**ANÁLISIS DEL RETO 2**

Juliana Rodríguez Morales – 202421552 – js.rodriguezm1234

Maria Clara Quijano - 202420069 - m.quijanoa

Juan Andrés Lozada - 202510410-j.lozadab

# **Requerimiento <<n>>**

## **Descripción**

Breve descripción de como abordaron la implementación del requerimiento

|  |  |
| --- | --- |
| **Entrada** | Parámetros necesarios para resolver el requerimiento. |
| **Salidas** | Respuesta esperada del algoritmo. |
| **Implementado (Sí/No)** | Si se implementó y quien lo hizo. |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Paso 1 | O(...) |
| Paso 2 | O(...) |
| Paso …. | O(...) |
| ***TOTAL*** | ***O(...)*** |

## **Análisis**

Análisis de resultados de la implementación, tener cuenta las pruebas realizadas y el analisis de complejidad.

# **Requerimiento Ejemplo**

## **Descripción**



Este requerimiento se encarga de retornar un dato de una lista dado su ID. Lo primero que hace es verificar si el elemento existe. Dado el caso que exista, retorna su posición, lo busca en la lista y lo retorna. De lo contrario, retorna None.

|  |  |
| --- | --- |
| **Entrada** | Estructuras de datos del modelo, ID. |
| **Salidas** | El elemento con el ID dado, si no existe se retorna None |
| **Implementado (Sí/No)** | Si. Implementado por Juan Andrés Ariza |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Buscar si el elemento existe (isPresent) | O(n) |
| Obtener el elemento (getElement) | O(1) |
| ***TOTAL*** | ***O(n)*** |

## **Análisis**

A pesar de que obtener un elemento en un *ArrayList,* dada su posición, tiene complejidad constante, la implementación de este requerimiento tiene un orden lineal O(n). Esto debido a que, lo primero que se hace es verificar si el elemento hace parte de la lista. Específicamente, a la hora de buscar un elemento en una lista, en el peor de los casos es necesario recorrer toda la lista, es decir, complejidad lineal.

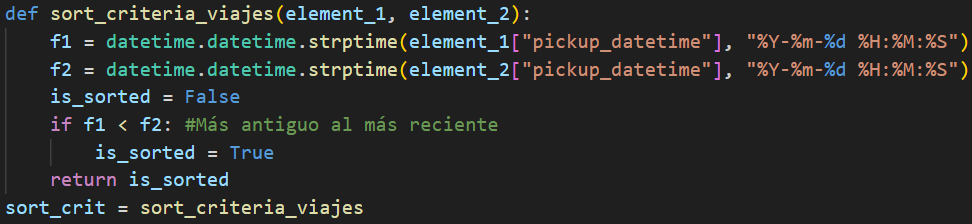
# **Requerimiento 1**

## **Descripción**

Función req\_1Texto

El contenido generado por IA puede ser incorrecto.

Función auxiliar de sort\_criteria para el quick\_sort del req\_1



A partir de un rango de fechas dado (fecha inicial y final de recogida) se deben mostrar los primeros n viajes y últimos n viajes previamente ordenados del más antiguo al más reciente.

La función de sort criteria viajes compara fecha y hora de recogida para organizar del más antiguo al más reciente al momento de utilizar quick\_sort.

|  |  |
| --- | --- |
| **Entrada** | Fecha y hora inicial de recogida  Fecha y hora final de recogida  Tamaño de muestra |
| **Salidas** | Tiempo de ejecución  # de trayectos dentro del rango de fecha/hora  N primeros y N últimos:  Fecha y hora de inicio  Coordenadas iniciales  Fecha y hora final  Coordenadas finales  Distancia recorrida (millas)  Costo total pagado |
| **Implementado (Sí/No)** | Sí, Juliana Rodríguez |

## **Análisis de complejidad**

Solo se analizará parte por parte el req\_1, no sort\_crit.

