Computer Studies of Turing Machine Problems

SHEN LIN*
Bell Telephone Laboratories, Inc., Murray Hill, New Jersey

AND

TIBOR RADO
The Ohio State University and Battelle Memorial Institute, Columbus, Ohio

Abstract. This paper solves a problem relating to Turing machines arising in connection with the Busy Beaver logical game [2]. Specifically, with the help of a computer program, the values of two very well-defined positive integers $\Sigma(3)$ and $SH(3)$ are determined to be 6 and 21 respectively. The functions $\Sigma(n)$ and $SH(n)$, however, are noncomputable functions.

I. *Introduction*

It is assumed that the reader is familiar with the discussion of Turing machines in Kleene [1]. We operate here with binary Turing machines with the alphabet 0, 1. In the way of illustration, consider the following Turing machine.

CARD 1	
0	1 1 2
1	1 1 3

CARD 2	
0	1 0 1
1	1 1 2

CARD 3	
0	1 0 2
1	1 1 0

Actually, a Turing machine is not a machine, but rather a program (set of instructions) spelled out in a fixed format, as illustrated above. The instructions are specified on a finite number of "cards;" thus the above illustration shows a 3-card Turing machine. The term "card" seems to be preferable to the term "state" or "internal configuration," since the idea of a Turing machine is not dependent upon physical computers. Let us also note that for reasons of convenience we deviate from Kleene [1] by not permitting a "center shift." On each card, the leftmost column contains the alphabet 0, 1. The next column to the right contains the "overprint by" instruction. The next column to the right contains the "shift" instruction, where 0 is the code for left shift, 1 is the code for right shift. The rightmost column shows the "call" instruction; it shows the index of the card to which control is transferred.

In the "call" positions, we may have any one of the card indices (now 1, 2, 3) or we may have 0, which is the code for "stop" (see the 1-line of card 3).

The Turing machine operates on a potentially both-ways infinite tape, divided into squares, each of which contains a 0 or 1. At any moment, one of these squares is scanned, and one of the cards is "in control" in the sense that the instructions on that card are to be executed.

The example below shows a situation where card 3 is in control and a 0 is scanned. (The $\cdots$ at either end means that all squares not shown contain 0's.)

This research was sponsored in part by the US Army Research Office (Durham) under grant DA-ARO(D)-31-124-G312 with The Ohio State University Research Foundation.

* The results of this paper appeared in part in this author's doctoral dissertation, The Ohio State University, 1963, under the guidance of Professor T. Rado.

```
· · · | 0 | 1 | 1 | 0 | 1 | 0 | · · ·
                    3
```

Now let us start on an all-0 tape with its card 1, the Turing machine described above. We find that we receive the stop instruction after four shifts; the final tape situation is

```
· · · | 0 | 1 | 1 | 0 | · · ·
                    0
```

Next, consider another 3-card Turing machine given below.

CARD 1	
0	1 1 2
1	1 0 3

CARD 2	
0	1 0 1
1	1 1 2

CARD 3	
0	1 0 2
1	1 1 0

Starting this machine on an all-0 tape with its card 1, we find that the stop instruction is received after 13 shifts. The final tape situation is

```
· · · | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | · · ·
                                0
```

As a last illustration, consider the 3-card Turing machine shown below.

CARD 1	
0	1 1 2
1	1 1 0

CARD 2	
0	1 0 3
1	1 1 2

CARD 3	
0	1 0 1
1	1 0 3

Starting this machine on an all-0 tape with its card 1, we find after a while that the machine fails to reach the situation required for stopping (see the 1-line of card 1). Now the question is: Will this machine ever stop? To get better insight, it is convenient to use the following diagram for the "operating record" of the Turing machine.

$$\begin{array}{ccccccc}
 & & & & 0_1 & & \\
 & & & & 1 & 0_2 & \\
 & & & & 1_3 & 1 & \\
 & & & 0_3 & 1 & 1 & \\
 & & 0_1 & 1 & 1 & 1 & \\
 & & 1 & 1_2 & 1 & 1 & \\
 & & 1 & 1 & 1_2 & 1 & \\
 & & 1 & 1 & 1 & 1_2 & \\
 & & 1 & 1 & 1 & 1 & 0_2 \\
 & & 1 & 1 & 1 & 1_3 & 1 \\
 & & 1 & 1 & 1_3 & 1 & 1 \\
 & & 1 & 1_3 & 1 & 1 & 1 \\
 & & 1_3 & 1 & 1 & 1 & 1 \\
 & 0_3 & 1 & 1 & 1 & 1 & 1 \\
0_1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1_2 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1_2 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1_2 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1_2 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1_2 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1_2
\end{array}$$

This diagram is obtained by showing the successive tape situations individually; it is very suggestive in formulating conjectures about the behavior of a machine. Each row of the diagram shows the tape only to the point (right and left) beyond which the tape contains 0's only. The subscripts in the various squares show the index of the card in control. The previous diagram shows the operating record through the first 20 shifts.

Looking at the operating record, we note that the tape situations which are framed there show a certain similarity; and so we surmise that the machine is in a "loop" and hence will never stop. We return to this point later on in the paper. For the moment, we merely observe that it may be difficult (or even impossible) to determine by inspection whether or not a given machine will ever stop.

As shown in the preceding discussion, the Turing machine

CARD 1	
0	1 1 2
1	1 1 3

CARD 2	
0	1 0 1
1	1 1 2

CARD 3	
0	1 0 2
1	1 1 0

(started on an all-0 tape with its card 1) prints two 1's on the tape by the time it stops. On the other hand, the Turing machine

CARD 1	
0	1 1 2
1	1 0 3

CARD 2	
0	1 0 1
1	1 1 2

CARD 3	
0	1 0 2
1	1 1 0

prints out six 1's by the time it stops.

