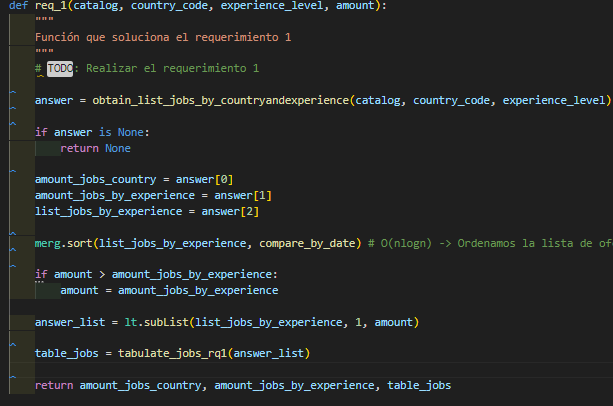
**ANÁLISIS DEL RETO**

Sofia Monzón Vergara-202317399-s.monzon@uniandes.edu.co

# **Requerimiento 1**

## **Descripción**

Este código aborda el requerimiento 1 que se trata de obtener una lista de trabajos por país y nivel de experiencia, limitada por una cantidad especifica. Aquí una breve descripción de cómo se implementó.



|  |  |
| --- | --- |
| **Entrada** | Numero (N) de ofertas a listar, código del país, nivel experticia |
| **Salidas** | Total de ofertas de trabajo ofrecidas según el país  Total de ofertas de trabajo ofrecidas según la condición (j.m.s)  Ofertas con información: Fecha de publicación de la oferta, título de la oferta, Nombre de la empresa de la oferta, nivel de experticia, país de la empresa, ciudad de la empresa, tamaño de la empresa, ubicación trabajo, disponible a contratar ucranianos. |
| **Implementado (Sí/No)** | Si se implementó, Sofia Monzón Vergara |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| PASO1    obtención de la lista de trabajos por país y experiencia:   * Acceder al catálogo y obtener información del país y nivel de experiencia: O(1) * Verificar si el país y nivel de experiencia existen en el catálogo: O(1) * Obtener la cantidad total de trabajos en el país: O(1) * Obtener la lista de trabajos por nivel de experiencia: O(1) | O(1) |
| PASO2    Ordenamiento de la lista de trabajos por fecha de publicación:   * Ordenar la lista de trabajos por fecha de publicación: O(n log n) donde n es el tamaño de la lista de trabajos por experiencia. Esto se debe al uso de “merg.sort”, que generalmente tiene complejidad O(n log n) | O(n log n) |
| PASO3    Selección de trabajos según la cantidad solicitada:   * Verificar si la cantidad solicitada es mayor que la cantidad de trabajos disponibles: O(1) debido a una comparación simple. * Crear una sublista de trabajos según la cantidad solicitada: O(1) | O(1) |
| ***TOTAL: En resumen, la complejidad total del algoritmo estaría dominada por el paso 2, paso de ordenamiento de la lista de trabajos por fecha de publicación, lo que resultaría en una complejidad de O(n log n) donde n es el tamaño de la lista de trabajos por experiencia.*** | ***O(n log n)*** |

## **Pruebas Realizadas**

Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

|  |  |
| --- | --- |
| Procesadores | AMD Ryzen 5 5500U with Radeon Graphics, 2100 Mhz, 6 procesadores principales, 12 procesadores lógicos |
| Memoria RAM | 8 GB |
| Sistema Operativo | Windows 11 Home Single Language |

SEPARATE CHAINING:

f

|  |  |
| --- | --- |
| **Entrada (factor de carga=2)** | **Tiempo (s)** |
| small | 1.4678821563720703 |
| medium | 2.4269039630889893 |
| Large | 2.6061503887176514 |
| 10 porciento | 0.3611266613006592 |
| 20 porciento | 0.7502973079681396 |
| 30 porciento | 1.1947011947631836 |
| 40 porciento | 1.750000238418579 |
| 50 porciento | 2.105433464050293 |
| 60 porciento | 2.425952434539795 |
| 70 porciento | 2.489713668823242 |
| 80 porciento | 2.4715754985809326 |
| 90 porciento | 2.479712724685669 |
| **Entrada (factor de carga=4)** | **Tiempo (s)** |
| small | 1.3018949031829834 |
| medium | 2.4414665699005127 |
| Large | 2.542673349380493 |
| 10 porciento | 0.3407588005065918 |
| 20 porciento |  |
| 30 porciento |  |
| 40 porciento |  |
| 50 porciento |  |
| 60 porciento |  |
| 70 porciento |  |
| 80 porciento |  |
| 90 porciento |  |
| **Entrada** | **Tiempo (s)** |
| small | 1.4678821563720703 |
| medium |  |
| Large |  |
| 10 porciento |  |
| 20 porciento |  |
| 30 porciento |  |
| 40 porciento |  |
| 50 porciento |  |
| 60 porciento |  |
| 70 porciento |  |
| 80 porciento |  |
| 90 porciento |  |
| **Entrada** | **Tiempo (s)** |
| small | 1.4678821563720703 |
| medium |  |
| Large |  |
| 10 porciento |  |
| 20 porciento |  |
| 30 porciento |  |
| 40 porciento |  |
| 50 porciento |  |
| 60 porciento |  |
| 70 porciento |  |
| 80 porciento |  |
| 90 porciento |  |

### **Tablas de datos**

Las tablas con la recopilación de datos de las pruebas.

### **Graficas**

Las gráficas con la representación de las pruebas realizadas.

## **Análisis**

Análisis de resultados de la implementación, tener cuenta las pruebas realizadas y el analisis de complejidad.

# **Requerimiento Ejemplo**

## **Descripción**



Este requerimiento se encarga de retornar un dato de una lista dado su ID. Lo primero que hace es verificar si el elemento existe. Dado el caso que exista, retorna su posición, lo busca en la lista y lo retorna. De lo contrario, retorna None.

|  |  |
| --- | --- |
| **Entrada** | Estructuras de datos del modelo, ID. |
| **Salidas** | El elemento con el ID dado, si no existe se retorna None |
| **Implementado (Sí/No)** | Si. Implementado por Juan Andrés Ariza |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Buscar si el elemento existe (isPresent) | O(n) |
| Obtener el elemento (getElement) | O(1) |
| ***TOTAL*** | ***O(n)*** |

## **Pruebas Realizadas**

Las pruebas realizadas fueron realizadas en una maquina con las siguientes especificaciones. Los datos de entrada fueron el ID 1.

|  |  |
| --- | --- |
| Procesadores | AMD Ryzen 7 4800HS with Radeon Graphics |
| Memoria RAM | 8 GB |
| Sistema Operativo | Windows 10 |

|  |  |
| --- | --- |
| **Entrada** | **Tiempo (ms)** |
| small | 0.05 |
| 5 pct | 0.33 |
| 10 pct | 1.28 |
| 20 pct | 2.54 |
| 30 pct | 4.98 |
| 50 pct | 7.51 |
| 80 pct | 13.81 |
| large | 25.97 |

### **Tablas de datos**

Las tablas con la recopilación de datos de las pruebas.

|  |  |  |
| --- | --- | --- |
| **Muestra** | **Salida** | **Tiempo (s)** |
| small | Dato1 | 0.05 |
| 5 pct | Dato2 | 0.33 |
| 10 pct | 33000 | 0,22 |
| 20 pct | Dato4 | 2.54 |
| 30 pct | Dato5 | 4.98 |
| 50 pct | Dato6 | 7.51 |
| 80 pct | Dato7 | 13.81 |
| large | 198000 | 1,67 |

### **Graficas**

Las gráficas con la representación de las pruebas realizadas.

## **Análisis**

A pesar de que obtener un elemento en un *ArrayList,* dada su posición, tiene complejidad constante, la implementación de este requerimiento tiene un orden lineal O(n). Esto debido a que, lo primero que se hace es verificar si el elemento hace parte de la lista. Específicamente, a la hora de buscar un elemento en una lista, en el peor de los casos es necesario recorrer toda la lista, es decir, complejidad lineal.

Este comportamiento se puede evidenciar experimentalmente en la gráfica. Ya que, gracias a que los datos no se encuentran tan dispersos con respecto a la línea de tendencia, la curva coincide con el comportamiento lineal esperado.