análisis de código

**Builder:**

* **CargarStopwords()** **O(n)**

Lee todas las palabras de un archivo y las guarda en un arreglo.

* **EsStopword(string token)** **O(n)**

Busca un token en el arreglo de *stopwords* (búsqueda lineal).

* **ProcesarDocumentos(string ruta, double zipf)** **O(n²)**

Llama a tokenización, creación de índice, TF-IDF, Zipf, vectores y guardado.

Lo más costoso es la construcción del índice y los vectores (O(n²)).

* **CargarTokenizador(string rutaCarpeta)** **O(n)**

Recorre los documentos en una carpeta y los tokeniza de manera lineal.

* **Tokenizador(string texto)** **O(n)**

Divide un texto en palabras y descarta *stopwords* en un recorrido lineal.

* **CrearIndiceInvertido(DocumentoActual[] documentos)** **O(n²)**

Para cada documento recorre todas sus palabras y las inserta en el índice.

* **CalculoTF\_IDF(IndiceInvertido index, int N)** **O(n)**

Calcula el peso TF-IDF de cada término de manera lineal.

* **AplicarLeyZipf(Termino[] terminos, double porc)** **O(n)**

Ordena o selecciona términos según su frecuencia (un recorrido lineal).

* **CalcularVectoresTFIDF(DocumentoActual[] documentos, Termino[] terminos)** **O(n²)**

Para cada documento calcula un vector con todos los términos doble recorrido.

* **GuardarIndicePreprocesado()** **O(n)**

Serializa los datos del índice y los escribe en archivo.

* **CargarIndicePreprocesado()** **O(n)**

Lee archivo y reconstruye el índice en memoria.

* **Consultar(string consulta)** **O(n²)**

Tokeniza la consulta, calcula similitudes con todos los documentos y ordena.

* **CalcularCoseno(double[] arr, double[] arr2)** **O(n)**
* Calcula el producto punto de dos vectores recorriéndolos una vez.

**Indice Invertido:**

* **AgregarTermino(string termino, int docId, int frecuencia)** O**(n)**

Primero recorre el arreglo de términos existentes para ver si ya está el término→ búsqueda lineal hasta terminosCont.

Si existe, recorre el subarreglo de documentos (hasta 1000) para insertar en la primera posición libre.

En el peor caso, recorre todos los términos y luego todo el subarreglo, lo que sigue siendo lineal respecto al número de términos y documentos por término.

* **ObtenerDocumentos(string termino)** **O(n)**

Recorre el arreglo de términos para encontrar el índice del término búsqueda lineal.

Si lo encuentra, recorre los documentos asociados hasta hallar la cantidad de documentos no nulos otro recorrido lineal.

Luego copia esos documentos a un nuevo arreglo también lineal.

Todos los pasos suman recorridos lineales, lo que se simplifica a O(n).

**Lista Circular:**

* **Add(T item) O(1)**

Crea un nuevo nodo lo cual es una operación constante y si la lista está vacía asigna las referencias de inicio y fin, que también es constante y al ser una lista circular no necesita recorrer la lista para insertar, todas son asignaciones directas, por ultimo incrementa el contador, lo cual también es constante.