DIMACS Technical Report 99-15
February 1999

Addition Chains as Test Trees and a Sequential Variant as the Huffman Problem

by

Julia Abrahams[1]
Center for Discrete Mathematics and Theoretical Computer Science (DIMACS)
Rutgers, The State University of New Jersey
Piscataway, NJ 08854-8018
abrahams@dimacs.rutgers.edu

[1]Permanent Member

DIMACS is a partnership of Rutgers University, Princeton University, AT&T Labs-Research, Bell Labs, Bellcore and NEC Research Institute.

DIMACS is an NSF Science and Technology Center, funded under contract STC–91–19999; and also receives support from the New Jersey Commission on Science and Technology.

ABSTRACT

An addition chain can be interpreted as a test tree to identify a single distinguished item from a set of candidates. In this diagnostic problem, there is a constant cost associated with setting up a test, distinct tests distinguish between different size subsets of items, and all tests are set up in advance. If the tests are set up sequentially as needed and if a probability is assigned to each item of its being the single item sought by the testing procedure, the Huffman algorithm finds the optimal test tree in the sense of minimizing the average cost of setting up the tests used in the testing procedure.

1. Addition Chains

The addition chain problem is reviewed extensively by Knuth [13], and many other papers have been written on this problem and its variants, some of which include [4-10, 15-18]. An addition chain for n is a sequence of integers

$$1 = a_0 < a_1 < a_2 < \ldots < a_r = n$$

with the property that

$$a_i = a_j + a_k, \qquad \text{for some} \quad k \leq j < i,$$

for all $i = 1, 2, \ldots, r$. An optimal addition chain is one for which r is minimum. Optimal addition chains are known for some but not all n, although many general methods are known for finding good but not necessarily optimal addition chains. Good addition chains provide efficient methods for the computation of X^n for arbitrary X.

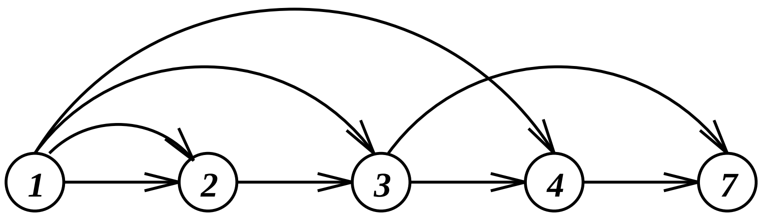
For example, two different addition chains for $n = 7$ are:

$$1 < 2 < 3 < 4 < 7, \tag{1}$$

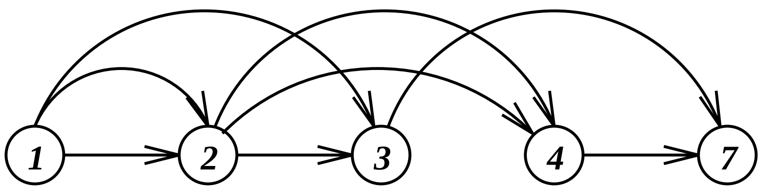
for which $r = 4$, and

$$1 < 2 < 4 < 5 < 6 < 7, \tag{2}$$

for which $r = 5$. If $a_i = a_j + a_k$ for more than one pair of arguments (a_k, a_j), for example $4 = 1 + 3 = 2 + 2$ in (1), it can be assumed that a definite pair (a_k, a_j) has been selected without loss of generality. A directed graph representation for the addition chain exhibits (a_k, a_j) explicitly when significant, so that (1) can be represented as either



in which $4 = 1 + 3$ is used or



in which $4 = 2 + 2$ is used.