The following problem arises: Consider, for a fixed positive integer n, the class K_n of all the n-card binary Turing machines (with the card format described above). Let M be a Turing machine in this class K_n. Start M, with its card 1, on an all-0 tape. If M stops after a while, then M is termed a *valid entry* in the BB-n contest (the n-card classification of the Busy Beaver logical game), and its score $\sigma(M)$ is the number of 1's remaining on the tape at the time it stops. Since K_n is a finite class (the number of n-card binary Turing machines is easily seen to be $[4(n + 1)]^{2n}$), the number of valid entries in the BB-n contest is also finite. Hence, the scores of these valid entries constitute a nonempty finite set of non-negative integers, and thus this set has a (unique) largest element which we denote by $\sum(n)$, to stress that this largest element depends upon the card-number n. It is practically trivial that this function $\sum(n)$ is not general recursive (see T. Rado [2, 3]). On the other hand, it may be possible to determine the value of $\sum(n)$ for particular values of n. Trivially, $\sum(1) = 1$. As an exercise in a seminar, it has been shown that $\sum(2) = 4$. The determination of the actual value of $\sum(3)$ presented, however, quite unexpected difficulties, even though it was soon conjectured that $\sum(3) = 6$. The problem mentioned above is to decide whether or not this conjecture is valid.

The solution of this quite special problem was attempted by several competent mathematicians and programmers, by means of increasingly elaborate computer programs. The first definite solution is contained in the present work. After some experimenting, one will readily observe that the crux of the matter is, for any card number n, the determination of the function $SH(n)$ defined as follows. Each valid entry M in the BB-n contest performs a certain number $s(M)$ of shifts by the time it stops; the function $SH(n)$ is the maximum of $s(M)$ for all valid entries in the BB-n contest. As shown in [2], the function $SH(n)$ is not general recursive either. However, if for some particular value of n the value of $SH(n)$ can be determined, then for the same value of n the value of $\sum(n)$ can also be effectively determined. Indeed, we merely run each n-card machine (starting with card 1 on an all-0 tape) through not more than $SH(n)$ shifts; we note the scores of those that stop, and the largest one of these scores is then $\sum(n)$. On the basis of extensive computer experiments, it has been conjectured that $SH(3) = 21$; and a 3-card Turing machine that shifted 21 times by the time it stopped has been found. In the present work, we verify that this conjecture is also valid.

Our interest in these very special problems was motivated by the fact that at present there is no formal concept available for the "effective calculability" of individual well-defined integers like $\sum(4)$, $\sum(5)$, $\cdots$. (We are indebted to Professor Kleene of the University of Wisconsin for this information.) We felt therefore that the actual evaluation of $\sum(3)$, $SH(3)$ may yield some clues regarding the formulation of a fruitful concept for the effective calculability (and noncalculability) of individual well-defined integers.

II. *The Method*

The total number of 3-card Turing machines can easily be seen to be $[4(3+1)]^6$ or about 17 million. We reduce this number by proper normalization (see below for details) to 82,944 which is then divided into four lots. For each lot, our computer program first generates the machines and stores their conveniently coded descriptions in a table which we call the machine table. Then the program finds and discards those machines that stop in not more than 21 shifts and at the same time takes note of their scores and shift numbers (when they stop). The list of the machines that were not discarded is then scrambled up in the machine table and the first 50 are printed out. (The purpose is to enable us to observe the behavior patterns of the undecided machines.) Their operating records are then made up and each is examined for some pattern of behavior indicating that the particular machine considered will never stop. From these, we observed a certain recurrence pattern (called below the partial recurrence) which we programmed. As a matter of luck, it turned out that this simple recurrence pattern disposed of all but 40 of the machines. When the operating records of the 40 "holdouts" were examined, it turned out that they all showed patterns (discussed below) which enabled us to decide that all the 40 holdouts were never-stoppers. We may stress here a certain point of interest. Even though only 40 holdouts were left, it was not clear a priori that it can be decided as to whether they are never-stoppers or not, for a given machine may exhibit such a bizarre operating record or exhibit patterns that occur only after a prohibitive number of shifts that no human being could be expected to decide that it will never stop. It is also entirely conceivable that we may have on our hands a machine which is undecidable for some logical reason. Luckily this did not happen in this particular case. In this manner it was established that those machines that stopped at all stopped in no more than 21 shifts. Since the program showed us a stopper in 21 shifts, we conclude that $SH(3) = 21$ and the BB-3 problem was solved.

We now proceed to some details of our work.

The four lots. The number of binary 3-card Turing machines is (see above) $(4 \cdot 4)^6 = 2^{24} = 16{,}777{,}216$. However, in searching for the actual values of $\sum(3)$

and SH(3), it is sufficient to consider a subset of these machines, obtained by the following considerations. First, let us observe that all the 3-card machines are of the form

CARD 1	
0	p_{10} s_{10} c_{10}
1	p_{11} s_{11} c_{11}

CARD 2	
0	p_{20} s_{20} c_{20}
1	p_{21} s_{21} c_{21}

CARD 3	
0	p_{30} s_{30} c_{30}
1	p_{31} s_{31} c_{31}

(1)

where $p_{ij} = 0$ or 1, $s_{ij} = 0$ or 1, $c_{ij} = 0$ or 1 or 2 or 3. Now consider one of these machines; denote it by M_0. Suppose M_0 is a valid BB-3 entry, with a score $\sigma(M_0)$ and shift number $s(M_0)$. Let M_0^* be the "mirror image" of M_0; that is, the machine obtained by replacing (in the cards for M_0) each right shift by a left shift and each left shift by a right shift. Evidently, M_0^* is again a valid BB-3 entry, and $\sigma(M_0^*) = \sigma(M_0)$, $s(M_0^*) = s(M_0)$. Accordingly, we can restrict ourselves to consider those 3-card machines for which

$$s_{10} = 1. \tag{2}$$

Next, we note that if M_0 is a valid entry such that

$$p_{10} = p_{20} = p_{30} = 0, \tag{3}$$

then clearly $\sigma(M_0) = 0$ and $s(M_0) \leqq 3$. Since we know that $\sum(3) \geqq 6$ and $SH(3) \geqq 21$, such a machine can be disregarded in searching for the actual value of $\sum(3)$ and $SH(3)$. Accordingly, it is sufficient to consider 3-card machines for which at least one of p_{10}, p_{20}, p_{30} is equal to one. It is also clear that such a machine M_0 is a valid BB-3 entry, then before M_0 stops, a card C_j with $p_{j0} = 1$ must have been used if the situation $\sigma(M_0) = 0$, $s(M_0) \leqq 3$ is to be avoided. Now let C_i be the card of M_0 which is in control when M_0 first overprints a 1; then $p_{i0} = 1$. Let M_0' be the machine obtained from M_0 by renumbering the cards of M_0 (and adjusting the call instructions c_{ij}) so that the original card C_i is renamed C_1. Clearly $\sigma(M_0') = \sigma(M_0)$, and $s(M_0) \leqq s(M_0') + 2$. After this modification, we can assume that

