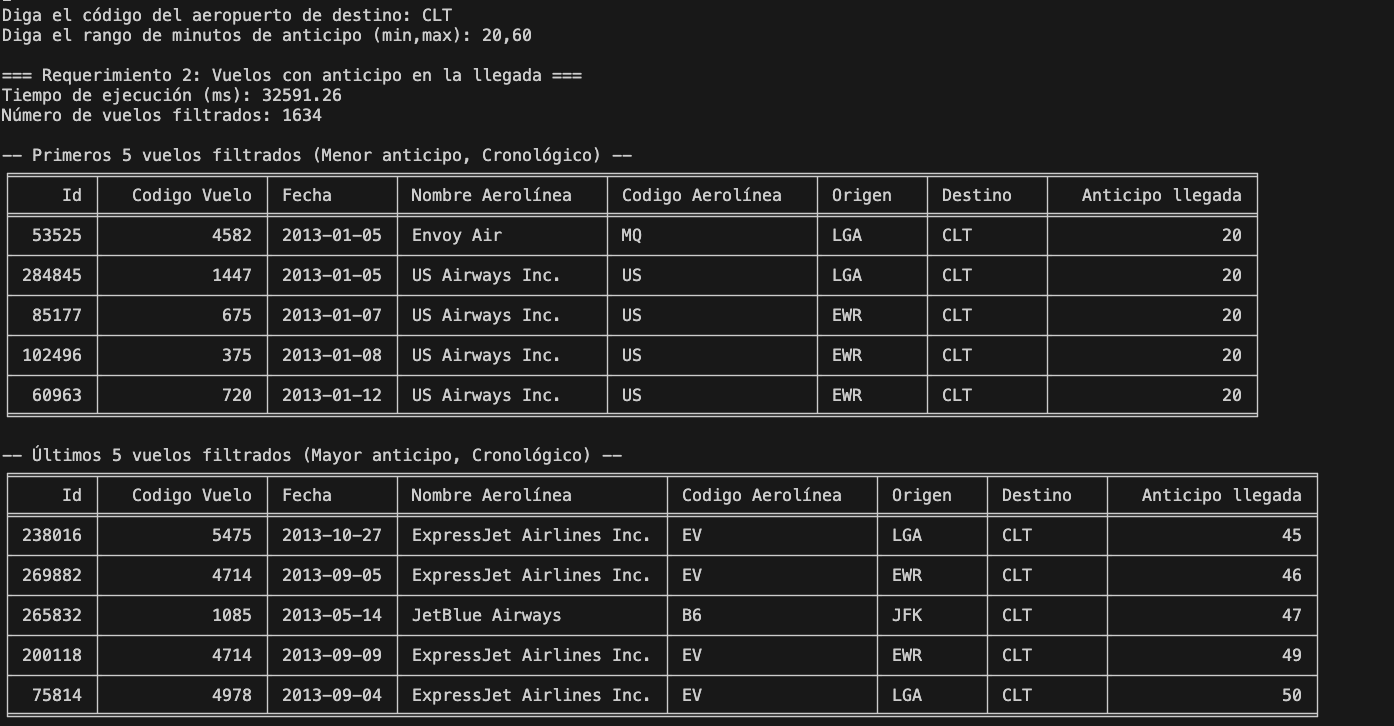
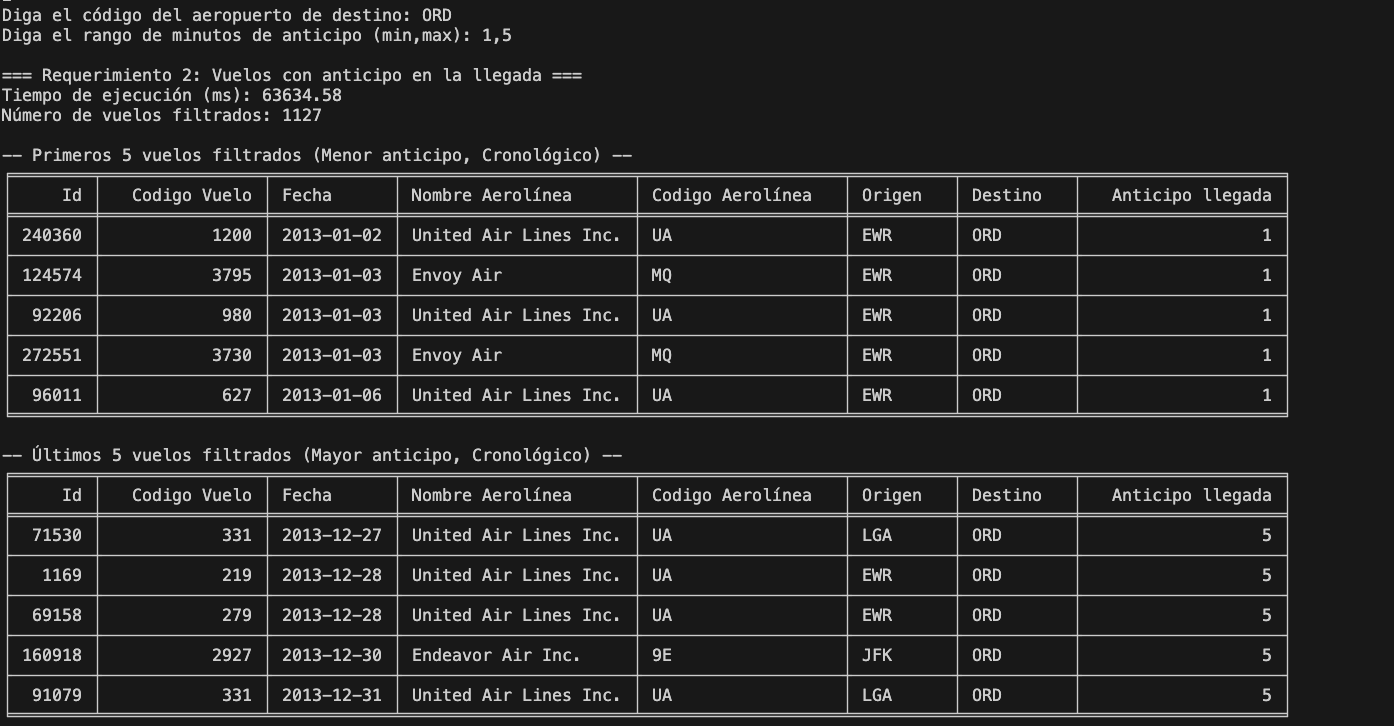
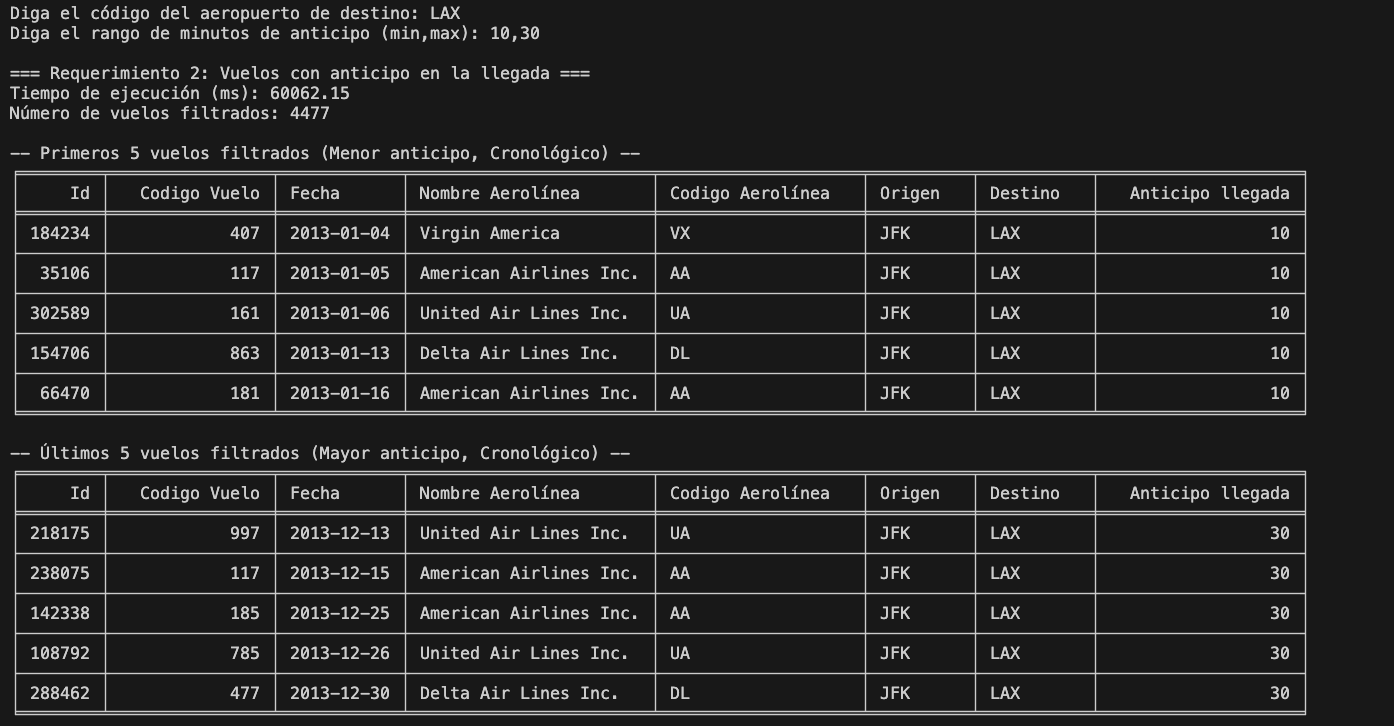
|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Busqueda por destinp | O(LogN) |
| Calculo anticipo | O(1) |
| While | O(m) |
| Get y put | O(Logk) |
| ***TOTAL*** | ***O(m\*logk)*** |

