**Proyecto 2 – Manejador de Taxis**

|  |  |
| --- | --- |
| **Nombre** | **1A- – Obtener el taxi que más servicios ha realizado iniciando en una zona dada para una determinada compañía** |
| **Resumen** | El usuario ingresará el nombre de la compañía y un identificador de zona de inicio con los cuales se retorna el o los taxis que más servicios empezaron en esa zona y son de esa compañía |
| **Entradas** | |
| Nombre de la Compañía | |
| Número de la zona | |
| **Resultados** | |
| Se mostrará el taxi (o los taxis) que más servicios han realizado para dicha compañía empezando en el número de zona dado. | |
| **Estimación de complejidad** | |
| **Peor Caso:** O (*N 2*) | ~ (*N 2*)  **Caso Promedio:** O (log *N*) | ~ (log *N*)  Dado que es un árbol binario, el peor caso es que el árbol sea una lista (O(N)); el método get() de la tabla de Hash es de O(1) en promedio (peor caso O(N)) por lo cual no es relevante. | |

|  |  |
| --- | --- |
| **Nombre** | **2A – Agrupar los servicios por duración, mostrar los servicios que hacen parte de un rango.** |
| **Resumen** | Se agrupan los servicios en rangos de 60 segundos según su duración en segundos y dado una duración muestra todos los servicios dentro del rango de dicha duración. |
| **Entradas** | |
| Duración en segundos | |
| **Resultados** | |
| Retorna todos los servicios de un rango dado una duración perteneciente a ese rango. | |
| **Estimación de complejidad** | |
| **Peor Caso:** O (*N 3*) | ~ (*N 3*)  **Caso Promedio:** O (*N 2*) | ~ (*N 2*)  Dado que para agrupar los servicios hay que recorrerlos todos (O(N)), añadir a la hashTable en el peor caso es O(N) y el peor caso sería que todos los servicios estén en un rango y sea necesario imprimirlos devuelta O(N) | |

|  |  |
| --- | --- |
| **Nombre** | **1B – Obtener el conjunto de servicios cuya distancia recorrida está en un rango** |
| **Resumen** | Se pide al usuario ingresar una distancia mínima y máxima, la cual será el rango para buscar el conjunto de servicios que están dentro de dicho rango. |
| **Entradas** | |
| Distancia Mínima | |
| Distancia Máxima | |
| **Resultados** | |
| Se mostrara el identificador y la distancia recorrida de cada servicio del conjunto. | |
| **Estimación de complejidad** | |
| O (log *N*)  ~ (2 log *N*)  **Operación más recurrente:** Inserción de los elementos en el árbol binario rojo-negro | |

|  |  |
| --- | --- |
| **Nombre** | **2B – Consultar los servicios por fecha/hora y zona de recogida/terminación** |
| **Resumen** | Se pide al usuario una zona de recogida, terminación, fecha y hora para luego hacer la búsqueda de los servicios que cumplen estos rangos dados. |
| **Entradas** | |
| Fecha/Hora | |
| Zona de recogida/terminación | |
| **Resultados** | |
| Se muestra el identificador, zona de recogida, zona de terminación, fecha inicial, hora inicial, fecha final y hora final de cada servicio. | |
| **Estimación de complejidad** | |
| O (*N* log *N*)  ~ (2 *N* log *N*)  **Operación más recurrente:** Inserción en la tabla de hash y la inserción de los elementos en el árbol binario rojo-negro  Al insertar un nuevo elemento en la tabla de hash este recorre en el peor de los casos toda la tabla en busca de un espacio libre para insertar el árbol. Luego se busca con la llave el árbol a la cual debe agregarse el servicio y en el peor de los casos el árbol binario le toma 2log N en insertarlo. | |

|  |  |
| --- | --- |
| **Nombre** | **1C – Ordenar los taxis por puntos** |
| **Resumen** | Se ordenan los taxis los taxis por puntos utilizando Heapsort. |
| **Entradas** |  |
| **Resultados** | |
| Se ordenaron los taxis por puntos en un Heap. | |
| **Estimación de complejidad** | |
| Peor caso: O(N log(N)))  ~ N log(N)  **Operación más recurrente:** Comparación para el ordenamiento. | |

|  |  |
| --- | --- |
| **Nombre** | **2C – Dar servicios de un taxi dentro de rango de distancia (latitud, longitud).** |
| **Resumen** | Retorna los servicios que estén dentro de un rango de distancia a partir de un punto dado. |
| **Entradas** |  |
| Latitud | |
| Longitud | |
| Límite de distancia | |
| Id del Taxi | |
| **Resultados** | |
| Se retornaron todos los servicios que estén por debajo del límite máximo de distancia desde un punto (longitud y latitud) dado. | |
| **Estimación de complejidad** | |
| **Peor Caso:** O (*N 3*) | ~ (*N 3*)  Dado que hay que buscar entre todos los taxis, el peor caso es que sea el último O(N), dado que hay que buscar entre todos sus servicios O(N) y el peor caso es que todos estén dentro del rango: O(N). | |

|  |  |
| --- | --- |
| **Nombre** | **3C – Retornar los servicios en el rango más cercano** |
| **Resumen** | Se pide al usuario que ingrese una fecha y hora para hacer la búsqueda de los servicios que están en el rango de 15 minutos más cercano. Además estos deben haber comenzado en una zona y terminado en otra. |
| **Entradas** |  |
| Fecha/Hora | |
| **Resultados** | |
| Se muestra el identificador, zona de recogida, zona de terminación, Fecha inicial y Hora inicial de cada servicio que se encuentra en el rango de 15 minutos más cercano a la fecha y hora dadas por el usuario. | |
| **Estimación de complejidad** | |
| O (log *N*)  ~ (4 log *N*)  **Operación más recurrente:** Inserción de los servicios en el árbol binario rojo-binario  Al insertar un elemento en el árbol binario a este le toma en el peor de los casos 2log *N* y luego cuando se quiere hacer la búsqueda del conjunto que está en el rango a este le toma 2 log *N* en el peor de los casos. | |