$$p_{10} = 1. \tag{4}$$

Next, if we have now $c_{10} = 0$, then clearly $\sigma(M_0) = 1$, $s(M_0) = 1$; hence any machine with $c_{10} = 0$ can be disregarded. Since then $c_{10} \neq 0$, by renumbering the cards 2 and 3 of M_0 (and adjusting the call numbers c_{ij}), we can assume that

$$c_{10} = 2. \tag{5}$$

Finally, if now $c_{20} = 0$, then clearly $\sigma(M_0) \leqq 2$, $s(M_0) = 2$. Hence, the machines with $c_{20} = 0$ can be disregarded. In view of (2), (4), (5) we can therefore assume that

$$p_{10} = 1, \quad s_{10} = 1, \quad c_{10} = 2, \quad c_{20} \neq 0, \tag{6}$$

without changing the actual value of $\sum(3)$. As regards $SH(3)$, it is clear from the preceding comments that on denoting by SH$^*(3)$ the maximum of $s(M)$ for valid BB-3 entries normalized in the manner shown in (6), then $SH(3) \leqq SH^*(3) + 2$.

Next, let M_0 be a valid BB-3 entry. Even though there may be several "stop-lines" in the cards for M_0, clearly only one of the several stop instructions will actually be used. Accordingly, we can assume that exactly one of c_{11}, c_{21}, c_{30}, c_{31} is equal to zero. Furthermore, the shift instruction in the unique stop-line of M_0 does not affect either $\sigma(M_0)$ or $s(M_0)$; hence we can assume that the stop-line orders a right shift. Finally, if we specify that the stop-line should issue the "overprint by 1" instruction, then clearly we do not diminish $\sigma(M_0)$. Hence, we can assume that the stop-line has the form 1 1 0. Now the unique stop-line may occur in just four locations; namely, as the 1-line of card 1, or as the 1-line of card 2, or the 0-line or 1-line of card 3. It follows that the machines that we have to investigate can be classified into four lots as shown below.

	CARD 1		CARD 2		CARD 3	
Lot 1	0	1 1 2	0	p_{20} s_{20} $\neq 0$	0	p_{30} s_{30} $\neq 0$
	1	1 1 0	1	p_{21} s_{21} $\neq 0$	1	p_{31} s_{31} $\neq 0$

	CARD 1		CARD 2		CARD 3	
Lot 2	0	1 1 2	0	p_{20} s_{20} $\neq 0$	0	p_{30} s_{30} $\neq 0$
	1	p_{11} s_{11} $\neq 0$	1	1 1 0	1	p_{31} s_{31} $\neq 0$

	CARD 1		CARD 2		CARD 3	
Lot 3	0	1 1 2	0	p_{20} s_{20} $\neq 0$	0	1 1 0
	1	p_{11} s_{11} $\neq 0$	1	p_{21} s_{21} $\neq 0$	1	p_{31} s_{31} $\neq 0$

	CARD 1		CARD 2		CARD 3	
Lot 4	0	1 1 2	0	p_{20} s_{20} $\neq 0$	0	p_{30} s_{30} $\neq 0$
	1	p_{11} s_{11} $\neq 0$	1	p_{21} s_{21} $\neq 0$	1	1 1 0

A simple computation shows that the number of machines in each one of these lots is equal to 20,736. Thus (as far as $\sum(3)$ is concerned) it is sufficient to investigate the $4 \cdot 20{,}736 = 82{,}944$ machines contained in the four lots. As regards $SH(3)$, a little more work is involved; we return to this point later.

We proceed to outline the procedures we followed in treating these four lots.

Description of the computer program. Each individual Turing machine is identified for the purpose of the program as follows. Each line of the Turing card is coded into a four-bit binary word (with the "call" instruction occupying two bits). They are then packed in sequence from the 0-line of card 1, 1-line of card 1, to the 1-line of card 3 into a single machine word. This enables us to identify each machine in terms of a single word. For example, the machine

CARD 1		CARD 2		CARD 3	
0	1 1 2	0	1 0 3	0	1 1 1
1	1 1 3	1	1 1 0	1	0 0 2

is coded as

1 1 1 0	1 1 1 1	1 0 1 1	1 1 0 0	1 1 0 1	0 0 1 0

For convenience we also use the octal representation of this binary number in referring to the Turing machine. Thus we identify the above machine also by its "serial number," 73736322. Since the number of machines in each lot is still too large to code by hand, we generate these machines in our computer program by a generalized counting process and store them in a machine table. For each

$$\begin{array}{llllll}
 & 0_1 & & & & \\
 & 1_1 & 0_2 & & & \\
 & 1 & 0 & 0_3 & & \\
 & 1 & 0_3 & 1 & & \\
 & 1_3 & 1 & 1 & & \\
0_1 & 1 & 1 & 1 & & \\
1 & 1_2 & 1 & 1 & & \\
1 & 1 & 1_2 & 1 & & \\
1 & 1 & 1 & 1_2 & & \\
1 & 1 & 1 & 1 & 0_2 & \\
1 & 1 & 1 & 1 & 0 & 0_3 \\
1 & 1 & 1 & 1 & 0_3 & 1 \\
1 & 1 & 1 & 1_3 & 1 & 1 \\
1 & 1 & 1_1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1_0 & 1 & 1
\end{array}$$

