SIAM J. COMPUT.
Vol. 11, No. 3, August 1982

0097-5397/82/1103-0001 $01.00/0

THE COUNTERFEIT COIN PROBLEM REVISITED*

NATHAN LINIAL† AND MICHAEL TARSI‡

Abstract. We find the optimal algorithm in the sense of average run time for the counterfeit coin problem: Given n coins, one of which is heavier or lighter than the rest. Using a balance scale, find the counterfeit coin and whether it is heavy or light. An interesting feature of the solution is that our algorithm is a straight line algorithm. We also find the optimal algorithm if a standard coin is available for the first weighing.

Key words. average-optimal algorithms, search problems, Huffman trees, straight-line algorithms

In this paper we consider the following algorithmic problem which is a variation on the ancient false coin problem. (Find the counterfeit one, out of 12, by 3 weighings.)

The Problem.

Input: $n \geqq 3$ coins, one of which is false, being either heavier or lighter than the other coins.

Output: Find the counterfeit coin and whether it is heavy or light. The output coin j is light (resp. heavy) is denoted jL (resp. jH).

Step: The elementary step is placing an equal number of coins on the two sides of a balance.

Assuming the possible $2n$ outcomes equally likely, we look for algorithms which take the smallest number of steps on the average.

(1) Straight line/nonstraight line, i.e., branching is prohibited or allowed. Branching prohibited means that the location of the coins for every weighing is given and cannot be changed according to the results of previous weighings. We show that this does not change the situation.

(2) A standard coin is available for use in the weighings in addition to the set of n coins. It turns out that the average number of steps can sometimes be reduced using this extra coin.

We define four functions:

$$F(n), \quad F_N(n), \quad S(n), \quad S_N(n).$$

$F(n)$ is defined as $2n$ times the smallest average number of steps that any algorithm takes to solve the problem. (Note that when n coins are given there are $2n$ possible outcomes). The other three functions are defined for the straight line case (no branching allowed), with a standard coin, and straight line with a standard coin, respectively.

Our main results are summarized in the following.

THEOREM. *For $n \geqq 3$, $F(n) = F_N(n)$, $S(n) = S_N(n)$ (i.e., branching cannot reduce the average number of steps). Let*

$$2n = 3^t + 2k + 1, \qquad (3^t > k \geqq 0).$$

* Received by the editors May 8, 1981 and in revised form June 23, 1981.

† Department of Mathematics, University of California, Los Angeles, California 90024 and Institute of Mathematics and Computer Science, Hebrew University, Jerusalem 91904 Israel. The work of this author was supported by the Chaim Weizman post-doctoral grant.

‡ Department of Mathematics, University of California, Los Angeles, California 90024. The work of this author was supported in part by the National Science Foundation under grant MCS 78-18924.

Then

$$F(n)=2nt+3k+\begin{cases}4, & k \text{ even},\\ 3, & k \text{ odd},\end{cases}$$

$$S(n)=2nt+3k+\begin{cases}2, & k \text{ even},\\ 3, & k \text{ odd}.\end{cases}$$

Before going into the proof, we need some definitions:

We call a tree *ternary* if each node has at most 3 sons, called the L-, N- and R-sons, respectively. For a ternary tree T rooted at r, the subtree of T rooted at the L son of r is called the *L-subtree* of T, and its leaves are the *L-leaves* of T. The N- and R-subtrees and N- and R-leaves are defined similarly. The definition extends in the obvious way to w-subtrees and w-leaves, w being any word over the alphabet $\{L, N, R\}$. Also we define $h(T)=\sum d(x, r)$, the sum taken over all leaves x of T and $d(x, r)$ being the distance in T from x to r.

An algorithm which solves the counterfeit coin problem can be represented by a rooted ternary tree: The problem has $2n$ possible outcomes $1H, 1L, \cdots, nH, nL$. Every possible outcome is discovered by a certain sequence of weighing results. We denote such a sequence by a word over the alphabet $\{L, N, R\}$. The ith letter is $R(L)$ if the right (left) side of the balance is heavier at the ith weighing and N if both sides are equal. Since the process stops when the counterfeit coin is discovered, the obtained set of words is prefix free and naturally defines a ternary tree with $2n$ leaves. This is the tree which we assign to the algorithm. The definitions make it clear that the average number of steps that the algorithm takes equals $(1/2n)h(T)$, where T is the corresponding tree.

