ESTRUCTURAS DE DATOS

UNIVERSIDAD DE LOS ANDES

Entrega de Diseño – Proyecto 2

**Autores**: Nicolás H. Orjuela Pava - 201913579

Nicolás González - 201713213

**Grupo**: 4

**Sección**: 3

**ÍNDICE**:

1. Requerimientos Carga de Información
2. Requerimientos – Parte A
3. Requerimientos – Parte B
4. Requerimientos – Parte C
5. Diseño de las Estructuras de Datos
6. Diseño de la Solución
7. Carga de Información

Al no haber una especificación en las instrucciones de entrega, se realizó la carga solamente en una Hash Table Separate Chaining. Con el fin de cumplir con los siguientes dos requerimientos.

|  |  |
| --- | --- |
| **Identificador** | 1CI |
| **Resumen** | Muestra el total de comparendos en el archivo |
| **Entradas** | |
| - | |
| **Resultados** | |
| Se muestra el total de comparendos en el archivo | |
| **Estructura de datos seleccionada** | |
| Los datos se cargan en cualquiera de las 3 estructuras de datos. Al irlos agregando a cada estructura, se va aumentando un contador en los métodos Add() o similares de cada estructura. | |
| **Desempeño** | |
| En las 3 estructuras hay un método agregar en el que hay un contador que se incrementa al haber un nuevo Objecto agregado a la misma. Por lo tanto, la complejidad durante la carga para mostrar el total de comparendos en el archivo es O(n), pues se va agregando cada comparendo uno por uno, luego, al agregar un solo comparendo, la complejidad de add() es O(1). | |

|  |  |
| --- | --- |
| **Identificador** | 2CI |
| **Resumen** | Muestra la información del comparendo con mayor OBJECTID encontrado. |
| **Entradas** | |
| - | |
| **Resultados** | |
| Se muestra la información (OBJECTID, FECHA\_HORA, INFRACCION, CLASE\_VEHICULO, TIPO\_SERVICIO, LOCALIDAD) del comparendo con mayor OBJECTID encontrado. | |
| **Estructura de datos seleccionada** | |
| Se puede usar cualquier estructura para cargar sus datos. En todas, lo que se hace es que durante la carga de cada comparendo (que es uno a uno) se va seleccionando el mayor OBJECTID, por lo tanto, al final de la carga ya se sabe cuál es el mayor y no es necesario hacer la búsqueda otra vez. Esto sea realiza directamente en el Controller. | |
| **Desempeño** | |
| Ocurre al final de la carga, por lo tanto, depende del tiempo que se demore en cargar los datos O(n). Luego, al ya haber una variable con el mayor comparendo, si se agregara otro, solo se tendría que comparar con esta variable para saber cuál es el mayor de todos, por lo tanto la complejidad de este nuevo proceso es O(1). | |

1. Requerimientos – Parte A

|  |  |
| --- | --- |
| **Identificador** | 1A – Obtener M comparendos con mayor gravedad |
| **Resumen** | Se obtienen los M comparendos con mayor gravedad |
| **Entradas** | |
| Int M | |
| **Resultados** | |
| Se muestran los M comparendos con mayor gravedad. | |
| **Estructura de datos seleccionada y desempeño** | |
| Es posible usar una priority queue con el fin de poder seleccionar los comparendos con mayor gravedad, luego de que se seleccionen se procede a ordenarlos para poder mostrarlos en consola. La complejidad de esta selección es O(nlog(n)). | |

|  |  |
| --- | --- |
| **Identificador** | 2A – Buscar los comparendos por mes y día de la semana |
| **Resumen** | Se buscan y muestran los N comparendos que cumplan los criterios de búsqueda. |
| **Entradas** | |
| Int Mes: (1-12)  String DíaSemana: (L,M,I,J,V,S,D) | |
| **Resultados** | |
| Se muestran los N comparendos de ese mes y día de la semana ingresado por el usuario. | |
| **Estructura de datos seleccionada y desempeño** | |
| La mejor estructura para este requerimiento es una tabla de hash (linear probing). Esto se debe a que es posible almacenar ciertos atributos (mes y día) en llaves con el fin de obtener el comparendo que cumpla el criterio de búsqueda. Luego se retornan los comparendos en consola. La complejidad de este proceso es O(n). | |