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Recorrer la lista con los trayectos. O(n) | O(n) |
| Obtener elemento de Array | O(1) |
| Comparación de fechas en cada viaje | O(1) - por estar dentro del for -> O(n) |
| Añadir elemento en un Array | O(n) |
| Quick sort | O(m logm) |
| ***TOTAL*** | **O(n + m logm)** |

## **Análisis**

Para que el código fuera mucho más rápido decidimos usar array\_list para que funciones de obtener un elemento fuera mucho más rápido. Así mismo,

# **Requerimiento 2**

## **Descripción**

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.Breve descripción de como abordaron la implementación del requerimiento

|  |  |
| --- | --- |
| **Entrada** | Coordenada inicial de latitud del rango.  Coordenada final de latitud del rango.  Tamaño de la muestra (N) de trayectos a mostrar al principio y al final del rango |
| **Salidas** | Tiempo de ejecución  # de trayectos dentro del rango de fecha/hora  N primeros y N últimos:  Fecha y hora de inicio  Coordenadas iniciales  Fecha y hora final  Coordenadas finales  Distancia recorrida (millas)  Costo total pagado |
| **Implementado (Sí/No)** | Sí. Por Maria Clara Quijano |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Filtrado por rango de latitud O(n) | Filtrado por rango de latitud O(n) |
| Ordenamiento O(k^2) | Ordenamiento O(k^2) |
| Construcción de salida O(n) | Construcción de salida O(n) |
| ***TOTAL*** | ***O(n+k^2)*** |

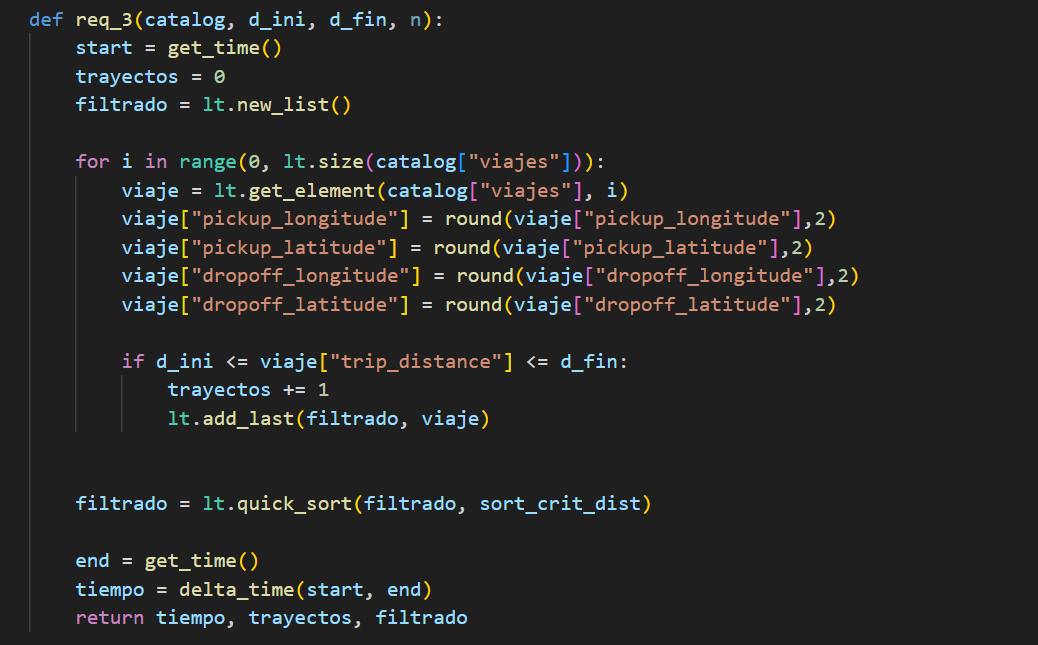
## **Análisis**

Se recorre toda la lista y luego se ordena usando quicksort todo lo que pasa el filtro. Cuando hay un rango mayor