## **Análisis**

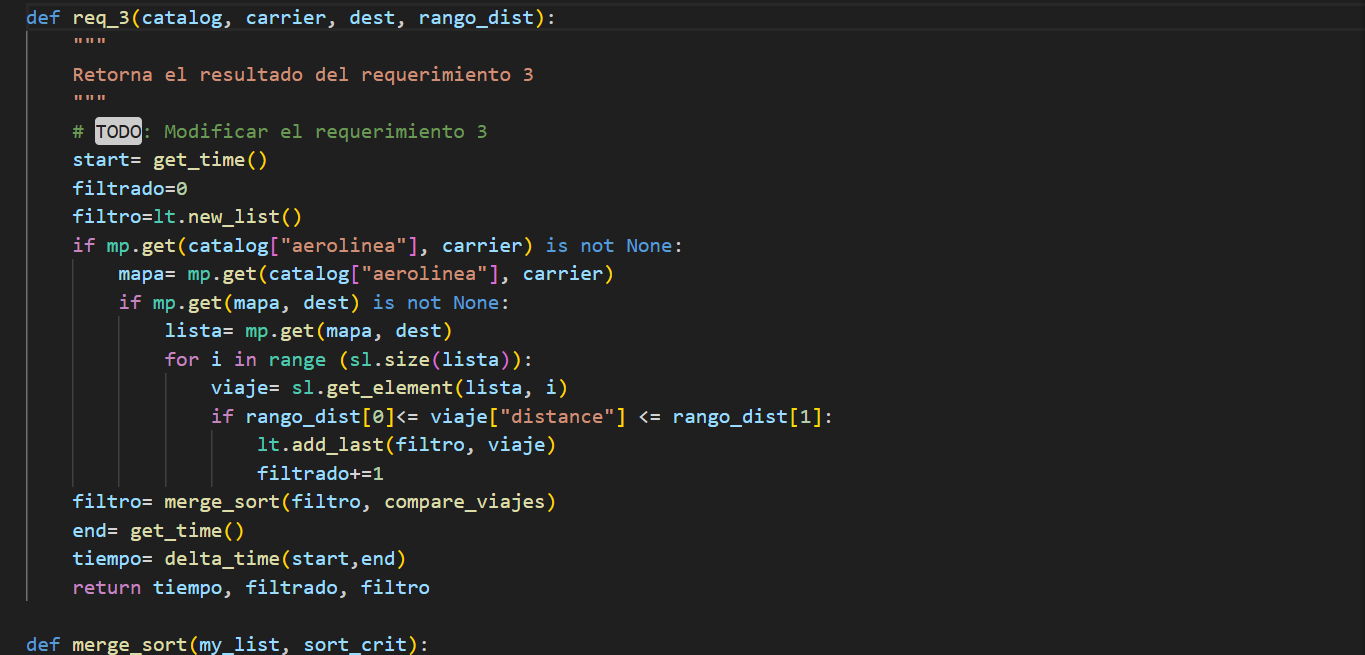
Se hace un filtro por busqueda, y luego entra al while, lo cual es lo que incrementa más su complejidad ya que itera sobre m vuelos, los que tienen el destino especificado. Normalmente, n será un valor bajo. Por el uso del Red Black Tree, la complejidad de insert y search es de O(logk), y mantiene un orden ascendente por anticipo de manera eficiente.

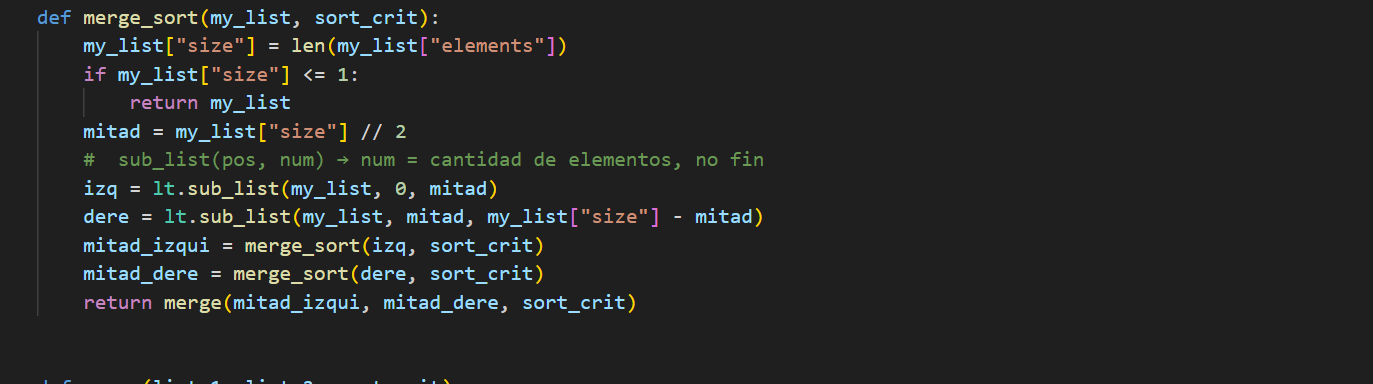
## **Pruebas (Al menos 5)**

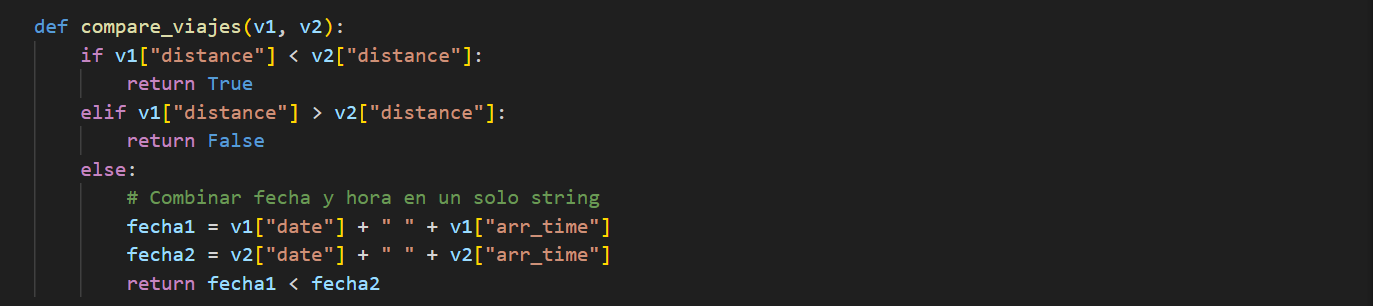
Hechos con large:



# **Requerimiento 3**







## **Descripción**

|  |  |
| --- | --- |
| **Entrada** | • Código del aeropuerto de destino a analizar (por ejemplo: “JFK”). • Rango de minutos de anticipo en la llegada a filtrar (por ejemplo: [10,30]). |
| **Salidas** | -Tiempo de la ejecución del requerimiento en milisegundos.  -Número total de vuelos que cumplen con el filtro del aeropuerto y del rango de anticipo.  -Teniendo en cuenta los vuelos que cumplen el filtro, presente los siguientes datos:   * o ID del vuelo. * o Código del vuelo. * o Fecha. * o Nombre de la aerolínea * o Código de la aerolínea. * o Aeropuerto de origen. o Aeropuerto de destino. * o Minutos de anticipo en la salida. |
| **Implementado (Sí/No)** | Si, por Juan Andrés Lozada Barragán |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

