Proyecto 2 de Estructuras de Datos

Requerimientos Funcionales

R1) Lectura de archivos:

Descripción: El código debe ser capaz de leer todos los archivos JSON de tránsito de un semestre seleccionado por el usuario de 2018 en Washington D.C y almacenarlos en estructuras de datos cola y fila.

Entradas: Selección del semestre escogido por el usuario.

Salidas: Estructuras de datos con infracciones de los archivos almacenados y número de infracciones de cada mes cargado.

Orden Estimado: N (lectura de archivo JSON por semestre y conteo por fecha)

R2) Zona geográfica mínima máxima:

Descripción: El código debe ser encontrar la zona de coordenadas mínimas capaz de encerrar todas las infracciones almacenadas en el semestre seleccionado.

Entradas: -

Salidas: Par de coordenadas (Xmin, Ymin) y (Xmax,Ymax).

Orden Estimado: N (un recorrido por elemento estudiando máximo y mínimo actual)

R3) Ranking de N franjas horarias:

Descripción: El código debe ser capaz de retornar las N franjas horarias con el mayor número de infracciones imprimiendo porcentaje con y sin accidentes, el valor total a pagar y el número de infracciones.

Entradas: N franjas (N<24).

Salidas: Estadísticas de N franjas horarias con mayor número de infracciones.

Orden Estimado: N (retorna las N-ésimas estadísticas de una Cola de prioridad ordenada por número de infracciones)

R4) Ordenar las infracciones por localización geográfica:

Descripción: Ordena las infracciones por el criterio de las coordenadas. En primera medida se evalúa coordX y en caso de que sean iguales, se mira el atributo coordY. Todas las infracciones que tengan la misma llave (coordX, coordY) se guardan en un mismo objeto.

Entradas: -

Salidas: -

Orden Estimado: N(tabla de hash con número óptimo de llaves y distribución uniforme)

R5) Buscar infracciones por localización geográfica:

Descripción: Busca las infracciones por localización geográfica. Se retorna el total de infracciones, el porcentaje de infracciones sin accidente, el porcentaje de infracciones con accidente, el valor total a pagar por las infracciones, el location, el AddressID y el StreetSegId.

Entradas: CoordX, CoordY. Cordenadas de localización geográfica a buscar.

Salidas: Retorna un objeto con los siguientes parámetros: total de infracciones, el porcentaje de infracciones sin accidente, el porcentaje de infracciones con accidente, el valor total a pagar por las infracciones, el location, el AddressID y el StreetSegId.

Orden Estimado: N (tabla de hash con número óptimo de llaves y distribución uniforme)

R6) Buscar infracciones por rango de fecha:

Descripción: Se representan las infracciones por fecha en un árbol balanceado por el TicketIssueDate. Para cada fecha que se encuentre en el rango y tenga al menos una infracción, Se debe informar la fecha, el total de infracciones, su porcentaje de infracciones sin accidente, su porcentaje de infracciones con accidente y el valor total a pagar.

Entradas: FechaInicial, FechaFinal. Intervalo de fechas a buscar. Las fechas deben estar en formato (Año/Mes/Día).

Salidas: retorna un arreglo con los objetos con los siguientes parámetros: Fecha, el total de infracciones, su porcentaje de infracciones sin accidente, su porcentaje de infracciones con accidente y el valor total a pagar.

Orden Estimado: logN(árbol binario balanceado)

R7) Obtener ranking de las N tipos de infracciones con más infracciones

Descripción: El código debe ser capaz de retornar el código, total, porcentaje con y sin accidente y valor total a pagar de cada tipo de infracción en una cola de prioridad dependiente del número de infracciones por tipo de infracción.

Entradas: Rango N.

Salidas: Cola de prioridad ordenada por número de infracciones con código, total, porcentaje con y sin accidente y valor total a pagar de cada tipo de infracción.

Orden Estimado: N (comparación sucesiva de infracciones ya ordenadas por cantidad)

R8) Ordenar infracciones por localización geográfica:

Descripción: El código ordena las infracciones ascendentemente por coordenada Xcoord y Ycoord en caso de no decisión en un árbol balanceado.

Entradas:-

Salidas: árbol balanceado por localización geográfica.

Orden Estimado: NlogN (ubicación de infracción en árbol binario por comparaciones inmediatas)

R9) Franja fecha-hora por acumulado:

Descripción: se deben buscar las franjas fecha-hora que contengan un valor acumulado de infracciones para cada fecha-hora, mediante un árbol balanceado por valor acumulado de infracciones.

Entradas: Valor inicial, Valor final

Salidas: Fecha-Hora con valor acumulado entre valores específicos con valor acumulado, total de infracciones, porcentaje con y sin accidente.

Orden Estimado: NlogN (recorridos n-ésimos con revisión en un árbol balanceado)

R10) Información de localización:

Descripción: Para una localización informar el total de infracciones, porcentaje con y sin accidente, valor total a pagar y el Id de la calle impreso por pantalla.

Entradas: AdressID.

Salidas: total de infracciones, porcentaje con y sin accidente, valor total a pagar y Streetsegid.

Orden Estimado: N (recorrido único)

R11) Infracciones en rango de horas:

Descripción: Para un rango de horas recibido por parámetro se tiene que dar su información general y por código

Entradas: HH:MM:SS inicial, HH:MM:SS final

Salidas: Gráfica ASCII.