**ANÁLISIS DEL RETO**

(Estudiante 1)= Daniel Pérez, Código:202224568 de.perez@uniandes.edu.co

(Estudiante 2 y 3)=Nicolas Barreto B., código:202320496, n.barretob@uniandes.edu.co

# **Requerimiento <<5>>**

## **Descripción**

Este requerimiento se encarga de retornar:

. El total de ofertas publicadas en la ciudad en el periodo de consulta.

• El total de empresas que publicaron por lo menos una oferta en la ciudad de consulta.

• Empresa con mayor número de ofertas y su conteo

• Empresa con menor número de ofertas (al menos una) y su conteo

• El listado de ofertas publicadas ordenadas cronológicamente por fecha y nombre de la empresa (v.gr.Para dos ofertas con la misma fecha, el orden lo decide la empresa de forma alfabética). Cada uno

de los elementos debe presentar la siguiente información:

o Fecha de publicación de la oferta

o Título de la oferta

o Nombre de la empresa de la oferta

o Tipo de lugar de trabajo de la oferta

o Tamaño de la empresa de la oferta

o Tipo de lugar de trabajo de la oferta

dado el Nombre de la ciudad, La fecha inicial del periodo a consultar (con formato "%Y-%m-%d"), La fecha final del periodo a consultar (con formato "%Y-%m-%d"), el tipo de lista en que es guardada la informacion, y la informacion en si.

**Pasos implementacion:**

**1.** # primero organiza la lista bajo el criterio necesitado

**2.** # luego la filtra y le saca el tamaño para obtener el total de ofertas publicadas en la ciudad en el periodo de consulta, un dato que se retorna al final de la función

**3.** después crea una lista fantasma en la que se guardan todas las empresas que publicaron, lo que luego es retornado retorna

**4,** # más tarde se hace una función iterativa para comparar el menor y el mayor de toda la lista, y así guardarlos para que luego sean retornados

**5.** # Y por último se retorna la lista ordenada inicial para sacar todos sus valores filtrados en el view

|  |  |
| --- | --- |
| **Entrada** | data\_structs, city, fecha\_inicial, fecha\_final, tipo |
| **Salidas** | total\_ofertas, dato\_empresas, conteo\_empresa, max\_empresa, conteo\_min, min\_empresa, lista\_ordenada |
| **Implementado (Sí/No)** | Sí, lo implemento Nicolas Barreto. |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Paso 1 (mergesort) | O(nlog(n)) |
| Paso 2,3 (Partes 1 y 2 del requerimiento) | O(n) |
| Paso 4,5,6 (partes 3,4, y 5 del requerimiento) | O(n) |
| ***TOTAL*** | ***O(nlog(n))*** |

Memoria utilizada extra = O(n)

## **Pruebas Realizadas**

Las pruebas realizadas fueron realizadas en una maquina con las siguientes especificaciones. Los datos de entrada fueron el data\_structs(carga datos), city (por usuario), fecha\_inicial(por usuario), fecha\_final(por usuario), tipo (por usuario).

### **Tablas de datos**

|  |  |
| --- | --- |
| Procesadores | Chip M2 Apple |
| Memoria RAM | 8 GB |
| Sistema Operativo | MacOS |

|  |  |
| --- | --- |
| **Entrada** | **Tiempo (ms)** |
| Data\_structs(10% del archivo con una muestra de 20,000 elementos), city(Warszawa),  fecha inicial(2022-04-14T14:57:00.000Z), fecha final(2023-08-30T09:00:00.000Z), tipo(0 = Array\_list) | 175.24 ms |
| Data\_structs(20% del archivo con una muestra de 40,000 elementos), city(Warszawa),  fecha inicial(2022-04-14T14:57:00.000Z), fecha final(2023-08-30T09:00:00.000Z), tipo(0 = Array\_list) | 348.29 ms |
| Data\_structs(20% del archivo con una muestra de 60,000 elementos), city(Warszawa),  fecha inicial(2022-04-14T14:57:00.000Z), fecha final(2023-08-30T09:00:00.000Z), tipo(0 = Array\_list) | 527.32 ms |

|  |  |
| --- | --- |
| Data\_structs(30% del archivo con una muestra de 80,000 elementos), city(Warszawa),  fecha inicial(2022-04-14T14:57:00.000Z), fecha final(2023-08-30T09:00:00.000Z), tipo(0 = Array\_list) | **690.69 ms** |
| Data\_structs(40% del archivo con una muestra de 120,000 elementos), city(Warszawa),  fecha inicial(2022-04-14T14:57:00.000Z), fecha final(2023-08-30T09:00:00.000Z), tipo(0 = Array\_list) | **1042.12 ms** |
| Data\_structs(50% del archivo con una muestra de 140,000 elementos), city(Warszawa),  fecha inicial(2022-04-14T14:57:00.000Z), fecha final(2023-08-30T09:00:00.000Z), tipo(0 = Array\_list) | **1216.62 ms** |
| Data\_structs(60% del archivo con una muestra de 160,000 elementos), city(Warszawa),  fecha inicial(2022-04-14T14:57:00.000Z), fecha final(2023-08-30T09:00:00.000Z), tipo(0 = Array\_list) | **1413.12 ms** |
| Data\_structs(70% del archivo con una muestra de 180,000 elementos), city(Warszawa),  fecha inicial(2022-04-14T14:57:00.000Z), fecha final(2023-08-30T09:00:00.000Z), tipo(0 = Array\_list) | **1557.15 ms** |
| Data\_structs(100% del archivo con una muestra de 200,000 elementos), city(Warszawa),  fecha inicial(2022-04-14T14:57:00.000Z), fecha final(2023-08-30T09:00:00.000Z), tipo(0 = Array\_list) | **1691.31 ms** |

### **Graficas**

## **Análisis**

Como nos damos cuenta en la grafica, los datos recopilados tienen una implementacion casi perfecta de O(nlogn) de manera que tienen la forma esperada y crecen de forma casi lineal, ya que es una linea multiplicada por un logaritmo. Dado que excel no tiene la funcionalidad de hacer una linea de regresion linearitmica, use una regresionl potencial al poder de 1,0068 que modela mejor esta caracteristica. En caunto a los valores de la tabla, se ve que crecen casi constantemente, ya que se le añade un poquito mas de cambio cada vez, lo que demuestra nuestro pensamiento teorico de que la funcion tiene una comlejidad O(nlogn).