**ANÁLISIS DEL RETO**

Juliana Rodríguez Morales – 202421552 – js.rodriguezm1234

Maria Clara Quijano - 202420069 - m.quijanoa

Juan Andrés Lozada - 202510410-j.lozadab

# **Requerimiento <<n>>**

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

## **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 análisis 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.

Este comportamiento se puede evidenciar experimentalmente en la gráfica. Ya que, gracias a que los datos no se encuentran tan dispersos con respecto a la línea de tendencia, la curva coincide con el comportamiento lineal esperado.

## **Requerimiento 1**

## **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

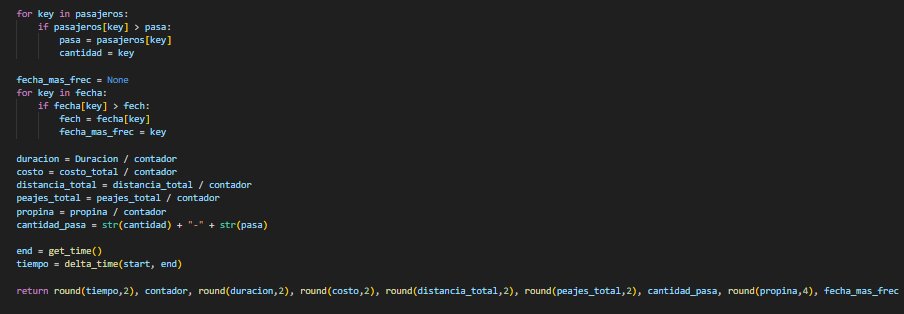
|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Paso 1 | O(...) |
| Paso 2 | O(...) |
| Paso …. | O(...) |
| ***TOTAL*** | ***O(...)*** |

## **Análisis**

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

# **Requerimiento 2**





### **Descripción**

Este requerimiento se encarga de filtrar los viajes según el método de pago y calcular estadísticas como duración promedio, costo, distancia, peajes, propina, cantidad de pasajeros más frecuente y fecha más común.

**Entradas:**

* Catálogo de viajes.
* Tipo de pago a filtrar (pago).

**Salidas:**

* Cantidad de viajes con ese pago.
* Promedio de duración.
* Promedio de costo, distancia, peajes y propina.
* Cantidad de pasajeros más frecuente.
* Fecha más común de finalización de viajes.

**Implementado (Sí/No):** Sí, implementado por Juan Andrés Lozada.

## **Análisis de complejidad**

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Recorrer la lista de viajes y filtrar por tipo de pago | O(n) |
| Calcular estadísticas de la lista filtrada | O(n) |
| Determinar pasajero y fecha más frecuentes | O(m), con *m* ≤ *n* |
| ***TOTAL*** | ***O(n)*** |

