**OBSERVACIONES DEL RETO 4**

María Catalina Ibáñez Piñeres Cod 201922462 Mail m.ibanez@uniandes.edu.co

María Alejandra Pérez Petro Cod 201923972 Mail ma.perezp@uniandes.edu.co

Análisis de complejidad de cada uno de los requerimientos en Notación O

*Carga de datos*

La complejidad de la carga de datos es O(Nlog(N)) donde N es el número de lineas (viajes) en el csv. Esta complejidad se debe a que primero se recorre cada linea del csv para filtrar los viajes y crear tres tablas de hash (“stations\_table”, “trip\_table”, “bike\_id”) y un arbol RBT ('dateTrips') en el catalogo. Luego, se recorre las llaves de “trip\_table”, para construir el grafo “connections” cuyos vertices son las estaciones de origen y de destino. Por lo que, la complejidad de la carga de datos es O(Nlog(N)), pues la operación que más aporta a la complejidad es la construcción del arbol RBT.

*Requerimiento 1*

La complejidad del primer requerimiento es O(N\*log(N)) donde N es el número de estaciones (vertices) del grafo “connections”. Esta complejidad se debe a que se recorren los vertices del grafo para calcular el outdegre de cada uno de ellos. Luego los vertices se organizan según el número de “Out trips” por medio de Merge sort y, finalmente, se seleccionan los primeros 5 vertices. El Merge Sort es la operación que más aporta a la complejidad, entonces la complejidad del requerimiento 1 es O(N\*log(N)).

*Requerimiento 2*

La complejidad del segundo requerimiento es O(E\*log(V)) donde V es el número de estaciones (vertices) y E es el número de arcos (de viajes sin repeticiones) del grafo “connections”. Esta complejidad se debe a que se realiza el algoritmo de Dijkstra sobre el grafo “connections” y luego se halla el PathTo para cada vertice del grafo, asi como, la suma de los pesos de los arcos que componen el PathTo. El algoritmo de Dijkstra es la operación que más aporta a la complejidad, entonces la complejidad del requerimiento 2 es O(E\*log(V)).

*Requerimiento 3*

La complejidad del tercer requerimiento es O(NM\*log(NM)) donde N es el número de índices (salarios) dentro del intervalo de salarios de interés y M es el número de jugadores que tienen un salario N. Esta complejidad se debe a que para poder conocer los jugadores que están dentro del intervalo del salario de interés, toca recorrer primero cada índice (cada salario dentro del intervalo) y luego cada jugador dentro de ese índice. Después de hacer esos recorridos y haber agregado cada uno de los jugadores a un array\_list que queda de tamaño NM, esta se organiza por salario del jugador por medio de Merge sort. El Merge Sort es la operación que más aporta a la complejidad, por lo que la complejidad del requerimiento 3 es O(NM\*log(NM)).

*Requerimiento 4*

La complejidad del cuarto requerimiento es O(E\*log(V)) donde V es el número de estaciones (vertices) y E es el número de arcos (de viajes sin repeticiones) del grafo “connections”. Esta complejidad se debe a que se realiza el algoritmo de Dijkstra sobre el grafo “connections” y luego se halla el PathTo hacia la estación de destino. Finalmente, se recorren los arcos que componen dicho PathTo y se suman sus pesos. El algoritmo de Dijkstra es la operación que más aporta a la complejidad, entonces la complejidad del requerimiento 4 es O(E\*log(V)).

*Requerimiento 5*

La complejidad del quinto requerimiento es O(NM) donde N es el número de intervalos y M es el número atributos en la propiedad dada por el usuario (es decir, si el usuario escoge wage\_eur, entonces M es el número de salarios existentes). Esta complejidad se debe a que se recorre el número de grupos/intervalos N que se van a armar y, luego, los M atributos de la propiedad que hay en cada intervalo. Por lo que la complejidad del requerimiento 5 es O(NM).

*Requerimiento 6*

La complejidad del sexto requerimiento es O(N) donde N es el número de jugadores en la posición de interés. Esta complejidad se debe a que primero se obtienen los jugadores que juegan en la posición solicitada con el RBT player\_positions. Luego, los jugadores que juegan en una posición de interés se recorren para calcular el valor representativo de cada jugador (vr) y posteriormente calcular la diferencia en valor adsoluto entre el vr del jugador y el vr del jugador a sustituir. El recorrido es la operación que más aporta a la conplejidad por lo que, la complejidad del sexto requerimiento es O(N).