Mini-Proyecto

Análisis y Diseño de Algoritmos I

Jhoan Sebastian Fernandez(2222772)

Facultad de Ingeniería
Escuela de Ingeniería de sistemas y computación.
Universidad del valle, Cali, Colombia 21 de abril del 2025

1. INTRODUCCION.

Este informe presenta el desarrollo de un miniproyecto de la asignatura *Análisis y Diseño de Algoritmos I*, cuyo objetivo es diseñar, implementar y analizar dos estrategias distintas para procesar los resultados de una encuesta compleja. Este problema involucra la lectura, organización y presentación ordenada de datos provenientes de varios encuestados que responden preguntas agrupadas por temas.

El enfoque principal del proyecto es aplicar los conocimientos adquiridos sobre estructuras de datos y algoritmos, evitando el uso de funciones predeterminadas de ordenamiento del lenguaje de programación, y en su lugar, implementando manualmente algoritmos clásicos como *insertion sort*, *bubble sort* y *merge sort*. Además, se evalúa el rendimiento de ambas soluciones mediante análisis teóricos, para comparar su eficiencia en tiempo de ejecución y complejidad.

2. DESCRIPCIÓN DEL PROBLEMA.

Se dispone de un conjunto de datos estructurado en tres componentes principales:

Una lista de encuestados, cada uno con un identificador único, nombre, nivel de experticia y una opinión.
Un conjunto de preguntas, agrupadas por temas, en las que se especifica qué encuestados participaron en cada pregunta.
Una estructura jerárquica de temas, cada uno con varias preguntas, y cada pregunta

con una lista de encuestados asociados.

El objetivo de procesar esta información es para:

Calcular estadísticas para cada pregunta, incluyendo: promedio de opinión, promedio de experticia, mediana, moda, extremismo y consenso.
Ordenar las preguntas dentro de cada tema según criterios múltiples (promedio de opinión, experticia y cantidad de encuestados).
Ordenar los temas en función de los promedios agregados de sus preguntas.
Generar una salida ordenada y legible que presente los temas y preguntas en el orden correcto, incluyendo los ID de encuestados en cada pregunta.
Mostrar al final estadísticas agregadas como la pregunta con mayor y menor promedio, mediana, moda, extremismo y consenso.

Para resolver este problema, se plantearon dos estrategias completamente distintas, cada una basada en diferentes estructuras de datos: la primera con árboles binarios de búsqueda (ABB), y la segunda con estructuras lineales como listas enlazadas y listas doblemente enlazadas.

3.SOLUCIONES.

3.1 SOLUCION 1.

La primera solución propuesta al problema de procesamiento de encuestas se basa en el uso de árboles binarios de búsqueda (ABB) como estructura principal para organizar los datos. Esta elección se justifica por su eficiencia en operaciones de inserción, búsqueda y recorrido ordenado, que son esenciales para procesar preguntas, temas y encuestados según múltiples criterios de ordenamiento.

En esta estrategia, se crean dos árboles binarios: uno para las preguntas y otro para los temas. Cada uno tiene un criterio específico de comparación y orden, tal como lo exige el enunciado del problema.

3.1.1 Estructura de datos utilizada:

ABBPreguntas: un árbol binario de búsqueda para almacenar preguntas dentro de un tema, ordenadas según:

1. Promedio de opinión (descendente),
2. Promedio de experticia (en caso de empate),
3. Cantidad de encuestados (en caso de empate persistente).

ABBTemas: otro árbol binario de búsqueda que contiene los temas, ordenados por:

1. Promedio del promedio de opinión de sus preguntas (descendente),
2. Promedio del promedio de experticia,
3. Número total de encuestados.

Encuestado: clase que representa una persona encuestada con atributos como nombre, opinión y experticia.

3.1.2 Algoritmos implementados

Bubble Sort para encuestados por pregunta

Cada pregunta contiene una lista de encuestados, que se ordenan manualmente usando Bubble Sort, según: Opinión (descendente) - Experticia (en caso de empate)

Complejidad teórica:

Mejor caso: O(n) (si ya está ordenado)
Peor caso: O(n^2) (orden completamente inverso)

Recorridos inorden en los ABB

Para mostrar los temas y preguntas ordenadas, se hace un recorrido inorden sobre los árboles binarios.

Complejidad teórica:

Recorrido completo: O(n)
Inserción: O(logn) en promedio, O(n) en el peor caso (si el árbol está desbalanceado)

Cálculo de estadísticas

Se calculan las siguientes métricas para cada pregunta: Promedio, mediana, moda, extremismo y consenso.

Estas se implementan manualmente sin usar librerías externas como dice la rúbrica.