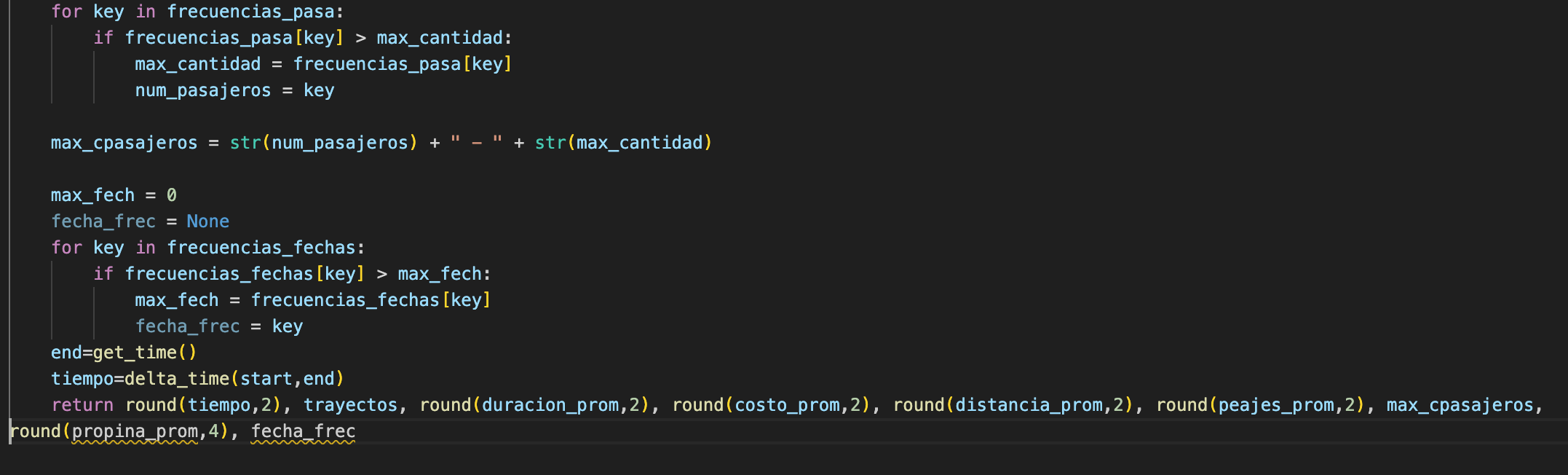
### **Análisis**

A pesar de que el requerimiento realiza varios cálculos (promedios, conteos de frecuencia y acumulaciones), todos dependen de recorrer los viajes una sola vez para filtrar y otra para acumular. El uso de diccionarios permite contar en tiempo constante promedio, por lo que no aumenta la complejidad.  
 El comportamiento es **lineal respecto al número de viajes** (O(n)), lo cual se refleja en un crecimiento proporcional al tamaño de los datos.

# **Requerimiento 3**

## **Descripción**





La función evalúa los trayectos que cumplen con el filtro de precios y luego se van sumando los valores que se tienen que sacar el promedio y retornar. Adicionalmente, se encuentra el número de pasajeros que es más frecuente en estos trayectos.

|  |  |
| --- | --- |
| **Entrada** | Catálogo de viajes, máxima cantidad que se paga por un viaje, y mínima cantidad que se paga por un viaje. |
| **Salidas** | * Tiempo promedio de la duración de los trayectos * precio total promedio de los trayectos * distancia promedio de los trayectos * precio promedio pagado en peajes de los trayectos * número y cantidad de pasajeros más frecuente en los trayectos * cantidad de propina promedio pagada en los trayectos * fecha de finalización de trayecto con mayor frecuencia. |
| **Implementado (Sí/No)** | Si. Implementado por Maria Clara Quijano |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

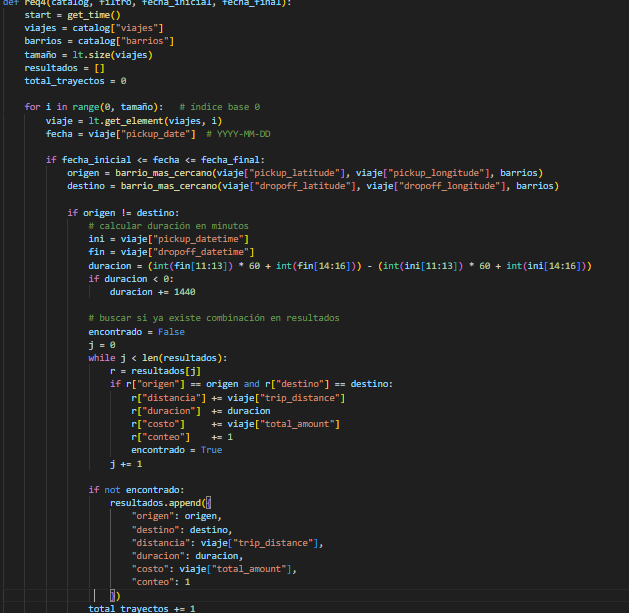
|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Recorrer lista y buscar si los elementos cumplen con el filtro | O(n) |
| Calculo Promedios | O(1) |
| Calcular las frecuencias de la cantidad de pasajeros | O(m) |
| Buscar fecha mas frecuente | O(p) |
| ***TOTAL*** | ***O(n) -> m y p en el peor caso son n*** |

