Algoritmos y Estructuras de Datos
Segundo Parcial Año 2024

Mayo 2024

Conteo y ordenamiento de palabras

Una forma simple de comprimir un archivo de texto es reemplazar cada palabra por un código binario que la represente. Por ejemplo, la palabra casa se reemplaza por el binario 1001 que ocupa 4 bit en lugar de 4 bytes. Para que la compresión sea efectiva, se debe reemplazar la palabra más repetida por un código mínimo, y así en forma creciente en el tamaño de código hasta la palabra menos repetida. Para ello es necesario contar cuantas veces se repite cada palabra en el texto.

La primera parte de la tarea consiste en contar cuantas veces se repite cada palabra. Para ello deberá leer un archivo de texto de al menos 1000 palabras con al menos 200 palabras distintas e identificar cuales son las palabras que lo forman y cuantas veces están repetidas cada una de ellas. El archivo es de libre elección.

Para contar las palabras deberá disponer de un nodo para cada palabra distinta del archivo que sea simultáneamente un nodo de lista y un nodo de árbol binario de búsqueda (abb), es decir, que contenga tres punteros, uno al siguiente nodo de la lista y dos para el abb, generando una estructura combinada de lista con abb. Además, el nodo tendrá como valor a la palabra y al contador de repeticiones que esta tenga. El criterio de búsqueda del abb es alfabético para poder buscar cada palabra leída desde el archivo. Si la palabra no existe, se crea el nodo respectivo y se agrega en el abb y se enlaza en la lista de palabras al comienzo de esta. Si la palabra ya existe, se suma uno a la cantidad de repeticiones que esta tiene. Al finalizar la carga, generar e imprimir un lista de palabras ordenadas alfabéticamente haciendo un recorrido inorden del abb. Preservar esta lista para un uso posterior.

La segunda parte es del ordenamiento de las palabras y sus repeticiones. Luego de construido el abb, deberá reordenar los nodos de la lista por la cantidad de repeticiones en forma decreciente utilizando el algoritmo de selección y generar un listado de las palabras ordenado de esta manera.

Para realizar el ordenamiento tendrá que utilizar *swaps* de los datos de los nodos de la lista original sin cambiar los punteros que enlazan estos y para ello deberá usar accesos posicionales sobre los nodos de la lista. En otras palabras, para el ordenamiento, utilizar swap(i,j) sobre los elementos de la lista, donde i

y j son las posiciones del nodo i-ésimo y j-ésimo de la lista intercambiando los datos de los nodos i y j, similar a como si fuera usando un arreglo.

Para que el acceso al i-ésimo valor de la lista no sea secuencial, deberá construir un árbol posicional binario semi-completo cuyas hojas sean punteros a los nodos de la lista. Este árbol no contiene los datos de las palabras, solo permite acceder en orden logarítmico a la j-ésima posición de la lista y trabaja de la siguiente forma:

- Para acceder a la posición k, deberá restar 1 a k y transformarlo es su equivalente número binario. Ej: si quiero ir a la posición 13, debo usar el binario de 12, 1100.
- el 0 y el 1 del número binario indican si debo tomar la rama izquierda (0) o derecha (1) del árbol, leyendo el binario de izquierda a derecha.

En el caso de 1100, la navegación es, partiendo de la raíz del árbol posicional, derecha, derecha, izquierda, izquierda.

Es decir que las posiciones 1 a n de los elementos de la lista se deben mapear de 0 a n-1 en el árbol posicional. La altura del árbol posicional deberá corresponderse con la menor potencia de 2 que sea mayor al número de palabras. Ej: Si tengo 300 palabras, el binario deberá ser de 9 dígitos, ya que 2 a la 8 es 256 y 2 a la 9 es 512.

Como tercera parte, debido a que el ordenamiento por número de repeticiones altera el orden alfabético de cada palabra, deberá volver a ordenar alfabéticamente la lista de nodos, en esta ocasión por medio del algoritmo Quick-sort, siguiendo las mismas restricciones del caso anterior, es decir, accediendo a la lista utilizando el árbol posicional e intercambiando los contenidos de los nodos, sin cambiar el enlace de los punteros. En cada ordenamiento (selección y quick-sort) haga un conteo de la cantidad de comparaciones que ambos algoritmos realizan para poder validar el orden de complejidad de ambos.

Como prueba de correctitud del ordenamiento, deberá controlar que el resultado de este ordenamiento debe coincidir con el listado generado en la primera parte del proyecto, es decir ambos resultados deben tener la misma cantidad de palabras y estas figurar en el mismo orden.

Tip

Considere un tipo de nodo de datos que solo tenga la palabra y la repetición y otro tipo de nodo para el árbol y la lista que en lugar de los datos tenga un puntero al nodo de datos. Esto a los fines de facilitar el swap de los datos, ya que solo intercambia punteros en lugar de datos.

Utilice los algoritmos dados en clase, modificandolos para implementar la solución.

Bonus

Se considerará como *bonus* en la evaluación que en lugar del abb se utilice un AVL balanceado.