Universidad de los Andes *Marzo, 2019*
Estructuras de Datos *ISIS 1206*
Daniel Yamin *Proyecto 2*
Sebastian Puerto

---

1. ***Requerimientos Funcionales:*** A continuación se presenta un resumen de los requerimientos funcionales del proyecto que se desarrollara:

Nombre	R1A: Obtener ranking de las N franjas horarias que tengas más infracciones
Resumen	Teniendo en cuenta que hay 24 franjas horarias en un día (franjas de 1 hora), se debe encontrar las N franjas con mayor cantidad de infracciones. Además, para cada una de las franjas, se informa al usuario sobre el total de infracciones, el porcentaje de infracciones sin accidente, el porcentaje de infracciones con accidente y el valor total a pagar por las infracciones.
Entradas	
N - El número de franjas horarias que se desean obtener	
Resultados	
Para cada una de las N franjas horarias, se obtiene: 1. Total de infracciones en la franja 2. El porcentaje de infracciones sin accidente 3. El porcentaje de infracciones con accidente 4. Valor total a pagar por las infracciones	
Complejidad	O(N log N) la primera vez, O(M) en el resto de llamadas.

Nombre	R2A: Realizar el ordenamiento de las infracciones por localización geográfica
Resumen	Se ordenan las infracciones por localización geográfica (Xcoord,Ycoord) en una tabla Hash. Así, el usuario puede realizar una consulta por localización geográfica en la cual obtiene el total de infracciones, el porcentaje de infracciones sin accidente, el porcentaje de infracciones con accidente, el valor total a pagar por infracciones, el location, el AddressID y el StreetSegId.
Entradas	
XCoord: La coordenada X que se desea buscar	
YCoord: La coordenada Y que se desea buscar	
Resultados	
Para la coordenada geográfica (X,Y) ingresada por el cliente, se informa: 1. Total de infracciones 2. Porcentaje de infracciones sin accidente	

3. Porcentaje de infracciones con accidente 4. Valor total a pagar por infracciones 5. El location 6. El ID de la dirección (Address ID) 7. El ID de la calle (StreetSegID)	
Complejidad	O(N)

Nombre	R3A: Buscar las infracciones por rango de fechas [Fecha Inicial (Año/Mes/Día), Fecha Final (Año/Mes/Día)]
Resumen	Dado un rango de fechas ingresado por el usuario, para cada una de las fechas contenidas en el rango y que tengan al menos una infracción, se informa sobre el total de infracciones, el porcentaje de infracciones sin accidente, el porcentaje de infracciones con accidente y el valor total a pagar por las infracciones de dicha fecha.
Entradas	
Fecha Inicial - [Fecha Inicial (Año/Mes/Día)]	
Fecha Final - [Fecha Final (Año/Mes/Día)]	
Resultados	
Se imprime en la consola cada una de las fechas en el rango con al menos una infracción. Se informa, para cada una de estas fechas: 1. El total de infracciones 2. El porcentaje de infracciones sin accidente 3. El porcentaje de infracciones con accidente 4. Valor total a pagar por las infracciones	
Complejidad	O(N)

Nombre	R1B: Ranking de los N tipos de infracción (Violation Code) que tengan más infracciones.
Resumen	Se obtienen los N tipos de infracción que tengan más infracciones, donde N es un número dado por el usuario. Los tipos de infracciones se ordenan de mayor a menor según el número de infracciones y para cada una se muestra: 1. Código del tipo de infracción 2. Total de infracciones 3. Porcentaje de infracciones sin accidente 4. Porcentaje de infracciones con accidente 5. Valor total a pagar por infracciones
Entradas	
N - El número de franjas horarias que se desean obtener	
Resultados	

Se imprime en la consola los N tipos de infracciones solicitados donde además se informa el código, el total de infracciones, el porcentaje de infracciones sin accidente, el porcentaje de infracciones con accidente y el valor total a pagar por infracciones correspondientes al tipo de infracción en cuestión.	
Complejidad	O(N)

