ANÁLISIS DEL RETO

Ana Fadul Ramos, 202320756, a.fadul

Manuela Galarza, 202320796, m.galarza

Juan Pablo Caicedo, código 3, email 3

Requerimiento 1

Descripción

Este requerimiento se encarga de listar las últimas N ofertas de trabajo ofrecidas en un país con un cierto nivel de experiencia (junior, mid o senior).

Entrada	Número de ofertas a listar, código del país, y nivel de experiencia de la oferta.
Salidas	Total de ofertas de trabajo según la condición, y por cada una de las ofertas de deben mostrar los siguientes criterios: fecha de publicación, titulo, nivel de experiencia, país, ciudad, tamaño de la empresa, tipo de ubicación y si está disponible a contratar ucranianos (True o False)
Implementado (Sí/No)	Sí. Implementado por todos.

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Crear la lista para los elementos que pasan los requerimientos y crear un contador. (newList() y variable)	O(1)
Recorrer todos los elementos de jobs. (for loop)	O(n)
Revisar si el experience level es igual (if statement) y agregar un valor al contador (variable)	O(1)
Revisar si el country code es igual (if statement)	O(1)
Agregar un valor en la última posición de la nueva lista.	O(1)
Organizar la lista por el orden de fecha de publicación (MergeSort)	O(n*log(n)) o O(k*log(k))
Retornar la lista y el contador	O(1)
TOTAL	***O(n*log(n))***

Pruebas Realizadas

Las pruebas realizadas se realizaron en una maquina con estas especificaciones. Los datos de entrada fueron: país SA, nivel de experiencia Senior, y número de trabajos 4.

Procesadores	**AMD Athlon Silver 3050U with Radeon Graphics 2.30 GHz**
Memoria RAM	8 GB
Sistema Operativo	Windows 11

Entrada	**Tiempo (s)**
small	0.057
10 pct	0.064

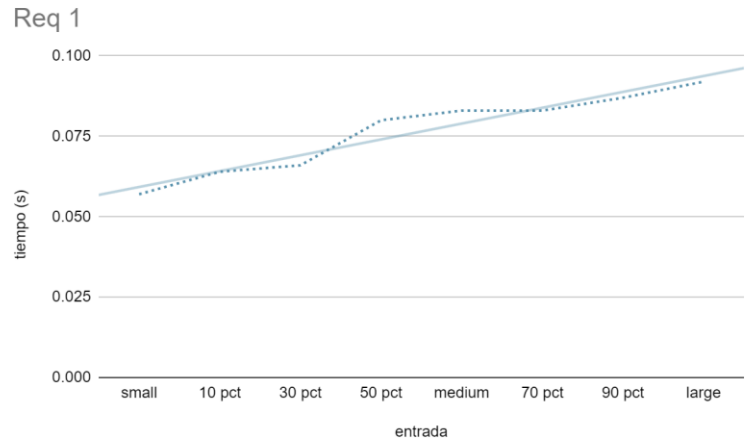
30 pct	0.066
50 pct	0.080
medium	0.083
70 pct	0.083
90 pct	0.087
large	0.092

Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

Muestra	Salida	Tiempo
small	Array_List y Variable	0.057
10 pct	Array_List y Variable	0.064
30 pct	Array_List y Variable	0.066
50 pct	Array_List y Variable	0.080
medium	Array_List y Variable	0.083
70 pct	Array_List y Variable	0.083
90 pct	Array_List y Variable	0.087
large	Array_List y Variable	0.092

Graficas



Análisis

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

A pesar de que obtener un elemento en un *ArrayList* y una variable, dada su posición, deberían dar una complejidad constante, la implementación de este requerimiento tiene un orden O(n*log(n)). Esto debido a que, aunque varias de las operaciones son de complejidad constante el ordenamiento de la lista a través de un MergeSort, y el for loop de todos los elementos aumentan la complejidad, está siendo determinada por el sort debido a su mayor complejidad.

Estos resultados se pueden presenciar en la gráfica, ya que, gracias a que los datos no se encuentran tan dispersos con la línea de tendencia podemos concluir que esta tiene la complejidad previamente mencionada. Es importante resaltar que cualquier divergencia en la gráfica también debe considerar que la complejidad

O(n*log(n)) es el peor caso en el cuál todos los elementos de la lista original aplican a la lista nueva, por esto deben existir divergencias pequeñas entre los dos gráficos, de esta manera mostrándose como O(k*log(k)). Es también relevante mencionar que sin importar las diferencias las pruebas de tiempo fueron efectivas, aumentando con el número de ofertas solicitadas. En el proceso, la implementación proporcionando resultados coherentes y eficientes para el requerimiento.