14 *shifts*

	C_1	C_2	C_3
0	1 1 2	0 1 3	10 3
1	1 1 0	1 1 2	1 0 1

$$\begin{array}{llllll}
 & & & 0_1 & & \\
 & & & 1 & 0_2 & \\
 & & & 1_1 & 1 & \\
 & & 0_3 & 1 & 1 & \\
 & 0_2 & 1 & 1 & 1 & \\
0_1 & 1 & 1 & 1 & 1 & \\
1 & 1_2 & 1 & 1 & 1 & \\
1 & 1 & 1_2 & 1 & 1 & \\
1 & 1 & 1 & 1_2 & 1 & \\
1 & 1 & 1 & 1 & 1_2 & \\
1 & 1 & 1 & 1 & 1 & 0_2 \\
1 & 1 & 1 & 1 & 1_1 & 1 \\
1 & 1 & 1 & 1_3 & 1 & 1 \\
1 & 1 & 1 & 1 & 1_0 & 1
\end{array}$$

13 *shifts*

	C_1	C_2	C_3
0	1 1 2	1 0 1	1 0 2
1	1 0 3	1 1 2	1 1 0

$$\begin{array}{lllllll}
 & & 0_1 & & & & \\
 & & 1 & 0_2 & & & \\
 & & 1_3 & 1 & & & \\
 & 0_2 & 0 & 1 & & & \\
0_3 & 1 & 0 & 1 & & & \\
1 & 1_1 & 0 & 1 & & & \\
1 & 1 & 0_3 & 1 & & & \\
1 & 1 & 1 & 1_1 & & & \\
1 & 1 & 1 & 1 & 0_3 & & \\
1 & 1 & 1 & 1 & 1 & 0_1 & \\
1 & 1 & 1 & 1 & 1 & 1 & 0_2 \\
1 & 1 & 1 & 1 & 1 & 1_3 & 1 \\
1 & 1 & 1 & 1 & 1_2 & 0 & 1 \\
1 & 1 & 1 & 1 & 1 & 0_0 & 1
\end{array}$$

13 *shifts*

	C_1	C_2	C_3
0	1 1 2	1 0 3	1 1 1
1	1 1 3	1 1 0	0 0 2

$$\begin{array}{llllll}
 & & 0_1 & & & \\
 & & 1 & 0_2 & & \\
 & & 1_3 & 1 & & \\
 & 0_2 & 1 & 1 & & \\
0_3 & 1 & 1 & 1 & & \\
1 & 1_1 & 1 & 1 & & \\
1 & 1 & 1_1 & 1 & & \\
1 & 1 & 1 & 1_1 & & \\
1 & 1 & 1 & 1 & 0_1 & \\
1 & 1 & 1 & 1 & 1 & 0_2 \\
1 & 1 & 1 & 1 & 1_3 & 1 \\
1 & 1 & 1 & 1_2 & 1 & 1 \\
1 & 1 & 1 & 1 & 1_0 & 1
\end{array}$$

12 *shifts*

	C_1	C_2	C_3
0	1 1 2	1 0 3	1 1 1
1	1 1 1	1 1 0	1 0 2

$$\begin{array}{llllll}
 & & & 0_1 & & \\
 & & & 1 & 0_2 & \\
 & & & 1 & 1 & 0_3 \\
 & & & 1 & 1_1 & 1 \\
 & & & 1_3 & 1 & 1 \\
 & & 0_2 & 0 & 1 & 1 \\
 & & 1 & 0_3 & 1 & 1 \\
 & & 1_1 & 1 & 1 & 1 \\
 & 0_3 & 1 & 1 & 1 & 1 \\
0_1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1_2 & 1 & 1 & 1 & 1 \\
1 & 1 & 1_0 & 1 & 1 & 1
\end{array}$$

11 *shifts*

	C_1	C_2	C_3
0	1 1 2	1 1 3	1 0 1
1	1 0 3	1 1 0	0 0 2

FIG. 1. Score champs and their operating records

$$\begin{array}{lllll}
 & 0_1 & & & \\
 & 1 & 0_2 & & \\
 & 1_2 & 1 & & \\
 & & 1_3 & & \\
 & 0_1 & 1 & & \\
 & 1 & 1_2 & & \\
 & 1 & 0 & 0_3 & \\
 & 1 & 0_3 & 1 & \\
 & 1_3 & 1 & 1 & \\
0_1 & 1 & 1 & 1 & \\
1 & 1_1 & 1 & 1 & \\
1 & 0 & 1_3 & 1 & \\
1 & 0_1 & 1 & 1 & \\
1 & 1 & 1_2 & 1 & \\
1 & 1 & 0 & 1_3 & \\
1 & 1 & 0_1 & 1 & \\
1 & 1 & 1 & 1_2 & \\
1 & 1 & 1 & 0 & 0_3 \\
1 & 1 & 1 & 0_3 & 1 \\
1 & 1 & 1_3 & 1 & 1 \\
1 & 1_1 & 1 & 1 & 1 \\
1 & 1 & 1_0 & 1 & 1
\end{array}$$

	C_1	C_2	C_3
0	1 1 2	1 0 2	1 0 3
1	1 1 0	0 1 3	1 0 1

21 *shifter*

$$\begin{array}{lllll}
 & & 0_1 & & \\
 & & 1 & 0_2 & \\
 & & 1_3 & & \\
 & 0_1 & 1 & & \\
 & 1 & 1_2 & & \\
 & 1 & 0 & 0_3 & \\
 & 1 & 0_3 & 1 & \\
 & 1_3 & 1 & 1 & \\
0_1 & 1 & 1 & 1 & \\
1 & 1_2 & 1 & 1 & \\
1 & 0 & 1_3 & 1 & \\
1 & 0_1 & 1 & 1 & \\
1 & 1 & 1_2 & 1 & \\
1 & 1 & 0 & 1_3 & \\
1 & 1 & 0_1 & 1 & \\
1 & 1 & 1 & 1_2 & \\
1 & 1 & 1 & 0 & 0_3 \\
1 & 1 & 1 & 0_3 & 1 \\
1 & 1 & 1_3 & 1 & 1 \\
1 & 1_1 & 1 & 1 & 1 \\
1 & 1 & 1_0 & 1 & 1
\end{array}$$