Merge Sort se utiliza para calcular la mediana.
Complejidad: O(nlogn)

Moda y **consenso**: se calculan con conteos directos.
Complejidad: O(n)

Flujo del procesamiento

1. Se cargan los encuestados y respuestas desde un archivo de entrada.
2. Se crean preguntas y se les asignan encuestados ordenados.
3. Las preguntas se insertan en el árbol ABB correspondiente al tema.
4. Se recalculan los promedios del tema y se insertan al ABB de temas.
5. Se recorren los árboles para producir la salida ordenada.
6. Se seleccionan las preguntas con métricas más representativas (mayor promedio, mediana, consenso, etc.).

Complejidad total estimada:

Ordenar encuestados por pregunta, Complejidad: O(k^2) (Bubble Sort)

Insertar preguntas en ABBPreguntas, Complejidad: O(nlogn)

Insertar temas en ABBTemas, Complejidad: O(mlogm)

Recorridos inOrden (Salida Ordenada), complejidad: O(n+m)

Cálculo de estadisticas por pregunta, complejidad: O(klogk) o sea la mediana.

Donde:

k => encuestados por pregunta.

n => número de preguntas.

m => número de temas.

Complejidad total:

O(nlogn + klogk + mgm)

3.2 SOLUCION 2.

La segunda estrategia propuesta para resolver el problema de análisis de encuestas se basa en el uso exclusivo de estructuras lineales: listas enlazadas simples y doblemente enlazadas. La idea es construir una estructura que permita almacenar los datos jerárquicos (temas → preguntas → encuestados) manteniendo un orden adecuado mediante técnicas manuales de inserción ordenada.

Cada tema se almacena en una lista enlazada, mientras que cada conjunto de preguntas de un tema se mantiene en una lista doblemente enlazada. Esto permite recorrer los datos en orden, insertar nuevos elementos conservando el orden y realizar análisis estadístico de forma eficiente en estructuras lineales.

3.2.1 Estructuras de datos utilizadas

NodoPregunta y ListaPreguntas: estructura de lista doblemente enlazada donde se insertan las preguntas ya ordenadas por criterios múltiples.

NodoTema y ListaTemas: lista enlazada simple donde se guardan los temas también en orden.
Tema: contiene una ListaPreguntas, estadísticas agregadas, y el total de encuestados del tema.

Estas estructuras permiten mantener el orden sin necesidad de algoritmos de ordenamiento externos, ya que la inserción se hace directamente en la posición correcta, comparando con los nodos existentes.

3.2.2 Algoritmos implementados

Insertion Sort para listas de encuestados

Este algoritmo ordena manualmente los encuestados por: Mayor opinión y en caso de empate, mayor experticia.

Se implementa sobre una lista de objetos Encuestado.

Complejidad teórica: *Peor caso:* O(n^2) - *Mejor caso:* O(n) si ya está ordenado.

Insertion Sort para impresión de encuestados (por experticia y ID)

Se usó un segundo insertion sort para ordenar la lista final de encuestados en la salida por: Mayor experticia y en caso de empate, mayor ID.

Complejidad teórica: igual al anterior.

Merge Sort para calcular la mediana

Para encontrar la mediana de las opiniones en cada pregunta, se usó un merge sort manual sobre la lista de opiniones.

Complejidad teórica: En todos los casos: O(nlogn)

Inserción ordenada en listas

La inserción de preguntas en ListaPreguntas y temas en ListaTemas se realiza comparando manualmente: Promedio de opinión (mayor es mejor), el promedio de experticia (si empatan en opinión) y la cantidad de encuestados (si también empatan en experticia).

Esto reemplaza la necesidad de ordenar posteriormente: la estructura queda siempre ordenada.

Complejidad teórica:

Por inserción: O(n)
Para insertar todas las preguntas o temas: O(n^2) en el peor caso

Flujo del procesamiento

El flujo general del programa sigue estos pasos:

1. Carga de datos desde archivo.
2. Creación de preguntas.
3. Cálculo de estadísticas por pregunta.
4. Inserción ordenada de preguntas en la lista doble del tema.
5. Recalculo de promedios del tema, con base en las preguntas asociadas.
6. Inserción ordenada del tema en la lista enlazada global:
7. Generación de salida ordenada:

Complejidad total estimada:

Ordenar encuestados por pregunta, Complejidad O(k^2)

Calcular estadisticas por pregunta (mediana), complejidad O(k log k)

Insertar preguntas en la lista doble del tema, Complejidad O(n^2).

Insertar temas en lista enlazada general, Complejidad O(m^2).

Recorrer listas, Complejidad O(n + m)

Selección de preguntas, complejidad O(n)

Donde:

k => encuestados por pregunta.

n => número de preguntas.