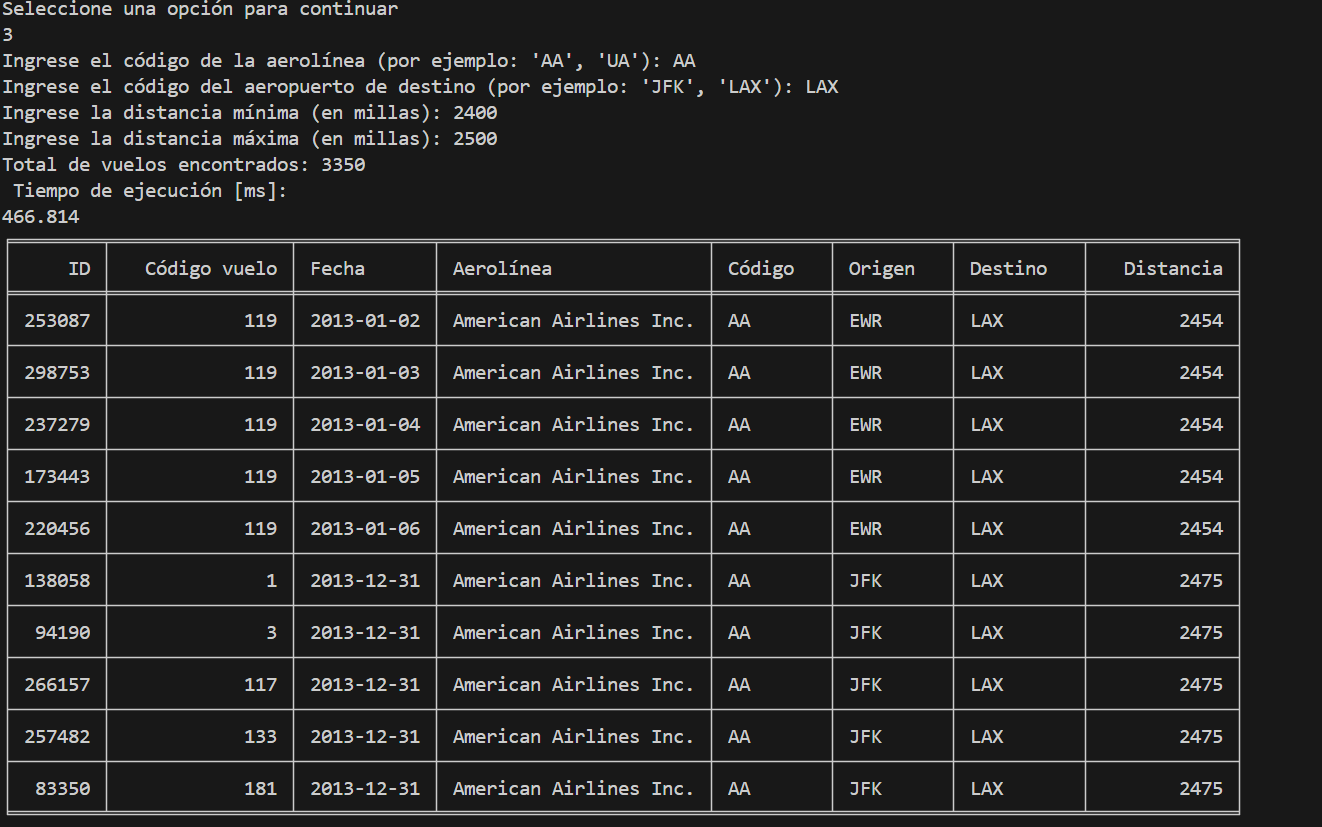
|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| mp.get(catalog["aerolinea"], carrier) y mp.get(mapa, dest) | O(1) |
| sl.size(lista) | O(1) |
| for i in range(sl.size(lista)) | O(n) |
| Comparación de rangos y adición a lista | O(1) |
| merge\_sort(filtro, compare\_viajes) | O(m log m) |
| delta\_time() y retornos | O(1) |
| ***TOTAL*** | O(n+m log m) |

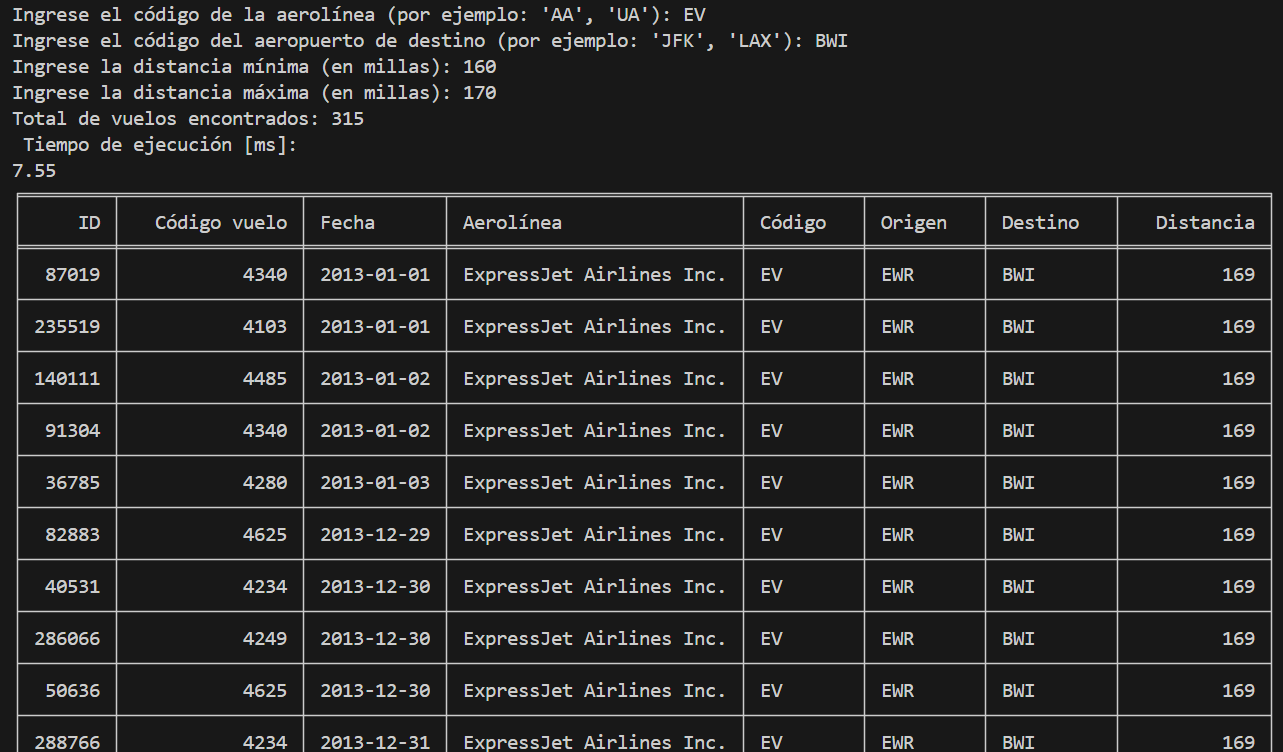
## **Análisis**

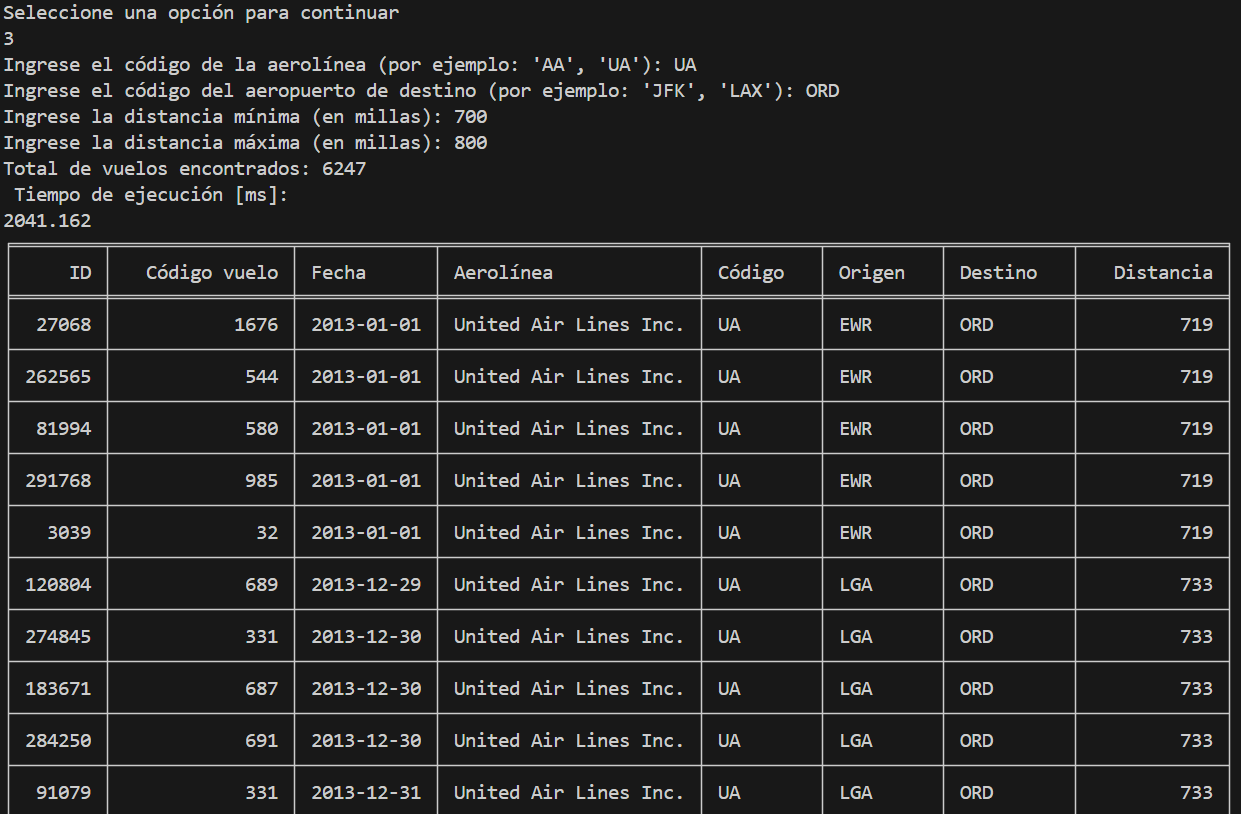
El algoritmo busca los vuelos de una aerolínea hacia un destino específico dentro del catálogo. Luego filtra los vuelos cuya distancia esté dentro de un rango dado y los almacena en una lista temporal. Una vez filtrados, ordena los vuelos por distancia y, en caso de empate, por fecha y hora de llegada usando ***merge sort***. Finalmente, devuelve el tiempo de ejecución, la cantidad filtrada y la lista ordenada.

