CPSC 2150 – Algorithms and Data Structures II

Lab6: Binary Tree – Huffman Encoding

Total - 70 Marks

Learning Outcomes

- Design and develop an appropriate binary tree
- Design and implement Huffman tree
- Extract the Huffman encoding tries from a given text
- Analyzing the Huffman algorithm
- Program with C++

Resources

- Chapter 12.4 of the text book

Description

Exercise 1 – Optimal encoding: Given the message EYEWITNESSNEWS, answer the following questions in **answers.pdf.**

1. **[7 marks]** Construct two different Huffman encoding tries for the given message. Show all the intermediate steps.

2. **[3 marks]** Use your two tries to produce the corresponding binary encodings for the message. Compare the numbers of bits of the two encoded messages. Are they of the same length?

3. **[4 marks]** This exercise is intended to let you know based on the same input data, we can construct several encodings tries in different ways (as you're doing in the first part). The problem is: Which one is the best? Does it depend on our choice in building the trie?

Exercise 2 - The tallest Huffman trie: Let F be a text file consisting of 128 characters, each in the set A = {a, b, c, d, e, f, g, h}, and let T be a Huffman encoding trie for F. The height of T depends on the frequencies the characters of A in F (the frequency of a character is the number of occurrences of that character in the file).

1. **[8 marks]** Assign a frequency to each character of A such that the height of T is maximum.

2. **[8 marks]** Assign a frequency to each character of A such that the height of T is minimum.

In each case, draw T and determine the length (number of bits) of the encoded file. Note that in each case, the sum of the frequencies of the characters must be 128. **(answers.pdf)**

Exercise 3 - Infinite-compression **[5 bonus marks]** X, who has completed his Ph.D on compression algorithms, has proposed an algorithm, which he claims, compresses ***any*** file by at least 10% of its original size. (i.e. If A file's length was t bytes before compression, its not greater than 0.9t after compression). Should he be shortlisted in the Google research team? Justify your conclusion. **(answers.pdf)**

Exercise 4 – Programming: Consider the following algorithm to implement and analyze a compression schema based on Huffman coding.

Algorithm Huffman(X):
Input: String X of length n with d distinct characters
Output: Coding tree for X
Compute the frequency $f(c)$ of each character c of X.
Initialize a priority queue Q.
for each character c in X **do**
  Create a single-node binary tree T storing c.
  Insert T into Q with key $f(c)$.
while Q.size() > 1 **do**
  $f_1 \leftarrow Q$.min()
  $T_1 \leftarrow Q$.removeMin()
  $f_2 \leftarrow Q$.min()
  $T_2 \leftarrow Q$.removeMin()
  Create a new binary tree T with left subtree T_1 and right subtree T_2.
  Insert T into Q with key $f_1 + f_2$.
return tree Q.removeMin()

1. **[22 marks]** write an efficient function named makeHuffmanTree that implements the above algorithm. It creates and returns the Huffman tree for a given input such as EYEWITNESSNEWS (**Huffman.cpp**).

   Note: (1) using STL for priority queue is allowed. (2) Huffman algorithm is case sensitive.

2. **[10 marks]** write a function named **printTrie** that given Huffman tree prints the Huffman encoding trie (**Huffman.cpp**).

3. **[4 marks]** find the complexity of **makeHuffmanTree** given X of length n and d distinct character. **(answers.pdf)**

4. **[4 marks]** find the complexity of **printTrie** given X of length n and d distinct character. **(answers.pdf)**

Submit to D2L

Make a **zip file** named **StudentNumber-lab6.zip** including all related files by the end of the lab time**.** For example, if your student number is 10023449, the submitted file must be named as **10023449-lab6.zip**.