It is clear now that the following Huffman problem [*Hu*] is closely related:

Evaluate $H(n)=\min h(T)$ over all ternary trees with $2n$ leaves, and describe all trees which attain this minimum.

We illustrate the situation in the following:

Observation 1.

$$F_N(n)\geqq F(n)\geqq H(n),$$

$$S_N(n)\geqq S(n)\geqq H(n),$$

$$F(n)\geqq S(n),$$

$$F_N(n)\geqq S_N(n).$$

Convention. Given an integer $n\geqq 3$, we represent

$$2n=3^t+2k+1, \qquad 3^t>k\geqq 0.$$

The proof of the main theorem obviously follows from the following four propositions:

PROPOSITION 1.

$$H(n)=2nt+3k+2.$$

The trees which achieve this minimum are the following: Starting from a complete ternary tree of height t, choose any $k+1$ leaves. Attach to k of them 3 *new leaves* ($\cdot\to\Vdash$) *and attach* 2 *new leaves to the* $(k+1)$st ($\cdot\to\Vdash$). *For any other tree T with* $2n$ *leaves,* $h(T)>H(n)$.

Proof. See [Hu] for the general Huffman tree problem. □

PROPOSITION 2.

$$S(n) \geqq H(n) + \begin{cases} 0, & k \text{ even}, \\ 1, & k \text{ odd}. \end{cases}$$

PROPOSITION 3.

$$F(n) \geqq H(n) + \begin{cases} 2, & k \text{ even}, \\ 1, & k \text{ odd}. \end{cases}$$

PROPOSITION 4.

$$F_N(n) \leqq 2nt + 3k + \begin{cases} 4, & k \text{ even}, \\ 3, & k \text{ odd}, \end{cases}$$

$$S_N(n) \leqq 2nt + 3k + \begin{cases} 2, & k \text{ even}, \\ 3, & k \text{ odd}. \end{cases}$$

The proof of Propositions 2, 3 is based on the fact that not every ternary tree corresponds to an algorithm for the counterfeit coin problem.

PROPOSITION 5. *In a ternary tree which corresponds to an algorithm for the S-problem (even if branching is allowed and a standard coin given), the following holds*:

(S1) *For all* $i \geqq 0$ *the number of* N^iL*-leaves equals the number of* N^iR*-leaves.*

In an F-problem ternary tree (branching still allowed, no standard coin):

(F1) *The number of L-leaves = number of R-leaves is even.*

Proof. According to the tree representation of an algorithm, the N^iR leaves correspond to the outcomes still possible after i "equal" weighings and one "right side heavier", meaning that one of the coins on the right-hand side in the $(i+1)$st step is heavier or one of those in the left side is lighter. The N^iL leaves correspond to the same coins just switching "heavier" and "lighter", and this proves (S1).

As for (F1), if b coins are placed on each side of the balance in the first step, there are $2b$ L-leaves as well as R-leaves in the tree. (Notice that it is not so if a standard coin participates in the first weighing.) □

Instead of Proposition 2, we prove the somewhat stronger:

PROPOSITION 2'. *If T is a rooted ternary tree with* $2n$ *leaves satisfying* (S1), *then*

$$h(T) \geqq H(n) + \begin{cases} 0, & k \text{ even}, \\ 1, & k \text{ odd}. \end{cases}$$

(We remind the reader that we always represent $2n = 3^t + 2k + 1$, $(3^t > k \geqq 0)$).

Proof. The definition of Huffman trees implies $h(T) \geqq H(n)$, and so we only want to prove that no Huffman tree with $2n$ leaves satisfies condition (S1) if k is odd. We prove it by induction on t. For $t = 1$ the only odd k to be considered is $k = 1$, so $2n = 6$. The only Huffman tree with 6 leaves (up to a permutation) is

This tree and any permutation of it violates condition (S1).