Nombre	R2B: Realizar el ordenamiento de las infracciones por localización geográfica
Resumen	Se ordena las infracciones por sus coordenadas geográficas, donde el criterio de ordenamiento es: (1) XCoord (2)YCoord. Luego, el usuario puede buscar una localización geográfica y obtiene la siguiente información sobre la misma: 1. Total de infracciones 2. Porcentaje de infracciones sin accidente 3. Porcentaje de infracciones con accidente 4. Valor total a pagar por las infracciones 5. El Location 6. El ID de la dirección (AddressID) 7. El ID de la calle (StreetSegID)
Entradas	
XCoord: La coordenada X que se desea buscar	
YCoord: La coordenada Y que se desea buscar	
Resultados	
Se imprime en consola, la siguiente información relacionada a las coordenada (XCoord, YCoord) ingresada por el usuario: ● Total de infracciones ● Porcentaje de infracciones sin accidente ● Porcentaje de infracciones con accidente ● Valor total a pagar por las infracciones ● El Location ● El ID de la dirección (AddressID) ● El ID de la calle (StreetSegID)	

Nombre	R3B:Buscar las franjas de fecha - hora donde se tiene un valor acumulado de infracciones en un rango dado
Resumen	A partir de una rango de valores dado por el usuario, se busca las fechas (hora) que tienen un valor acumulado de infracciones en el rango dado por el cliente. Para cada fecha - hora, se le informa al cliente sobre: valor ($) acumulado de infracciones, el total de infracciones, el porcentaje de infracciones sin accidente y el porcentaje de infracciones con accidente.
Entradas	
Valor Inferior - [US$ Valor Inicial]	

Valor Superior - [US$ Valor Final]	
Resultados	
Se imprime en la consola, la siguiente información para cada una de las fecha - hora cuyo valor acumulado de infracciones este en el rango suministrado por el cliente: 1. Valor total acumulado de infracciones 2. Total de infracciones 3. Porcentaje de infracciones sin accidente 4. Porcentaje de infracciones con accidente	
Complejidad	O(N)

Nombre	R1C: Obtener la información de una localización dada (AddressID)
Resumen	Para una localización dada por el usuario, se debe informar el total de infracciones, el porcentaje de infracciones sin accidentes, el porcentaje de infracciones con accidente, el valor total a pagar po las infracciones y el ID de la calle (StreetSegID).
Entradas	
La localización que se desea buscar - AddressID	
Resultados	
Para la localización (AddressID) ingresada por el cliente, se imprime en consola la siguiente información: 1. Total de infracciones 2. Porcentaje de infracciones sin accidente 3. Porcentaje de infracciones con accidente 4. .Valor total a pagar por las infracciones 5. EL ID de la calle (StreetSegID)	
Complejidad	O(N) en la primera llamada, O(1) en el resto.

Nombre	R2C: Obtener las infracciones en un rango de horas. .
Resumen	El usuario ingresa un rango de horas y para este se genera la siguiente información: ● **Información General:** Total de infracciones, porcentaje de infracciones sin accidente, porcentaje de infracciones con accidente y el valor total a pagar por las infracciones ● **Información por Código (ViolationCode):** Para las infracciones resultantes, se agrupan por código y se informa el total de infracciones con respectivo código.
Entradas	
Hora Inicial - [HH:MM:SS]	
Hora Final - [HH:MM:SS]	

Resultados	
Para el rango dado por el usuario, se imprime la información general y la información por código en la consola.	
Complejidad	O(N)

