Requerimientos C

|  |  |
| --- | --- |
| Nombre | ViajesConHoraYZonaOrigenDada |
| Descripción | Retornar los tiempo promedio de los viajes que salen de una zona dada a una hora dada |
| Datos de entrada | Int Zona de inicio, int hora |
| Datos de salida | Zona de origen, zona de destino, hora y tiempo promedio de cada viaje |
| Complejidad | On |

|  |  |
| --- | --- |
| Nombre | VijeasEnRangoDeHorasYZonaDestinoDada |
| Descripción | Retornar los tiempos promedio de los viajes en un rango de horas y que tengan una zona de destino dada |
| Datos de entrada | Int zona de llegada, int hora inferior, int hora superior |
| Datos de salida | Zona de origen, zona de destino, hora y tiempo promedio de cada viaje. |
| Complejidad | On |

|  |  |
| --- | --- |
| Nombre | ObtenerNZonasPriorizadasPorMayoríaDeNodos |
| Descripción | Obtiene las N zonas con mayor cantidad de nodos que definen su frontera |
| Datos de entrada | int N |
| Datos de salida | Por cada zona mostrar el nombre de la zona y la cantidad de nodos en su frontera |
| Complejidad | On |

|  |  |
| --- | --- |
| Nombre | ASCIIDePorcentajeFaltante |
| Descripción | Crear una gráfica ASCII que muestre por cada zona de origen el porcentaje de datos faltantes |
| Datos de entrada |  |
| Datos de salida | Gráfico ASCII por zonas y con \* que representan 2% de datos faltantes |
| Complejidad | On |

1C

HashLP

Se usará para usar los módulos de los id de las zonas para acotar la búsqueda de los viajes.

2C

Árbol Rojo Negro.

Permite hacer el mejor uso del rango de horas, dado que un árbol rojo negro no solo puede funcionar con rangos, sino que siempre se sabe cuáles son menores y mayores a otros, por lo que se puede delimitar con mayor facilidad desde donde se toma el rango y hasta adonde.

3C

MaxHeapCP

Este permite encontrar con mayor facilidad los elementos con mayor valor. Entonces, al buscar las N zonas más priorizadas, el heap es la opción que debería encontrarlos con mayor facilidad en la mayoría de los casos, si organizamos de tal manera que la zona más priorizada por nodos que la definen como frontera es la máxima.

4C

HashLP

Para este proceso, es necesario revisar todas y cada una de las zonas de origen y revisar si tienen los datos que se esperan que tengan. No importa si una de estas zonas es mayor que otra, solo se busca tener el proceso más rápido para buscar la zona que se quiere revisar y saber sus viajes a distintas zonas, para lo que el hash parece ser el más conveniente. También la existencia del método keys() es conveniente para saber que viajes a que zonas tiene que tener cada zona.