Programming Assignment 8

Due 16 NOV @ 11:59pm

Finish the implementation of Huffman compression and decompression using provided template. Several methods are already provided and ideally should be used. You need to provide implementations for the compress, decompress, makeTrie, makeCodingTable, and determineFrequencies methods (currently empty). You are also responsible for providing the documentation for these methods. You only need to modify the methods in the Huffman.java file.

The main method takes two parameters, <+ | -> <filename>. If you want to compress a file, use - <filename>, to decompress use + <filename>. Compress creates a new file with name of original appended with “zip”, e.g. “input.txt.zip”. The decompress prints to standard out to avoid overwriting the original file.

Notes on using the BitStreamOutput/Input in the compress/decompress methods:

In compress, to write the number of symbols use the writeBits(...) method, for example:

```
out.writeBits(text.length, 31);
```

To write out a code (stored in the coding table as a String, e.g. "0101"), go through each char and use writeBit(boolean):

```
if(      code.charAt(j) == '0' ) out.writeBit(false);
else if( code.charAt(j) == '1' ) out.writeBit(true);
```

In decompress, the input should be performed similarly, to read in the number of symbols:

```
int n = in.readBits(31);
```

The codes were written out a bit at a time, so you need to read in a single bit at a time. You use the boolean returned to traverse the trie to a leaf which has the corresponding symbol. This is best accomplished with an iterative while loop.

```
if( in.readBit() ) // go right
else               // go left
```

Grading Notes

You must:

- Use the template provided for you
- Have a style (indentation, good variable names, etc.)
- Comment your code well (no need to over do it, just do it well)

You may not:

- Make your program part of a package.
- Use *code* from anywhere except your own brain.

Submission Instructions:

- Name a folder with your gmu username
- Put your java files in the folder (but not your .class)
- Zip the folder (not just the files) and name the zip “username-pa8.zip”
- Submit to blackboard

Grading Rubric

No Credit:

- Non-submitted assignments
- Late assignments
- Non-compiling assignments
- Non-independent work

1pt	Submission Format
1pt	Style and Comments
2pt	compress
2pt	decompress
2pt	makeTrie
1pt	makeCodingTable
1pt	determineFrequencies

Example Compress

```
> java Huffman - algorhyme.txt
> ls -la
total 80
drwxr-xr-x. 2 user user 4096 Nov  3 14:35 .
drwxrwxr-x. 4 user user   67 Nov  3 14:10 ..
-rw-r--r--. 1 user user  386 Nov  3 14:31 algorhyme.txt
-rw-rw-r--. 1 user user  262 Nov  3 14:35 algorhyme.txt.zip
-rw-rw-r--. 1 user user 1972 Nov  3 14:34 BitStreamInput.class
-rw-rw-r--. 1 user user 2642 Sep  6 20:40 BitStreamInput.java
-rw-rw-r--. 1 user user 4268 Nov  3 14:34 BitStreamOutput.class
-rw-r--r--. 1 user user 5817 Sep  6 20:41 BitStreamOutput.java
-rw-rw-r--. 1 user user 4759 Nov  3 14:34 Huffman.class
-rw-rw-r--. 1 user user 9756 Nov  3 14:07 Huffman.java
-rw-rw-r--. 1 user user 1842 Nov  3 14:34 Huffman$Node.class
-rw-rw-r--. 1 user user 2426 Nov  3 14:34 MinHeap.class
-rw-rw-r--. 1 user user 3528 Sep  5 22:32 MinHeap.java
-rw-rw-r--. 1 user user 4065 Nov  3 14:34 Stdio.class
-rw-rw-r--. 1 user user 6950 Sep 13 16:53 Stdio.java
```

Example Decompress

```
> java Huffman + algorhyme.txt.zip
Decompressed: I think that I shall never see
A graph more lovely than a tree.
A tree whose crucial property
Is loop-free connectivity.
A tree that must be sure to span
So packets can reach every LAN.
First, the root must be selected.
By ID, it is elected.
Least-cost paths from root are traced.
In the tree, these paths are placed.
A mesh is made by folks like me,
Then bridges find a spanning tree.

>
```