COP 3503 - Programming Assignment #5
A Text Processing System
Total: *12 points.*
Assigned: Feb 21, 2012 (Tuesday)
Due: April 12, 2012 (Thursday) at 11:55 PM WebCourses time

Objective
Text searching and text compression.

Problem: Develop a system for
1. Text searching using: Brute force, Knuth-Morris-Pratt and Boyer-Moore-Horspool algorithm. In each case compute the position of all occurrence of pattern in the text, and the number of character comparison the algorithm used for the computation. [6 points]
2. Text Compression/decompression using Huffman Coding: (a) Computation of Huffman code (b) creating compressed code (c) decompression using the generated Huffman code. [6 points]

Instructions:
1. Text Searching: For each algorithm you must compute and print the following:
    - "the position of all occurrence of pattern in the text, and the number of character comparison the algorithm used for the computation."
        - Position: If you assume that the text is an array of characters, then the position will be the index in the array at which the pattern is found during the search.
        - Number of comparisons: How many character comparisons your algorithm makes for this search?
        - All occurrences: The pattern may appear multiple times in the text. So for each occurrence you must compute the position and number of comparisons.
    - For occurrence of overlapping patterns in the text use the approach followed by standard search engines:
        - find the overlapping occurrence of the pattern separately. For example, if the text was “abababab” and the pattern was “abab” then each of your algorithms must find 3 occurrences:
            - First at location 0
            - Next at location 2
            - Finally at location 4.
        - For implementing this approach you should start searching at i = 0, and after a pattern is found at location i, you should start the next search at i = i+1 (unless you are at the end of the file).
2. Huffman Encoding/Decoding: The algorithm should have the following component.
    - Compute frequency of each character in the supplied text
    - Create a node for each character with character and the frequency as the node elements.

- Construct a min-Heap H for the nodes, with frequency as the key
- Create Optimal Coding tree (see text book for detail)
    - for i = 1 to n-1 // n = the number of characters in the alphabet
        - node1 = H.removeMin()
        - node2 = H.removeMin()
        - node = new Node
        - node. frequency = node1.frequency + node2.frequency
        - node.left = node2
        - node.right = node1 // Smallest one to the right as our text book does
        - H.insert(node) // insert the node to heap
- Generate code for the characters: Carry out any of the binary tree traversal technique (in-, pre-, or post- order). Accumulate an ordered list of binary characters as you traverse.
    - Start at root. Initialize the ordered list to null.
    - if the current node is leaf: then the ordered list of binary characters represent the Huffman code for the character at the node. Save that code with the character. You will use it for encoding.
    - if you are traversing down
        - if the traversal leads to left child then append 1 to the list
        - if the traversal leads to right child then append 0 to the list
    - If you traverse back up the tree (during traversal) then remember to remove the binary character from the end of the list.
    - If you have completed the traversal then the code creation is complete.
- Coding: Read text character at a time. For every character in the text, output its Huffman code.
- Decoding: The code tree is required for decoding. The algorithm is simple:
    - While(not end of code text)
        - Start at the root of the tree. i.e. node is root
        - while (node is not leaf)
            - Read encoded text a character at a time.
            - if character is 1 then traverse to left. i.e. node =node.left
            - else traverse to right. i.e. node = node.right
        - output the character at the leaf node
        - reset the node to root and continue

Input

The program should be designed accept text from file **Assignment05.txt** and a pattern from standard input

Output: (see sample file for data and output)

1: For searching:

Occurrences and comparison numbers printed to standard output.

2: For compression: Write to standard output:

Huffman code for each character, (one character followed by its code per line)

encoded data (as text of binary characters)
and
decoded output as normal text. (This text should match the text from the input file.)

Deliverables

You must submit the source code (The .java files, not the .class files) for your programs, and report over WebCourses by 11:55 PM on Thursday, April 12, 2012. You must upload your source files as an attachment using the "Add Attachments" button. Assignments that are typed into the submission box will not be accepted.
The package must contain:
- Assignment05.java, the main java files and any additional for your assignment.
- Assignment05.pdf: the writing component of your assignment. This document must contain three sections.

(a) Status of the project

(b) Cost analysis for each part of the assignment: (cost in asymptotic notation and the justification of the cost)

- For search cost for each search algorithms with analysis.
- For encoding/decoding: Cost analysis for each major component listed in the instruction section.

(c) A sample text and corresponding output.

Restrictions

- Follow posted assignment submission guideline.
- Your program must accept the input file (**Assignment05.txt**) from the same directory as where the Java source file is placed. NOTE: Do not code any absolute directory path.

Grading Policy: Total 12 points

You program must compile and run. (0 credit for code not compiling or not running.)
This submission will be graded from a total of 12 points using the following criteria:

- Correct implementation of search algorithm + associated report material (2 point for each algorithm. Total 6 points)
- For Encoding/Decoding + associated report material (Total 6 points)
  - Computation of Huffman code (3 points)
  - Encoding Part (1 point)
  - Decoding Part (2 points)