By the description of Huffman trees in Proposition 1, the number of L, N, R-leaves are all between 3^{t-1} and 3^t. Also, two of these numbers are odd and one is even. Together with condition (S1), this implies that

$$\text{number of } L\text{-leaves} = \text{number of } R\text{-leaves} = 3^{t-1} + 2l, \qquad 3^{t-1} \geqq l \geqq 0,$$

and

$$\text{number of } N\text{-leaves} = 3^{t-1} + 2m + 1, \qquad m \text{ is odd}, \quad 3^{t-1} - 2 \geqq m \geqq 1.$$

Using once again our knowledge about Huffman trees, we see that the L, N, R-subtrees of a Huffman tree are all Huffman trees. But the N-subtree satisfies condition (S1) and so cannot be a Huffman tree by the induction hypothesis. □

Proof of Proposition 3. In view of Proposition 2 and the obvious fact that $F(n) \geqq S(n)$, it suffices to discuss the case of even k. For any tree T satisfying (S1) and (F1) with $2n$ leaves and $2n = 3^t + 2k + 1$ (k even, $3^t - 1 \geqq k \geqq 0$), we construct another ternary tree with $2n$ leaves T' so that $h(T) \geqq h(T') + 2$. This will prove our clairn. (F1) implies that the number of L-leaves in T which equals the number of R-leaves in T is even. We call it $2l$. The number of N-leaves is even, too, and we call it $2m$. Replace the L, N, R-subtrees of T by Huffman trees with $2l$, $2m$, $2l$ leaves, respectively. We claim that

$$2l, 2m \geqq 3^{t-1}.$$

By Proposition 1 each of the L, N, R-subtrees of T has a node with exactly two sons which are leaves. Let x, y, z be these nodes with sons x_1, x_2, y_1, y_2 and z_1, z_2 respectively. Assume without loss of generality that $d(x, r) \geqq d(y, r)$. Delete x_1, x_2 and add a new son y_3 to y. If $d(x, r) > d(y, r)$, this already reduces $h(T)$ by at least two. So we may assume that $d(x, r) = d(y, r) = d(z, r)$, which by Proposition 1 implies

$$2l, 2m \geqq 3^{t-1}.$$

Since $2n = 4l + 2m$ and k is even, it follows that

$$2m = 3^{t-1} + 2s + 1, \qquad s \text{ odd}, \quad 3^{t-1} - 2 \geqq s \geqq 1.$$

Now we show how to reduce $h(T)$ by at least two: transfer leaves from the L-subtree to the R-subtree as in the above-described procedure. As mentioned above, this reduces $h(T)$ by one. Now the N-subtree satisfies (S1) and s is odd, so on replacing the N-subtree by a Huffman tree with $2m$ leaves at least another one is gained, as shown in Proposition 2′. □

Proof of Proposition 4. The proof is constructive and inductive on n. Straight line algorithms will be represented by means of a table having n rows: one for each coin and a column for each weighing step. The (i, j) entry of the table being L, R or N indicates that the ith coin is placed on the left, placed on the right, or not placed on the balance at the jth weighing step. Not all the rows must be of equal length. Empty entries at the end of a row indicate that if the corresponding coin is false, then it will be known by the last step in whose column there is an $\{N, L, R\}$ entry in the row.

Associating L with 1, R with -1 and N with 0, it is obvious how arithmetic is done on the "row vectors" of the table. The vectors table constitutes an algorithm to solve the false coin problem if and only if the following 3 conditions hold:

(T1) For $\mathbf{x}$, $\mathbf{y}$ rows of the table, $\mathbf{x}$ is not a prefix of $\mathbf{y}$.

(T2) For $\mathbf{x}$, $\mathbf{y}$ rows of the table, $-\mathbf{x}$ is not a prefix of $\mathbf{y}$.

(T3) The sum of all row vectors is the zero vector.

Necessity. If (T1) is violated we cannot tell between the outcomes $\mathbf{x}H$ and $\mathbf{y}H$. If (T2) fails to hold, then an $\mathbf{x}H$ outcome cannot be told from a $\mathbf{y}L$ outcome. Failure of (T3) implies that one of the weighing steps is worthless since we place a different number of coins on the right side and left side of the balance.