==========

Requerimiento 2

Descripción

Este requerimiento se encarga de listar las últimas N ofertas de trabajo por una empresa dado su nombre y ciudad.

Entrada	Número de ofertas a listar, nombre completo de la empresa, y ciudad de la oferta.
Salidas	Total, de ofertas por la empresa y la ciudad, y por cada oferta deben mostrar estos criterios: fecha de publicación, título, nivel de experiencia, país, ciudad, nombre de la empresa, formato de aplicación y tipo de trabajo (remoto o no).
Implementado (Sí/No)	Sí. Implementado por todos.

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Crear la lista para los elementos que pasan los requerimientos y crear un contador. (newList() y variable)	O(1)
Recorrer todos los elementos de jobs. (for loop)	O(n)
Revisar si el experience level y ciudad es igual (if statement)	O(1)
Agregar un valor en la última posición de la nueva lista.	O(1)
Organizar la lista por el orden de fecha de publicación (MergeSort)	O(n*log(n)) o O(k*log(k))
Retornar la lista y el tamaño (get_size())	O(1)
TOTAL	***O(n*log(n))***

Pruebas Realizadas

Las pruebas realizadas se realizaron en una maquina con estas especificaciones. Los datos de entrada fueron: nombre de empresa CloudTalk, ciudad Bratislava, y el número de trabajos 4.

Procesadores	**AMD Athlon Silver 3050U with Radeon Graphics 2.30 GHz**
Memoria RAM	8 GB
Sistema Operativo	Windows 11

Entrada	**Tiempo (s)**
small	0.036
10 pct	0.032
30 pct	0.041

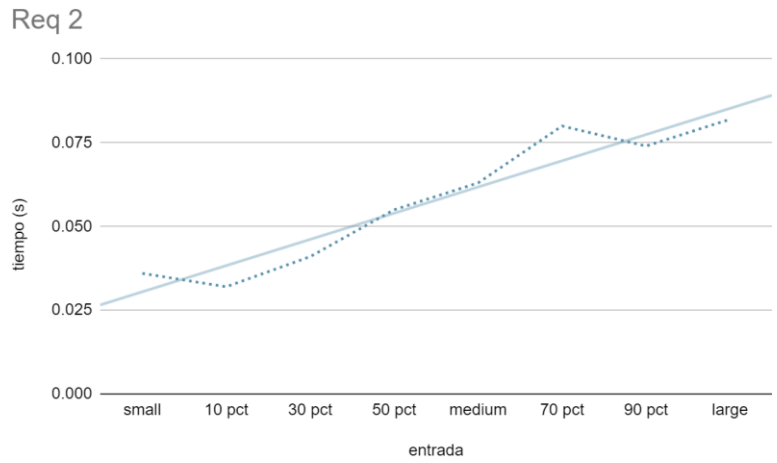
50 pct	0.055
medium	0.063
70 pct	0.080
90 pct	0.074
large	0.082

Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

Muestra	Salida	Tiempo
small	Array_List y Variable	0.036
10 pct	Array_List y Variable	0.032
30 pct	Array_List y Variable	0.041
50 pct	Array_List y Variable	0.055
medium	Array_List y Variable	0.063
70 pct	Array_List y Variable	0.080
90 pct	Array_List y Variable	0.074
large	Array_List y Variable	0.082

Graficas



Análisis

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

A pesar de que obtener un elemento en un *ArrayList* y una variable, dada su posición, deberían dar una complejidad constante, la implementación de este requerimiento tiene un orden O(n*log(n)). Esto debido a que, aunque varias de las operaciones son de complejidad constante el ordenamiento de la lista a través de un MergeSort, y el for loop de todos los elementos aumentan la complejidad, está siendo determinada por el sort debido a su mayor complejidad.

Estos resultados se pueden presenciar en la gráfica, ya que, gracias a que los datos no se encuentran tan dispersos con la línea de tendencia podemos concluir que esta tiene la complejidad previamente mencionada. Es importante resaltar que cualquier divergencia en la gráfica también debe considerar que la complejidad O(n*log(n)) es el peor caso en el cuál todos los elementos de la lista original aplican a la lista nueva, por esto

deben existir divergencias pequeñas entre los dos gráficos, aunque en este caso son casi imperceptibles, de esta manera mostrándose como O(k*log(k)). Sin importar las diferencias, las pruebas de tiempo fueron efectivas, siendo coherentes con el gráfico.

Requerimiento 3

Descripción

Este requerimiento se encarga de consultar las ofertas de trabajo publicadas por una empresa en un rango de fechas.