## **Análisis**

La complejidad de la función depende del tamaño de la lista debido a que se deben recorrer todos los viajes para poder implementar el filtro. Adicionalmente, para encontrar las frecuencias, el peor caso que puede llegar a ser es n ya que esa es la cantidad de elementos en la lista. Los elementos que pasen el filtro son n o una cantidad menor.

# **Requerimiento 4**

### **Descripción**





Este requerimiento busca, en un rango de fechas dado, las combinaciones de barrios de origen y destino, acumulando distancia, duración y costo de los viajes. Finalmente, selecciona la combinación con **mayor o menor costo promedio** según el filtro indicado.

**Entradas:**

* Catálogo de viajes y barrios.
* Filtro (MAYOR o MENOR).
* Fecha inicial y fecha final.

**Salidas:**

* Trayecto con mayor/menor costo promedio.
* Distancia y duración promedio de ese trayecto.
* Cantidad total de trayectos considerados.

**Implementado (Sí/No):** Sí, implementado por Juan Andres Lozada Baragan.

## **Análisis de complejidad**

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Recorrer todos los viajes y verificar rango de fechas | O(n) |
| Determinar barrio más cercano (origen y destino) | O(k) k=num de barrios |
| Actualizar/insertar combinación en la lista de resultados | O(r), siendo r ≤ n |
| Seleccionar combinación con mejor costo promedio | O(r) |
| ***TOTAL*** | O(n·k + r) ≈ O(n·k) |

