Proyecto 1 --- Lista de requerimientos

|  |  |
| --- | --- |
| **Nombre** | R1. Cargar los datos del archivo **GeoJSON**. |
| **Resumen** | Se leen los datos del archivo GeoJSON y se crea una lista doblemente encadenada donde se guardan los comparendos. La lectura del archivo debe realizarse según el formato esperado (véase especificaciones de diseño en la guía). |
| **Entradas** | |
| **Ruta** donde será leído el archivo en formato GeoJSON. | |
| **Salidas** | |
| Este requerimiento no genera ninguna salida. | |
| **Resultados** | |
| Se crea una lista doblemente encadenada con los comparendos. | |
| **Complejidad** | |
| Se estima una complejidad , es decir, una **complejidad lineal**. | |

|  |  |
| --- | --- |
| **Nombre** | R2. Consultar el primer comparendo que aparezca en el archivo que tiene una localidad dada. |
| **Resumen** | Se busca en la lista doblemente encadenada (con el algoritmo más eficiente y de menor complejidad) el primer comparendo que empiece por una localidad dada. En caso de que no se encuentre un comparendo por dicha localidad se debe reportar que NO existe información al respecto. |
| **Entradas** | |
| **Localidad** del comparendo buscado. | |
| **Salidas** | |
| **Primer comparendo** por la localidad dada. Si no se encuentra, se informa al usuario. | |
| **Resultados** | |
| Se busca en la lista el comparendo y se retorna al usuario el comparendo. En caso contrario se le informa que no se encontró. | |
| **Complejidad** | |
| Se estima una complejidad , es decir, una **complejidad lineal**. | |

|  |  |
| --- | --- |
| **Nombre** | R3. Consultar los comparendos registrados dados una fecha y hora |
| **Resumen** | Se recorre la lista y se crea una lista doblemente encadenada donde se guardan los comparendos ocurridos en la fecha y hora dada. Al final la lista se ordena de mayor a menor por el código de infracción, este ordenamiento debe tener el algoritmo más eficiente y óptimo; se debe mostrar los datos en el formato especificado (véase especificaciones de diseño). |
| **Entradas** | |
| **Fecha y hora** para buscar los comparendos. Se busca la cantidad de infracciones en esa fecha y hora. | |
| **Salidas** | |
| Retorna una **Lista** doblemente encadenada con los comparendos que se encuentran en dicha fecha y hora. | |
| **Resultados** | |
| Se retorna una Lista doblemente encadenada al usuario con los comparendos ordenados de mayor a menor según infracción que se encuentran en dicha hora y fecha. | |
| **Complejidad** | |
| Se estima una complejidad , es decir, una **complejidad linearítmica**. | |

|  |  |
| --- | --- |
| **Nombre** | R4. Consultar los comparendos registrados por dos fechas y horas dadas. |
| **Resumen** | Se recorre la lista y se crea una lista doblemente encadenada donde se guardan los comparendos en las fechas y horas dadas; las infracciones deben quedar ordenadas de mayor a menor en orden alfabético utilizando el algoritmo más óptimo y eficiente. |
| **Entradas** | |
| **Dos fechas y horas** para buscar los comparendos en ambas fechas. Se busca la cantidad de infracciones en esas fechas. | |
| **Salidas** | |
| Retorna un **String** según el formato dado en las especificaciones de diseño. | |
| **Resultados** | |
| Se retorna String al usuario (una tabla) con los comparendos en ambas fechas y en el formato especificado según diseño. Para ello se crea una lista encadenada que guardará localmente los comparendos para luego solicitar información, esta lista se encuentra ordenada alfabéticamente de mayor a menor según infracción. Si no hay comparendos en ambas fechas no se reporta una infracción. | |
| **Complejidad** | |
| Se estima una complejidad , es decir, una **complejidad cuadrática**. | |
| **Nombre** | R5. Consultar el primer comparendo que aparezca en el archivo que tiene una infracción dada. |
| **Resumen** | Se busca en la lista doblemente encadenada (con el algoritmo más eficiente y de menor complejidad) el primer comparendo que empiece por una infracción dada. En caso de que no se encuentre un comparendo por dicha infracción se debe reportar que NO existe información al respecto. |
| **Entradas** | |
| **Infracción** del comparendo buscado. | |
| **Salidas** | |
| **Primer comparendo** por la infracción dada. Si no se encuentra, se informa al usuario. | |
| **Resultados** | |
| Se busca en la lista el comparendo y se retorna al usuario el comparendo. En caso contrario se le informa que no se encontró. | |
| **Complejidad** | |
| Se estima una complejidad , es decir, una **complejidad lineal**. | |

