Multimedia

Assignment 3 – Standard Huffman Compression Algorithm

- GUI and files are mandatory.
- The compression code MUST be saved in bits.
- The deadline is two weeks.
- If you submit the assignment next week you will get bonus grades.

Clarification for saving in bits requirement

The Huffman coding algorithm compresses the input stream into a compression code of 0 and 1 bits.

Assume that after doing compression you have

String compressionCode ="11010110010011000100101001110010010011010101100110";

According to Huffman algorithm the compressed file size = Number of bits = 50 bit.

The problem is, As you know, the string is a sequence of characters so when you save the above compression code to a file, it will deal with every 0 and 1 as a character (7 bits) not 1 bit.

So, the compressed file size will be = 50 * 7 = 350 bit rather than 50.

So, you are required to find a more efficient way to save the compression code.

Hints:

- In Java the smallest unit that you can read/write is byte.

- The Most efficient way is
  - The size of byte is 7 bits (the eights bit is reserved for the sign), so you can merge each 7 bits of the code into a bytes and write these bytes to a file.

- Another acceptable way

  - The size of Integer is 31 bit (the 32 bit is reserved for the sign), so you can merge each 31 bits of the code into Integers and write these Integers to a file.

  - Use the method ParseInt(String, radix) from class Integer to convert the stream of bits to an integer.

    ```
    Integer.parseInt("11010011111110101010",2) the
    result is 868266 saving this integer will require
    4 bytes only rather than 20 bytes if you saved
    the stream directly as a string.
    ```

  - Use method toBinaryString(int) from class Integer to convert the read integer value to string of 0 and 1 so you can do the decompression.

    ```
    Integer.toBinaryString(868266) the result is the
    string 11010011111110101010
    ```

- The following code snippet writes/ reads integers to/from a file.

```
      FileOutputStream fos = new FileOutputStream("code.txt");
     ObjectOutputStream oos = new ObjectOutputStream(fos);
      oos.writeInt(1768005546);
      oos.writeInt(50);
      oos.close();
      fos.close();

// you can check the file size, it will increase by 4 bytes after
adding each integer
      File f = new File("code.txt");
       System.out.println(f.length());
      f.close();

FileInputStream fis = new FileInputStream("code.txt");
ObjectInputStream ois = new ObjectInputStream(fis);
System.out.println(ois.readInt());
       System.out.println(ois.readInt());
ois.close();
fis.close();
```