	C_1	C_2	C_3
0	1 1 2	0 0 3	1 0 3
1	1 1 0	0 1 3	1 0 1

20 *shifter*

$$\begin{array}{llllll}
 & & 0_1 & & & \\
 & & 1 & 0_2 & & \\
 & & 1 & 0 & 0_3 & \\
 & & 1 & 0_3 & 1 & \\
 & & 1_3 & 1 & 1 & \\
 & 0_1 & 0 & 1 & 1 & \\
 & 1 & 0_2 & 1 & 1 & \\
 & 1 & 0 & 1_3 & 1 & \\
 & 1 & 0_1 & 0 & 1 & \\
 & 1 & 1 & 0_2 & 1 & \\
 & 1 & 1 & 0 & 1_3 & \\
 & 1 & 1 & 0_1 & & \\
 & 1 & 1 & 1 & 0_2 & \\
 & 1 & 1 & 1 & 0 & 0_3 \\
 & 1 & 1 & 1 & 0_3 & 1 \\
 & 1 & 1 & 1_3 & 1 & 1 \\
 & 1 & 1_1 & 0 & 1 & 1 \\
 & 1_1 & 1 & 0 & 1 & 1 \\
0_1 & 1 & 1 & 0 & 1 & 1 \\
1 & 1_2 & 1 & 0 & 1 & 1 \\
1 & 1 & 1_0 & 0 & 1 & 1
\end{array}$$

	C_1	C_2	C_3
0	1 1 2	0 1 3	1 0 3
1	1 0 1	1 1 0	0 0 1

20 *shifter*

FIG. 2. High shifters and their operating records

machine in a fixed lot, we have two fixed lines, namely the 0-line of card 1, 112 (coded 1110) and the stop-line 110 (coded 1100) which occupy the same bit positions in every Turing machine coded in the lot. These are set up first in the storage locations assigned for the machine table. Each of the four other lines can have 12 possible cases. The program sets up these 12 cases of one line in the corresponding bit locations and OR's them into the machine table consecutively, repeating this procedure 1728 times. Then the second line is set up, this time with each case repeated 12 times and the whole configuration of 144 entries repeatedly OR'ed into the machine table 144 times. The third line is set up with each case first repeated 144 times and the whole configuration of 1728 entries repeatedly OR'ed into the machine table 12 times. Finally, the last line is set up with each of the 12 possibilities repeated 1728 times and OR'ed into the machine table. In this way all possible machines in a lot are obtained and their coded descriptions in the machine table are now ready for examination.

Previous work on the *BB*-3 problem led to the conjecture that $SH(3) = 21$. We therefore simulate the operation of each Turing machine in the four lots through 21 shifts in our computer. If a machine stops in less than or equal to 21 shifts, its shift-number and score are noted in a table and the machine is then discarded. It is our hope that we can show later that all those machines that do

Turing Machine

CARD 1	
0	1 1 2
1	1 1 0

CARD 2	
0	0 1 3
1	1 0 2

CARD 3	
0	1 0 1
1	0 1 2

Operating Record

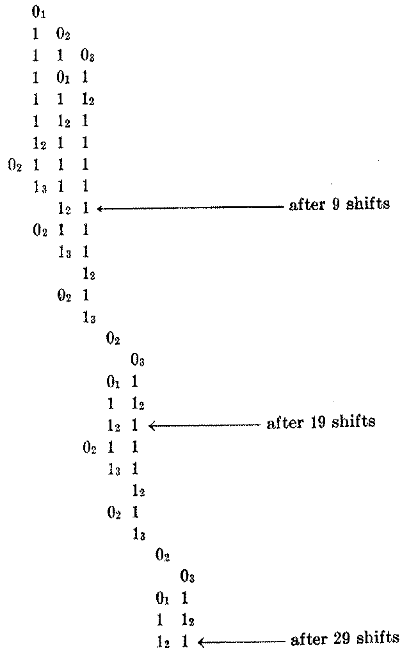


FIG. 3. Operating record of the Turing machine whose serial number is 73075226 (octal) showing the total recurrence pattern

not stop in less than or equal to 21 shifts will never stop. Furthermore, descriptions of machines that score six (or more) or shifted 20 or 21 times are printed out. The collected statistics reveal the following: In all four lots, we have 26,073 stoppers in less than or equal to 21 shifts (out of a total of 82,944), five machines which scored six, one machine which shifted 21 times and two machines that shifted 20 times (see Figures 1 and 2 for their descriptions).

In order to reduce further the machine table size, we discard all machines in lot 1 with no 1's in the "call" positions of cards 2 and 3, and all machines in lots 3 and 4 with no 3's in the "call" positions of cards 1 and 2. These are obvious never-stoppers since the stop-lines can not be reached. In all four lots, 27,774 of these machines are discarded.

The next step in the investigation is to discard those never-stoppers which exhibit a recurrence pattern. The idea may be described briefly as follows. Suppose we operate a given Turing machine M and observe that card i scans a tape square S_m containing the digit d after m shifts. Later, suppose the same card i scans a square S_n containing the same digit d after n shifts. If, relative to the scanned squares S_m and S_n, the tape conditions in both instances are identical, it is clear that the same pattern of operation must repeat from then on and hence the Turing machine M is a never-stopper. We call this a *total recurrence* (see Figure 3). Further analysis reveals that we need not have to consider the total tape conditions in most cases. Suppose the square S_n is to the right of the square S_m and that, during the operation from m shifts to n shifts, the leftmost

Turing Machine

CARD 1	
0	1 1 2
1	1 1 0

CARD 2	
0	1 0 2
1	0 0 3

CARD 3	
0	1 0 1
1	1 1 1

Operating Record

$$
\begin{array}{llllll l}
 & & 0_1 & & & & \\
 & & 1 & 0_2 & & & \\
 & & 1_2 & 1 & & & \\
 & 0_3 & 0 & 1 & & & \\
0_1 & 1 & 0 & 1 & & & \\
1 & 1_2 & 0 & 1 & & & \\
1_3 & 0 & 0 & 1 & & & \\
1 & 0_1 & 0 & 1 & & & \\
1 & 1 & 0_2 & 1 & & & \\
1 & 1_2 & 1 & 1 & & & \\
1_3 & 0 & 1 & 1 & & & \\
1 & 0_1 & 1 & 1 & & & \\
1 & 1 & 1_2 & 1 & & & \longleftarrow \text{after 12 shifts} \\
1 & 1_3 & 0 & 1 & & & \\
1 & 1 & 0_1 & 1 & & & \\
1 & 1 & 1 & 1_2 & & & \\
1 & 1 & 1_3 & & & & \\
1 & 1 & 1 & 0_1 & & & \\
1 & 1 & 1 & 1 & 0_2 & & \\
1 & 1 & 1 & 1_2 & 1 & & \longleftarrow \text{after 19 shifts} \\
1 & 1 & 1_3 & 0 & 1 & & \\
1 & 1 & 1 & 0_1 & 1 & & \\
1 & 1 & 1 & 1 & 1_2 & & \\
1 & 1 & 1 & 1_3 & & & \\
1 & 1 & 1 & 1 & 0_1 & & \\
1 & 1 & 1 & 1 & 1 & 0_2 &
\end{array}
$$