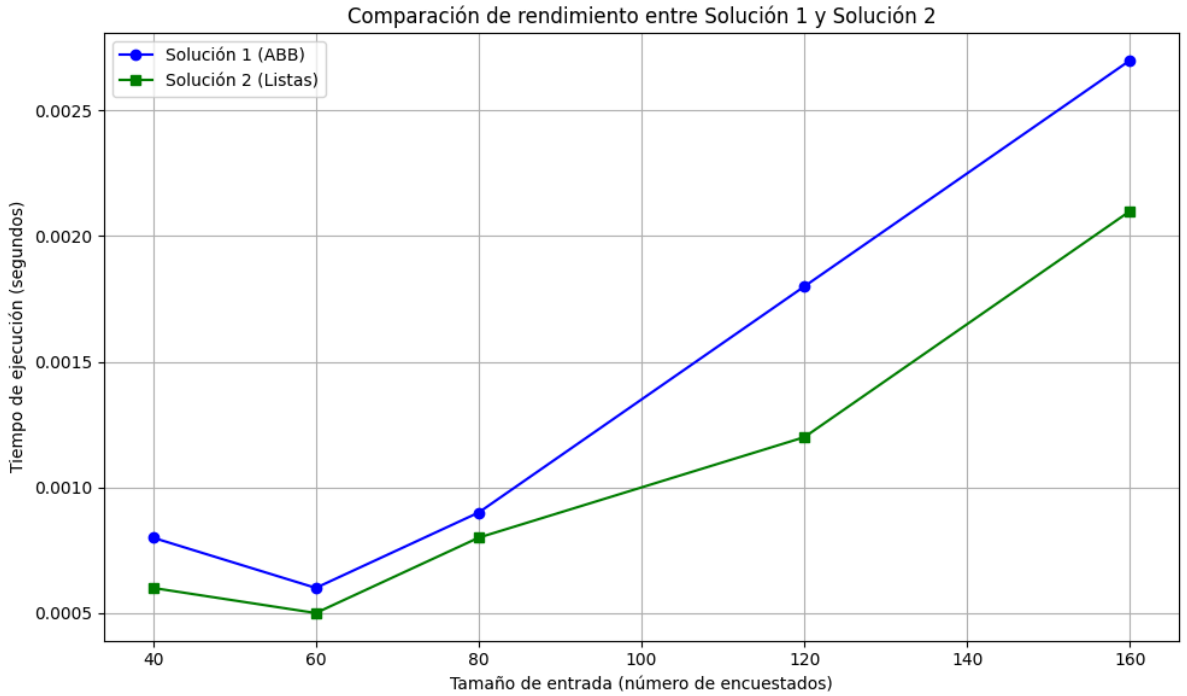
m => número de temas.

Complejidad Total

O(n^2 + k^2 + m^2)

4. PRUEBAS Y RESULTADOS.

Archivo	Tiempo S1 (s)	Tiempo S2 (s)
Test1.txt	0.0008	0.0006
Test2.txt	0.0006	0.0005
Test3.txt	0.0009	0.0008
Test4.txt	0.0018	0.0012
Test5.txt	0.0027	0.0021



A pesar de que teóricamente la Solución 1 (que emplea árboles binarios de búsqueda) presenta una complejidad más eficiente (O(nlogn) en comparación con la Solución 2 O(n^2), los tiempos de ejecución registrados muestran que Solución 2 son prácticamente iguales en todos los casos de prueba realizados.

Esto se debe a que, para tamaños de entrada relativamente pequeños como los utilizados (máximo 160 encuestados), el impacto de la complejidad cuadrática aún no se hace evidente. Además, factores como el menor overhead de estructuras lineales en Python, así como las diferencias de implementación, pueden generar estos resultados.

Sin embargo, en escenarios con volúmenes de datos significativamente mayores, se espera que la complejidad cuadrática de Solución 2 se refleje en tiempos de ejecución mucho más altos, validando la superioridad teórica de Solución 1.

5. CONCLUSIONES.

Durante el desarrollo del proyecto se implementaron dos soluciones al problema propuesto, cada una con estructuras de datos distintas: una basada en árboles binarios de búsqueda (ABB) y otra en listas y algoritmos de ordenamiento manual como insertion sort.

A pesar de la diferencia en complejidad teórica (O(nlogn) para ABB vs. O(n^2) para listas), los resultados experimentales mostraron tiempos de ejecución similares o incluso levemente mejores para la segunda solución en entradas pequeñas. Esto refleja la importancia de considerar no solo la complejidad teórica, sino también factores como implementación, lenguaje y tamaño de entrada.

Se evidenció que la solución con ABB, aunque teóricamente más eficiente dió los mismos resultados. Sin embargo, se espera que en entradas mayores la eficiencia de ABB sea notoria.

Se cumplió el desarrolló de ambos códigos cumpliendo las estructuras de datos distintas y ambas corriendo el código de forma correcta.