Nombre	R3C: Obtener el ranking de las N localizaciones geográficas con la mayor cantidad de infracciones
Resumen	Dado un valor N ingresado por el usuario, se obtienen las N localizaciones geográficas (XCoord, YCoord) con la mayor cantidad de infracciones ordenadas de mayor a menor. Para cada localización se reporta: el total de infracciones, el porcentaje de infracciones sin accidente, el porcentaje de infracciones con accidente, el location, el ID de la dirección (AddressID) y el ID de la calle (StreetSegID).
Entradas	
N - El número de localización que se desean obtener	
Resultados	
Se imprime en la consola las N localizaciones. Para cada una de ellas, se informa: 1. El total de infracciones 2. El porcentaje de infracciones sin accidente 3. El porcentaje de infracciones con accidente 4. El location 5. El ID de la dirección (AddressID) 6. El ID de la calle (StreetSegID)	
Complejidad	O(M) ya creada la tabla de hash del requerimiento 2A.

Nombre	R4C: Mostrar una gráfica ASCII con la información de las infracciones por código (ViolationCode).
Resumen	Se muestra un gráfica la cual para cada uno de los tipos de infracción (ordenados de mayor a menor) se gráfica el porcentaje de infracciones que ocurrieron de dicho tipo con respecto al total.
Entradas	
Resultados	
Se imprime en la consola una gráfica del siguiente estilo:	

Reporte Agregado de Infracciones por Código 2018 TXXX | ********** TYYY | ********** TZZZ | ********** ... Cada * representa un porcentaje K% (a definir). En la gráfica debe indicarse el porcentaje (%) al que corresponde cada *.	
Complejidad	O(N)

2. Estimación de Complejidad y Generalidades de los Requerimientos:

- **Requerimiento 1A.**
  - Input: N.
  - Output: cola con los M objetos de tipo (FRANJA, total infr., porcentaje sin acc., porcentaje con acc., valor total a pagar) con mayor infracciones totales.
  - Complejidad: O(N log N) la primera vez, O(M) en el resto de llamadas.
    - Se ordenan los datos cargados por hora: O(N log N)
    - Se crean los datos (FRANJA, total infr., porcentaje sin acc., porcentaje con acc., valor total a pagar): O(N)
    - Se crea una priority queue con los datos: O(24 log 24) = O(1) (El numero de franjas es 24)
    - Se extraen los M primeros: O(M)
- **Requerimiento 2A.**
  - Input: (xcoord, ycoord).
  - Output: objeto (total infr., porcentaje sin acc., porcentaje con acc., valor total a pagar, location, AddressID, StreetSegId).
  - Complejidad: O(N)
    - Crear tabla de hash de (xcoord, ycoord):(total infr., porcentaje sin acc., porcentaje con acc., valor total a pagar, location, AddressID, StreetSegId): O(1)
    - Recorrer la lista actualizando la tabla de hash con la informacion de la infracción actual: O(N log 5000) = O(N) (Hay un numero constante de AddressIDs, alrededor de 5000, que no crece al crecer el numero de datos, así que la inserción de elementos tiene complejidad máxima de log 5000)
- **Requerimiento 3A.**
  - Input: rango de fechas
  - Output: Arreglo Dinámico de objetos (total infr., porcentaje sin acc., porcentaje con acc., valor total a pagar)
  - Complejidad: O(N)
    - Crear árbol balanceado de FECHA: (total infr., porcentaje sin acc., porcentaje con acc., valor total a pagar) : O(1)

