CS 430 Lecture 19 Activities

Fractional Knapsack Problem

1. Prove that the Fractional Knapsack Problem has optimal substructure.
2. Try various "common sense" greedy approaches that divide the problem into a sub-problem(s) and try to come up with counter-examples or prove the greedy choice is correct using the "cut and paste" proof.

Huffman Codes Problem

Data Encoding Background

- Data is a sequence of characters
- Fixed Length – Each character is represented by a unique binary string Easy to encode, concatenate the codes together. Easy to decode, break off 3-bit codewords and decode each one.
- Variable Length – Give frequent characters shorter codewords, infrequent characters get long codewords. However, how do we decode if the length of the codewords are variable?
- Prefix Codes – No codeword is a prefix of another codeword Easy to encode, concatenate the code together. Easy to decode, since no codeword is a prefix of another, strip off one bit at a time and match to the unique prefix code

Huffman Codes

- Are a Data Compression technique using a greedy algorithm to construct an optimal variable length prefix code
- Use frequency of occurrence of characters to build an optimal way to represent each character as a binary string
- Use a Binary tree method – 0 means go to left child, 1 means go to right child (not a binary search tree).
- Cost of Tree in bits:

$$B(T) = \sum_{\text{for all } c \in C} \text{FREQ}(c) \times \text{DEPTH}(c)$$

Example

Table 19.1: A character-coding problem. A data file of 100,000 characters containes only the characters a–f, with the frequencies indicated. If each character is assigned a 3-bit codeword, the file can be encoded in 300,000 bits. Using the variable-length code shown, the file can be encoded in 224,000 bits.

	a	b	c	d	e	f
Frequency (in thousands)	45	13	12	16	9	5
Fixed-length codeword	000	001	010	011	100	101
Variable-length codeword	0	101	100	111	1101	1100