2. Test Trees

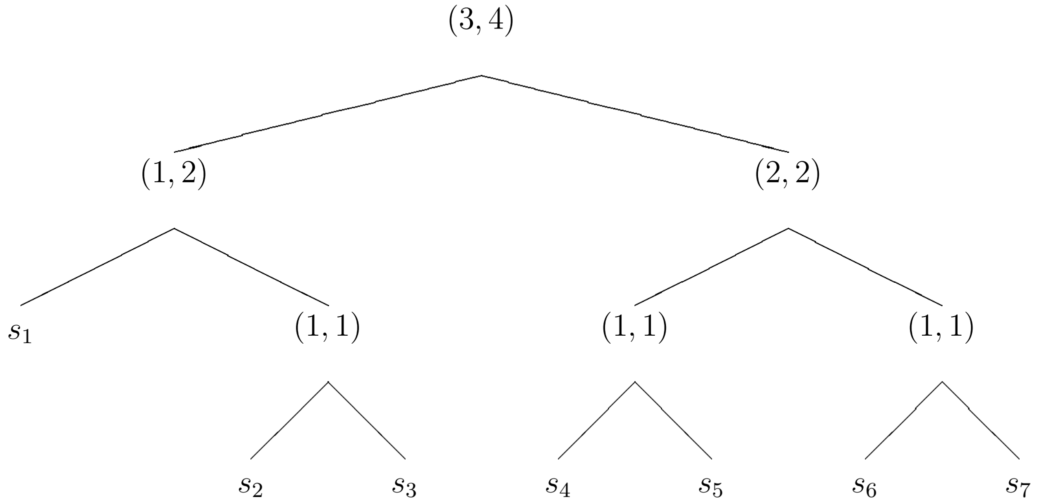
The test tree problem, also called the search or decision tree problem, has an extensive literature [2-3, 14], including many variants of the specific version of interest here. The goal of the test problem is to identify one of a set of n items, $\{s_m\}, m = 1, 2, \ldots, n$, as the single distinguished item, say s^*, by performing a sequence of binary tests on successively smaller subsets of $\{s_m\}$, each made up of the union of two disjoint sets, where the result of each test reports which of the two disjoint sets contains s^*.

The sequence of tests can be described by a binary tree, each of whose interior nodes is labeled by a test and each of whose leaf nodes is labeled by one of the s_m. The result of a test indicates whether s^* labels a leaf node in the left subtree of the test node or the right subtree. Starting with the test labeling the root, the path from the root to the leaf node labeled s_m describes the sequence of tests performed and their results to identify $s^* = s_m$.

Denote a test by a pair of sets, the labels of the leaf nodes in the left subtree of the test node and the labels of the leaf nodes in the right subtree. In the case that the items are interchangeable with respect to the test performed, it is sufficient to denote a test by the pair (a_k, a_j) where there are a_k leaf nodes in the left subtree of the test node and a_j in the right, where $a_k \leq a_j$ without loss of generality.

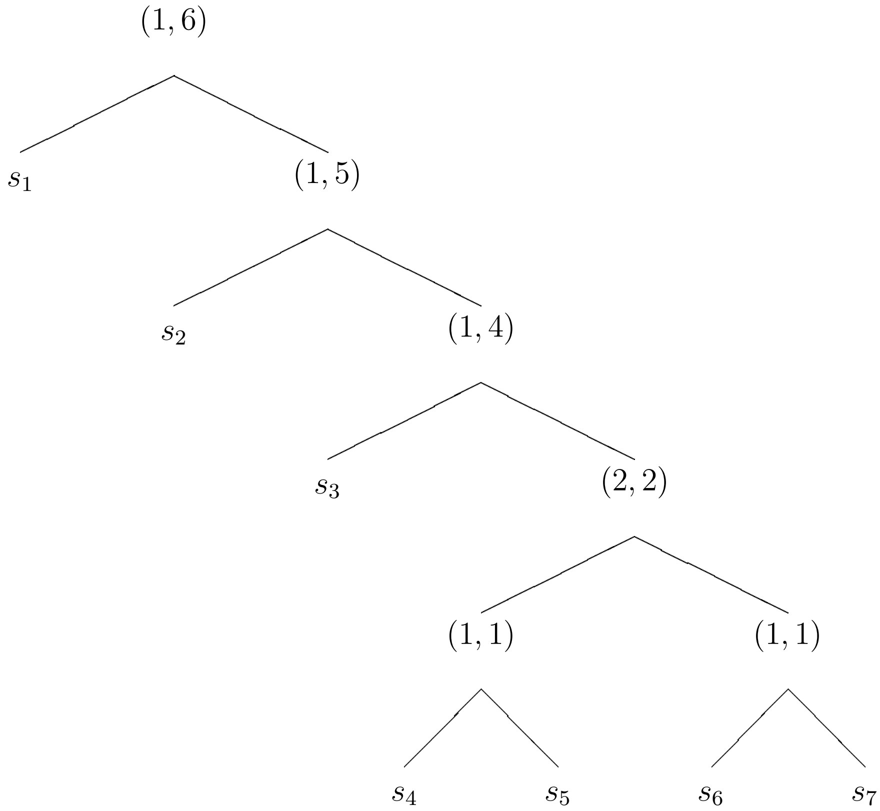
Consider now the particular version of this problem in which the items are interchangeable with respect to all the tests, the tests are all set up in advance, there is a constant cost associated with setting up a test, and a test must be set up for each (a_k, a_j) pair.

