Ejercicio Teórico 1

**El ejercicio nos pide que propongamos alguna estructura alternativa que mejore la complejidad espacial y temporal de las representaciones: matricial, de índice invertido, con tabla hash y con trie. Sin embargo, debido al hecho de que la tabla hash tiene una complejidad tanto espacial como temporal constante no se analizará para este ejercicio.**

El almacenamiento en matrices es uno de los más pesados que existen ya que depende tanto del número de términos como del número de documentos, almacenando muchos términos repetidos por cada vez que aparece en el documento dicha palabra.

El almacenamiento en índice invertido realiza una indexación de cada palabra una única vez, indicando en qué documento/s aparece/n. Para esto, se utiliza un vector desordenado en la que se almacena una pareja (término, idDocs), donde idDocs es un Array con el identificador del documento en el que aparece dicha palabra. Esto permite el acceso más rápido a la palabra en caso de que el Array esté ordenado por lo que deberíamos realizar las inserciones de manera ordenada o añadirle algún algoritmo de ordenado, como el quicksort, a la complejidad de este algoritmo.

El almacenamiento en Trie se basa en la creación de un árbol de prefijos de modo que, dependiendo de si se trata de una palabra completa o no, podemos acotar la región en la que se encuentra una palabra, acelerando el proceso de búsqueda aunque, dependiendo de cómo se implementara, podría hacer que la complejidad espacial se disparara ya que, al tener que almacenar prefijos de todas las palabras, los cuales podrían ser una única letra, llevaría a un árbol de altura igual a la longitud de la palabra más larga almacenada.

Un método de acceso alternativo a todos los mencionados anteriormente, sería la implementación de un sistema de indexado en el que se almacenen las palabras en una lista enlazada, compuesta de pares {término, documentos} como en el índice invertido, de modo que las inserciones se realizaran de manera alfabética, acelerando el proceso de búsqueda, minimizando, de este modo, el tiempo necesario para buscar cada una de las palabras.

Sería un Array que almacenara las listas enlazadas, ordenando las posiciones del Array según la letra inicial de cada una de las palabras, almacenadas en una lista con los índices, a modo de diccionario.

La complejidad tanto espacial como temporal dependería únicamente del número de palabras existentes ya que no depende de cuán larga sea la palabra ni del número de documentos a indexar como los métodos anteriores, así como del número de letras iniciales indexadas.

Inserción de las palabras:

|  |  |
| --- | --- |
| ALGAS | ALAS |
| ALGO | ALEMANES |
| ALGUNOS | ALEMAÑES |
| ALEMAN | ALGA |
| ZOOM | ALGOS |
| AL | ALS |
| ALA | LAURA |
| ZORRO | ZO |
| ZORROS | ZOOMS |

ÍNDICE DE LETRAS MAYÚSCULAS

|  |  |
| --- | --- |
| **LETRA** | **POS** |
| A | 1 |
| L | 2 |
| Z | 3 |

INDICE DE LISTAS ENLAZADAS

|  |  |
| --- | --- |
| **POS** | **LISTA** |
| 1 | {AL,ALA,ALAS,ALEMAN,ALEMANES,ALEMAÑES,ALGA,ALGAS,ALGO,ALGOS,ALGUNOS,ALS} |
| 2 | {LAURA} |
| 3 | {ZO,ZOOM,ZOOMS,ZORRO,ZORROS} |

La inserción se realiza de manera ordenada y cada palabra almacena junto a ella el/los documento/s en el/los que aparece/n.