FIG. 4. Operating record of the Turing machine whose serial number is 73121635 (octal) showing the partial recurrence with left barrier

square scanned is S, which is, say k squares to the left of the square S_m. We call the square which is $k + 1$ squares to the left of S_m the *left barrier* relative to S_m. Similarly, the left barrier relative to S_n will be the square which is $k + 1$ squares to the left of the square S_n. It is clear then that if the tape conditions to the right of the left barrier relative to S_m after m shifts is identical to the tape condition to the right of the left barrier relative to S_n after n shifts, the same sequence of operations must repeat and the Turing machine M will never stop. We call this a *partial recurrence pattern.*

As an illustration, consider the Turing machine and its operating record in Figure 4. Card 2 scans a 1 after 12 shifts and again a 1 after 19 shifts, during which the portion of the tape scanned is never more than one square to the left of S_{12}, the scanned square after 12 shifts. Since the portion of the tape to the right of the left barrier relative to S_{12} is identical to the portion of the tape to the right of the left barrier relative to S_{19}, we see that the same sequence of operations must repeat from 19 to 26 shifts, and so on, progressing to the right. It is obvious therefore that this machine will never stop.

If S_n is to the left of S_m, we may consider a right barrier similar to the left barrier described above. An illustration of this case is given in Figure 5.

Turing Machine

CARD 1	
0	1 1 2
1	1 1 0

CARD 2	
0	1 0 3
1	1 1 1

CARD 3	
0	1 0 1
1	0 0 3

Operating Record

```
                              0_1
                              1    0_2
                              1_3  1
                         0_3  0    1
                    0_1  1    0    1
                    1    1_2  0    1
                    1    1    0_1  1
                    1    1    1    1_2
                    1    1    1    1    0_1
                    1    1    1    1    1    0_2
                    1    1    1    1    1_3  1
                    1    1    1    1_3  0    1
                    1    1    1_3  0    0    1
                    1    1_3  0    0    0    1
                    1_3  0    0    0    0    1
               0_3  0    0   |0    0    0    1
          0_1  1    0    0   |0    0    0    1
          1    1_2  0    0   |0    0    0    1
          1    1    0_1  0   |0    0    0    1
          1    1    1    0_2 |0    0    0    1
          1    1    1_3  1   |0    0    0    1
          1    1_3  0    1   |0    0    0    1
          1_3  0    0    1   |0    0    0    1
     0_3  0    0   |0    1    0    0    0    1
0_1  1    0    0   |0    1    0    0    0    1
1    1_2  0    0   |0    1    0    0    0    1
1    1    0_1  0   |0    1    0    0    0    1
1    1    1    0_2 |0    1    0    0    0    1
1    1    1_3  1   |0    1    0    0    0    1
1    1_3  0    1   |0    1    0    0    0    1
1_3  0    0    1   |0    1    0    0    0    1
```

FIG. 5. Operating record of the Turing machine whose serial number is 73136623 (octal) showing the partial recurrence with right barrier

If S_n happens to be the same square as S_m, we may use both the right barrier and the left barrier. If the portion of the tape between the right and the left barriers after m shifts is identical to that after n shifts, then a recurrence must appear and the machine will never stop.

Lot 1	*Lot 2*	*Lot 3*	*Lot 4*
73037233	73676261	70537311	70513754
73137233	73736122	70636711	70612634
73137123	71536037	70726711	70712634
73136523	73336333	72737311	72377034
73133271	71676261	71717312	72377234
73133251	73336133	72211715	72613234
73132742	73236333	72237311	
73132542	73236133	72311715	
73032532		72317716	
73032632		72331715	
73033132		72337311	
73033271		72337315	
73073271			
73075221			

