CISC235
Winter 2007
Homework for week 6
in preparation for quiz 3
Solutions

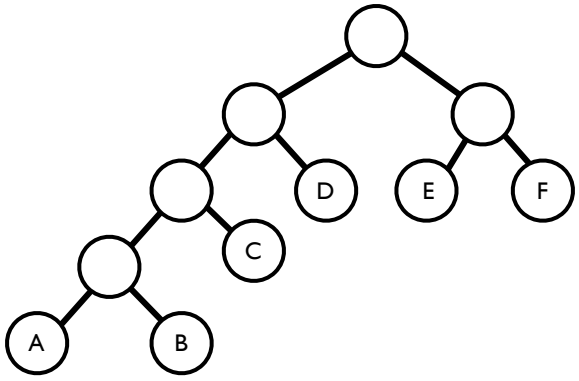

Figure 1: A Huffman tree.

1 Suppose you are given a text consisting of the characters A,B,C,D,E,F with probabilities A= 0.05, B = 0.05, C = 0.10, D = 0.25, E = 0.30, and F = 0.25. Draw a Hufman tree corresponding to these characters and specify the code obtained for each character. Here is one possibility. Note that the Huffman tree is not unique.

The code words obtained from the tree in Figure 1 are:

A - 0000; B - 0001; C - 001; D - 01; E - 10; F - 11;

2 One possible solution is shown in Figure 2. Note that the heap obtained from these values is not unique.

11.6 - 1 The average length $L_{ave} = \sum P(m_i) \log_2(1/P(m_i))$. A maximal code occurs when all symbols have the same probability. A minimal code occurs when one symbol has probability almost 1, and the rest share the miniscule remaining probability.

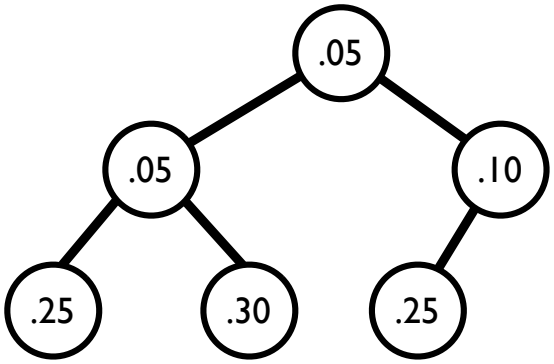

Figure 2: A heap.

11.6 - 2 I get $L_{ave} = .05\log_2(1/.05)*2+.9\log_2(1/.9) = 0.5690$, and $L_{Huf} = 4(.05)+.9 = 1.1$ Thus diff(L_{Huf}, L_{ave}) = (1.1 - 0.5690)/1.1 $\approx$ .482 or 48%.

The so called problem arises from the fact that we can't allocate fractional bits to a code. The small number of symbols exacerbates the problem. One way to remedy the situation is to follow the books lead and use pairs of symbols. In the text the probability of an XX pair is estimated by $(.05)^2$. (Note: An accurate probability of an XX pair can be determined by doing a pair of symbols count on the input file.) The result of looking at all 9 distinct character pairs gives us the probabilities

XX = 0.0025; XY = 0.0025; YX = 0.0025; YY = 0.0025; XZ = 0.045; YZ = 0.045; ZX = 0.045; ZY = 0.045; ZZ = 0.81.

Using these probabilities we get $L_{ave} = 1.138$. The Huffman code obtained using pairs of symbols is:

XX = 0000; XY = 0001; YX = 0010; YY = 0011; XZ = 0100; YZ = 0101; ZX = 0110; ZY = 0111; ZZ =1.

So $L_{Huf} = 1.57$

So now diff(L_{Huf}, L_{ave}) = (1.57 - 1.138)/1.57 $\approx$ .275 or 27%.

11.6 - 4 The codes for the least frequent symbol will be identical except for their rightmost bits, so the lengths of their codes are the same.

6.12 - 24 I will describe both algorithms using moveUp and moveDown.

```
William's
for(int last = 1; last < n; last ++)
      moveUp(data,0,last)
```

In the best case we have already have a heap so no swaps are needed. There will be one comparison per call to moveUp, for a total of n-1 comparisons.

```
Floyd's
for(int first = (n-2)/2 ; first >= 0, first--)
      moveDown(data,first, n-1)
```

In the best case no swaps are needed and (n-2)/2 (rounded down) calls to moveDown. Each iteration of moveDown compares children to find the largest and then compares to the parent. Thus there are two comparisons are used for each call so we have a total of about n-2 comparisons.

Nevertheless both algorithms are O(n) in this best case.