CSc 3102: Huffman Trees

Encoding Data

- Huffman Codes

1 Introduction

Finding a binary tree of minimum weighted leaf path length has many important applications including Huffman codes and merge patterns. In this lecture, we focus on merge codes. Let λ_i denote the length of the path in T from the root to the leaf node corresponding to some symbol α_i, $i = 1, \ldots, n$. The expected length of the coded symbols determined by T is closely related to a generalization of the leaf path length called the *weighted leaf path length* $\Pi_{wl}(T)$ *of* T, defined by

$$\Pi_{wl}(T) = \sum_{i=1}^{n} \lambda_i f_i$$

From information theory, entropy is the lower bound on the average number of bits required to encode the symbols in a source. The entropy, denoted H, of the codes for a source is given by the formula below, where w_i is the weight or relative frequency of each of its n distinct symbols:

$$H = -\sum_{i=1}^{n} w_i \lg\, w_i$$

Note: lg denotes log_2 and $w_i = \frac{f_i}{\sum_{i=1}^{n} f_i}$.

2 Algorithm

```
ALGORITHM: Huffman(A,Freq,hc)
   Input: Freq[1:n] - an array of non-negative frequencies:
          Freq[i] = fi
   Output: hc[1:n] - an array of binary strings for Huffman
           code. hc[i] is the binary string encoding symbol
           ai, i=1,...,n.
   for i <- 1 to n do
      AllocateNode(P)
      P.symbolIndex <- i
      P.frequency <- Freq[i]
      P.left <- NULL
      P.right <- NULL
      leaf[i] <- P
   CreatePQueue(Leaf,Q)
   for i <-1 to n-1 do
      RemovePQueue(Q,L)
      RemovePQueue(Q,R)
      AllocateNode(root)
      root.left <- L
      root.right <- R
      root.frequency <- L.frequency + R.frequency
      InsertPQueue(Q,Root)
   root.BinaryString <- " "
   GenerateCode(root,hc)
```

Proposition 1. Given a set of frequencies $f_i, i = 1, \ldots, n$, the Huffman tree T constructed by *Huffman* has minimal weighted Π_{wl} over all 2-trees whose edges are weighted by these frequencies.

Suppose a text to be encoded using Huffman encoding has frequencies of its characters (symbols) as shown in Table 1: a$\rightarrow$9, b$\rightarrow$8, c$\rightarrow$5, d$\rightarrow$3, e$\rightarrow$15 and f $\rightarrow$ 2. We now trace the action of the Huffman encoding algorithm, showing the initial forest (contents of the min-priority-queue) and its contents during each stage of the algorithm.

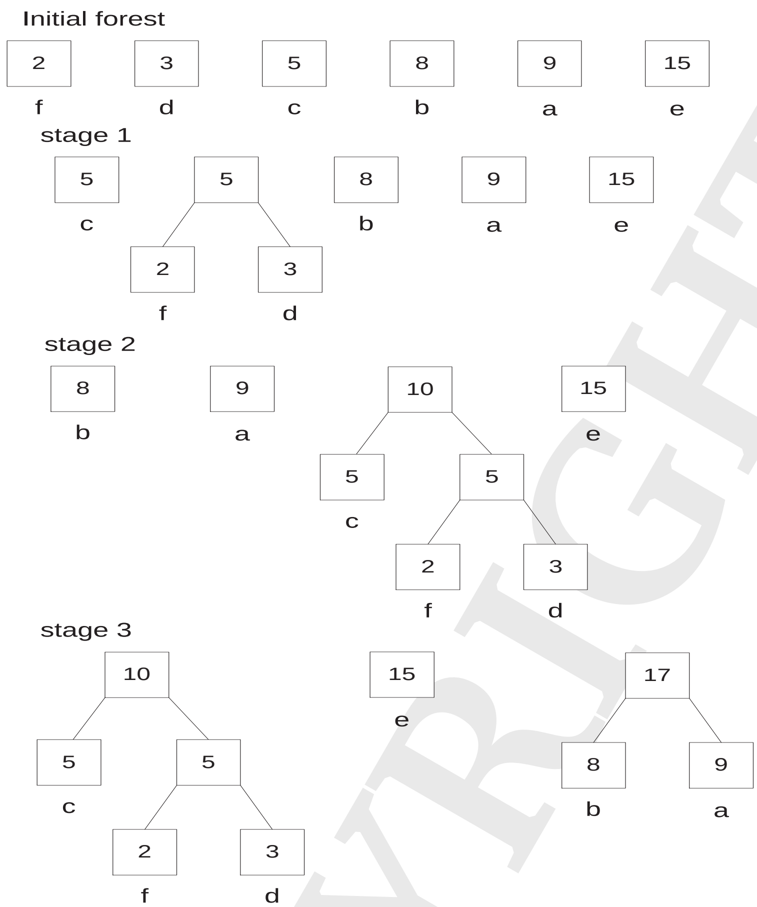

Figure 1: Action of Huffman for the frequencies shown in Table 1

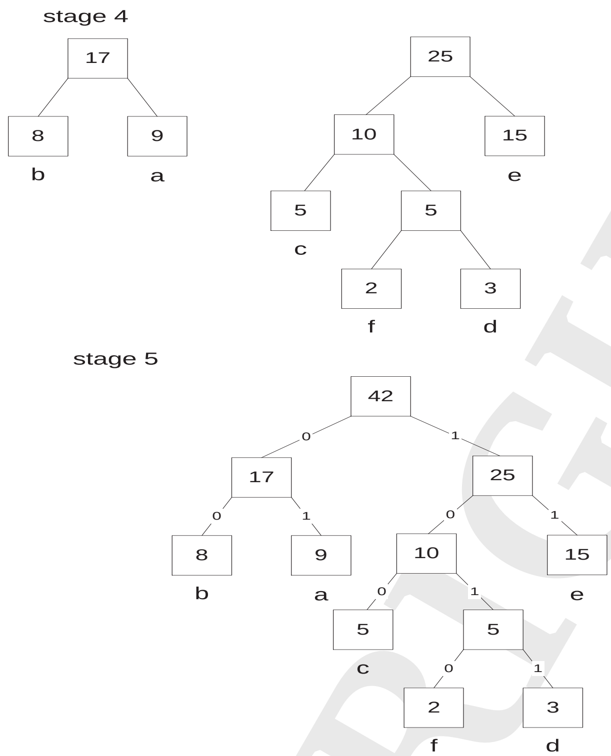

Figure 2: Action of Huffman for the frequencies shown in Table 1 continues

Symbol	Frequency	Codeword
a	9	01
b	8	00
c	5	100
d	3	1011
e	15	11
f	2	1010

Table 1: Frequency table of symbols

$$\Pi_{wl}(T) = \sum_{i=1}^{n} \lambda_i f_i = (2)(9) + (2)(8) + (3)(5) + (4)(3) + (2)(15) + (4)(2) = 99$$

Problem 1.

1. What is the weighted leaf path length of the tree?
2. How many bits are required to encode the text with the frequency table given in Table 1?
3. What is the average code length?
4. Using the codes in Table 1, give the encoding for the phrase *bed face*, leaving a white space between the code for each symbol.
5. Using the codes in Table 1, calculate the entropy of the text. How does the entropy compare to the average code length?