FIG. 6. The forty holdouts

0_1

$1\ 0_2$

$1_3\ 1$

$0_3\ 1\ 1$

$0_1\ 1\ 1\ 1$

$1\ 1_2\ 1\ 1$

$1\ 1\ 1_2\ 1$

$1\ 1\ 1\ 1_2$

$1\ 1\ 1\ 1\ 0_2$

$1\ 1\ 1\ 1_3\ 1$

$1\ 1\ 1_3\ 1\ 1$

$1\ 1_3\ 1\ 1\ 1$

$1_3\ 1\ 1\ 1\ 1$

$0_3\ 1\ 1\ 1\ 1\ 1$

$0_1\ 1\ 1\ 1\ 1\ 1\ 1$

$1\ 1_2\ 1\ 1\ 1\ 1\ 1$

$1\ 1\ 1_2\ 1\ 1\ 1\ 1$

$1\ 1\ 1\ 1_2\ 1\ 1\ 1$

$1\ 1\ 1\ 1\ 1_2\ 1\ 1$

$1\ 1\ 1\ 1\ 1\ 1_2\ 1$

$1\ 1\ 1\ 1\ 1\ 1\ 1_2$

$1\ 1\ 1\ 1\ 1\ 1\ 1\ 0_2$

$1\ 1\ 1\ 1\ 1\ 1\ 1_3\ 1$

$1\ 1\ 1\ 1\ 1\ 1_3\ 1\ 1$

$1\ 1\ 1\ 1\ 1_3\ 1\ 1\ 1$

$1\ 1\ 1\ 1_3\ 1\ 1\ 1\ 1$

$1\ 1\ 1_3\ 1\ 1\ 1\ 1\ 1$

$1\ 1_3\ 1\ 1\ 1\ 1\ 1\ 1$

$1_3\ 1\ 1\ 1\ 1\ 1\ 1\ 1$

$0_3\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1$

$0_1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1$

$1\ 1_2\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1$

$1\ 1\ 1_2\ 1\ 1\ 1\ 1\ 1\ 1\ 1$

$1\ 1\ 1\ 1_2\ 1\ 1\ 1\ 1\ 1\ 1$

$1\ 1\ 1\ 1\ 1_2\ 1\ 1\ 1\ 1\ 1$

$1\ 1\ 1\ 1\ 1\ 1_2\ 1\ 1\ 1\ 1$

$1\ 1\ 1\ 1\ 1\ 1\ 1_2\ 1\ 1\ 1$

$1\ 1\ 1\ 1\ 1\ 1\ 1\ 1_2\ 1\ 1$

$1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1_2\ 1$

$1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1_2$

$1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0_2$

CARD 1	
0	1 1 2
1	1 1 0

CARD 2	
0	1 0 3
1	1 1 2

CARD 3	
0	1 0 1
1	1 0 3

FIG. 7. Operating record of the Turing machine whose serial number is 73137233.

Next, we construct a computer routine to discard never-stoppers showing the recurrence patterns described above. For the Turing tape we use a machine word of 36 bits with each bit representing a square and the starting square at bit 18. We further identify the squares on the tape by their "deviation" from the starting square: the starting square has deviation 0, the square to the right of the starting square has deviation 1, the square to the left of the starting square has deviation -1, and so on. Thus a square with a deviation D is represented by the bit $18 + D$. After each shift, the tape condition T, herein represented by a single machine word of 36 bits, is stored in an appropriate tape table TB_{ij} corresponding to the card index i called and the digit j in the scanned square. The shift-number at that time and the deviation of the scanned square are also stored in the accompanying tables. Meanwhile the deviations of the scanned square after each shift are further stored in another table (called the deviation table), so that the maximum deviation D_{MAX} and the minimum deviation D_{MIN} may be determined for any portion of the operation of the Turing machine, say between S_1 shifts and S_2 shifts. This is to find out how far to the right and to the left the scanning head has moved during this portion of the operation (for use in finding the right and the left barriers). Whenever an entry T is made into a tape table and the tape table was previously nonempty, tests are made for recurrence as follows. If T_0 is a previous entry in the table with associated shift-number s_0 and deviation D_0, and s is the shift-number and D the deviation associated with the present entry T, D_0 and D are compared. If $D_0 < D$, minimum deviation D_{MIN} is determined from the deviation table for the operation between s_0 and s shifts. T_0 is shifted left $18 + D_{\mathrm{MIN}}$ bits and T shifted left $18 + D_{\mathrm{MIN}} + D - D_0$ bits and compared. If the resulting logical words are equal, the Turing machine operated on is discarded. Otherwise, T is tested against another previous entry in the same tape table TB_{ij} until all previous entries in the tape table TB_{ij} are checked. If no recurrence pattern is found, the Turing machine is given one more shift and the same procedure goes on. Symmetrical procedures hold when $D_0 > D$. If $D_0 = D$, both D_{MAX} and D_{MIN} are determined and T_0 and T are compared from bits $18 + D_{\mathrm{MIN}}$ to $18 + D_{\mathrm{MAX}}$ by the use of a mask.

A bound of 50 is set for the shift-number with a check for spill provided whenever the magnitude of the deviation exceeds 17. This is to insure that the portion of the tape scanned can be contained entirely in a single machine word; and the both-ways infinite portions of the tape to the right and to the left of the squares represented by the 36-bit machine word which have never been scanned can therefore be assumed to contain all 0's in all instances. If the machine does not show the recurrence pattern after 50 shifts, it is retained in the machine table and printed out later as a "holdout".

The results of this modest effort were quite unexpected. In all four lots, only 40 holdouts were left. That these 40 holdouts are all never-stoppers will be shown in Section III. In Figure 6, we give the descriptions of these 40 holdouts in terms of their octal "serial numbers."

III. *The Forty Holdouts*

As stated in Section II there remained 40 Turing machines which the computer program failed to eliminate. According to our plan, these 40 holdouts were checked by hand, and they were all recognized to be never-stoppers by inspection of their operating records. The Figures 7, 8 and 9 show some typical cases. To illustrate the methods used to show that they are never-stoppers, we discuss in detail two additional cases below.

As our first case, we consider the holdout whose operating record is shown in Figure 10. The cards of this machine are as follows.

CARD 1	
0	1 1 2
1	1 1 0

CARD 2	
0	0 0 3
1	1 1 2

CARD 3	
0	1 0 1
1	1 0 3

By inspecting its operating record (Figure 10), we observe that the following tape situation appears repeatedly.

· · ·	1	1	1	1	1_3	0	· · ·

This leads to the question of what happens next when we have this type of tape situation. A glance at card 3 reveals that the string of 1's is first extended to the left by one. Let us use the code name XTNDL for this operation. After this, a left shift is made (code name GOTOL), and control is transferred to card 1. Card 1 orders printing a 1 over a 0 (MARK); there follows a sequence of shifts to the right, after which control is transferred to card 3 at the right of the string

