# Huffman Codes

Radu Trîmbiţaş

November 11, 2012

Huffman invented a greedy algorithm that constructs an optimal prefix code called a Huffman code. In the pseudocode that follows (Algorithm 1), we assume that $C$ is a set of n characters and that each character $c \in C$ is an object with an attribute $c.freq$ giving its frequency. The algorithm builds the tree T corresponding to the optimal code in a bottom-up manner. It begins with a set of $|C|$ leaves and performs a sequence of $|C| - 1$ merging operations to create the final tree. The algorithm uses a min-priority queue $Q$, keyed on the freq attribute, to identify the two least-frequent objects to merge together. When we merge two objects, the result is a new object whose frequency is the sum of the frequencies of the two objects that were merged.

**Algorithm 1** HUFFMAN(C)

```
1: n := |C|;
2: Q := C;
3: for i := 1 to n − 1 do
4:    allocate a new node z
5:    z.left := x := EXTRACT-MIN(Q);
6:    z.right := y := EXTRACT-MIN(Q);
7:    z.freq := x.freq + y.freq;
8:    INSERT(Q, z);
9: end for
10: return EXTRACT-MIN(Q); {return the root of the tree}
```

| | 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 |

Table 1: A character-coding problem. A data file of 100,000 characters contains only the characters af, with the frequencies indicated. If we assign each character a 3-bit codeword, we can encode the file in 300,000 bits. Using the variable-length code shown, we can encode the file in only 224,000 bits.

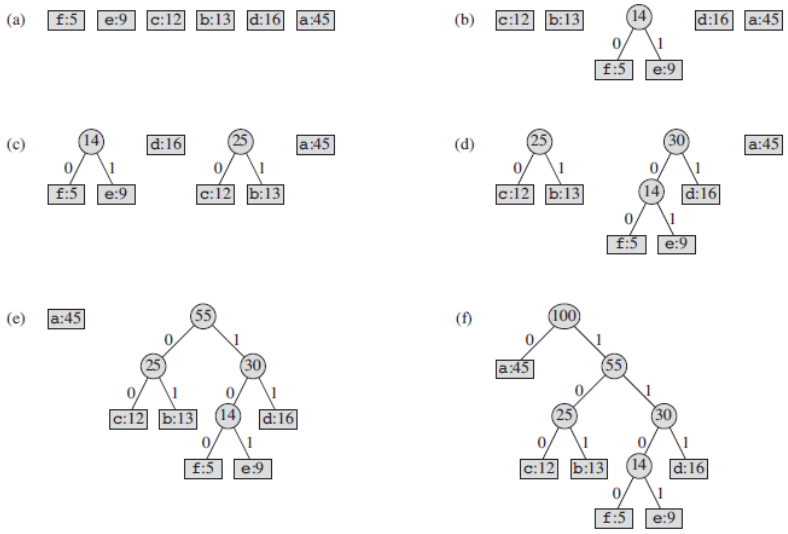


Figure 1: The steps of Huffmans algorithm for the frequencies given in Table 1. Each part shows the contents of the queue sorted into increasing order by frequency. At each step, the two trees with lowest frequencies are merged. Leaves are shown as rectangles containing a character and its frequency. Internal nodes are shown as circles containing the sum of the frequencies of their children. An edge connecting an internal node with its children is labeled 0 if it is an edge to a left child and 1 if it is an edge to a right child. The codeword for a letter is the sequence of labels on the edges connecting the root to the leaf for that letter. (a) The initial set of $n = 6$ nodes, one for each letter. (b)(e) Intermediate stages. (f) The final tree.

For our example, Huffmans algorithm proceeds as shown in Figure 1. Since the alphabet contains 6 letters, the initial queue size is $n = 6$, and 5 merge steps build the tree. The final tree represents the optimal prefix code. The codeword for a letter is the sequence of edge labels on the simple path from the root to the letter.

Line 2 initializes the min-priority queue $Q$ with the characters in $C$. The for loop in lines 38 repeatedly extracts the two nodes $x$ and $y$ of lowest frequency from the queue, replacing them in the queue with a new node $z$ representing their merger. The frequency of $z$ is computed as the sum of the frequencies of $x$ and $y$ in line 7. The node $z$ has $x$ as its left child and $y$ as its right child. (This order is arbitrary; switching the left and right child of any node yields a different code of the same cost.) After $n - 1$ mergers, line 9 returns the one node left in the queue, which is the root of the code tree.

To analyze the running time of Huffmans algorithm, we assume that $Q$ is implemented as a binary min-heap (see [1, Chapter 6]). For a set $C$ of $n$ characters, we can initialize $Q$ in line 2 in $O(n)$ time using the BUILD-MIN-

HEAP procedure discussed in [1, Section 6.3]. The for loop in lines 3–8 executes exactly $n - 1$ times, and since each heap operation requires time $O(\log n)$, the loop contributes $O(n \log n)$ to the running time. Thus, the total running time of HUFFMAN on a set of $n$ characters is $O(n \log n)$. We can reduce the running time to $O(n \log \log n)$ by replacing the binary min-heap with a van Emde Boas tree (see [1, Chapter 20]).

# References

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, *Introduction to Algorithms*, Third Edition, MIT Press, 2009