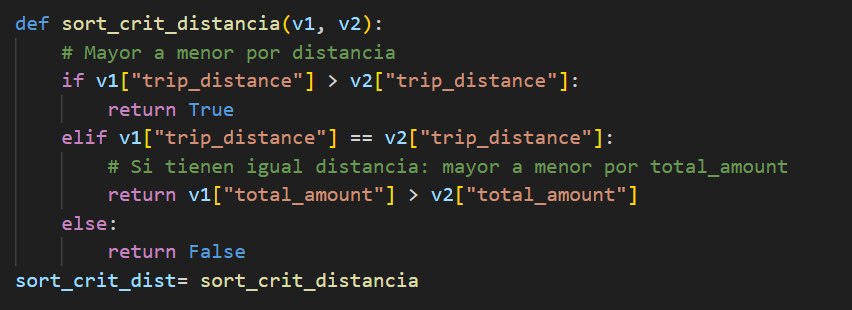
# **Requerimiento 3**

## **Descripción**

Req 3



Función auxiliar de criterio de ordenamiento en el req 3



El **req 3** filtra los trayectos cuya **distancia recorrida** esté entre un valor mínimo y máximo ingresado por el usuario. Luego, aplica una **función de criterio de ordenamiento** que compara los valores de la clave "trip\_distance" para organizar los viajes **de mayor a menor distancia**. Finalmente, muestra el tiempo de ejecución, la cantidad de trayectos encontrados, y presenta los **primeros y últimos viajes** del rango según su distancia.

|  |  |
| --- | --- |
| **Entrada** | Parámetros necesarios para resolver el requerimiento. |
| **Salidas** | Tiempo de ejecución  # de trayectos dentro del rango de fecha/hora  N primeros y N últimos:  Fecha y hora de inicio  Coordenadas iniciales  Fecha y hora final  Coordenadas finales  Distancia recorrida (millas)  Costo total pagado |
| **Implementado (Sí/No)** | Si, Juan Andres Lozada Barragan |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Paso 1: Recorrer todos los viajes y filtrar por distancia | O(n) |
| Paso 2: Agregar los viajes que cumplen el filtro a una nueva lista | O(n) |
| Paso 3: Ordenar la lista filtrada con quick sort | O(n log n) |
| ***TOTAL*** | O(n log n) |

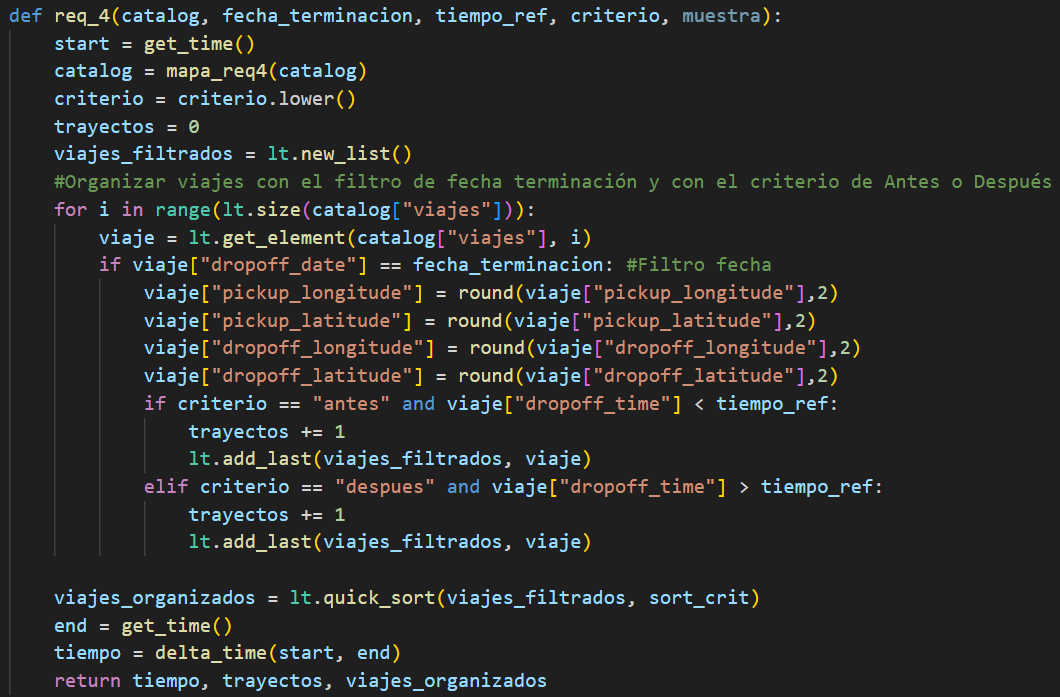
## **Análisis**

El algoritmo tiene una complejidad total **O(n log n)**, dominada por el proceso de ordenamiento.  
 El filtrado inicial de trayectos por distancia recorre la lista completa una vez (**O(n)**), mientras que quick sort organiza los resultados según la distancia y el costo, haciendo el proceso eficiente incluso con grandes volúmenes de datos. En pruebas, el tiempo de ejecución crece moderadamente al aumentar el número de trayectos.