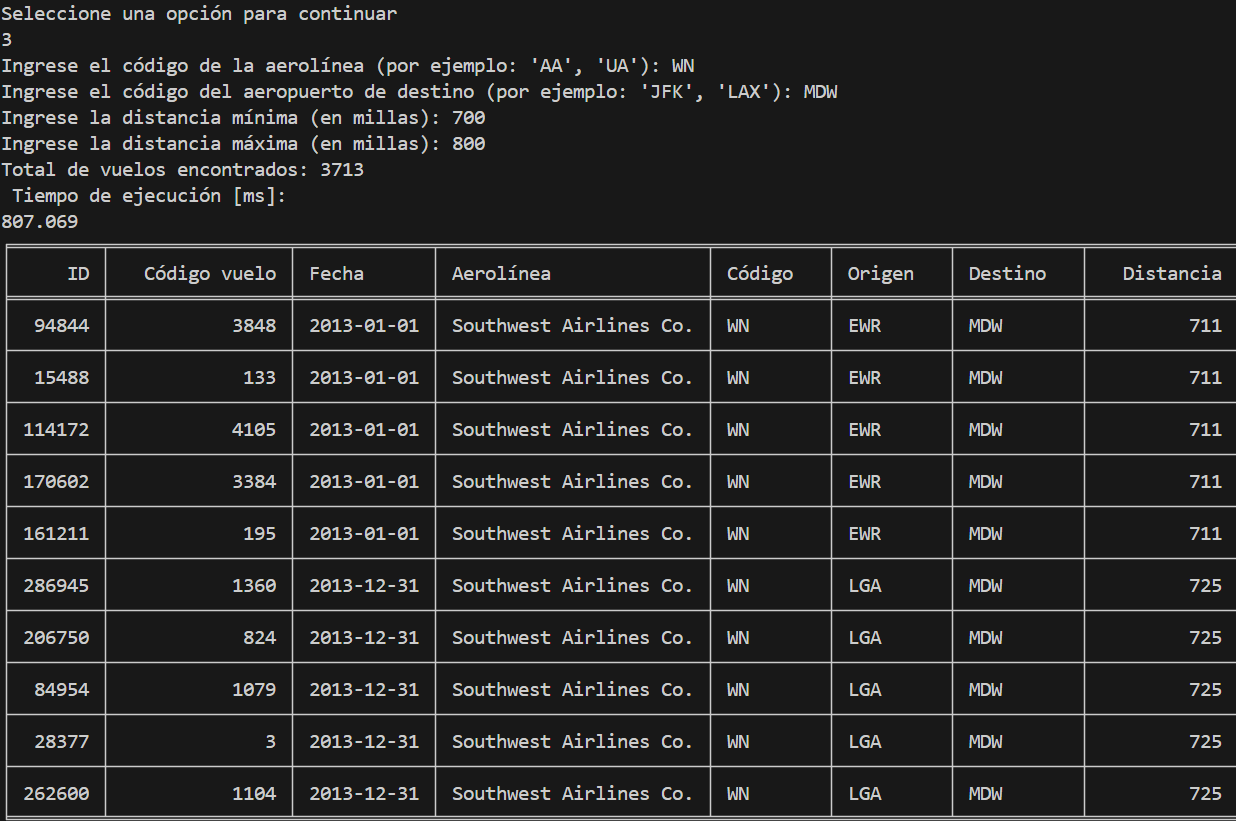
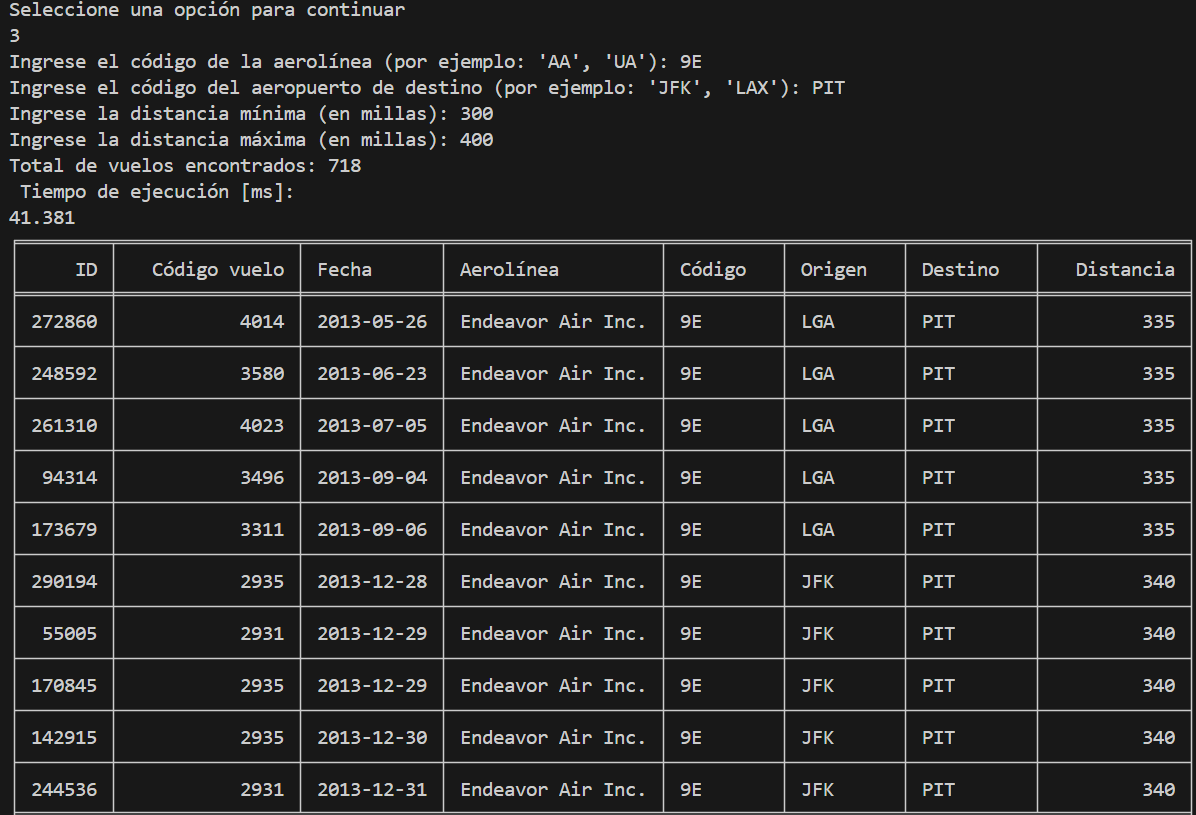
## **Pruebas (Al menos 5)**

Hechos con large:

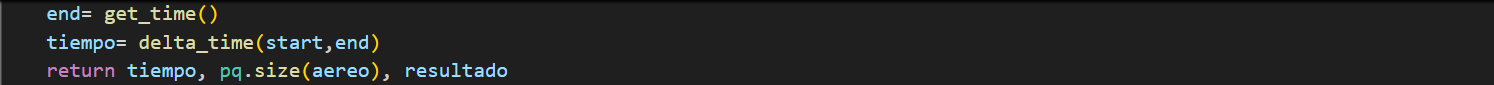








# **Requerimiento 4**



## **Descripción**

|  |  |
| --- | --- |
| **Entrada** | • Código de la aerolínea a analizar (por ejemplo: “UA”).  • Rango de minutos de retraso en salida a filtrar (por ejemplo: [10,30]) |
| **Salidas** | -Tiempo de la ejecución del requerimiento en milisegundos.  -Número total de vuelos que cumplen con el filtro de la aerolínea y rango de retraso.  -Teniendo en cuenta los vuelos que cumplen el filtro, presente los siguientes datos:   * o ID del vuelo. * o Código del vuelo. * o Fecha. * o Nombre de la aerolínea. * o Código de la aerolínea. * o Aeropuerto de origen. * o Aeropuerto de destino. * o Minutos de retraso en la salida. |
| **Implementado (Sí/No)** | Si, por Juan Andres Lozada Barragan |

## **Análisis de complejidad**

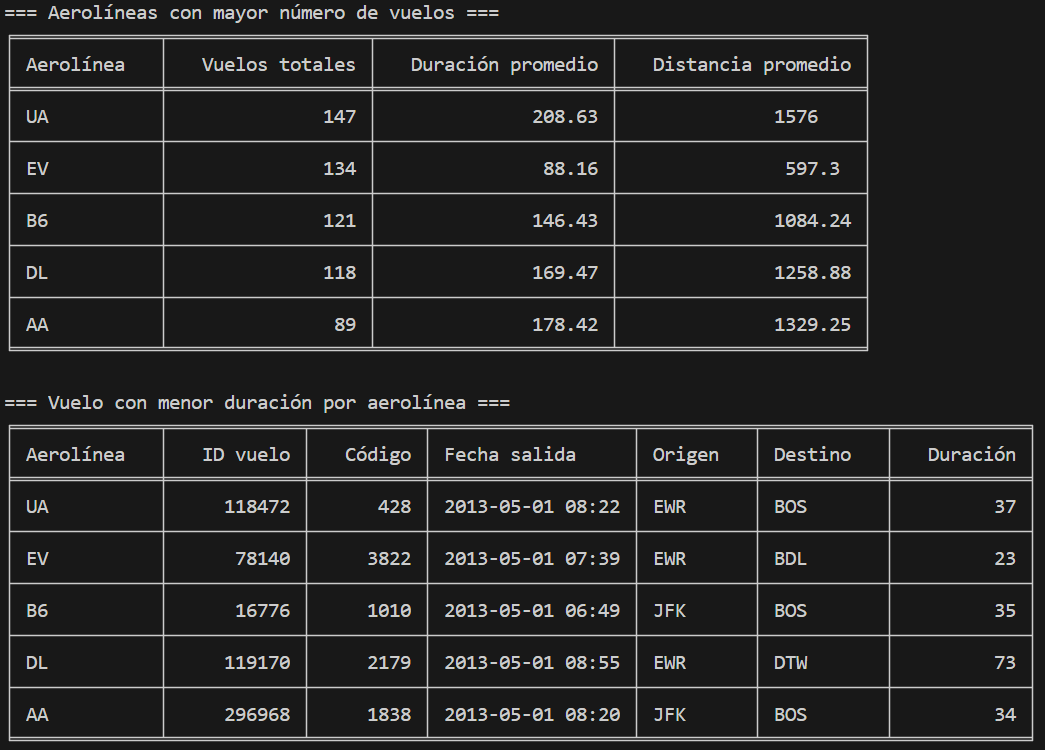
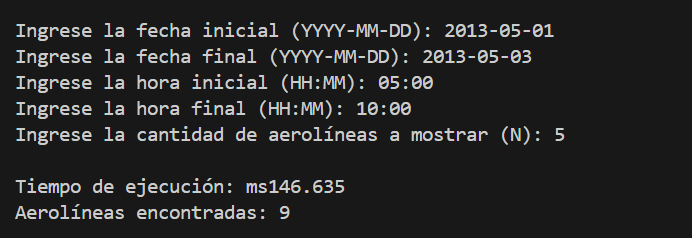
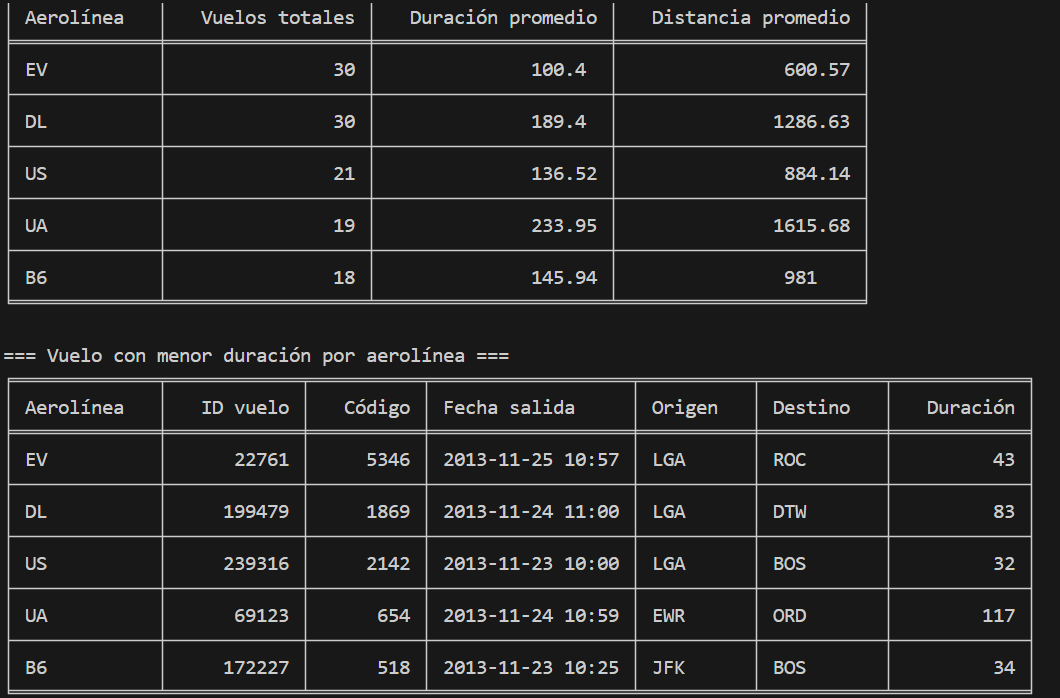
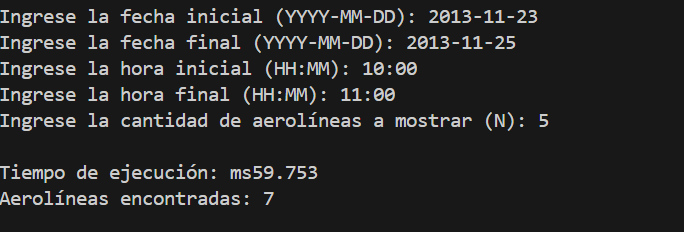
Análisis de complejidad de cada uno de los pasos del algoritmo