Entrada	Nombre de la empresa, fecha inicial del periodo a consultar y fecha final del periodo a consultar en el formato "%Y-%m-%d".
Salidas	Número total de ofertas, número de ofertas con experiencia junior, número de ofertas con experiencia mid, y número de ofertas con experiencia senior. Además de las ofertas ordenadas cronológicamente (y después alfabéticamente por país) que incluyen la información de fecha de publicación, titulo, nivel de experiticia, país, tamaño de la empresa, tipo de lugar de trabajo, y disponibilidad para contratar ucranianos.
Implementado (Sí/No)	Sí. Implementado por Ana Fadul.

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Creación de variables (num_ofertas_junior, num_ofertas_mid, num_ofertas_senior), creación de listas (newList()),y conversión de fechas (datetime.strptime)	O(1)
Recorrer todos los elementos de jobs	O(n)
Agregar un valor en la última posición de la nueva lista.	O(1)
Comparar el experience level con las 3 opciones (senior, mid, y junior) y sumar a una variable si alguna es True	O(1)
Retornar la lista y las 3 variables (num_ofertas_senior, num_ofertas_mid y num_ofertas_junior)	O(1)
Organizar la nueva lista por la fecha de publicación y después por el país alfabeticamente	O(n*log(n)) o O(k*log(n))
TOTAL	***O(n*log(n))***

Pruebas Realizadas

Las pruebas realizadas se realizaron en una maquina con estas especificaciones. Los datos de entrada fueron: nombre de empresa CloudTalk, con la fecha minima de 2022-04-01, y la fecha máxima de 2022-05-01.

Procesadores	**AMD Athlon Silver 3050U with Radeon Graphics 2.30 GHz**
Memoria RAM	8 GB
Sistema Operativo	Windows 11

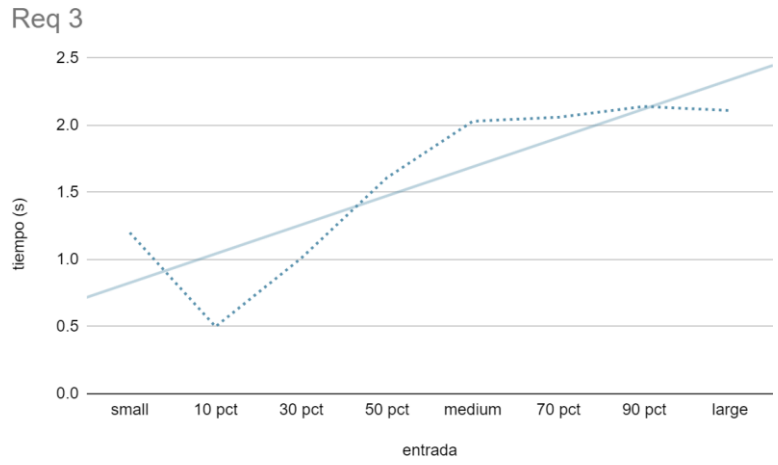
Entrada	Tiempo (ms)
small	1.20
10 pct	0.50
30 pct	1.01
50 pct	1.61
medium	2.03
70 pct	2.06
90 pct	2.14
large	2.11

Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

Muestra	Salida	Tiempo
small	Array_List y 3 Variables	1.20
10 pct	Array_List y 3 Variables	0.50
30 pct	Array_List y 3 Variables	1.01
50 pct	Array_List y 3 Variables	1.61
medium	Array_List y 3 Variables	2.03
70 pct	Array_List y 3 Variables	2.06
90 pct	Array_List y 3 Variables	2.14
large	Array_List y 3 Variables	2.11

Graficas



Análisis

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

A pesar de que obtener un elemento en un *ArrayList* y 3 variables, dadas su posición, deberían dar una complejidad constante, la implementación de este requerimiento tiene un orden O(n*log(n)). Esto debido a que, aunque varias de las operaciones son de complejidad constante el ordenamiento de la lista a través de un

MergeSort, y el for loop que recorre todos los elementos aumentan la complejidad, está siendo determinada por el sort debido a su mayor complejidad.

Estos resultados se pueden presenciar en la gráfica, ya que, gracias a la línea de tendencia podemos concluir que esta tiene la complejidad previamente mencionada, tomando en cuenta la similitud entre esta y la gráfica de n*log(n). Es importante resaltar que cualquier divergencia en la gráfica también debe considerar que la complejidad O(n*log(n)) es el peor caso en el cuál todos los elementos de la lista original aplican a la lista nueva, por esto deben existir divergencias pequeñas entre los dos gráficos, ya que la lista ordenada no es la lista original de elementos N, de esta manera mostrándose como O(k*log(k)). Es también importante resaltar que el incremento de tiempos es coherente con la complejidad delimitada.