|  |  |
| --- | --- |
| **Nombre** | R6. Consultar los comparendos registrados dados una infracción ordenados por fecha y hora. |
| **Resumen** | Se recorre la lista y se crea una lista doblemente encadenada donde se guardan los comparendos que concuerden con la infracción. Al final la lista se ordena de menor a mayor por la fecha y hora, este ordenamiento debe tener el algoritmo más eficiente y óptimo; se debe mostrar los datos en el formato especificado (véase especificaciones de diseño). |
| **Entradas** | |
| **Infracción** para buscar los comparendos. Se busca la cantidad de comparendos con dicha infracción. | |
| **Salidas** | |
| Retorna una **Lista** doblemente encadenada con los comparendos que se encuentran con dicha infracción. | |
| **Resultados** | |
| Se retorna una Lista doblemente encadenada al usuario con los comparendos ordenados de menor a mayor según fecha y hora que se encuentran en dicha infracción. | |
| **Complejidad** | |
| Se estima una complejidad , es decir, una **complejidad linearítmica**. | |

|  |  |
| --- | --- |
| **Nombre** | R7. Comparar comparendos por tipo de servicio. |
| **Resumen** | Se recorre la lista y se crea una lista doblemente encadenada donde se guardan los comparendos de tipo “Particular” y tipo “Público” de menor a mayor por orden alfabético según el tipo de infracción. |
| **Entradas** | |
| **No hay entradas para este requerimiento** | |
| **Salidas** | |
| Retorna un **String** según el formato dado en las especificaciones de diseño. | |
| **Resultados** | |
| Se retorna String al usuario (una tabla) con los comparendos en ambos tipos y en el formato especificado según diseño. Para ello se crea una lista encadenada que guardará localmente los comparendos para luego solicitar información, esta lista se encuentra ordenada alfabéticamente de menor a mayor según tipo de infracción. Si no hay comparendos en ambos tipos no se reporta una infracción. | |
| **Complejidad** | |
| Se estima una complejidad , es decir, una **complejidad cuadrática**. | |

|  |  |
| --- | --- |
| **Nombre** | R8. Dar número comparendos por infracción según localidad y rango de fecha. |
| **Resumen** | Se recorre la lista y se crea una lista doblemente encadenada donde se guardan los comparendos de la localidad dada y que se encuentran en el rango de fecha, se retornan de menor a mayor por orden alfabético según el tipo de infracción. |
| **Entradas** | |
| **Tipo de infracción** que será buscada. **Fecha y hora inicial, Fecha y hora final** son las dos fechas que tendrán el rango. | |
| **Salidas** | |
| Retorna un **String** según el formato dado en las especificaciones de diseño. | |
| **Resultados** | |
| Se retorna String al usuario (una tabla) con el número de comparendos encontrados en la infracción de la localidad en el rango de fecha y hora y en el formato especificado según diseño. Si no hay comparendos en ambos tipos no se reporta una infracción. | |
| **Complejidad** | |
| Se estima una complejidad , es decir, una **complejidad cuadrática**. | |
| **Nombre** | R9. Dar N ranking de comparendos con más infracciones en un rango de fecha y hora. |
| **Resumen** | Se recorre la lista y se crea una lista doblemente encadenada donde se guardan los comparendos según código con mayor número de infracciones, es decir, de mayor a menor. Posteriormente se da al usuario un número N de la lista comparendos, mostrando el código de la infracción y el número de veces que se repitió. |
| **Entradas** | |
| **N** posiciones a mostrar al usuario. **Fecha y hora inicial, Fecha y hora final** son las dos fechas que tendrán el rango. | |
| **Salidas** | |
| Retorna un **String** según el formato dado en las especificaciones de diseño. | |
| **Resultados** | |
| Se retorna String al usuario (una tabla) con el número código de la infracción y el número de veces que dicha infracción fue repetida en el rango de fecha y hora determinado. Por otro lado, se muestra un número N de infracciones en ranking de mayor a menor, según cantidad de repeticiones. | |
| **Complejidad** | |
| Se estima una complejidad , es decir, una **complejidad cuadrática**. | |

|  |  |
| --- | --- |
| **Nombre** | R10. Generar tabla ASCII que de el número total de comparendos por cada localidad. |
| **Resumen** | Se recorre la lista y se crea un arreglo dinámico de enteros que guarda el número de comparendos por localidad. Luego se pasa dicho número a un número determinado de “\*” según especificaciones de diseño. |
| **Entradas** | |
| No tiene entradas. | |
| **Salidas** | |
| Retorna un **String** según el formato dado en las especificaciones de diseño. | |
| **Resultados** | |
| Se retorna String al usuario (una tabla) con la localidad y un número determinado de “\*” según el número de comparendos encontrados. Si no existen comparendos en una localidad se muestra “sin comparendos”. Finalmente se muestra el número total de comparendos. | |
| **Complejidad** | |
| Se estima una complejidad , es decir, una **complejidad cuadrática**. | |

**ACLARACIÓN ADICIONAL:**

Se comenta que muchos de estos requerimientos funcionales pueden no estar completamente optimizados (como si lo estarán al final del proyecto). Por ende, algunos algoritmos tienen complejidades demasiado elevadas como ; al llegar al proceso de implementación del código muchos de estos aspectos se “pulirán” y darán un terminado más eficiente y elegante al proyecto.