|  |  |
| --- | --- |
| **Identificador** | 3A – Buscar los comparendos que tiene una fecha-hora en un rango y que son de una localidad data. |
| **Resumen** | Se buscan y muestran los N comparendos que cumplan los criterios de búsqueda. |
| **Entradas** | |
| Rango de fecha-hora: límite\_bajo, límite\_alto en formato YYY/MM/DD-HH:ss  Localidad | |
| **Resultados** | |
| Se muestran los N comparendos que cumplan los criterios de búsqueda. Se muestra el id, tipo de servicio, infracción, fecha-hora y clase de vehículo. | |
| **Estructura de datos seleccionada y desempeño** | |
| Se puede usar una cola de prioridad para este requerimiento con el fin de recorrer cada posición del arreglo (hoja del heap), con el fin que encontrar todos los comparendos que cumplan el criterio de búsqueda. Esta operaci´no toma O(nlog(n)) según nuestras aproximaciones. | |

1. Requerimientos – Parte B

|  |  |
| --- | --- |
| **Identificador** | 1B – Buscar los M comparendos más cercanos a la estación de policía. |
| **Resumen** | Se busca y muestra los M comparendos más cercanos a la estación de policía del Campín. |
| **Entradas** | |
| - | |
| **Resultados** | |
| Se muestra el id, tipo de servicio, infracción, fecha-hora, clase de vehículo, longitud y latitud geográficas de los M comparendos más cercanos a la estación de policía del Campín. | |
| **Estructura de datos seleccionada y desempeño** | |
| Es posible usar una priority queue con el fin de encontrar los comparendos que se encuentra más cerca a la estación de policía. Se comparan los comparendos por medio de un comparator que compara por longitud y latitud, para calcular la distancia se usa el cógido Haversine entre dos puntos. La complejidad de este proceso es O(nlog(n)). | |

|  |  |
| --- | --- |
| **Identificador** | 2B- Buscar los comparendos por medio de detección, clase de vehículo, tipo de servicio y localidad. |
| **Resumen** | Se buscan y muestran los N comparendos que cumplan los criterios de búsqueda ordenados por fecha. |
| **Entradas** | |
| Medio de detección, clase de vehículo, tipo de servicio, localidad. | |
| **Resultados** | |
| Se muestra en consola los N comparendos que cumplan los criterios de búsqueda ordenados por fecha. Se muestra el id, tipo de servicio, infracción, fecha-hora, clase de vehículo y localidad para cada comparendo. | |
| **Estructura de datos seleccionada y desempeño** | |
| Se puede usar una tabla de hash (linear probing) pues permite almacenar los atributos del comparendo necesarios para hacer la comparaciones del requerimiento en llaves. Luego, se accede a la llave teniendo en cuenta los datos ingresados por el usuario y se imprime en consola la información de los comparendos que cumplan con los criterios. La complejidad es O(1). | |

|  |  |
| --- | --- |
| **Identificador** | 3B – Buscar los comparendos que tienen una latitud en un rango dado y que involucraron un tipo de vehículo particular. |
| **Resumen** | Se muestra y busca los N comparendos que cumplan los criterios de búsqueda. |
| **Entradas** | |
| Rango de latitudes: límite\_bajo, límite\_alto  Clase de vehículo: automóvil, camioneta,, moto | |
| **Resultados** | |
| Se muestran los N comparendos que cumplan los criterios de búsqueda. De cada comparendo se muestra el id, tipo de servicio, infracción, fecha-hora, clase de vhículoy latitud geográfica. | |
| **Estructura de datos seleccionada y desempeño** | |
| Se puede usar una hash table (linear probing) con el fin de encontrar los comparendos que cumplan con los criterios de búsqueda y establecer en las llaves los atributos que permiten realizar este proceso para poder acceder a los mismos. La complejidad es de O(1). | |

1. Requerimientos – Parte C

|  |  |
| --- | --- |
| **Identificador** | 1C – Visualizar Datos en una Tabla ASCII |
| **Resumen** | Se muestran los datos en una tabla ASCII |
| **Entradas** | |
| Número de Días D | |
| **Resultados** | |
| Se muestran los datos del requerimiento en una tabla ASCII. | |
| **Estructura de datos seleccionada y desempeño** | |
| Se puede usar una tabla de hash con el fin de mostrar los comparendos del número de días según el requerimiento para luego pasar a mostrarlos en consola por medio de una tabla ASCII. La complejidad de este proceso es O(nlog(n)). | |