Requerimiento 4

Descripción

Breve descripción de como abordaron la implementación del requerimiento

Entrada	control Dict: Diccionario con la información de las ofertas de trabajo. country_code str: Código del país. start_date str: Fecha de inicio del periodo. end_date str: Fecha de fin del periodo.
Salidas	jobs_found list: Lista de ofertas encontradas. qty_jobs_found int: Cantidad de ofertas encontradas. qty_companies int: Cantidad de empresas que publicaron ofertas. qty_cities int: Cantidad de ciudades con ofertas. city_most_offers str: Ciudad con más ofertas. city_less_offers str: Ciudad con menos ofertas. qty_city_most_offers int: Cantidad de ofertas en la ciudad con más ofertas. qty_city_less_offers int: Cantidad de ofertas en la ciudad con menos ofertas.
Implementado (Sí/No)	Si, Implementado por Juan Caicedo

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Paso 1(ordenar ofertas)	O(n*logn)
Paso 2 (convertir fechas)	O(1)
Paso 3 (inicializar con algunas variables)	O(1)
Paso 4 (Recorrer las ofertas para encontrar las que cumplen con los criterios de búsqueda de país y fecha)	O(n)
Paso 4(Encontrar las ciudades con más y menos ofertas. (Agregar las ofertas encontradas a una lista, las empresas a un conjunto, y las ciudades con ofertas al diccionario cities_with_jobs)	O(n)
Paso 6 Encontrar las ciudades con más y menos ofertas.	O(n)
Paso 7 (convertir a lista Python)	O(1)
Paso 8 (retornar resultados en tupla	O(1)
TOTAL	***O(nlogn)***

Con los valores país SA, y fechas entre 2022-04-01 y 2022-05-01

Procesadores	**AMD Athlon Silver 3050U with Radeon Graphics 2.30 GHz**
Memoria RAM	8 GB
Sistema Operativo	Windows 11

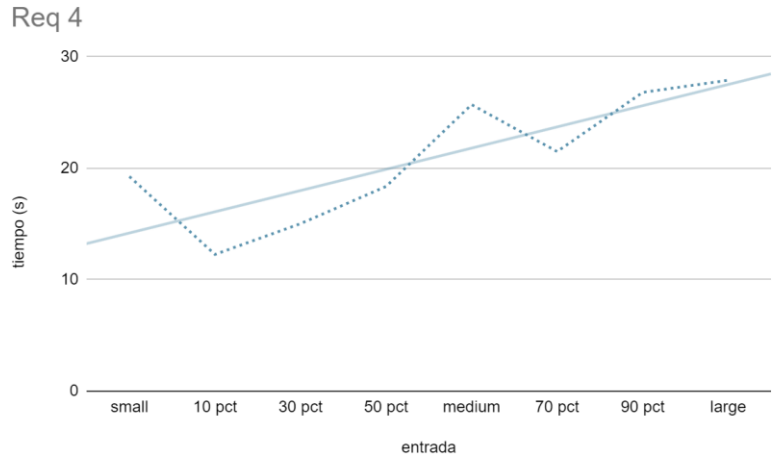
Entrada	**Tiempo (ms)**
small	19.26
10 pct	12.27
30 pct	15.03
50 pct	18.37
medium	25.71
70 pct	21.50
90 pct	26.81
large	27.90

Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

Muestra	**Salida**	**Tiempo**
small	Array_List	19.26
10 pct	Array_List	12.27
30 pct	Array_List	15.03
50 pct	Array_List	18.37
medium	Array_List	25.71
70 pct	Array_List	21.50
90 pct	Array_List	26.81
large	Array_List	27.90

Graficas



Análisis

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

A pesar de que obtener un elemento en un *ArrayList* dadas su posición, deberían dar una complejidad constante, la implementación de este requerimiento tiene un orden O(n*log(n)). Esto debido a que, aunque varias de las operaciones son de complejidad constante el ordenamiento de la lista a través de un MergeSort, y los for loop que recorre todos los elementos aumentan la complejidad, está siendo determinada por el sort debido a su mayor complejidad.

Estos resultados se pueden presenciar en la gráfica, ya que, gracias a la línea de tendencia podemos concluir que esta tiene la complejidad previamente mencionada, tomando en cuenta la similitud entre esta y la gráfica de n*log(n). Esto considerando que en el sort inicial se recorren todos los elementos de la lista.

Requerimiento 5

Descripción

Este requerimiento se encarga de consultar las ofertas de trabajo publicadas por una empresa en un rango de fechas.

Entrada	Nombre de la ciudad, fecha inicial del periodo a consultar y fecha final del periodo a consultar en el formato "%Y-%m-%d".
Salidas	Número total de ofertas, número de empresas que publicaron una oferta en la ciudad. La empresa con mayor número de ofertas y su conteo y la empresa con menor número de ofertas y su conteo. Ademas de imprimir la fecha de publicacion , el título, el nombre de la compañia, el lugar de trabajo, y el tamaño de la compañia, de las 3 primeras y 3 ultimas ofertas de la lista.
Implementado (Sí/No)	Sí. Implementado por Manuela Galarza.