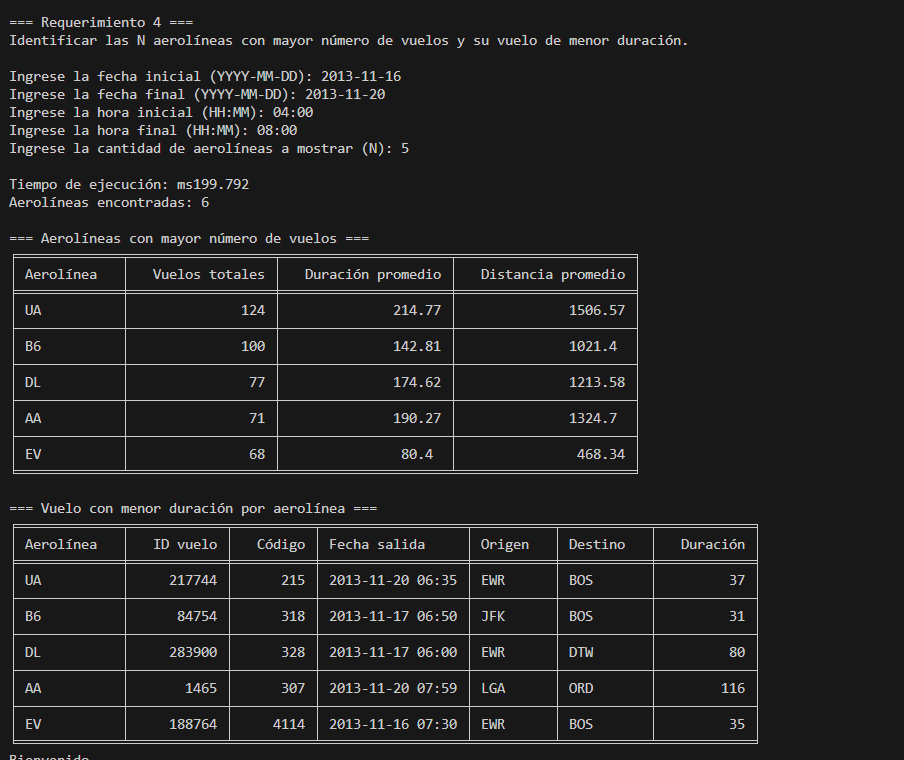
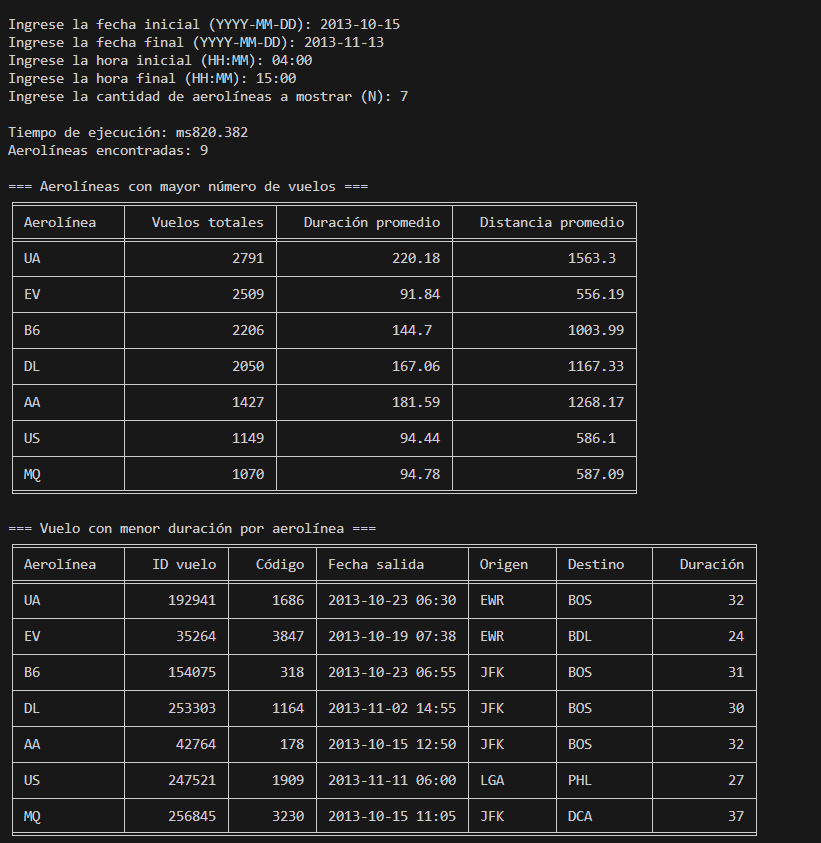
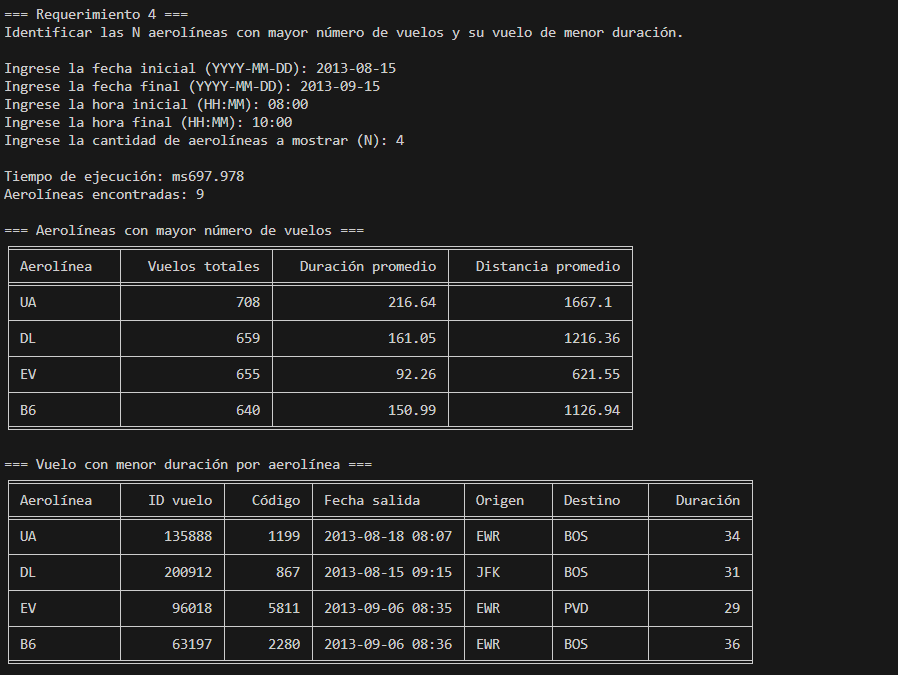
|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Filtrar los vuelos por rango de fechas y horas | O(n) |
| Agrupar los vuelos por aerolínea y acumular valores | O(n) |
| Insertar las aerolíneas en un heap limitado por cantidad de valores | O(k log k) |
| Calcular promedios y hallar la aerolínea con menor duración | O(k) |
| Cálculo final y retorno de resultados | O(1) |
| ***TOTAL*** | O(n + k log k) |

## **Análisis**

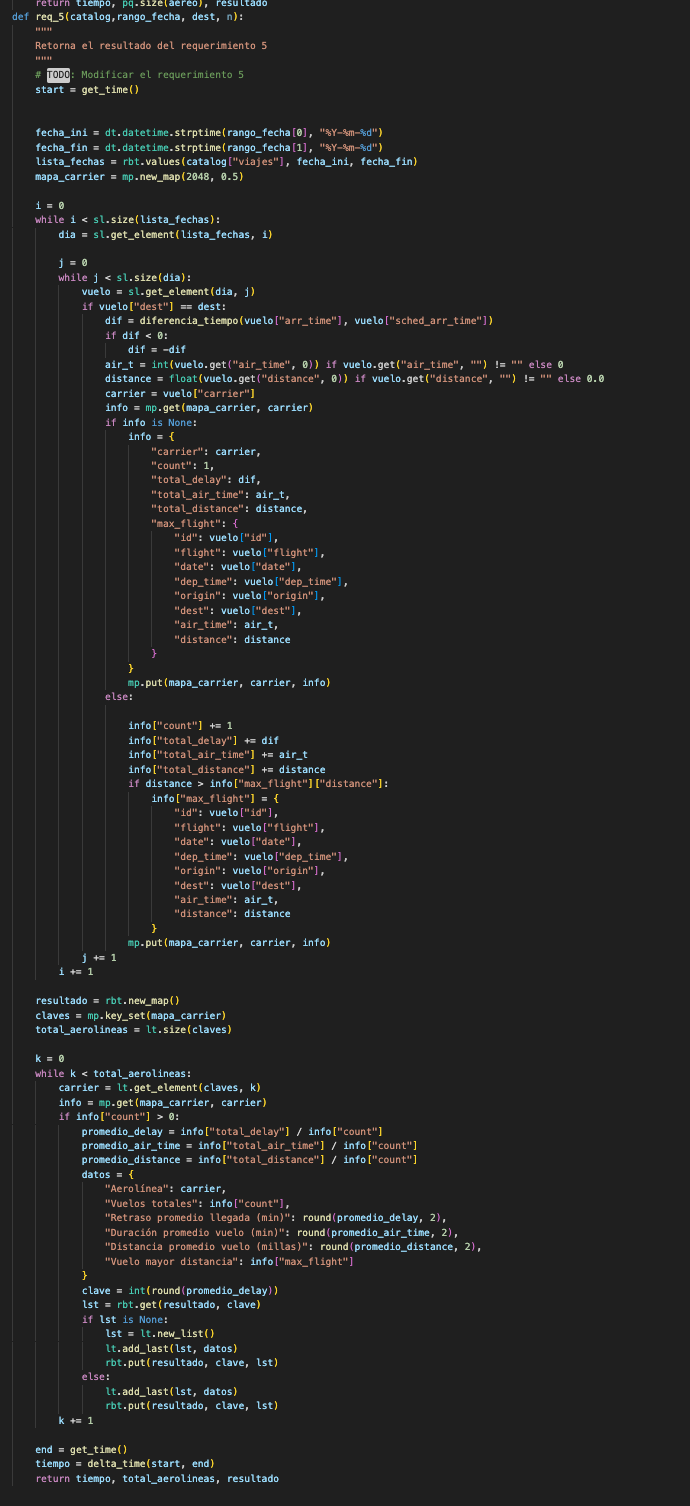
El algoritmo analiza los vuelos en un rango de fechas y horas, agrupándolos por aerolínea para calcular promedios de duración y distancia. Luego usa un heap para obtener las aerolíneas con más valores registrados y determinar la de menor duración promedio. Recorre todos los vuelos una vez (O(n)), realiza operaciones logarítmicas en el heap (O(k log k)) y devuelve estadísticas resumidas. Es eficiente, ya que su costo principal crece linealmente con el número de vuelos.