# **Requerimiento 4**

## **Descripción**

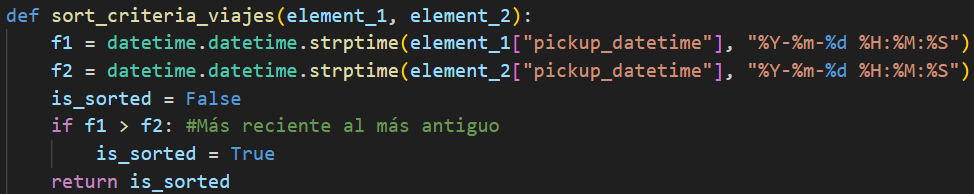
Función req\_4



Función creación de la Tabla Hash para el req\_4

Pantalla de computadora con letras

El contenido generado por IA puede ser incorrecto.

Función sort\_crit para la organización en el quick\_sort del req\_4

A partir de una fecha de terminación de trayecto dada debo filtrar todos los viajes que hayan terminado en esa fecha y mostrar todos los trayectos antes o después (criterio dado por el usuario) de una hora de terminación dada. Deben mostrarse los primeros n trayectos y últimos n trayectos previamente ordenados del más reciente al más antiguo.

La función mapa\_req4 crea la tabla de hash a utilizar para el cumplimiento del req\_4. La llaves son las fechas de terminación de los trayectos, y su valor una array\_list con todos los viajes que terminaron en esa fecha.

La función de sort criterio viajes compara fecha y hora de recogida para organizar del más antiguo al más reciente al momento de utilizar quick\_sort.

|  |  |
| --- | --- |
| **Entrada** | Fecha de terminación  Criterio de comparación (ANTES, DESPUES)  Tiempo de terminación  Tamaño de muestra |
| **Salidas** | Respuesta Tiempo de ejecución  # de trayectos dentro del rango de fecha/hora  N primeros y N últimos:  Fecha y hora de inicio  Coordenadas iniciales  Fecha y hora final  Coordenadas finales  Distancia recorrida (millas)  Costo total pagado |
| **Implementado (Sí/No)** | Sí, Juliana Rodríguez |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Paso 1 | O(...) |
| Paso 2 | O(...) |
| Paso …. | O(...) |
| ***TOTAL*** | ***O(...)*** |

## **Análisis**

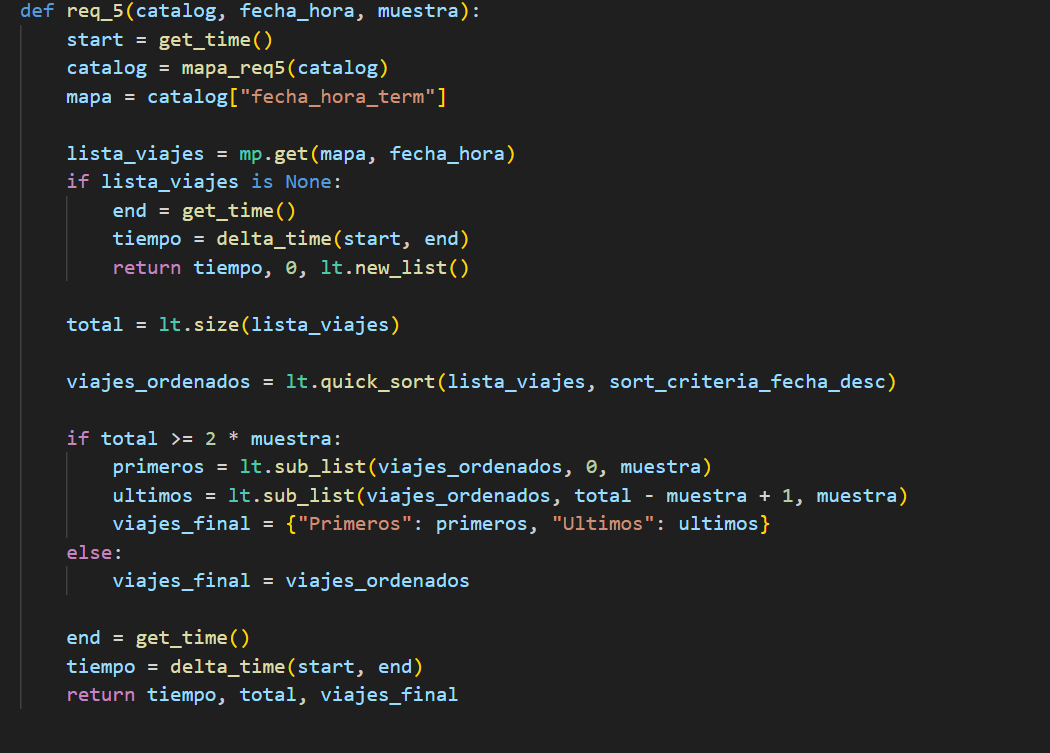
Análisis de resultados de la implementación, tener cuenta las pruebas realizadas y el analisis de complejidad.