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Convertir las fechas de entrada a objetos datetime.	O(1)
Iterar sobre todos los elementos de ‘jobs’ en ‘data_structs’ y verificar si está en el rango de fechas. Si está en el rango y en la ciudad se agrega a ‘en_ciudad’ y	O(n)

mantener un diccionario para contar las ocurrencias de cada empresa.	
Sobre el diccionario de empresa se busca la mayor y menor y su conteo.	O(k)
Ordenamiento de la lista en_ciudad usando el algoritmo TimSort.	O(n*log n) (complejidad promedio de TimSort) o O(k*logk)
Iterar sobre los elementos de p_u (3 primeros y 3 últimos de 'en_ciudad') y agregar cada elemento a una nueva fila, que se agrega a la lista que se va a tabular.	O(n) o O(k)
TOTAL	***O(n*log(n))***

Pruebas Realizadas

Las pruebas realizadas se realizaron en una maquina con estas especificaciones. Los datos de entrada fueron: la ciudad Warszawa, con la fecha minima de 2020-04-01, y la fecha máxima de 2022-05-01.

Procesador	**11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz 2.42 GHz**
Memoria RAM	16,0 GB (15,8 GB utilizable)
Sistema Operativo	Windows 11

Entrada	**Tiempo (ms)**
small	1483.6928999871016
10 pct	372.1645999997854
30 pct	1172.0998999923468
50 pct	1928.4745000004768
medium	2299.1335999965668
70 pct	2136.104999989271
90 pct	2559.236200004816
large	2882.954899996519

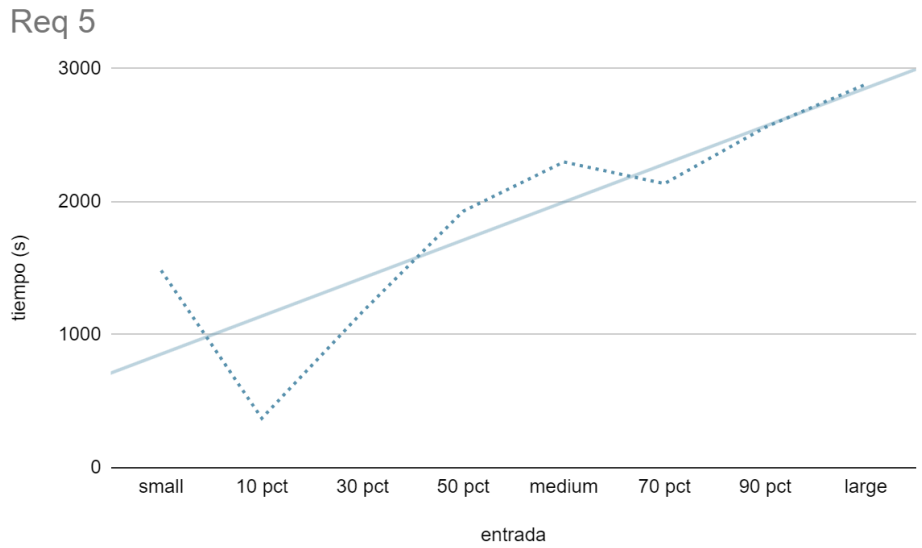
Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

Muestra	**Salida**	**Tiempo (ms)**
small	Array_List (para tabulate) y 3 Variables	1483.6928999871016
10 pct	Array_List (para tabulate) y 3 Variables	372.1645999997854
30 pct	Array_List (para tabulate) y 3 Variables	1172.0998999923468
50 pct	Array_List (para tabulate) y 3 Variables	1928.4745000004768

medium	Array_List (para tabulate) y 3 Variables	2299.1335999965668
70 pct	Array_List (para tabulate) y 3 Variables	2136.104999989271
90 pct	Array_List (para tabulate) y 3 Variables	2559.236200004816
large	Array_List (para tabulate) y 3 Variables	2882.954899996519

Graficas



Análisis

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

El análisis de resultados de la implementación del Requerimiento 5 refleja una eficiencia general en términos de tiempo de ejecución. La complejidad del algoritmo se evaluó detalladamente, mostrando que la mayoría de las operaciones tienen una complejidad constante O(1), mientras que la ordenación de la lista final presenta una complejidad O(n*log(n)) debido al algoritmo de ordenamiento utilizado. Las pruebas mostraron que el tiempo que tarda el programa en ejecutarse es razonable, incluso cuando se trabajó con más datos. Esto sugiere que el programa funciona bien para encontrar y mostrar las ofertas de trabajo como se esperaba.