## **Pruebas (Al menos 5)**

Hechos con large:



# **Requerimiento 5**



## **Descripción**

Este requerimiento toma datos de vuelos dentro de un rango de fechas y destino especifico y los agrupa por aerolinea y los ordena por el retraso promedio de llegada.

|  |  |
| --- | --- |
| **Entrada** | * Código del aeropuerto de destino a analizar * Rango de minutos de anticipo en la llegada a filtrar |
| **Salidas** | * Tiempo de ejecución * Número total de vuelos que cumplen con el filtro del aeropuerto y del rango de anticipo * Vuelos que cumplen el filtro:   + Id   + Código   + Fecha   + Nombre   + Codigo aero   + Aeropuerto origen   + Aeropuerto destino   + Minutos anticipo en salida |
| **Implementado (Sí/No)** | Si, por Maria Clara Quijano |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

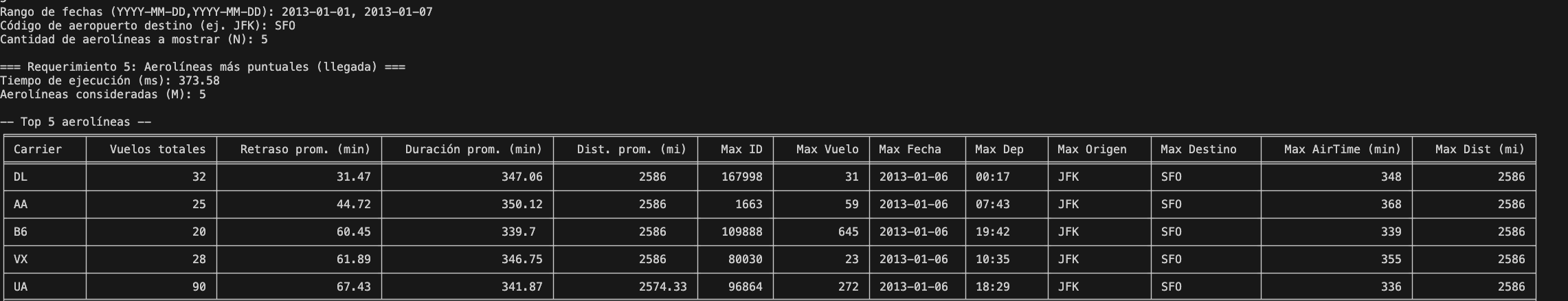
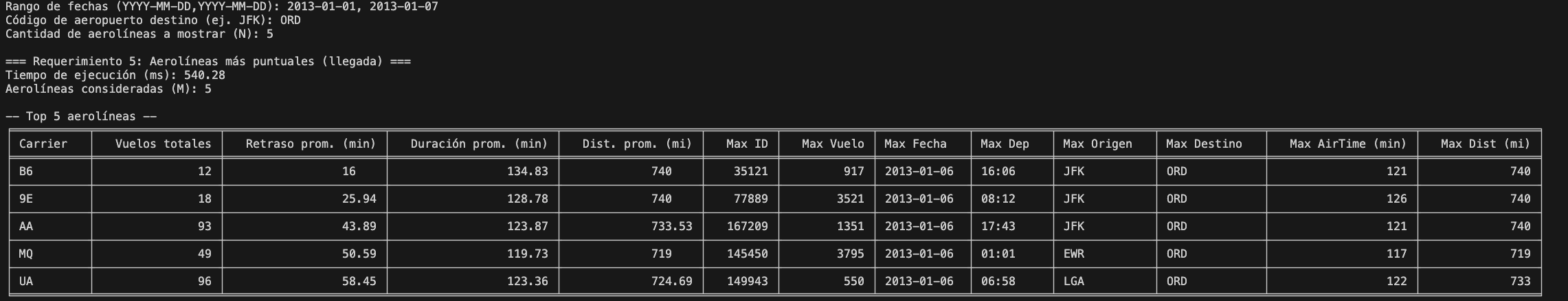
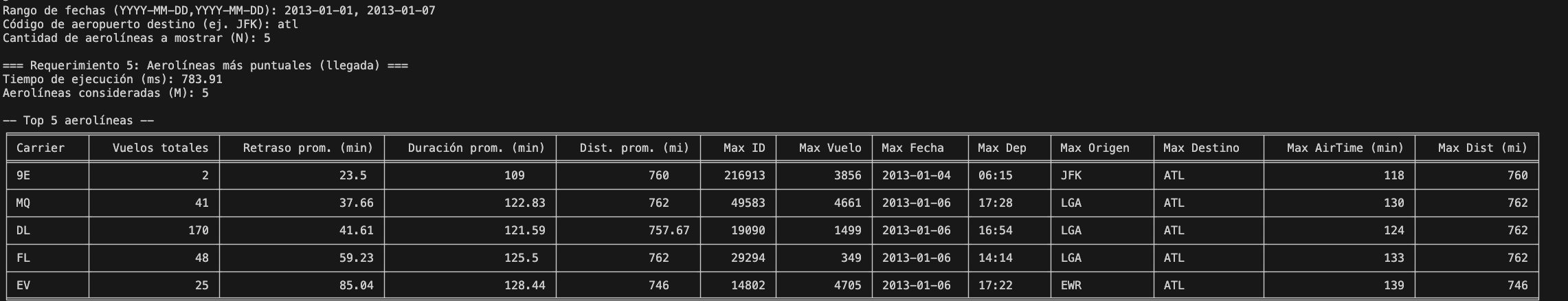
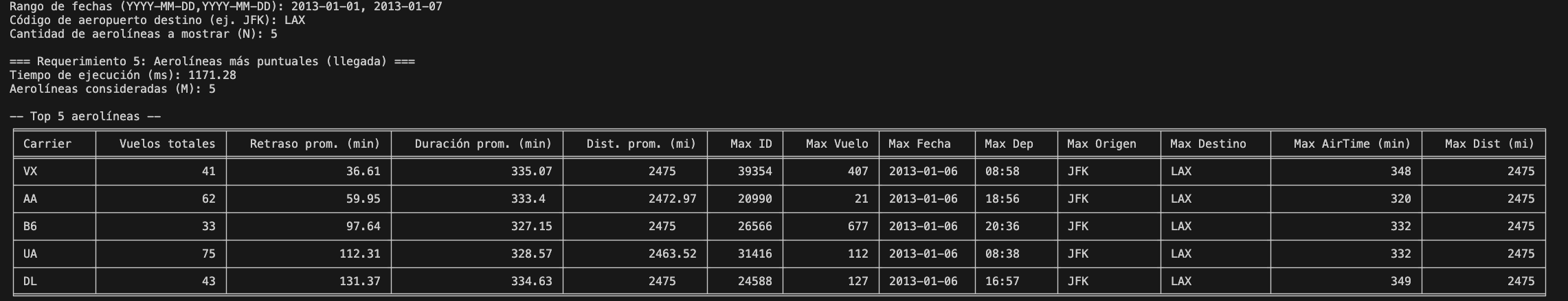
|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Vuelos dentro de rango del RBT | O(d+N) |
| Inicialización/búsqueda/Inserción hash map | O(1) |
| Bucle sobre días | O(d) |
| *Filtro por destino* | O(1) |
| *Obtener aerolíneas únicas y bucle* | O(A) |
| *Búsqueda/put rbt* | O(LogR) |
| ***TOTAL*** | **O(N+A\*LogR)** |

## **Análisis**

Se usa un hash map para tener mayor eficiencia al algregar datos por su complejidad promedio de O(1). Se usa un RBT para ordenar automaticamente los datos por el retraso promedio, y tiene una complejidad de O(LogR).