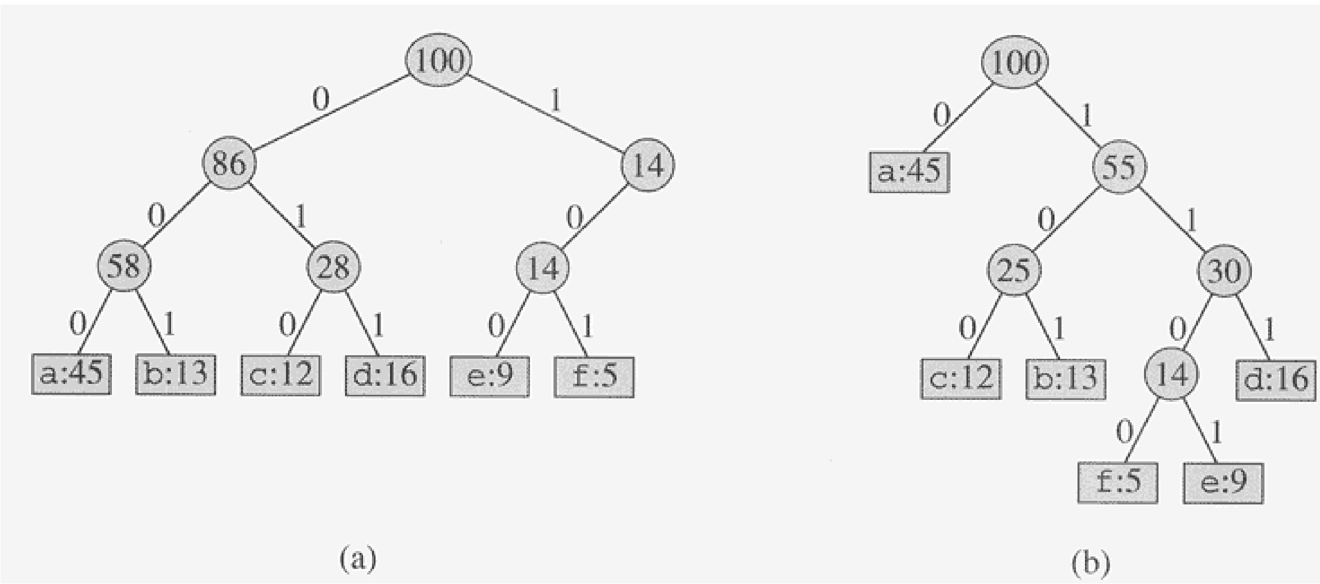
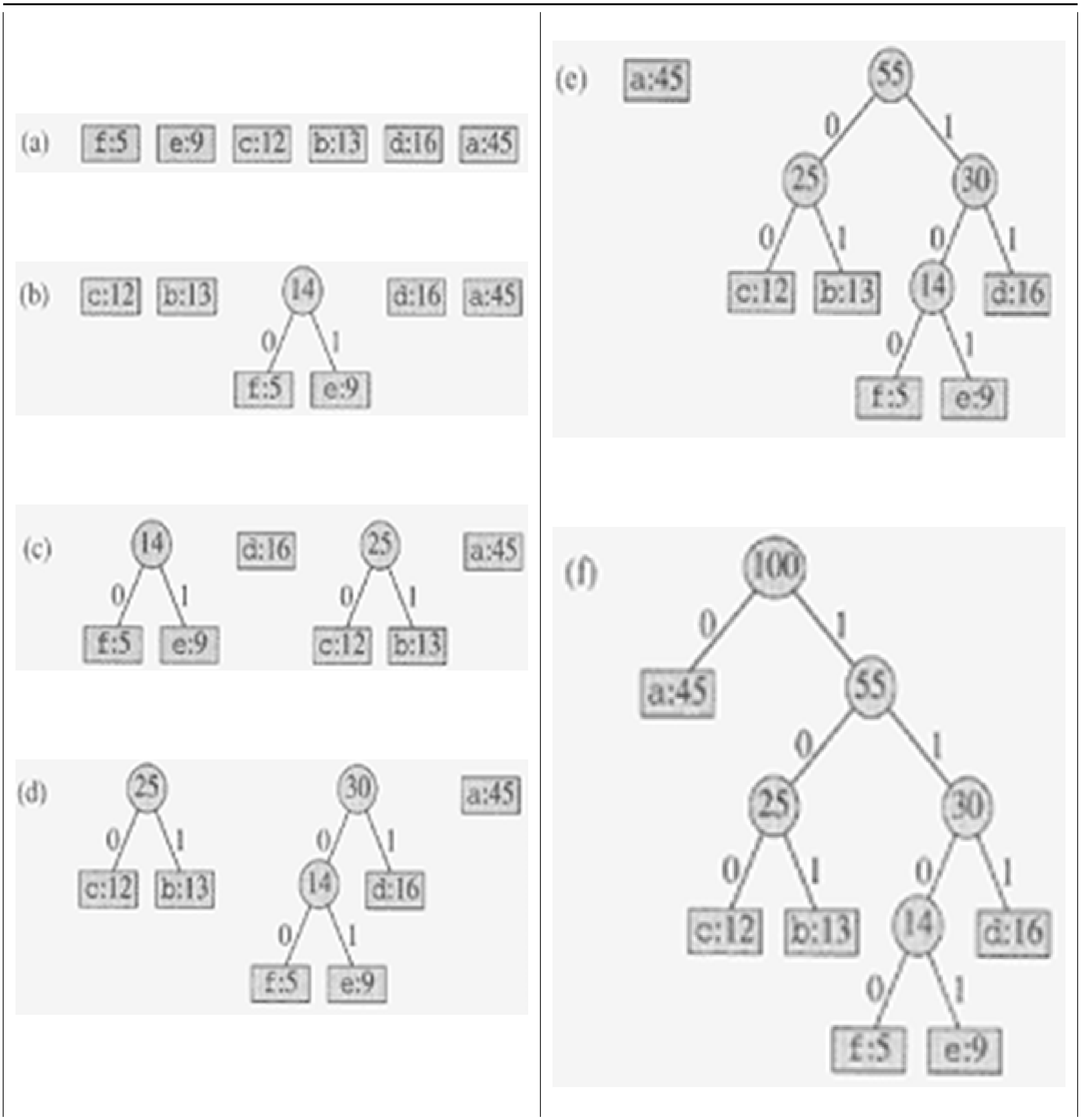


Figure 19.1: Trees corresponding to the coding schemes in Table 19.1. Each leaf is labeled with a character and its frequency of occurence. Each internal node is labeled with the sum of the frequencies of the leaves in its subtree. **(a)** The tree corresponding to the fixed-length code $a = 000, \ldots, f = 101$. **(b)** The tree corresponding to the optimal prefix code $a = 0, b = 101, \ldots, f = 1100$.

3. Prove that the Huffman Codes Problem has optimal substructure.

The greedy approach is: Build the tree bottom up by using a minimum priority queue to merge 2 least frequent objects (objects are leaf nodes or other subtrees) together into a new subtree.



Huffman http://www.cs.auckland.ac.nz/software/AlgAnim/huffman.html

4. Proof this greedy algorithm leads to an optimal Huffman Code tree: Build the tree bottom up by using a minimum priority queue to merge 2 least frequent objects (objects are leaf nodes or other subtrees) together into a new subtree.