        - Recorrer lista e ir actualizando los datos del árbol balanceado: O(N) en peor caso
        - Buscar fechas en el rango dado: O(log 180) = O(1) (son alrededor de 180 fechas a en un semestre, esto no cambia con el número de datos)
        - Agregar a ArregloDinamico: O(180) = O(1)
        - Ordenar este arreglo: O(180 log 180) = O(1)
- **Requerimiento 1B**
    - Input:
    - Output:
    - Complejidad: O(N)
        - Se ordenan los datos por VOCode: O(N log #VOCodes) = O(N) usando un algoritmo estable que aproveche la redundancia de VOCodes)
        - Crear cola de prioridad para M primeros (VOCode, total infr., porcentaje sin acc., porcentaje con acc., valor total a pagar) con prioridad total infr.: O(#VOCodes log M) = O(log M)
- **Requerimiento 2B**
    - Input:
    - Output:
    - Complejidad: O(N)
        - Se crea un arbol balanceado de tipo (xcoord, ycoord):(total infr., porcentaje sin acc., porcentaje con acc., valor total a pagar, StreetSegId): O(1)
        - Se recorre la lista de datos, actualizando el árbol: O(N log 5000) = O(N)
- **Requerimiento 3B**
    - Input:
    - Output:
    - Complejidad: O(N)
        - Ordenar por hora: O(N log 24) = O(N)
        - Se agregan a árbol balanceado si están en el rango: O(N)
- **Requerimiento 1C.**
    - Estructuras a utilizar: La hashtable se usa ya que permite un rápido acceso e inserción a datos organizados de la forma llave, valor. Se usan tablas de hash, con AddressID como llave y objeto (total infr., porcentaje sin acc., porcentaje con acc., valor total a pagar, StreegtSegId) como valor, pues es una estructura que se construirá una vez, que permite una inserción y búsqueda de nuevos datos con complejidad O(log N) en el peor caso y constante en el promedio; el uso de espacio para las tablas de hash con linear probing y separate chaining es del mismo orden, pero es mayor para la implementación con separate chaining pues se hace uso de listas encadenadas, lo cual ocupa un mayor espacio que los arreglos con la misma información de los que hace uso la implementación con linear probing.
    - Input: AdressID
    - Output: (total infr., porcentaje sin acc., porcentaje con acc., valor total a pagar, StreegtSegId)
    - Complejidad: O(N) en la primera llamada, O (1) en el resto.

        - Se crea una tabla de hash con linear probing, con llave AdressID y con valores de tipo Objeto (total infr., porcentaje sin acc., porcentaje con acc., valor total a pagar, StreegtSegId): O(1)
        - Se recorre la lista de infracciones para actualizar la tabla de hash: O(N log 5000) = O(N) (el número de AddressIDs para la ciudad de Nueva York es constante, alrededor de 5000, así que la inserción de cada elemento tendrá complejidad máxima de log 5000)
        - Se extrae la información del AddressID deseado: O(1)
- **Requerimiento 2C**.
    - Estructuras a utilizar:
        - Información general: Hashtable con Linear Probing con rango de hora como llave y con valor de tipo objeto (total infr., porcentaje sin acc., porcentaje con acc., valor total a pagar). La hashtable se usa ya que permite un rápido acceso e inserción a datos organizados de la forma llave, valor. Se usan tablas de hash con linear probbing debido a su menor consumo de memoria en comparación con una implementación que haga uso de separate chaining.
        - Información por código: Hashtable con Linear Probing con rango de hora como llave y con valor de tipo Hashtable con ViolationCode como llave y objeto de tipo (total infr., VIOLATIONDESC)
    - Input: Hora inicial, hora final para definir
    - Output: objeto de tipo (total infr., porcentaje sin acc., porcentaje con acc., valor total a pagar, hashtable<vocode, (total infr., VIOLATIONDESC)>)
    - Complejidad: O(N)

- **Requerimiento 3C**
    - Estructura a utilizar: priority queue implementada mediante binary heap. Se utiliza esta estructura ya que permite la inserción ordenada de datos y el uso mínimo de memoria guardándose solo los M primeros datos si se desea.
    - Input:
    - Output:
    - Complejidad: O(M) ya creada la tabla de hash del requerimiento 2A.
- **Requerimiento 4C**
    - Estructura a utilizar: árbol balanceado negro-rojo de tipo #infracciones:ArregloDinamico<VOCode> o cola de prioridad de objetos (VOCode, #infracciones)
    - Input:
    - Output:
    - Complejidad: O(N)

Nota: *Obsérvese que para los requerimientos C, se menciona y justifica la estructura utilizada. Dicha estructura se elige con base al análisis teórico pero está sujeta a cambios en la implementación futura del proyecto.*