Capítulo 1

Proyecto

1.1 Descripción del problema

El programa implementa un analizador de texto que procesa un archivo `.txt` y construye tres estructuras principales para organizar la información:

- Una lista ligada que almacena todas las líneas del archivo en orden.
- Un hash map que almacena:
  - La frecuencia total de cada palabra.
  - El índice invertido, es decir, las líneas donde aparece cada palabra.
- Un árbol binario de búsqueda (BST) donde cada nodo contiene:
  - Una palabra
  - Su frecuencia
  - Una lista de líneas donde aparece

Con estas estructuras, el programa permite al usuario:

- Buscar palabras y sus frecuencias.
- Obtener las líneas donde aparece una palabra.
- Obtener el Top-K de palabras más frecuentes.
- Ver el índice invertido.
- Mostrar todas las líneas procesadas.

El programa simula lo que hacen ciertos motores de búsqueda o editores avanzados, pero implementado manualmente con estructuras de datos clásicas.

1.2 Justificación de las estructuras utilizadas

1.2.1 Lista ligada (ListaLigada)

Se usa para almacenar las líneas del archivo en orden. Permite:

- Guardar las líneas en el orden original.
- Mostrarlas sin reprocesar el archivo.

1.2.2 Hash Map (HashMap)

En este proyecto se usa para:

- Acceso rápido a la frecuencia de una palabra.
- Búsqueda instantánea de todas sus líneas.

1.2.3 Árbol binario de búsqueda (ArbolPalabras)

Se utiliza para:

- Guardar palabras con su frecuencia y líneas.
- Permitir búsquedas ordenadas.
- Facilitar el recorrido para obtener todas las palabras.

1.3 Arquitectura del programa

El programa se organiza en los siguientes componentes.

1.3.1 Estructuras de datos

NodoLinea y ListaLigada

Guardan las líneas del archivo:

$$[\text{Línea 1}] \rightarrow [\text{Línea 2}] \rightarrow [\text{Línea 3}] \rightarrow \cdots$$

HashMap

Contiene dos diccionarios:

frecuencias[“palabra”] = cantidad

indice_invertido[“palabra”] = [líneas...]

Árbol Binario de Palabras

Cada nodo contiene:

- palabra
- frecuencia
- lista de líneas
- hijo izquierdo
- hijo derecho

1.3.2 Módulos funcionales

`procesar_archivo()`

- Abre el archivo.
- Limpia palabras (solo alfanuméricas).
- Inserta cada palabra en:
  - Árbol
  - Hash map
  - Lista ligada (la línea completa)

Es el "motor" del analizador.

1.3.3 Funciones de consulta

`top_k()`

Obtiene las palabras más frecuentes usando un recorrido en-orden y luego ordenamiento.

`lineas_de()`

Devuelve las líneas donde aparece una palabra usando el árbol.

1.3.4 Interfaz de usuario (menú)

El menú permite:

- Procesar archivo
- Buscar palabra
- Mostrar top-k
- Ver índice invertido
- Ver frecuencias
- Ver líneas completas

1.4 Complejidad temporal y computacional

1.4.1 Lista ligada

- Insertar una línea: $O(1)$
- Recorrer todas las líneas: $O(L)$, donde L es el número de líneas.

1.4.2 Hash Map

- Insertar palabra:
  - Promedio: $O(1)$
  - Peor caso: $O(n)$
- Buscar frecuencia: $O(1)$
- Agregar línea al índice invertido: $O(1)$ amortizado

1.4.3 Árbol binario

No está balanceado, por lo que depende de la forma del árbol.

- Insertar palabra:
  - Mejor caso: $O(\log n)$
  - Peor caso: $O(n)$
- Buscar palabra: mismo análisis.
- Recorrido para top-K:
  - Recorrido en-orden: $O(n)$
  - Ordenamiento: $O(n \log n)$
  - Total: $O(n \log n)$