PRÁCTICA 1

ALGORITMIA BÁSICA

Diego Martínez Baselga 735969

David Solanas Sanz 738630

En esta práctica se ha desarrollado un programa compresor y descompresor que permite la compresión y descompresión de ficheros mediante el método de códigos Huffman. Se han desarrollado los ficheros Arbol_caracteres.h, Monticulo_arboles.h y p1.cpp, que constituyen el código fuente; junto con un fichero Makefile y un conjunto de pruebas.

Arbol_caracteres.h

En primer lugar se ha desarrollado el fichero Arbol_caracteres.h, que implementa el tipo de dato correspondiente a un árbol de Huffman. El TAD tiene los siguientes atributos:

- Frecuencia (entero sin signo): Almacena la frecuencia de aparición del árbol.
- Caracter (carácter sin signo): Byte al que corresponde ese nodo del árbol. Solo estará definido si corresponde con una hoja.
- Der (Puntero a Arbol_caracteres): Hijo derecho del árbol. Si es hoja será puntero a null.
- Izq (Puntero a Arbol_caracteres): Hijo izquierdo del árbol. Si es hoja será puntero a null.

Tiene 3 constructores distintos. Si se llama al constructor con un entero sin signo y un carácter sin signo (byte), se crea una hoja con esos valores. Si se le llama con dos punteros a árboles, se crea un árbol cuya suma es la suma de frecuencias de los árboles y esos árboles pasarán a ser sus hijos. Si se le llama con un solo árbol, este pasará a ser su hijo izquierdo (caso en el que en el fichero hay un solo tipo de byte).

Tiene un procedimiento, que tiene como parámetro un flujo de salida, que sirve para escribir el árbol en ese flujo. De cada nodo, escribe el byte al que corresponde (indefinido si no es hoja) y un booleano indicando si es hoja o no. Para escribir el árbol, se recorre por anchura con ayuda de una cola, escribiendo primero la raíz, después sus hijos, los hijos de estos y así sucesivamente. No se escribe la frecuencia del carácter ya que no es necesaria para recuperar el árbol (solo es necesario recuperar su estructura y el valor de los bytes), y se escribe si es hoja porque el árbol no es completo y de esta manera se posibilita su reconstrucción.

Monticulo_arboles.h

Tipo de dato utilizado para crear el árbol resultante de la fusión de todos los árboles iniciales correspondiente al algoritmo de Huffman. Tiene los siguientes atributos:

- Vector (array de tamaño 256 de punteros a árboles): Almacena los punteros a los árboles ordenados por frecuencias formando un montículo, de manera de que los hijos de cada nodo del montículo son, para la posición i, 2 * i +1 y 2 * i + 2. Tiene tamaño 256 porque un byte puede tomar 256 valores distintos y, por lo tanto, puede haber 256 valores como máximo.
- Ultimo (entero): Posición en el vector del último nodo que forma parte del montículo. Si está vacío toma el valor -1.

Este TAD tiene implementadas las funciones de insertar un nuevo árbol en el montículo y borrar y devolver el árbol que está en la cima.

Comprimir

El fichero p1.cpp contiene la función main que corresponde con el programa principal. Comprueba si los parámetros introducidos son correctos y si la opción es -c comprime el archivo de nombre <nom_fichero> a <nom_fichero>.huf. La función de comprimir sigue el siguiente esquema:

- Leer el fichero: Se lee el fichero y se almacena su contenido en un vector de caracteres sin signo, almacenando su contenido byte a byte. Se realiza minimizando las llamadas al sistema realizadas.
- Calcular frecuencias de cada carácter: Se almacena en un vector de 256 enteros la frecuencia de cada byte en el fichero, almacenando 0 para los bytes que no aparecen. Se reacliza recorriendo el vector del contenido.
- Calcular árbol de Huffman: En primer lugar, se utiliza el vector de frecuencias recién calculado para crear un montículo de árboles de un solo nodo, cada árbol siendo una hoja que es un byte del fichero con su frecuencia de aparición. A continuación, mientras el montículo tenga más de un árbol, se retiran los dos primeros, se fusionan utilizando el constructor explicado en el fichero Arbol_caracteres.h y se inserta el árbol resultante. Se tienen en cuenta los casos en el que el fichero solo tiene un tipo de byte o es un fichero vacío. Cuando solo queda un árbol en el montículo, se devuelve un puntero a ese árbol.
- Calcular códigos: Se utiliza una función recursiva para recorrer el árbol añadiendo al llegar a cada hoja el código del byte de esa hoja en un vector. La función toma como parámetros el árbol de Huffman, el vector de códigos y el código acumulado del nodo que se visita. Este código la primera vez que se llama a la función es una cadena vacía, mientras que se le añade un ‘0’ si se accede al hijo izquierdo y un ‘1’ si se accede al derecho. Los códigos en este nivel se almacenan como cadenas, para facilitar su obtención y posterior manipulación.
- Escribir fichero comprimido: Se abre un fichero con nombre el del fichero a comprimir añadiendo la extensión .huf. En este fichero se escriben los siguientes elementos en el siguiente orden:
  - Bits extra: El tamaño del código resultante cifrar el archivo normalmente no es divisible por 8. Como se almacena a nivel de bits, es necesario añadir al código con un número de bits que completen el último byte (se añaden bits 0). Este número de bits se guarda en el fichero comprimido porque será necesario al descomprimirlo para descartarlos.
  - Número de nodos del árbol: Facilita la recuperación del árbol del fichero.
  - Tamaño del archivo original: Utilizado para reservar espacio en memoria para el vector donde se almacena el contenido al descifrar.
  - Árbol de Huffman: Se escribe utilizando la función explicada en el apartado de Arbol_caracteres.h.

- Fichero cifrado: Se recorre el vector que almacena los bytes del fichero original. Para cada uno, se obtiene su código y se añade a una variable que acumula los códigos. Cuando la longitud de esa variable es mayor que 8, se utiliza desplazamiento de bits para obtener el byte correspondiente con sus primeros 8 elementos, se escribe y se borran los 8 digitos escritos de la variable. Se realiza esta secuencia hasta que ya no queden más bytes por recorrer. Cuando termina, es posible que la variable acumuladora tenga una longitud mayor que 0 (no se ha llegado a formar el último byte). Se completa ese byte con 0 y se escribe.

Descomprimir

Al descomprimir, se lee el fichero <nom_fichero>.huf y se escribe el resultado de la descompresión en <nom_fichero>. Se siguen los siguientes pasos:

- Se leen del fichero, en el orden explicado en el apartado de comprimir, el número de bits extra, el número de nodos del árbol, el tamaño del archivo original, el árbol y el código. Para recuperar el árbol se utiliza la ayuda de una cola, que permite recorrer el árbol por anchura. El código se almacena un un vector de caracteres sin signo (bytes).
- Se recorre el vector del código , recorriendo cada uno de sus elementos bit a bit y siguiendo la siguiente estrategia: Empezando en la raíz, cada bit 0 implica acceder al hijo izquierdo y 1 al derecho. Cuando se accede a una hoja, se añade al vector con el contenido descifrado el byte correspondiente y se vuelve a acceder a la raíz. Se repite hasta que falten por recorrer el número de bits extra que se ha leído como primera variable del archivo.
- Finalmente, se escribe el vector de bytes resultante en el fichero de salida, obteniendo así el fichero descomprimido.