TPPmark2012

October 18, 2012

Recall the Huffman tree construction below (from [1]), an algorithm for computing a binary code tree that is optimal among so-called “prefix condition” trees for a given non-empty set of finite source letters with known probabilities:

1. Let L be a list of the probabilities of the source letters corresponding to the leaves of the tree.
2. Take two smallest probabilities in L, make the corresponding nodes siblings, generate an intermediate node as their parent, and label the link from parent to one child with zero and the other with one.
3. Replace the above two probabilities in L with their sum, associated with the new intermediate node. If the new L contains one element, stop, and otherwise return to step 2.

Problem: Suppose we only consider the source letters having positive (i.e., non-zero) probabilities. And we define “sibling property” (also from [1]) as follows:

> **Definition:** A binary code tree has the *sibling property* if each node (except the root) has a sibling and if the nodes can be listed in order of nonincreasing probability with each node being adjacent in the list to its sibling.

Then prove that:

1. Every Huffman tree has the sibling property.
2. Every binary code tree having the sibling property is a Huffman tree.

Hints:

- We can formulate a binary code tree using positive integer values instead of real probability values as follows:
  - Each leaf has a positive integer assigned to it. The integer value represents the number of occurrences of a source letter. You may also need to assign some label to each leaf, especially if you don’t use paths to distinguish leaves having the same occurrence number.

  - Each internal node has exactly two children and also has a positive integer associated with it. The integer value should be equal to the sum of the positive integers assigned to its descendant leaves.

- Informal proofs are given in [1] (as the proof of Theorem 1).
- There are some existing formalization works about the optimality of Huffman trees [2, 3]. The predicate `build` in [2] corresponds to "being a Huffman tree" (Strictly speaking, there is a restriction that the integer associated with the left child of a parent node cannot be greater than the one associated with the right child of the parent. If you adopt this definition, then you'll need to add a similar restriction to the definition of "sibling property" in order to show 2).

References

[1] Robert G. Gallager, Variations on a Theme by Huffman, IEEE Transactions on Information Theory, IT-24(6), pp. 668–674, 1978.

[2] Laurent Théry, Formalising Huffman's Algorithm, Tech. report TRCS 034, Dept. of Informatics, Univ. of L'Aquila, 2004. Coq proof is available from http://coq.inria.fr/pylons/pylons/contribs/view/Huffman/v8.4 (the "Download" link at the right column.)

[3] Jasmin Christian Blanchette, Proof Pearl: Mechanizing the Textbook Proof of Huffman's Algorithm, Journal of Automated Reasoning, Volume 43 Issue 1, pp. 1–18, 2009. Isabelle/HOL proof is available from http://afp.sourceforge.net/browser_info/current/HOL/Huffman/Huffman.html