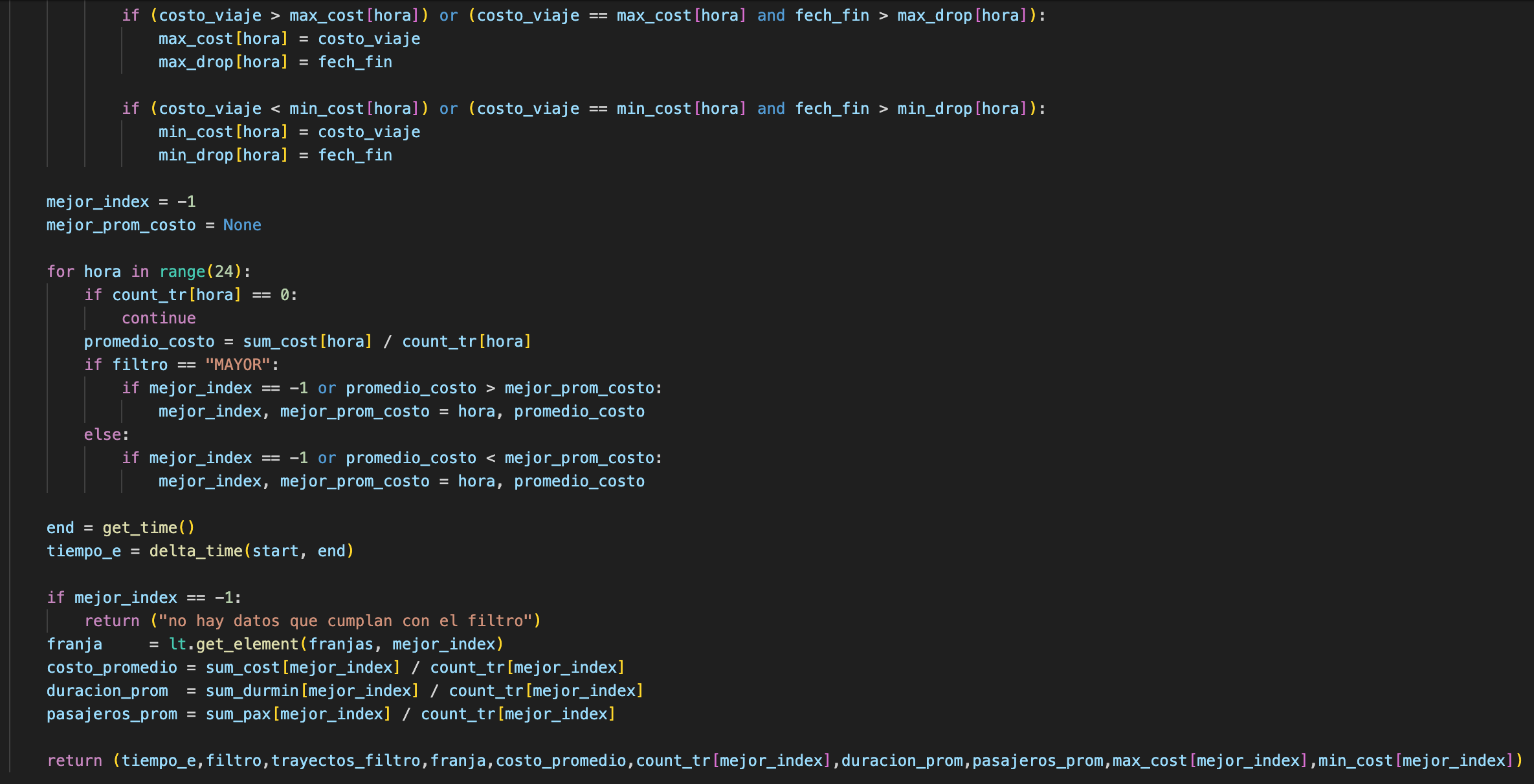
### **Análisis**

El paso más costoso es la búsqueda de barrios más cercanos, ya que para cada viaje se comparan coordenadas contra todos los barrios (*k*). Así, la complejidad total se aproxima a **O(n·k)**. La acumulación en resultados y la búsqueda de la mejor combinación aportan un costo adicional **O(r)**, pero dominado por el término principal.

En resumen, la complejidad depende linealmente del número de viajes y barrios, siendo más pesada que el requerimiento 2, pues involucra operaciones anidadas sobre las dos estructuras.

# **Requerimiento 5**

## **Descripción**



La función busca la franja de hora del día en el cual tiene un costo promedio mayor o menor, dependiendo del filtro que se busca. Retorna la información promedio de esa franja horaria.

|  |  |
| --- | --- |
| **Entrada** | Catálogo de viajes, filtro de selección del costo, fecha inicial del periodo a consultar, fecha final del periodo a consultar |
| **Salidas** | * Tiempo de la ejecución del requerimiento * Filtro de selección del costo * Número total de trayectos que cumplieron el filtro de fechas * Franja horaria * Costo promedio de los trayectos de la franja horaria * Número de trayectos incluidos en la franja horaria * Tiempo promedio de duración de los trayectos en la franja horaria * Cantidad promedio de pasajeros transportados por trayecto * Costo total del trayecto con mayor costo en la franja horaria * Costo total del trayecto con menor costo en la franja horaria |
| **Implementado (Sí/No)** | Si. Implementado por Maria Clara Quijano |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Asignación variables | O(1) |
| Recorrer viajes y verificar fechas | O(n) |
| Seleccionar la franja con mayor o menor costo promedio | O(1)-> se recorren 24 veces |
| ***TOTAL*** | ***O(n)*** |

## **Análisis**

Se tiene una complejidad O(n) por el filtro inicial que se le tiene que hacer a la lista, pero después se hace un recorrido 24 veces, una por cada franja de tiempo para seleccionar la franja que tiene mayor o meenor costo promedio, dependiendo de lo que le pidan. Los arreglos tienen un tamaño fijo (24), debido a que se guarda información para determinadas franjas de tiempo.