Sufficiency. Let $\mathbf{b}$ be a row vector over $\{N, L, R\}$ indicating the results of the steps given by the balance. The jth entry being L, R or N indicated that on the jth step the left (resp. right) hand was heavy or they were balanced (resp.). Now either a prefix of $\mathbf{b}$ is a row in the table and this row is unique by (T1), or a prefix of $-\mathbf{b}$ is a row in the table and there is only one such row by (T2). In either case we know

the false coin, being heavy in the first case and light in the second case. It is impossible that both situations should occur, by (T2). Since one of the coins is false, **b** or $-\mathbf{b}$ should appear on the table. So we have:

$$F_N(n) = 2 \times \begin{array}{l}\text{smallest number of nonempty entries in an} \\ n\text{-row table satisfying (T1), (T2), (T3).}\end{array}$$

Before we can construct the tables we need some more terminology: If B is table with entries L, N, R and empty, we denote by RB the array resulting from affixing an R at the beginning of each row in B. Similar definitions hold for NB, LB. Also, $-B$ is the array which we obtain on replacing each L entry in B by R and each R by L. For arrays B, C, we let

$$\begin{matrix}\mathrm{B} \\ \mathrm{C}\end{matrix}$$

stand for the array whose row set is the union of the row set of B with that of C. If $\mathbf{w}$ is a row in B and $\mathbf{w}_1, \cdots, \mathbf{w}_p$ are words over $\{L, N, R\}$, we denote by

$$\mathbf{w} \rightarrow \begin{cases}\mathbf{w}_1 \\ \vdots \\ \mathbf{w}_p\end{cases}$$

the operation where the row $\mathbf{w}$ is deleted and the rows $\mathbf{w}_1, \cdots, \mathbf{w}_p$ are introduced into the array. Also N^r is the row of r consecutive N's, etc. Finally, the sum of all row vectors in B is denoted by $\sum B$, and $|B|$ is twice the number of nonempty entries in B.

We construct our tables for the F_N problem first. As usual, we represent $2n = 3^t + 2k + 1$ $(3^t - 1 \geqq k \geqq 0)$, and the corresponding table is called $A_{t,k}$. We start with the case of odd k. We describe a construction for $k = 1$ which is done by induction on t. The other tables for odd k are obtained by a simple procedure.

The case of even k is a little more involved. Using induction on t, we take care of the cases $k = 0, 2$. Most of the remaining cases can be handled easily using the $A_{t,2}$ table. The case $k = 3^t - 1$ is again handled separately by induction on t. Most of the (routine) calculations to ensure that conditions (T1), (T2), (T3) are satisfied and that the definitions make sense are omitted.

The solution for the S_N problem follows from the F_N case in a simple way.

We start the construction by defining a sequence of arrays B_t as follows:

$$B_1 = L \text{ (a one-by-one array with an } L \text{ entry).}$$

For $t \geqq 1$, define

$$B_{t+1} = \begin{cases}RB_t \\ LB_t \\ N(-B_t) \\ LN^t.\end{cases}$$

B_t is a $(3^t - 1)/2$ by t array satisfying (T1), (T2); it does not satisfy (T3) since $\sum B_t = L^t$. Note also that L^t is a row in B_t. Define now the array $A_{t,1}$ which is obtained from B_t by

$$L^t \rightarrow \begin{cases}L^t N \\ R^{t+1} \\ N^t L.\end{cases}$$

$A_{t,1}$ supplies an algorithm for $F_N(n)$, where $n=(3^t+3)/2$ and $|A_{t,1}|=2(tn+3)$ as needed.

For $2n=3^t+2k+1$, $k\geqq 3$ odd, say $k=2l+1$, we start from $A_{t,1}$ changing it into $A_{t,k}$ by picking l distinct rows $\mathbf{x}_1,\cdots,\mathbf{x}_l$ of length t and replacing each one of them:

$$\mathbf{x}_i\to\begin{cases}\mathbf{x}_iR\\ \mathbf{x}_iL\\ (-\mathbf{x}_i)N.\end{cases}$$

Note that $A_{t,k}$ satisfies (T1), (T2), (T3) and

$$|A_{t,k}|=F_N(n),$$

so $A_{t,k}$ supplies an optimal algorithm for the n coin problem.