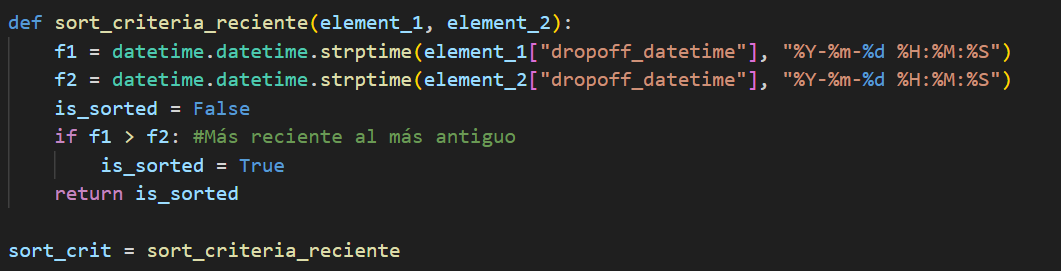
# **Requerimiento 5**

## **Descripción**

Función auxiliar de creacion del mapa del req 5.







Breve descripción de como abordaron la implementación del requerimiento

|  |  |
| --- | --- |
| **Entrada** | Parámetros necesarios para resolver el requerimiento. |
| **Salidas** | Tiempo de ejecución  # de trayectos dentro del rango de fecha/hora  N primeros y N últimos:  Fecha y hora de inicio  Coordenadas iniciales  Fecha y hora final  Coordenadas finales  Distancia recorrida (millas)  Costo total pagado |
| **Implementado (Sí/No)** | Si se implementó y quien lo hizo. |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Paso 1 | O(...) |
| Paso 2 | O(...) |
| Paso …. | O(...) |
| ***TOTAL*** | ***O(...)*** |

## **Análisis**

Análisis de resultados de la implementación, tener cuenta las pruebas realizadas y el analisis de complejidad.

# **Requerimiento 6**

## **Descripción**

Identificar los trayectos con recogida en un barrio de entrada y en un rango de horas de recogida. Solo se tiene en cuenta la hora de recogida. Se organizan de los trayectos más antiguos al más reciente.

|  |  |
| --- | --- |
| **Entrada** | Barrio de recogida (ej.: “Tribeca”, “Midtown”).  • Hora inicial del rango de recogida (formato "%H" ej.:"09").  • Hora final del rango de recogida (formato "%H" ej.:"12")  • Tamaño de la muestra (N) de trayectos a mostrar al principio y al final del rango |
| **Salidas** | · Tiempo de la ejecución del requerimiento en milisegundos.  · Número total de trayectos que cumplieron el filtro de barrio y hora de recogida.  · Mostrar la siguiente información de cada uno de los N primeros trayectos y de los N últimos trayectos en el rango:  o Fecha y tiempo de recogida  o Latitud y longitud de recogida  o Fecha y tiempo de terminación  o Latitud y longitud de terminación  o Distancia recorrida (millas)  o Costo total pagado |
| **Implementado (Sí/No)** | Sí, Maria Clara Quijano |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Mapa\_req6 | O(nxb) |
| Bucle principal | O(nxb) |
| Ordenamiento | O(k^2) |
| Construcción de salida | O(k) |
| ***TOTAL*** | ***O(nxb+k^2)*** |

## **Análisis**