The purpose of this assignment is to practice your knowledge of trees and Huffman's algorithm for data compression. This assignment must be completed in pairs.

1. The Problem
   We have been asked to construct a program that takes in a text file and generates a compressed version of the file. We decide to use Huffman's algorithm to achieve the compression. The algorithm uses a binary trie to encode each character in the input file. Frequently occurring characters are encoded with fewer bits than less frequently occuring characters.

   Our program should also take in a file compressed using Huffman's algorithm and generate the decompressed version. This will be handy for testing our compression strategy.

2. Requirements

   - `HuffmanTree` class

     The `HuffmanTree` class is provided and contains the following three methods left for you to fill in.

     - `int compareTo(Node rhs)` of the nested `Node`class
       Recall that Huffman's algorithm repeatedly merges the two smallest-weight trees into a new tree. In order to do that, tree nodes must be Comparable (and thus, have a `compareTo` method). The priority queue used in the next part will use compareTo to determine which node is smallest.

       **Remember to use a tie-breaker**. (See lecture 22)

     - `void createTree()`
       This method constructs a Huffman tree to encode each character in the original file according to Huffman's algorithm. See the algorithm and examples in Lecture 22.

     - `int[] getCode(int ch)`
       This method returns the bit code (represented as an array of 0s and

1s) for the character given as input by traversing the path from the character's leaf node up to the root of the tree. Encountering a left child causes a 0 to be pre-appended to the bit code, and encountering a right child causes a 1 to be pre-appended. See the algorithm and examples in Lecture 22.

Add your own private methods as needed, but DO NOT alter the signatures of the provided methods.

- `CompressionDemo` class

  The `CompressionDemo` class is provided and demonstrates how to use the `HuffmanTree` class to compress a file and to decompress a file.

- `BitInputStream` class

  The `BitInputStream` class is provided and is used to read bit codes from a compressed file.

- `BitOutputStream` class

  The `BitOutputStream` class is provided and is used to write bit codes to a compressed file.

Create your own tests and submit them with your program.

3. When preliminary coding is complete and your program compiles without error or warning, test the program thoroughly and systematically.

   Your code should be well-commented (Javadoc comments are recommended) and formatted such that it is clear and easy to read. Be sure to put the names of both programming partners in the header comment of each file.

   Zip your source code files (`.java` only) and **upload the zip file here by 5p on April 11**. Please submit just one solution per pair (i.e., one partner should upload the zip file, the other should not upload anything).

4. Analysis Document (must be written and submitted by each programming partner) ***due April 11 at 5p***

| **Due date:** | Thursday, 11 April 2013, 5:10 PM |
|---|---|

## Submission feedback

Daniel Kopta
Monday, 29 April 2013, 7:44 PM

Grade: 95.00 / 100.00

```
------------------------------

-----Small File Compression----

PASSED: "inClass1.txt" compression resulted in large file.

PASSED: "inClass1.txt" successfully decompressed

PASSED: "helloWorld.txt" compression resulted in large file.

FAILED: "helloWorld.txt" failed decompression.

PASSED: "tie_1.txt" compression resulted in large file.

PASSED: "tie_1.txt" successfully decompressed

PASSED: "abcs.txt" compression resulted in large file.

PASSED: "abcs.txt" successfully decompressed

PASSED: "tie_2.txt" compression resulted in large file.

PASSED: "tie_2.txt" successfully decompressed

PASSED: "tie_3.txt" compression resulted in large file.

PASSED: "tie_3.txt" successfully decompressed



------------------------------

----Medium File Compression----

PASSED: "modernMajor.txt" compression resulted in smaller file.
```

```
PASSED: "modernMajor.txt" successfully decompressed

PASSED: "shakespeare.txt" compression resulted in smaller file.

PASSED: "shakespeare.txt" successfully decompressed




-------------------------------

----Large File Compression----

PASSED: "guliversTravels.txt" compression resulted in smaller file.

PASSED: "guliversTravels.txt" successfully decompressed

PASSED: "senseAndSensi.txt" compression resulted in smaller file.

PASSED: "senseAndSensi.txt" successfully decompressed




-------------------------------

--------Scoring Results--------



5/5  TA spot check.

56/60  Compression/decompression tests.

/10  Quality of tests and style.

24/25  Analysis document.

95/100  Total points.

-------------------------------

----------TA Comments----------
```

Nice work.

See attached pdf for addition comments.

analysis.pdf

## Submission

analysis.pdf

**No further submissions are allowed.**