Estos resultados se pueden presenciar en la gráfica, ya que, gracias a que los datos no se encuentran tan dispersos con la línea de tendencia podemos concluir que esta tiene la complejidad previamente mencionada de n*log(n). Es importante resaltar que cualquier divergencia en la gráfica también debe considerar que la complejidad O(n*log(n)) es el peor caso en el cuál todos los elementos de la lista original aplican a la lista nueva, por esto deben existir divergencias pequeñas entre los dos gráficos, aunque en este caso son casi imperceptibles, de esta manera mostrándose como O(k*log(k)). Sin importar las diferencias, las pruebas de tiempo fueron efectivas, siendo coherentes con el gráfico.

Requerimiento 6

Descripción

Este requerimiento se encarga de consultar las ofertas de trabajo publicadas por una empresa en un rango de fechas.

Entrada	Nombre de la empresa, fecha inicial del periodo a consultar y fecha final del periodo a consultar en el formato "%Y-%m-%d".
Salidas	Número total de ofertas, número de ofertas con experiencia junior, número de ofertas con experiencia mid, y número de ofertas con experiencia senior. Además de las ofertas ordenadas cronológicamente (y después alfabéticamente por país) que incluyen la información de fecha de publicación, titulo, nivel de experiticia, país, tamaño de la empresa, tipo de lugar de trabajo, y disponibilidad para contratar ucranianos.
Implementado (Sí/No)	Sí. Implementado por todos.

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Creación de variables(ofertas_publicadas, salario_promedio_total, y conteo) y listas (ciudades_nombres, ciudades_info, empresas, y salatios_total)	O(1)
Recorrer cada elemento del diccionario jobs	O(n)
Establecer las variables (primero, fecha) y ver si cumple las condiciones del nivel de experiencia, fechas, y país (if statement)	O(1)
Establecer la variable (Info), agregar información al último elemento de uno sus items que es un array list(addLast()) y ver si la ciudad existe dentro de una lista previamente creada (isPresent() y if statement()	O(1)
Dentro del if statement se agregan elementos al final de un array list (addLast), y cambiar variables en el diccionario previamente creado	O(1)
Ver si varios condiciones son verdaderas utilizando comandos de ADT listas (isPresent, getElement) y comparaciones de valores.	O(1)
Dentro de los if statements agregar elementos al final de listas (addLast), establecer variables, y cambiar valores en diccionarios	O(1)
Recorrer elementos agregados previamente (for loop) para agregarlo a una variable	O(n) o O(m)
Crear listas de typo array list (newList), ver si un condicional es verdad para cambiar una variabe, y cambiar otra variable	O(1)
Recorrer los elementos en la lista previamente creada de información de ciudades (for loop) dentro de este se establecen variables, cambia información en listas, y cambian variables basando en si condiciones son verdad (if statement). Después se recorren los elementos en una	O(n^2) o O(k^2)

lista ADT dentro del mismo diccionario para cambiar una información dentro de este.	
Ver si un condicional en caso de una lista vacia es verdad, y en este caso agregar al final de un arrayList información (addLast())	O(1)
Organizar la nueva lista por el número de ofertas publicadas, el salario promedio de las ofertas, y el nombre de la ciudad.	O(n*log(n)) o O(m*log(m))
Agregar el primer y último elemento de la lista ordenada al último elemento de una lista diversa si un condicional es verdadero con el elemento get_size()	O(1)
TOTAL	***O(k^2 + m*log(m) + n)***

Pruebas Realizadas

Las pruebas realizadas se realizaron en una maquina con estas especificaciones. Los datos de entrada fueron: 3 ciudades a visualizar, entre las fechas 2022-04-10 y 2022-04-20, con nivel de experiencia senior en Polonia (código de país PL).

Procesadores	**AMD Athlon Silver 3050U with Radeon Graphics 2.30 GHz**
Memoria RAM	8 GB
Sistema Operativo	Windows 11

Entrada	**Tiempo (ms)**
small	1.77
10 pct	0.42
30 pct	1.61
50 pct	2.52
medium	3.24
70 pct	3.08
90 pct	3.43
large	3.24