Next we handle the case of even k. For $k=0$,

$$n=(3^t+1)/2,\qquad t\geqq 2,$$

change B_t into $A_{t,0}$ by the operations

$$L^t\to\begin{cases}L^tN\\ R^{t+1}\end{cases}\qquad \mathbf{x}\to\mathbf{x}(L),$$

where $\mathbf{x}$ is any row of length t in B_t and (L) indicates that this coin is placed on the left side of the balance on step $(t+1)$ only to have an equal number of coins on the right and left hands of the balance. If this coin is false, it will be known after t steps since conditions (T1), (T2) hold also if the (L) is omitted. Therefore we do not count the (L) in $|A_{t,0}|$. Hence $|A_{t,0}|=2(tn+2)$ and so it gives the optimal algorithm for $k=0$, $n=(3^t+1)/2$. For $k=2$ and $n=(3^t+5)/2$, change $A_{t,0}$ into $A_{t,2}$ by

$$\mathbf{x}(L)\to\begin{cases}\mathbf{x}L\\ (-\mathbf{x})L\\ \mathbf{x}N\end{cases}\qquad L^tN\to L^tR.$$

Now $|A_{t,2}|=|A_{t,0}|+2(2t+3)=2(nt+5)$, so $A_{t,2}$ solves the problem for $n=(3^t+5)/2$. For even k, $3^t-3\geqq k\geqq 4$, $k=2l+4$, we pick l distinct rows $\mathbf{x}_1,\cdots,\mathbf{x}_l$ of length t in $A_{t,2}$ and replace each of them:

$$\mathbf{x}_i\to\begin{cases}\mathbf{x}_iR\\ \mathbf{x}_iL\\ (-\mathbf{x}_i)N.\end{cases}$$

It is again routine to check that $A_{t,k}$ is an optimal solution table.

For the last remaining case, $n=(3^t-1)/2$, start from B_t, and to have condition (T3) hold replace

$$L^t\to N^tR,\qquad \mathbf{x}\to\mathbf{x}(L),$$

$\mathbf{x}$ being any row of length t and (L) is as explained above.

To complete the proof we only need to show that

$$S_N(n)\leqq 2nt+3k+2\quad\text{for even } k.$$

(The case of odd k is already settled, since $F_N(n)\geqq S_N(n)$ is obvious.) To construct a table for this problem, if $n\neq(3^{t+1}-1)/2$, start with an $A_{t,k+1}$ table, which has $n+1$ rows and satisfies $|A_{t,k+1}|=2+(n+1)+3(k+1)+3$. Two of the rows in $A_{t,k+1}$ are L^tN

and R^{t+1}. Let $S_{t,k}$ be the table which results on performing

$$L^tN \to (L^tR), \qquad R^{t+1} \to R^t.$$

The row vector (L^tR) is the row for the standard coin. Now $S_{t,k}$ violates only condition (T2) in that $-R^t = L^t$ and L^t is a prefix of L^tR. But if the results of the weighings are R^t or L^t we know that the R^t coin is counterfeit, as the (L^tR) coin is known to be a standard coin. Thus we do not count the (L^tR) row, and we get

$$|S_{t,k}| = |A_{t,k+1}| - 2(t+1) - 2 = 2nt + 3k + 2,$$

as required.

At last, if $n = (3^{t+1}-1)/2$, the table $\{^{B_{t+1}}_{(R^{t+1})}\}$, where (R^{t+1}) comes for the standard coin, supplies the required algorithm.

Acknowledgment. The first author wishes to express his thanks for the generous support of the Chaim Weizman postdoctoral grant.

REFERENCE

[Hu] D. A. HUFFMAN *A method for the construction of minimum redundancy codes*, Proc. IRE, 40, (Sept. 1952), p. 1098.

[Me] D. G. MEAD, *The average number of weighings to locate a counterfeit coin*, IEEE Trans. Inform. Theory, IT-25 (1974), pp. 616–617.