CS211 ALGORITHMS & DATA STRUCTURES II

LAB 3

Dr. Phil Maguire

QUICKSORT - HUFFMAN CODES

Pen and Paper Exercise

Show how the following numbers would be sorted by quicksort, first using ordinary quicksort (with the rightmost element as pivot) and then again using median of 3 quicksort.

Show the pivots and the order of swaps that take place. For example, you should indicate which elements are already sorted, which one is the pivot, and where the left and right scans are.

7 8 2 5 1 9 3 6

Programming Exercise

You can add to last week's code if you like. If you did not fully complete last week's lab then you can download the following three files:

i)	Huffman.java	(reads in the sentence and breaks up the letters)
ii)	Tree.java	(implements all the binary tree methods)
iii)	Node.java	(simple Tree node class)

The task is to complete the Huffman algorithm so that it takes in a sentence and outputs Huffman codes. Several chunks of the code are missing in Huffman.java:

i) Fill in the code that creates the forest of trees for each letter and adds them to the Priority Queue

ii) Fill in the code that implements the Huffman algorithm

iii) Fill in the code that prints out all the results, including the rate of compression achieved

When you have all that finished the output of your program should look like this:

```
Enter your sentence: The cats did not sit on the mat

1010100 1101000 1100101 0100000 1100011 1100001 1110100 1110011
0100000 1100100 1101001 1100100 0100000 1101110 1101111 1110100
0100000 1110011 1101001 1110100 0100000 1101111 1101110 0100000
1110100 1101000 1100101 0100000 1101101 1100001 1110100

' ' appeared 7 times
't' appeared 5 times
'a' appeared 2 times
'd' appeared 2 times
'e' appeared 2 times
'h' appeared 2 times
'i' appeared 2 times
'n' appeared 2 times
'o' appeared 2 times
's' appeared 2 times
'T' appeared 1 time
'c' appeared 1 time
'm' appeared 1 time


01010 1110 0110 00 11010 1011 111 1101 00 0011 1001 0011 00 0001
0101 111 00 1101 1001 111 00 0101 0001 00 111 1110 0110 00 0010
1011 111

Compressed size is 107 bits / 217 bits = 49 %
```

Advanced Programming Exercise

The advanced programming challenge is to finish your algorithm off by creating a real viable compressor application which reads in a file and compresses it into a smaller file. You will have to store the translation table in the compressed file in order to be able to decompress it, and store the 1s and 0s as true binary, rather than ASCII symbols!