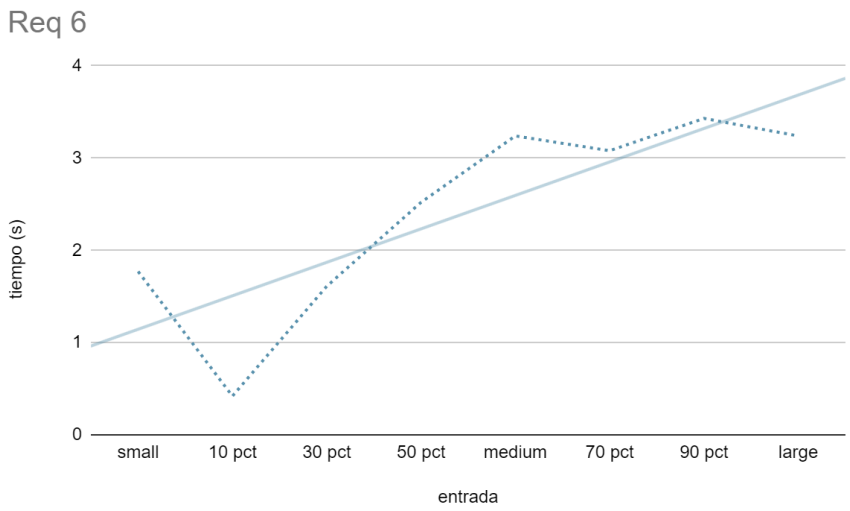
Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

Muestra	**Salida**	**Tiempo**
small	3 Array_List y 5 Variables	1.77
10 pct	3 Array_List y 5 Variables	0.42
30 pct	3 Array_List y 5 Variables	1.61
50 pct	3 Array_List y 5 Variables	2.52
medium	3 Array_List y 5 Variables	3.24
70 pct	3 Array_List y 5 Variables	3.08
90 pct	3 Array_List y 5 Variables	3.43

large	3 Array_List y 5 Variables	3.24

Graficas



Análisis

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

A pesar de que obtener un elemento en un *ArrayList* y 3 variables, dadas su posición, deberían dar una complejidad constante, la implementación de este requerimiento tiene un orden O(k^2 + m*log(m) +n). Esto debido a que, aunque varias de las operaciones son de complejidad constante, el ordenamiento de la lista a través de un MergeSort, y los varios for loops aumentan su complejidad, está mostrándose en la sumatoria de variables. K, m, y n representando listas diferentes, aunque todas derivadas de N, aunque solo la variable m pasando por el sort causando esto su complejidad de m*log(m).

Estos resultados se pueden presenciar en la gráfica, ya que, gracias a la línea de tendencia podemos concluir que esta tiene la complejidad similar a la previamente mencionada. Es importante resaltar que cualquier divergencia entre la gráfica nuestra y la de la complejidad, también debe considerar que la complejidad O(n^2) es el peor caso de todos, en el cual todos los elementos de la lista están ubicados en una ciudad diversa y una empresa diversa de esta manera teniendo esta complejidad. En la mayoría de los casos esta va a ser reducida, mostrada como O(k^2 +m*log(m) +n) causando una divergencia entre las gráficas, pero que igual va a mantener su estado exponencial.

Requerimiento 7

Descripción

Este requerimiento se encarga de consultar las ofertas de trabajo publicadas por una empresa en un rango de fechas.

Entrada	Nombre de la empresa, fecha inicial del periodo a consultar y fecha final del periodo a consultar en el formato "%Y-%m-%d".
Salidas	Número total de ofertas, número de ofertas con experiencia junior, número de ofertas con experiencia mid, y número de ofertas con experiencia senior. Además de las ofertas ordenadas cronológicamente (y después alfabéticamente por país) que incluyen la información de fecha de publicación, titulo, nivel de experiticia, país,

	tamaño de la empresa, tipo de lugar de trabajo, y disponibilidad para contratar ucranianos.
Implementado (Sí/No)	Sí.

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Convertir las fechas de entrada a objetos datetime.	O(1)
Iterar sobre los trabajos en data_structs['jobs'] para filtrar aquellos dentro del rango de fechas. Ademas, dentro del mismo ciclo, para cada trabajo, extrae información y actualiza los diccionarios: paises, ciudades, id_compania, habilidades, niveles, empresas_por_nivel y sedes_por_nivel.	O(n)
Crear una lista de tuplas tuplas_paises y ordenarla usando el algoritmo ShellSort	O(n) o O(n*log n) en el peor de los casos.
Iterar sobre las habilidades en data_structs['skills'] y actualizar el diccionario habilidades y la lista niveles.	O(m) (m es todas las habilidades)
Itera sobre las sedes en data_structs[‘multilocations’] y cuenta cuantas sedes hay por empresa	O(p) (p es todas las locaciones)
Calcular varias métricas, como total de ofertas, ciudades más grandes, etc.	O(k) Depende de la cantidad de datos recopilados anteriormente.
Ordenamiento de la lista en_ciudad usando el algoritmo TimSort.	O(n*log(n))
TOTAL	***O(n + m + p + n*log(n))***

Pruebas Realizadas

Las pruebas realizadas se realizaron en una maquina con estas especificaciones. Los datos de entrada fueron: numero de paises 2, con la fecha minima de 2020-04-01, y la fecha máxima de 2022-05-01.