For example, two different test trees to identify one item from a set of $n = 7$ interchangeable items are:



(3)

for which a total of $r = 4$ different tests are set up, and

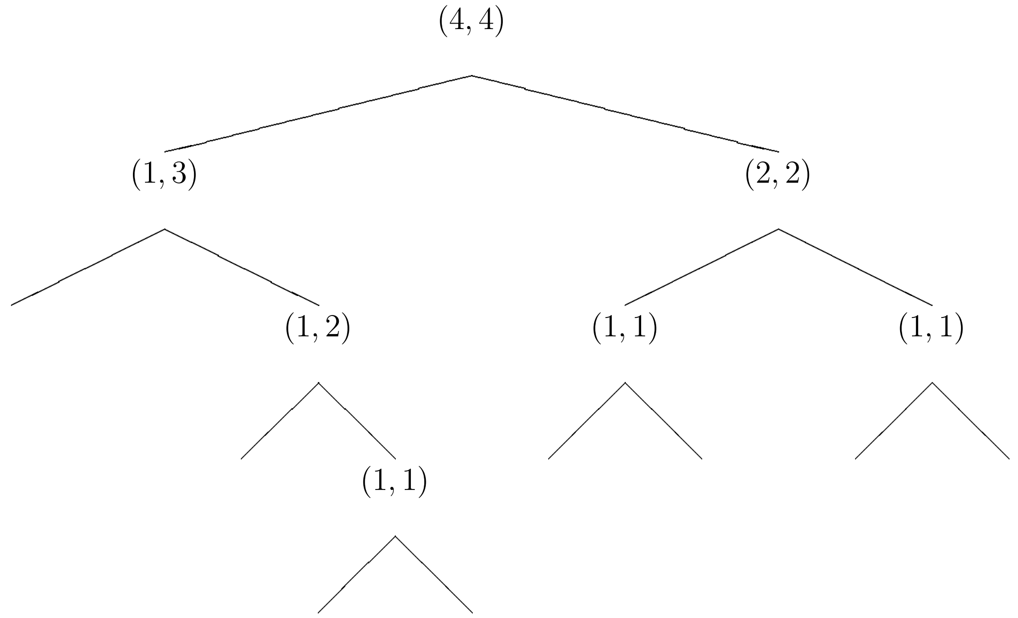


(4)

for which a total of $r = 5$ different tests are set up. The leaf nodes in each tree are labeled arbitrarily with the s_m. The interpretation of the test tree in (3), for example, is as follows. First, 4 units of some "laboratory apparatus" are set up, one to perform a test on two subsets of distinct items in $\{s_m\}$, one of size 3 and one of size 4, another to perform a test on two subsets of distinct items in $\{s_m\}$, one of size 1 and one of size 2, another to perform a test on two subsets of distinct items in $\{s_m\}$, one of size 2 and one of size 2, and another to perform a test on two subsets of distinct items in $\{s_m\}$, one of size 1 and one of size 1. Then, first the (3,4) test is performed on 7 items to determine if the single distinguished item is among the 3 distinct items placed in one "chamber" of the apparatus set up for the (3,4) test or among the 4 distinct items placed in the other chamber, that is whether $s^* \in \{s_1, s_2, s_3\}$ or $\{s_4, s_5, s_6, s_7\}$.

Depending on the result of the $(3, 4)$ test, the next test performed is either the $(1, 2)$, that is whether $s^* \in \{s_1\}$ or $\{s_2, s_3\}$, or the $(2, 2)$, that is whether $s^* \in \{s_4, s_5\}$ or $\{s_6, s_7\}$, test using the corresponding apparatus set up. The tests continue in this fashion until there are exactly two candidate items remaining. The final (1,1) test identifies the item labeling the selected leaf node of the test tree. Note that the (1,1) test is applied to distinguish between different pairs of items each time it is used depending on the previous test sequence, either testing $\{s_2\}$ versus $\{s_3\}$ or $\{s_4\}$ versus $\{s_5\}$ or $\{s_6\}$ versus $\{s_7\}$.

