Caso Práctico

Estructuras de datos y algoritmos

Francisco José Ruiz Segovia
NIA: 100363894
Laura Álvarez Flórez
NIA: 100363965

Índice:

Fase 1

a) ¿Qué atributos son necesarios para implementar este diccionario?

Este diccionario implementa una lista doble, como atributos contiene dos nodos centinela de tipo Word: *header* y *trailer*.
El tamaño del diccionario almacenado en la variable de tipo integer *size* y dos listas dobles: *sortedFrequencyAscend* y *sortedFrequencyDescend*, donde se irán almacenando los elementos del diccionario ordenados de manera ascendente y descendente, respectivamente.

Fase 2

a) Para almacenar las palabras en orden alfabético, en los nodos, ¿qué utilizarás como clave?, ¿y cómo valor asociado?

Utilizaría como clave la propia palabra y como valor asociado su frecuencia.
Mediante el recorrido *inOrder*, se recorrería en orden alfabético.

c) Calcula la complejidad de cada uno de los métodos de la estructura diccionario.
Compara los resultados con los de la fase1, puedes crear una tabla con las funciones tiempos de ejecución de los métodos de la interfaz y analiza las diferencias, si las hay.

- Tabla comparativa:

Métodos	Complejidad Fase 1	Complejidad Fase 2
add (queue)	$O(n^2)$	$O(\text{nlog}(n))$
add (word)	$O(n)$	$O(\log(n))$
show ()	$O(n^2)$	$O(\log(n))$
search ()	$O(n)$	$O(\log(n))$
getTop ()	$O(n^2)$	$O(n^2)$
getLow ()	$O(n^2)$	$O(n^2)$

Analizando la tabla podemos observar que la Fase 1 tiene mayor complejidad en todos sus métodos, excepto en *getTop* y *getLow* donde no hemos conseguido reducirla más y se mantiene en n^2.

d) Supón que el diccionario DictionaryTree tiene también un método que recibe como argumento un objeto de tipo DictionaryList desarrollado en la fase 1 (es decir, una lista de palabras ordenadas alfabéticamente y sus frecuencias), y que recorre las palabras en dicho orden para ser insertadas en el árbol. Sin necesidad de implementar el método responde a las siguientes preguntas ¿Qué forma tendrá el árbol resultante?, ¿cuál será la complejidad de insertar una nueva palabra en ese árbol?, ¿qué técnica podríamos aplicar para para disminuir la complejidad de dicha operación?

Si insertáramos las palabras en orden alfabético, obtendríamos un árbol degenerado con forma de lista.
La complejidad de insertar una nueva palabra sería $O(n)$, al tener una estructura de lista.
Podríamos aplicar un equilibrado ABS para poder disminuir la complejidad hasta obtener una de orden logarítmico $O(\log(n))$.

Fase 3

- Complejidad de los métodos:

Métodos Fase 3	Complejidad
getWords ()	$O(n \log(n))$
getSinks ()	$O(n^2)$
searchLongChain ()	$O(n^3)$
search ()	$O(n^2)$