Procesador	**11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz 2.42 GHz**
Memoria RAM	16,0 GB (15,8 GB utilizable)
Sistema Operativo	Windows 11

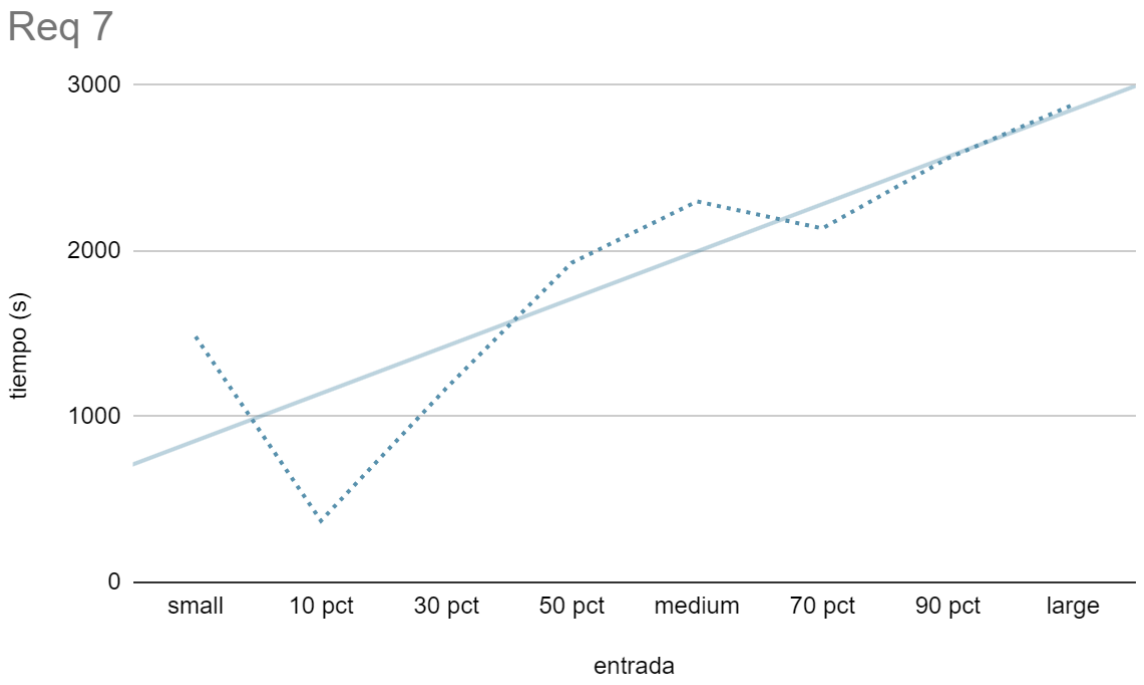
Entrada	**Tiempo (ms)**
small	1013.179800003767
10 pct	385.4611999988556
30 pct	925.2760999947786
50 pct	1925.4756000041962
medium	1851.0448999851942
70 pct	1891.8961000144482
90 pct	1836.5967000126839
large	1846.9098000079393

Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

Muestra	Salida	Tiempo (ms)
small	Array_List (para tabulate) y Variables	1013.179800003767
10 pct	Array_List (para tabulate) y Variables	385.4611999988556
30 pct	Array_List (para tabulate) y Variables	925.2760999947786
50 pct	Array_List (para tabulate) y Variables	1925.4756000041962
medium	Array_List (para tabulate) y Variables	1851.0448999851942
70 pct	Array_List (para tabulate) y Variables	1891.8961000144482
90 pct	Array_List (para tabulate) y Variables	1836.5967000126839
large	Array_List (para tabulate) y Variables	1846.9098000079393

Graficas



Análisis

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

==========

Análisis

A pesar de que obtener un elemento en un *ArrayList* y varias variables, dada su posición, tiene complejidad constante, la implementación de este requerimiento tiene un orden lineal O(n + m + p + n*log(n). Esto debido a

que a través de varios for loops de diferentes listas, representados por n, m, y p, además de un sort por Tim Sort representado por n*log(n), la complejidad incrementa.

Estos resultados se pueden presenciar en la gráfica, ya que, gracias a la línea de tendencia podemos concluir que esta tiene la complejidad similar a n*log(n), esta siendo la mayor de todas. Es importante resaltar que cualquier divergencia entre la gráfica nuestra y la de la complejidad, también debe considerar que la complejidad O(n*log(n) es el peor caso de todos, en el cual todos los elementos de la lista están ubicados en una ciudad diversa y una empresa diversa de esta manera teniendo esta complejidad. En la mayoría de los casos esta va a ser reducida, mostrada como O(k^2 +m*log(m) +n) causando una divergencia entre las gráficas, pero que igual va a mantener su estado n*log(n).