ANÁLISIS DEL RETO

Samuel Ovalle Arenas

Estudiante 2, código 2, email 2

Estudiante 3, código 3, email 3

Requerimiento <<n>>

Plantilla para el documentar y analizar cada uno de los requerimientos.

Descripción

Breve descripción de como abordaron la implementación del requerimiento

Entrada	Parámetros necesarios para resolver el requerimiento.
Salidas	Respuesta esperada del algoritmo.
Implementado (Sí/No)	Si se implementó y quien lo hizo.

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Paso 1	O(N/2)
Paso 2	O(N/2)
Paso ….	O(N/2)
TOTAL	***O(***N/2***)***

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).

Entrada	**Tiempo (s)**

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

```
def get_data(data_structs, id):
    """
    Retorna un dato a partir de su ID
    """
    pos_data = lt.isPresent(data_structs["data"], id)
    if pos_data > 0:
        data = lt.getElement(data_structs["data"], pos_data)
        return data
    return None
```

Requerimiento 1

Entrada	Estructuras de datos del modelo, código país, experiencia, rango.
Salidas	Elementos que cumplen con el código del país y experiencia,
Implementado (Sí/No)	Si.

Análisis de complejidad
Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Extraer Elemento `me.getValue(paises)`	O(N/M)
Extraer elemento del mapa `mp.get(catalog['code'], codigo_final)`	O(N/M)
TOTAL	***O(N/M)***

Requerimiento 3

Entrada	`catalog, empresa, fechainicial, fechafinal` y estructura de datos

Salidas	Elementos que fueron publicados entre la fecha inicial y final, numero de ofertas, diccionario con informacion de las ofertas
Implementado (Sí/No)	Si.

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
`quk.sort(codigos, compareDate)`	O(n^2)

Extraer elemento del mapa
`mp.get(catalog['code'], codigo_final)`
`me.getValue(datos)["jobs"]`
	O(N/M)

TOTAL	O(n^2)

Requerimiento 6

Entrada	`catalog, numero, experiencia, ano` y estructura de datos
Salidas	`respuesta_lista, total_empresas["empresas"], total_ofertas_publicadas, numero_ofertas`
Implementado (Sí/No)	Si.

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
`for x in lt.iterator(nombre_ciudades):`	O(N)

Extraer elemento del mapa
`datos_ciudad = mp.get(ciudades, x)`
`if datos_ciudad:`
`ofertas_en_ciudad = me.getValue(datos_ciudad)["jobs"]`
	O(N/M)

TOTAL	O(N)
Entrada	`catalog, rank, ano, mes y estructura de datos`

Salidas	`numero_ofertas, ciudad_mayor, ciudad_mayor_conteo, paises, paises_mayor_conteo, junior, mid, senior`
Implementado (Sí/No)	Si.

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	Complejidad
`for x in lt.iterator(nombre_ciudades):`	O(N)

`paises = merg.sort(catalog["paises_req7"], compareOfertas2)`
`datos = mp.get(catalog['country'], i)`
`paises = me.getValue(datos)["jobs"]`
	log n * n * O(1)
	
	O(N/M)

TOTAL	O(N)

Pruebas Realizadas

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

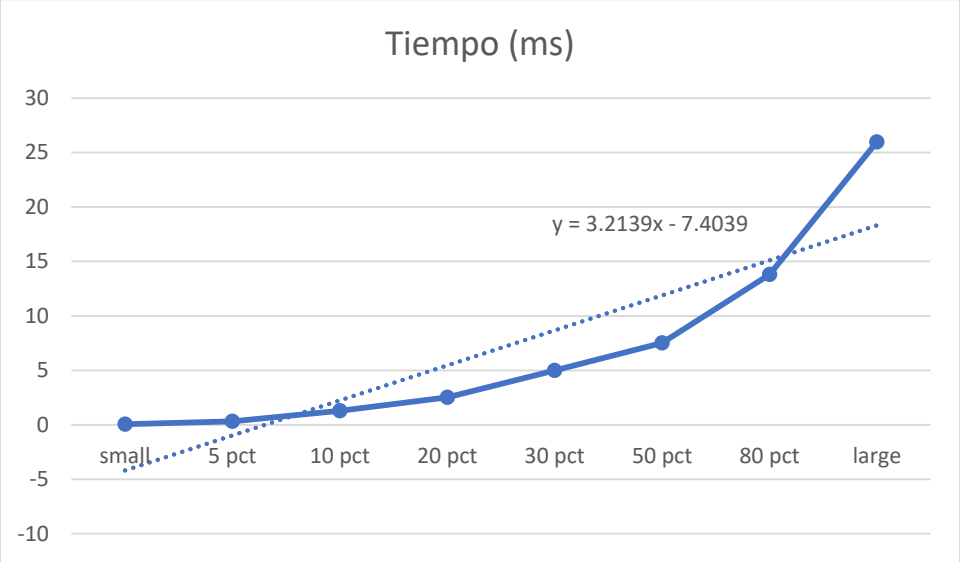
Las tablas con la recopilación de datos de las pruebas.

Muestra	Salida	Tiempo (ms)
small	Dato1	0.05
5 pct	Dato2	0.33

10 pct	Dato3	1.28
20 pct	Dato4	2.54
30 pct	Dato5	4.98
50 pct	Dato6	7.51
80 pct	Dato7	13.81
large	Dato8	25.97

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.