Algorithms Illuminated Ch. 14

Andrew Doss

14.1. Find average encoding length for optimal prefix free code

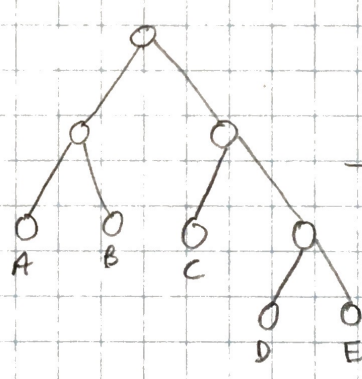
A	B	C	D	E
0.32	0.25	0.2	0.18	0.05

A	B	C	DE
0.32	0.25	0.2	0.23

A	B	CDE
0.32	0.25	0.43

AB	CDE
0.57	0.43

ABCDE
1.0



→ $2(0.32+0.25+0.2) + 3(0.18+0.05)$

$= 2(0.77) + 3(0.23)$

$= 1.54 + 0.69$

$= 2.23$ (a)

14.2

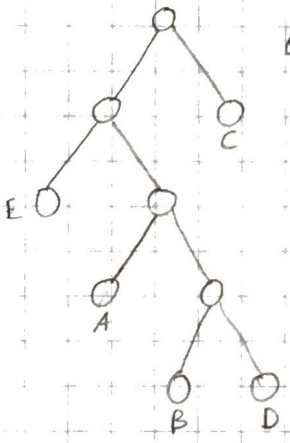
A	B	C	D	E
0.16	0.08	0.35	0.07	0.34

A	B D	C	E
0.16	0.15	0.35	0.34

A B D	C	E
0.31	0.35	0.34

A B D E	C
0.65	0.35

A B D E C
1.0



1(0.35) + 2(0.34) + 3(0.16) + 4(0.15)

0.35 + 0.68 + 0.48 + 0.60

1.03 1.08

2.11 (a)

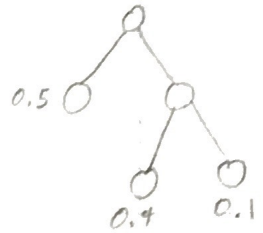
14.3

Determining the maximum encoding length for the symbol in an alphabet is equivalent to determining the maximum tree depth that is possible for a given alphabet size that has been optimally encoded with Huffman's algorithm. Using Huffman's algorithm, there are no leaves at the root level, at least one leaf at each intermediate level, and at least two leaves at the lowest level (assuming the alphabet has at least two symbols). This means that $1 + l + 2 \leq n$, where l represents the number of intermediate levels. Rearranging, this gives $l \leq n-2$, so l is at most $n-2$. The max encoding length is equal to 1 plus the number of intermediate levels, so the maximum encoding length for an alphabet of size n is $n-1$.

(C)

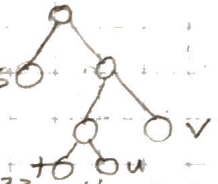
14.9

a) False. See counter-example.

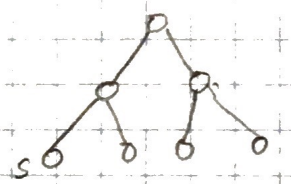


b) True. If all symbols sum to a frequency of 1.0, the sum of all other symbols is 0.5, so there is no subset of the other symbols that could be merged with 0.5 and remain < the sum of all other symbols, therefore the 0.5 symbol cannot merge before the final merge.

c) (True) The only way for a symbol to have an encoding length <2 is for the symbol to merge on the final iteration as shown below. If all symbols have



frequency < 0.33, there are at least 4 symbols so the top 3 levels of the tree must look like the tree drawn above. Alternatively, in a tree where all symbols have encoding lengths of at least 2, the last few levels are as shown below:



The later structure happens when the frequency of s is less than the frequency of the symbols t and u. V has frequency at most 0.33, so tu combined as frequency at least 0.34. Therefore, the former structure is impossible with $s < 0.33$.

14.5

First, sort the input array. Will call this A_I. Then, initialize an empty array called A_M to hold merge results. Initialize pointers at the start of each array. At each iteration, compare the values at each pointer (if the merge array is non-empty) and store the smaller as min_1 while incrementing the corresponding pointer. Repeat to get min_2. Sum min_1 and min_2 and add the result to the end of the merge array. Iterate until A_I is exhausted and the A_M pointer reaches the end of A_M.

The sorting procedure is $O(n \log n)$ while each iteration does constant work, making the remaining work $O(n)$.