A Derivation of Huffman's Algorithm

Rob R. Hoogerwoord

Eindhoven University of Technology, department of Mathematics and Computing Science,
PO Box 513, 5600 MB Eindhoven, The Netherlands

Abstract. We present a semi-formal derivation of Huffman's well-known algorithm for the construction of an *optimal encoding tree.*

0 Specification

In this paper we study *(binary) trees*, the leaves of which are labelled with *real numbers.* In the application area of source coding, these numbers constitute a probability distribution, but that is irrelevant here. Throughout this paper we use the following type conventions for our variables:

b, c : real number
B, B' : bag of real number
s, s', t, t' : tree
k, l : natural number

Trees are defined recursively, as follows:

$\langle b \rangle$ is a (singleton) tree , and
$\langle s, t \rangle$ is a (composite) tree .

Trees composed of 2 singletons we call *pairs*; for pairs we use the abbreviation $\langle b, c \rangle$, so we have:

$$\langle b, c \rangle \;=\; \langle \langle b \rangle \,,\, \langle c \rangle \rangle$$

With each tree s is associated a (real) number $w{\cdot}s$; informally, function w is defined as follows, where dummy x identifies the leaves of the tree:

$$w{\cdot}s \;=\; (\Sigma x : \text{“}x \text{ is a leaf of } s\text{”} : d{\cdot}x * v{\cdot}x) \;\;,$$

where functions d and v are defined informally by:

$d{\cdot}x$ = “the distance of x to the root of s”
$v{\cdot}x$ = “the number associated with x”

Each tree s also represents a bag $bag{\cdot}s$, defined by —where $\{b\}$ denotes the singleton bag containing b and where $+$ denotes bag summation—:

$$\begin{array}{lcl} bag{\cdot}\langle b \rangle & = & \{b\} \\ bag{\cdot}\langle s, t \rangle & = & bag{\cdot}s + bag{\cdot}t \end{array}$$

The remainder of this paper is devoted to a derivation of an algorithm for the computation of a solution to the following

problem 0: for given finite and nonempty bag B of real numbers, solve

$$s \;:\; bag{\cdot}s = B \;\wedge\; (\forall t : bag{\cdot}t = B : w{\cdot}s \leq w{\cdot}t)$$

□

We say that "s is optimal for B" for any solution s to this problem.

1 Solution

The key to the design of the algorithm is provided by the following observations, which lead to a recursive algorithm, based on induction on the number of elements of the bag.

For a singleton bag $\{b\}$, exactly one tree s exists for which $bag{\cdot}s = \{b\}$, namely $\langle b \rangle$; because this tree is unique, it is optimal.

For a composite —that is: non-singleton— bag all trees representing it are composite as well. For composite tree s containing a pair $\langle b, c \rangle$ as a subtree we define s' to be a tree obtained from s by replacement of one occurrence of $\langle b, c \rangle$ by the singleton tree $\langle b{+}c \rangle$. We call the act of constructing s' from s in this way *pruning*, whereas we call the reverse operation *grafting*. Trees s and s' thus related have the following properties:

$$\begin{aligned} bag{\cdot}s' &= bag{\cdot}s - \{b, c\} + \{b{+}c\} \\ w{\cdot}s' &= w{\cdot}s - b - c \;, \end{aligned}$$

where the latter equality follows from the following little calculation, in which $l{+}1$ denotes the distance to the root (in s) of b and c; the first formula in this calculation represents the contribution of $\langle b, c \rangle$ to $w{\cdot}s$, whereas $l{*}(b{+}c)$ is the contribution of $\langle b{+}c \rangle$ to $w{\cdot}s'$:

$$\begin{aligned} & (l{+}1){*}b + (l{+}1){*}c \\ = & \quad \{ \text{ algebra } \} \\ & l{*}(b{+}c) + b + c \;. \end{aligned}$$

Pruning a composite tree is always possible, as is shown by the following lemma, which can be proved by straightforward structural induction.

lemma 1: every composite tree contains a pair as subtree

□

Now we show how an optimal tree for a composite bag B can be constructed, by means of grafting, from an optimal tree for a smaller bag B'; to specify the numbers b, c to which the grafting operation pertains, we need auxiliary variables s and s'. We assume that:

s is optimal for bag B and s contains $\langle b, c \rangle$,
$B' = B - \{b, c\} + \{b{+}c\}$,
s' is obtained from s by pruning and $bag{\cdot}s' = B'$,
t' is optimal for B', so $bag{\cdot}t' = B'$,
t is obtained from t' by grafting and $bag{\cdot}t = B$;

and we calculate as follows:

$$
\begin{array}{ll}
 & w \cdot t \\
= & \quad \{\ t \text{ is obtained from } t' \text{ by grafting } \} \\
 & w \cdot t' + b + c \\
\leq & \quad \{\ t' \text{ is optimal and } bag \cdot t' = bag \cdot s' \ \} \\
 & w \cdot s' + b + c \\
= & \quad \{\ s' \text{ is obtained from } s \text{ by pruning } \} \\
 & w \cdot s \\
\leq & \quad \{\ s \text{ is optimal and } bag \cdot s = bag \cdot t \ \} \\
 & w \cdot t \ .
\end{array}
$$

Hence, $w \cdot t = w \cdot s$ which implies that t is optimal as well.

So, an optimal tree for composite bag B can be constructed as follows. Select in B elements b, c and obtain the (smaller) bag B' by replacing b, c by $b+c$. Construct an optimal tree for B'; grafting this tree then yields an optimal tree for B. The only difficulty here is that b and c may not be chosen arbitrarily: we have assumed that an optimal tree for B exists in which pair $\langle b, c \rangle$ occurs as subtree. To show that this assumption is realistic we need a constructive solution to our remaining

problem 2: for composite bag B, solve
$b, c : (\exists s :: \text{“}s \text{ is optimal for } B\text{”} \ \wedge \ \text{“}s \text{ contains } \langle b, c \rangle\text{”})$

□

To solve this problem we need the following stronger version of lemma 1; again, it can be proved by structural induction.

lemma 3: every composite tree contains a pair as subtree, whose two leaves have *maximal* (and equal) distances to the root.

□

The relevance of this lemma follows from the observation that a tree might contain only one pair; in that case the leaves of this one and only pair are at maximal distance from the root. A corollary of this lemma is that each composite tree has at least two leaves at maximal distance.

We now investigate what we can conclude about the numbers in leaves at maximal distance. We have:

$$
\begin{array}{ll}
 & b*k + c*l \ < \ c*k + b*l \\
\equiv & \quad \{\ \text{algebra } \} \\
 & 0 \ < \ (c-b)*(k-l) \\
\Leftarrow & \quad \{\ \text{algebra } \}
\end{array}
$$

$$b<c \wedge l<k \ .$$

From this derivation it follows that a tree containing a larger number at a larger distance and a smaller number at a smaller distance is not optimal. Hence, in every optimal tree the smallest two numbers have maximal distances to the root. By lemma 3, the maximal distance occurs at least twice. Hence, in every optimal tree the smallest two numbers have *maximal and equal* distances to the root. Because swapping numbers at equal distances does not change the w-value of a tree and because, by lemma 3, every tree contains a pair at maximal distance, we conclude that an optimal tree exists in which the smallest two numbers occur in a pair.

So, a solution to problem 2 is the pair of the smallest two numbers in B. This concludes our derivation of the algorithm.

acknowledgement

To the Eindhoven Tuesday Afternoon Club, for some useful comments on the presentation.

(Eindhoven, 18 november 1992)