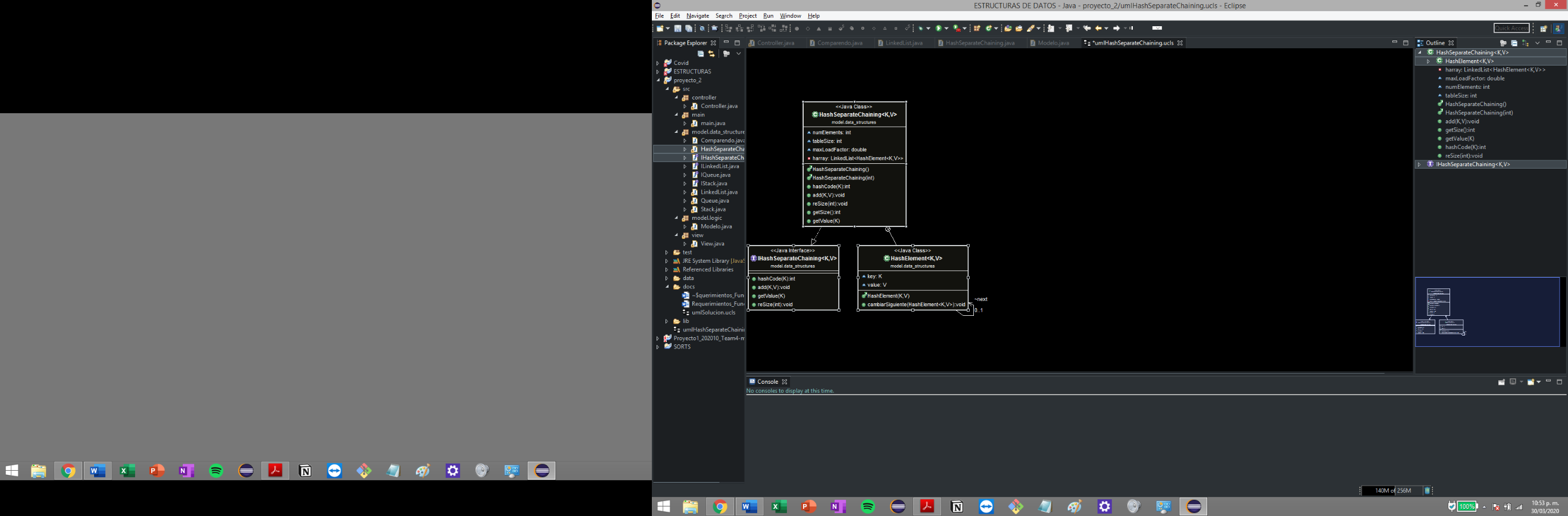
|  |  |
| --- | --- |
| **Identificador** | 2C – El costo de los tiempos de espera hoy en día (cola) |
| **Resumen** | Se muestran en consola los datos del requerimiento. |
| **Entradas** | |
| - | |
| **Resultados** | |
| Se muestra en consola el costo total que generan las penalizaciones en 2018  El número de días en promedio que debe esperar un comparendo.  Un histograma ASCII con el número de comparendos procesados por día y el número de comparendos que están esperando. Por día solo se pueden procesar manualmente máximo 1500 comparendos. | |
| **Estructura de datos seleccionada y desempeño** | |
| Se puede usar una priority queue para establecer cuales comparendos son los que generan las penalizaciones en 2018, teniendo en cuenta la prioridad de los mismos para que o se cobre la multa, y la cantidad máxima de comparendos que se pueden procesar por día (1500). | |

|  |  |
| --- | --- |
| **Identificador** | 3C – El costo de los tiempos de espera usando el nuevo sistema |
| **Resumen** | Se hace lo mismo que el requerimiento 2C pero con el nuevo sistema. |
| **Entradas** | |
| - | |
| **Resultados** | |
| Mismos resultados que el requerimiento 2C pero usando el nuevo sistema. | |
| **Estructura de datos seleccionada y desempeño** | |
| Se puede usar una priority queue con el fin de establecer cuales comparendos tienen prioridad para que no se les cobre la multa de acuerdo con el tipo de infracción y la cantidad máxima de comparendos (1500) al día. Complejidad de O(nlog(n)). | |

4C – Conclusión Final: (requerimiento de la entrega 2 del proyecto).

1. Diseño de Estructuras de Datos

Como para la primera entrega (diseño) solo se usó para cargar los datos una Hash Table Separate Chaining, pues para la carga de los datos y sus requerimientos el enunciado no especificaba implementar todas las estructuras, se mostrará a continuación el diagrama UML de esta estructura.



1. Diseño de la Solución

Se presentará a continuación un diagrama se clases UML relacionado con la primera entrega del proyecto.

