

Universidad Nacional de Córdoba

Facultad de Ciencias Exactas, Físicas y Naturales

**Algoritmos y Estructuras de Datos**

Parcial N°2

Conteo de repetición de palabras en un archivo de texto

Alumno:

Heredia, Marco

Matricula:

93560445

Fecha:

11 de junio de 2015

La consigna pedía analizar un archivo de texto, contar cuantas veces se repite cada palabra y devolver una lista con las palabras ordenadas del que mayor veces se repite a la palabra con menos repeticiones.

Para ello había que utilizar un árbol binario de búsqueda respetando las propiedades de AVL utilizando un arreglo de 1000 elementos para crear el árbol.

*Breve descripción de cada estructura y clase implementada*:

Estructura nodoAVL:

En los nodos del árbol AVL en lugar de utilizar punteros a un mismo tipo de nodo para llamar a los hijos del nodo, utilizo números enteros que son las ubicaciones de los hijos en el arreglo. Además utilizo un string para guardar la palabra y un entero para ir contando el número de veces que se repite la palabra.

Clase Lista:

Utilice la clase lista vista en las clases prácticas y le implemente 2 métodos: el método impre() se encarga de recorrer todos los nodos de la lista partiendo desde el comienzo hasta que encuentra un puntero \*next NULL e imprime en pantalla la palabra y el número de repeticiones de esa palabra en el texto. Además le implemente para que cada 50 palabras que muestra en pantalla se detenga para permitir observar todos los valores de la lista.

En la clase lista también implemente el método de ordenamiento inserción el cual se encarga de ordenar los valores de mayor número de repetición al de menor número realizando una subdivisión de la lista en la cual tiene una lista ordenada y otra desordenada. En cada pasada toma el primer valor de la lista desordenada y lo ubica en la posición correcta de la lista ordenada hasta que toda la liste este correctamente ordenada.

Clase arbolAVL:

Esta clase es la clase más importante del programa, la misma se encarga de crear el árbol a partir de una arreglo. Para armar esta clase use como base el código de la clase arbolAVL visto en las clases prácticas y lo adapte para que en lugar de trabajar con nodos de memoria dinámica utilice un arreglo de 1000 elementos. El método esta, se fija si una determinada palabra pasada como parámetro se encuentra en la lista y si se encuentra le suma uno al contador de repeticiones.

Además cree ciertos métodos que me permitieron una vez que el árbol este creado armar una lista con punteros a todos los elementos del arreglo.

Y por ultimo un método que se encarga de calcular la altura del árbol.

Función main:

En la función main me encargo de crear un objeto de la clase arbolAVL y de ingresar todas las palabras de un archivo de texto al a través de la función getline. Una vez que llego al final del archivo muestra en pantalla el árbol, aunque por el tamaño de este árbol no se lo llega a apreciar bien, luego crea una lista con punteros al arreglo para evitar duplicación de datos, ordena la lista con el método de la inserción y muestra la lista en pantalla ordenada de mayor a menor.

Por último llama a un método para calcular la altura del árbol y la muestra en pantalla, en este caso la altura del árbol es 11.

Opción de promoción

Para la opción de promoción, tenía que realizar una segunda versión del proyecto que tenga la misma funcionalidad pero esta vez en lugar de utilizar un arreglo como estructura de soporte del árbol AVL, tenía que utilizar una lista simplemente enlazada que tenga acceso casi directo a través de un segundo árbol de direccionamiento. Y el algoritmo de ordenamiento para esta parte debía ser el algoritmo de quicksort.

*Breve descripción de cada estructura y clase implementada*:

Estructura nodoLISTA:

Esta estructura es similar a la estructura nodoAVL de la versión anterior: en los nodos del árbol AVL en lugar de utilizar punteros a un mismo tipo de nodo para llamar a los hijos del nodo, utilizo números enteros que son las posiciones de los hijos en la lista. Además utilizo un string para guardar la palabra, un entero para ir contando el número de veces que se repite la palabra y un entero FB el cual simboliza la altura del subárbol izquierdo menos la altura del subárbol derecho, lo cual me ayuda a mantener el árbol balanceado con las propiedades del AVL.

Estructura nodoAVL:

Esta estructura es la estructura del nodo del árbol de direccionamiento y es similar a los nodos de árbol AVL vistos en clase con la particularidad que en lugar de almacenar valores enteros tiene un puntero a nodoLISTA el cual es el nodo de la lista y un valor entero llamado pos, el cual simboliza la posición de el nodoLISTA en la lista.

Clase Nodo y Lista:

Los utilice como clases auxiliares y simplemente utilice las clases vistas en clases prácticas sin realizar modificaciones.

Clase arbolAVL:

Esta clase es el árbol de direccionamiento a las posiciones en la lista, utilice como base lo visto en clases y únicamente le agregue una clase para que me devuelva el puntero de un nodoLISTA al pasarle una posición y otra clase para agregar un nodoLISTA a una posición dada.

Clase listaAVL:

Esta clase utiliza una lista en la cual va almacenando las palabras de un archivo de texto y para obtener acceso a esas palabras utiliza como soporte la clase árbol AVL explicada anteriormente.

Nuevamente modifique el código visto en clase para arboles AVL para que funcione con una lista y luego le cree los siguiente métodos:

El método esta, se fija si una determinada palabra pasada como parámetro se encuentra en la lista y si se encuentra le suma uno al contador de repeticiones.

En esta clase cree un método para realizar el ordenamiento QuickSort, el cual en primer lugar crea un arreglo con todas las posiciones de la lista y luego ordena las posiciones de acuerdo al valor que representan para dejar ordenado las palabras desde la que mas repeticiones tengan a la que menor número de veces se repite.

El método de Quicksort lo que realiza es elegir un pivot, en este caso siempre elige como pivot el último elemento y divide la lista en 2: los que son mayores que el pivot los deja a la izquierda y los que son mayores a la derecha del mismo. Luego repite este paso con cada una de las sublistas hasta tener todos los elementos ordenados correctamente.

Luego implemente un método que muestra en pantalla todos las palabras con su número de repeticiones ordenados de mayor a menor, nuevamente le implemente que pare cada 50 palabras para permitir una mejor visualización

Por último implemente un método para calcular la altura del árbol.

Función main:

En la función main al igual que en la versión anterior creo un objeto de la clase listaAVL y le agrego todas las palabras de un archivo de texto al a través de la función getline. Una vez que llego al final del archivo muestra en pantalla el árbol, aunque por el tamaño de este árbol no se lo llega a apreciar bien, luego llamo al método Quicksort() de la clase listaAVL para realizar el ordenamiento y muestro en pantalla la lista de las palabras ordenada de mayor a menor.

Por último llama a un método para calcular la altura del árbol y la muestra en pantalla.