It is evident that this test tree problem for interchangeable items, with advance test set up and costs given by the number of different size tests set up, is equivalent to the addition chain problem. When the goal is to minimize the cost of the test procedure given by number of test set ups, the optimal test tree is given by the optimal addition chain. Note that if $a_i = a_j + a_k$ for more than one pair of arguments (a_k, a_j), it can be assumed that a definite pair (a_k, a_j) has been selected without loss of generality. That is, for example, the test tree (5) for $n = 8$ can never be an optimal tree:



(5)

since in an optimal tree both tests $(1, 3)$ and $(2, 2)$ would not appear. Note that the addition chain in (1) and the test tree in (3) both represent the same sequence of integers as do (2) and (4). For both the addition chain problem and the test tree problem, the optimal solution minimizes r. In fact, the minimum value of r for $n = 7$ is 4.

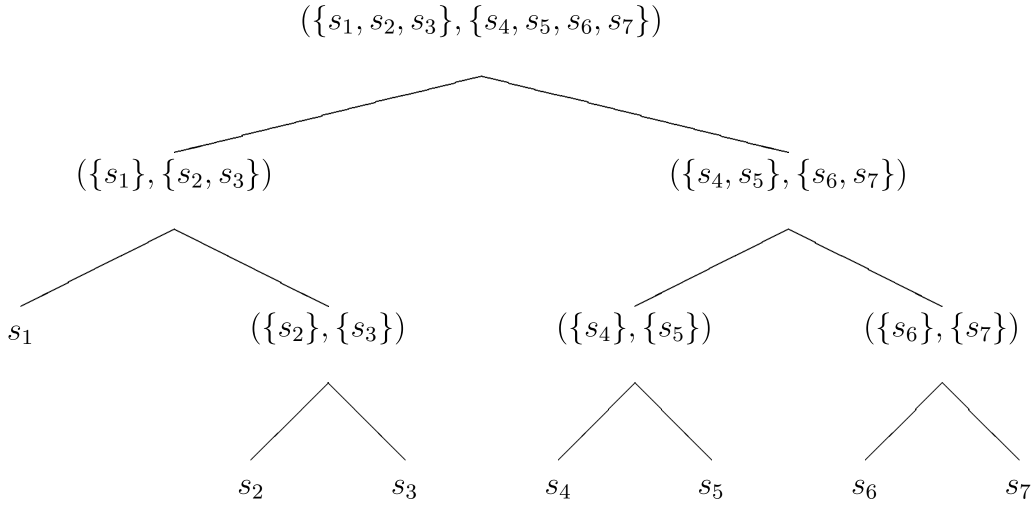
3. Sequentially Set-up Tests and Non-interchangeable Items

Suppose now that s_m, $m = 1, \ldots, m$, is the single distinguished item with probability p_m, that is, the items are no longer interchangeable with respect to the likelihood with which they will be selected by the sequence of tests. Without loss of generality assume:

$$p_1 \geq p_2 \geq \ldots \geq p_n. \tag{6}$$

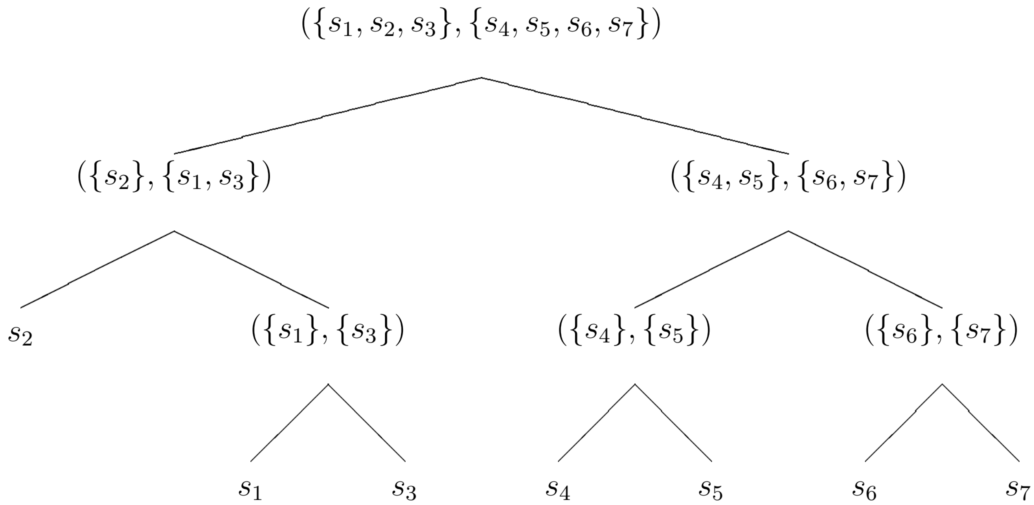
Suppose further that the units of apparatus are set up sequentially as needed so that the cost of identifying the single distinguished item is only the number of tests labeling the interior nodes on the path through the test tree to the leaf node labeled with the distinguished item. That is, if a test does not fall on the path from the root to the leaf node associated with s^*, the apparatus need not be set up for that test.

Now consider the problem finding the optimal diagnostic test tree, optimal in the sense of minimizing the average cost of the testing procedure or, that is, the average number of distinct tests set up when the tests are set up sequentially. For example, if $p_m \neq 1/n$ for all p_m, two different test trees to identify one item from a set of $n = 7$ items are:



(7)

and



(8)

The average cost of (8) is greater than or equal to the average cost of (7) because $2p_1 + 3p_2 \leq 2p_2 + 3p_1$ from (6).

Clearly, for a fixed unlabeled test tree, the average cost is less if s_1 labels the leaf node with the shortest path from the root, s_2 the next shortest, and so on. Denote the length of the path from the root to s_m by c_m. Then the average cost of a tree is

$$C = \sum_{m=1}^{n} p_m c_m. \tag{9}$$

since the number of tests set up to identify s^* as s_m is given by the length of the path from the root to s_m, a distinct test is set up for each node along that path, involving successively smaller subset sizes, and no other tests are set up.

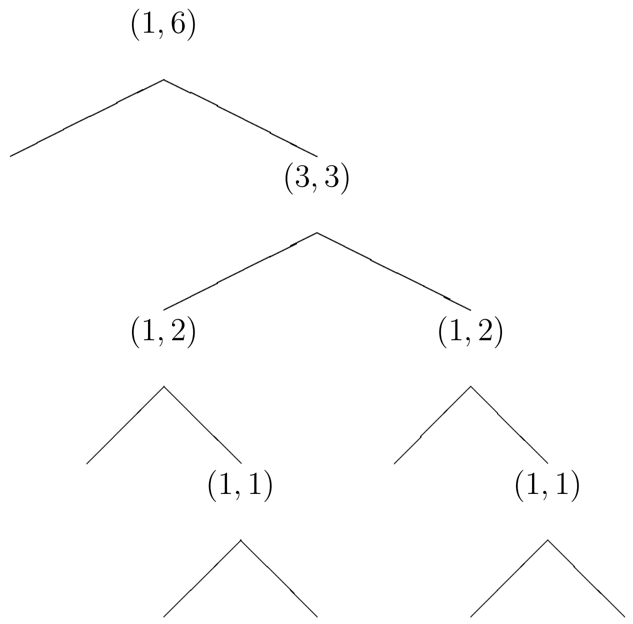
The optimal test tree, in the sense of minimizing the average cost of test set up, for the test procedure in which tests are set up sequentially and each different size test costs a constant set-up cost can be obtained by selecting path lengths c_m to minimize (9). However, this is exactly the binary Huffman coding problem [12], and see the bibliography in [1], to select path lengths through a tree to minimize average path length. Thus the Huffman algorithm finds the optimal test tree for constant test set-up costs and sequential test set up.

It is well known that the Huffman algorithm solves another optimal test tree problem as well. This is the problem of minimizing the average number of tests performed to identify a single distinguished item. Each test performed carries constant cost. Thus it is not surprising that in the sequential test set-up problem, where the only tests set up are the tests performed, the Huffman algorithm should provide the optimal test tree.

