**Codificacion\_Huffman**

Explicación

Estructura del Nodo Huffman:

HuffmanNode representa cada nodo del árbol de Huffman.

Clase HuffmanCoding:

Método codificar:

Calcula las frecuencias de los caracteres.

Construye el árbol de Huffman.

Genera los códigos binarios para cada carácter.

Codifica el texto de entrada usando los códigos generados.

Método decodificar:

Usa los códigos binarios para reconstruir el texto original.

Métodos Privados:

buildTree: Construye el árbol de Huffman usando una cola de prioridad.

buildCodes: Genera los códigos binarios para cada carácter recorriendo el árbol de Huffman.

**Compresion\_LZ**

Explicación

Estructura Trie:

searchLongestPrefix: Devuelve tanto la posición como la longitud del prefijo más largo encontrado.

Clase LZCompression:

Método comprimir:

Usa el trie para buscar el substring más largo que ya ha aparecido.

Añade un par a la lista comprimida: (posición, longitud, siguiente carácter) si el substring es encontrado; de lo contrario, solo el carácter actual.

Inserta el substring actual en el trie.

Método descomprimir:

Recorre la lista de pares comprimidos y reconstruye el texto original.

Explicación

Nodo del Trie:

TrieNode representa cada nodo del trie. Contiene un mapa de sus hijos y la posición del substring.

Clase Trie:

insert: Inserta una cadena en el trie con su posición.

searchLongestPrefix: Encuentra el prefijo más largo en el trie que coincide con una parte del texto actual.

Clase LZCompression:

Método comprimir:

Usa un trie para encontrar el substring más largo que ya ha aparecido.

Añade un par a la lista comprimida: (posición, longitud, siguiente carácter).

Método descomprimir:

Recorre la lista de pares comprimidos y reconstruye el texto original.