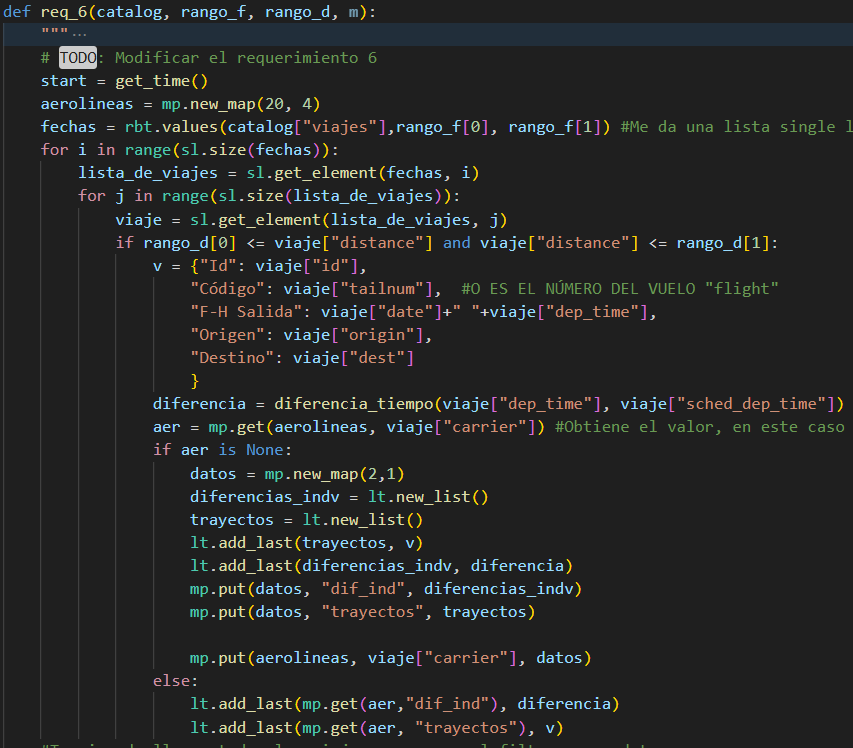
## **Pruebas (Al menos 5)**

Hechos con large:



# **Requerimiento 6**

## **Descripción**

Dado un rango de fechas y distancias, determinar las M aerolíneas más estables en su hora de salida, considerando la puntualidad de sus vuelos.Texto

El contenido generado por IA puede ser incorrecto.

|  |  |
| --- | --- |
| **Entrada** | • Rango de fechas a analizar.  • Rango de distancias (en millas) a analizar.  • Cantidad M de aerolíneas a mostrar. |
| **Salidas** | Tiempo de la ejecución del requerimiento en milisegundos.  Número total de aerolíneas analizadas (M).  Para cada una de las aerolíneas:   * Código de la aerolínea. * Número total de vuelos analizados en el rango especificado. * Promedio del retraso/anticipo en la salida de sus vuelos, en minutos. * Estabilidad de la salida (desviación estándar del retraso/anticipo en la salida) en minutos. * Vuelo con el retraso más cercano al promedio:   + ID del vuelo   + Código del vuelo   + Fecha-Hora de salida del vuelo   + Aeropuerto de origen   + Aeropuerto de destino |
| **Implementado (Sí/No)** | Sí, por Juliana Rodríguez |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Lista de valores según un rango (values) | O(1) |
| Primer recorrido for (lista de listas) | O(n) |
| Obtener elemento de una lista single | O(n) |
| Segundo recorrido for (viajes) | O(m) |
| Obtener elemento dentro de lista single | O(m) |
| Get de un mapa x3 | O(r/R) |
| Recorrido del nuevo mapa con los vuelos filtrados | O(r) |
| Recorrido for para hallar promedio | O(l) |
| Recorrido for para hallar desviación | O(l) |
| Insert de cola de prioridad | O(1+log r) |
| ***TOTAL*** | ***O(n\*m\*r/R + r/R\*l\*(1+log r))*** |

## **Análisis**

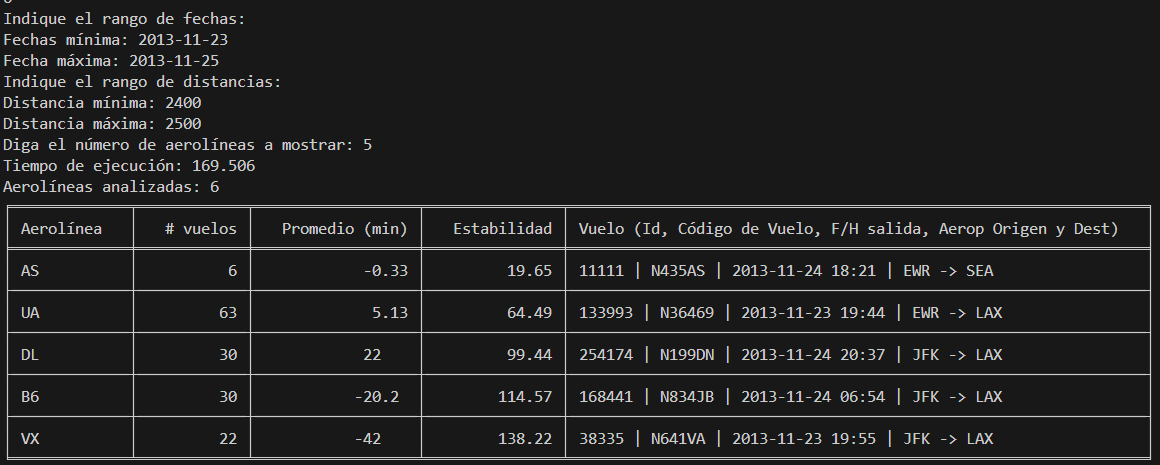
La función de divide en dos partes, la primera analiza los vuelos que cumplen los criterios y guarda la información de estos y en la segunda parte se realiza el cálculo del promedio y desviación estándar ya que con esto se presentarán los datos (además de ser el propósito de la función).

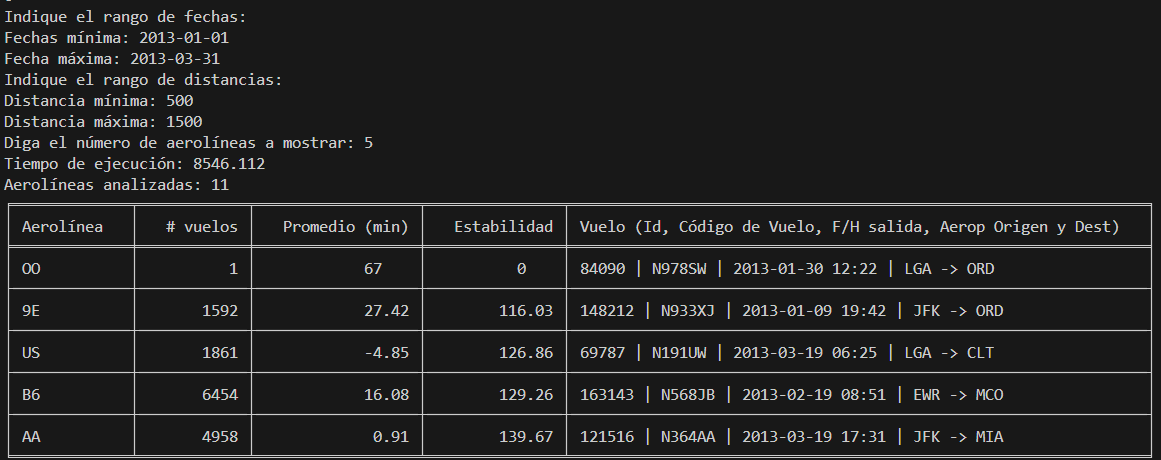
En la primera parte del árbol con las fechas primero se realiza una extracción de las fechas dentro del rango dado por parámetro, lo que genera una single\_linked, es decir una lista de listas. Para poder recorrer todas las listas (que son los valores de las fechas) y recorrer todos los trayectos dentro de estas se hizo dos recorridos for (por eso n x m); de allí se analizaron todos los vuelos filtrando por el rango de distancias (pero como este se hizo con un if y llamado de llaves no aporta a mayor complejidad), acá se guardó la información necesaria según lo que pedía el req. Se creó un mapa para poder guardar dos listas, una con la lista de las diferencia entre el tiempo de salida real y programado (para usarlo más adelante en el cálculo del promedio y desviación) y otra con la información del viaje.

En la segunda parte se recorre el mapa antes creado para poder recorrer las listas de las diferencias, así, en el primero for se calculaba el promedio por aerolínea, después se hizo otro for para poder calcular la desviación estándar –La razón de no haber hecho el cálculo en un mismo for para ahorrar complejidad es porque la desviación estándar se calcula por cada viaje y se necesita del promedio. En el mismo se calculó el viaje que tenía menos diferencia con el promedio, pues era parte del req. Finalmente se ajustaron los datos para su presentación y se añadieron en una cola de prioridad para poder organizarlo más fácil.

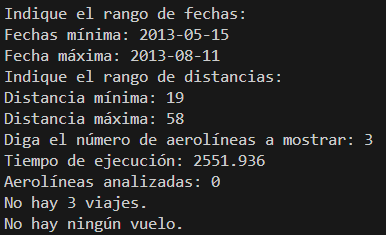
## **Pruebas**

Hechas con large:





Texto

El contenido generado por IA puede ser incorrecto.

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

Promedio de tiempo: 3530.057 ms