4. Concluding Remarks

What is surprising is the connection between two such well-studied problems, addition chains and Huffman codes. The Huffman problem is a non-uniform, sequentially set-up version of the addition chain problem. The addition chain problem is a particular binary diagnostic test tree problem with a constant test set-up cost for distinct tests.

This connection suggests other interesting problem variants. For example, Graham, Yao, and Yao [11] give an algorithm to find the optimal addition chain under a different model for the cost of a test set up. The cost of the test (a_k, a_j) is given by $a_k \times a_j$. For example, the cost of the optimal test tree for 7 items under this model,



is 18. An interesting open question is to give an algorithm to resolve the sequentially set-up version of this test problem even for uniform $\{p_m\}$. Similarly, consider a sequentially set-

up version of Zantema's [19] addition chain problem variant in which the cost of (a_k, a_j) is $a_k + a_j$.

References

1. J. Abrahams, *Code and parse trees for lossless source encoding*, in: Proceedings Compression and Complexity of Sequences 1997, B. Carpentieri, A. DeSantis, U. Vaccaro and J. A. Storer, eds. (IEEE Computer Society, Los Alamitos, CA, 1998) 145–171.

2. R. Ahlswede, I. Wegener, Search Problems, Wiley, NY, 1987.

3. M. Aigner, Combinatorial Search, Wiley, NY and Teubner, Stuttgart, 1988.

4. F. Bergeron, J. Berstel and S. Brlek, *Efficient computation of addition chains*, J. de Theorie des Nombres **6** (1994) 21–38.

5. J. Bos and M. Coster, *Addition chain heuristics*, in: Advances in Cryptology - CRYPTO '89 Proceedings, Lecture Notes in Computer Science, Vol. 435, G. Brassard, ed. (Springer Verlag, NY, 1990) 400–407.

6. Y. -J. Chen, C. -C. Chang and W. -P. Yang, *Some properties of vectorial addition chains*, Intern. J. Computer Math. **54** (1994) 185–196.

7. Y. -J. Chen, C. -C. Chang and W. -P. Yang, *Shortest weighted length addition chains*, J. of Information Science and Engineering, **11** (1995) 295–305.

8. V. S. Dimitrov, G. A. Jullien and W. C. Miller, *Algorithms for multi-exponentiation based on complex arithmetic*, in: Proceedings 13th IEEE Symposium on Computer Arithmetic, T. Lang, J. -M. Muller, N. Takagi, eds. (IEEE Computer Society, Los Alamitos, CA, 1997) 208–215.

9. D. Gollmann, Y. Han, C. J. Mitchell, *Redundant integer representations and fast exponentation*, Designs, Codes and Cryptography **7** (1996) 135–151.

10. D. M. Gordon, *A survey of fast exponentation methods*, Journal of Algorithms **27** (1998) 129–146.

11. R. L. Graham, A. C. -C. Yao, F. -F. Yao, *Addition chains with multiplicative cost*, Discrete Mathematics **23** (1978) 115–119.

12. D. A. Huffman, *A method for the construction of minimum-redundancy codes*, Proceedings of the IRE, **40** (1952) 1098–1101.

13. D. E. Knuth, Seminumerical Algorithms, 3rd ed., The Art of Computer Programming, Vol. 2, Addison-Wesley, Reading, MA, 1998.

14. B. M. E. Moret, *Decision trees and diagrams*, Computing Surveys **14** (1982) 593–623.

15. M. V. Subbarao, *Addition chains - some results and problems*, in: Number Theory and Applications, R. A. Mollin, ed. (Kluwer, Boston, 1989) 555–574.

16. Y. H. Tsai and Y. H. Chin, *On addition chains*, Intern. J. Computer Math. **45** (1992) 145–160.

17. H. Volger, *Some results on addition/subtraction chains*, Information Processing Letters **20** (1985) 155–160.

18. Y. Yacobi, *Exponentiating faster with addition chains*, in: Advances in Cryptology - EUROCRYPT '90 Proceedings, Lecture Notes in Computer Science, Vol. 473, I. B. Damgard, ed. (Springer-Verlag, NY, 1991) 222–229.

19. H. Zantema, *Minimizing sums of addition chains*, Journal of Algorithms **12** (1991) 281–307.