$$\begin{array}{ccccccccc}
 & & & & 0_1 & & & & \\
 & & & & 1 & 0_2 & & & \\
 & & & & 1_3 & 1 & & & \\
 & & & 0_1 & 1 & 1 & & & \\
 & & & 1 & 1_2 & 1 & & & \\
 & & & 1 & 0 & 1_2 & & & \\
 & & & 1 & 0 & 0 & 0_2 & & \\
 & & & 1 & 0 & 0_3 & 1 & & \\
 & & & 1 & 0_2 & 1 & 1 & & \\
 & & & 1_2 & 1 & 1 & 1 & & \\
 & & 0_1 & 1 & 1 & 1 & 1 & & \\
 & & 1 & 1_2 & 1 & 1 & 1 & & \\
 & & 1 & 0 & 1_2 & 1 & 1 & & \\
 & & 1 & 0 & 0 & 1_2 & 1 & & \\
 & & 1 & 0 & 0 & 0 & 1_2 & & \\
 & & 1 & 0 & 0 & 0 & 0 & 0_2 & \\
 & & 1 & 0 & 0 & 0 & 0_3 & 1 & \\
 & & 1 & 0 & 0 & 0_2 & 1 & 1 & \\
 & & 1 & 0 & 0_3 & 1 & 1 & 1 & \\
 & & 1 & 0_2 & 1 & 1 & 1 & 1 & \\
 & & 1_3 & 1 & 1 & 1 & 1 & 1 & \\
 & 0_1 & 1 & 1 & 1 & 1 & 1 & 1 & \\
 & 1 & 1_2 & 1 & 1 & 1 & 1 & 1 & \\
 & 1 & 0 & 1_2 & 1 & 1 & 1 & 1 & \\
 & 1 & 0 & 0 & 1_2 & 1 & 1 & 1 & \\
 & 1 & 0 & 0 & 0 & 1_2 & 1 & 1 & \\
 & 1 & 0 & 0 & 0 & 0 & 1_2 & 1 & \\
 & 1 & 0 & 0 & 0 & 0 & 0 & 1_2 & \\
 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0_2 \\
 & 1 & 0 & 0 & 0 & 0 & 0 & 0_3 & 1 \\
 & 1 & 0 & 0 & 0 & 0 & 0_2 & 1 & 1 \\
 & 1 & 0 & 0 & 0 & 0_3 & 1 & 1 & 1 \\
 & 1 & 0 & 0 & 0_2 & 1 & 1 & 1 & 1 \\
 & 1 & 0 & 0_3 & 1 & 1 & 1 & 1 & 1 \\
 & 1 & 0_2 & 1 & 1 & 1 & 1 & 1 & 1 \\
 & 1_3 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0_1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{array}$$

CARD 1	
0	1 1 2
1	1 1 0

CARD 2	
0	1 0 3
1	0 1 2

CARD 3	
0	1 0 2
1	1 0 1

FIG. 8. Operating record of the Turing machine whose serial number is 73133251_8

$$\begin{array}{ccccccccccc}
 & & & & & 0_1 & & & & & \\
 & & & & & 1 & 0_2 & & & & \\
 & & & & & 1_3 & 1 & & & & \\
 & & & & 0_2 & 0 & 1 & & & & \\
 & & & 0_3 & 1 & 0 & 1 & & & & \\
 & & & 1 & 1_2 & 0 & 1 & & & & \\
 & & & 1 & 0 & 0_1 & 1 & & & & \\
 & & & 1 & 0 & 1 & 1_2 & & & & \\
 & & & 1 & 0 & 1 & 0 & 0_1 & & & \\
 & & & 1 & 0 & 1 & 0 & 1 & 0_2 & & \\
 & & & 1 & 0 & 1 & 0 & 1_3 & 1 & & \\
 & & & 1 & 0 & 1 & 0_2 & 0 & 1 & & \\
 & & & 1 & 0 & 1_3 & 1 & 0 & 1 & & \\
 & & & 1 & 0_2 & 0 & 1 & 0 & 1 & & \\
 & & & 1_3 & 1 & 0 & 1 & 0 & 1 & & \\
 & & 0_2 & 0 & 1 & 0 & 1 & 0 & 1 & & \\
 & & 0_2 & 0 & 1 & 0 & 1 & 0 & 1 & & \\
 & 0_3 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & & \\
 & 1 & 1_2 & 0 & 1 & 0 & 1 & 0 & 1 & & \\
 & 1 & 0 & 0_1 & 1 & 0 & 1 & 0 & 1 & & \\
 & 1 & 0 & 1 & 1_2 & 0 & 1 & 0 & 1 & & \\
 & 1 & 0 & 1 & 0 & 0_1 & 1 & 0 & 1 & & \\
 & 1 & 0 & 1 & 0 & 1 & 1_2 & 0 & 1 & & \\
 & 1 & 0 & 1 & 0 & 1 & 0 & 0_1 & 1 & & \\
 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1_2 & & \\
 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0_1 & \\
 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0_2 \\
 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1_3 & 1 \\
 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0_2 & 0 & 1 \\
 & 1 & 0 & 1 & 0 & 1 & 0 & 1_3 & 1 & 0 & 1 \\
 & 1 & 0 & 1 & 0 & 1 & 0_2 & 0 & 1 & 0 & 1 \\
 & 1 & 0 & 1 & 0 & 1_3 & 1 & 0 & 1 & 0 & 1 \\
 & 1 & 0 & 1 & 0_2 & 0 & 1 & 0 & 1 & 0 & 1 \\
 & 1 & 0 & 1_3 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
 & 1 & 0_2 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
 & 1_3 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0_2 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1
\end{array}$$

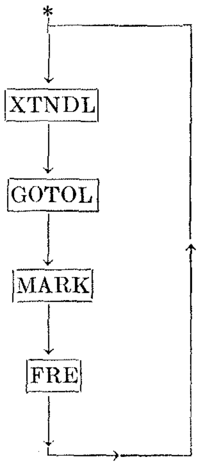
CARD 1	
0	1 1 2
1	1 1 0

CARD 2	
0	1 0 3
1	0 1 1

CARD 3	
0	1 1 2
1	0 0 2

FIG. 9. Operating record of the Turing machine whose serial number is 73132742_8

(FRE for find right end of the string of 1's). This verbal description becomes more intuitive by the use of the following flowchart, which indicated clearly that the machine has entered into a *loop* without exit (the string merely gets longer and longer to the left).



$$\begin{array}{cccccccccc}
 & & & & & & & & 0_1 & \\
 & & & & & & & & 1 & 0_2 \\
 & & & & & & & & 1_3 & \\
 & & & & & & & 0_3 & 1 & \\
 & & & & & & 0_1 & 1 & 1 & \\
 & & & & & & 1 & 1_2 & 1 & \\
 & & & & & & 1 & 1 & 1_2 & \\
 & & & & & & 1 & 1 & 1 & 0_2 \\
 & & & & & & 1 & 1 & 1_3 & \\
 & & & & & & 1 & 1_3 & 1 & \\
 & & & & & & 1_3 & 1 & 1 & \\
 & & & & & 0_3 & 1 & 1 & 1 & \\
 & & & & 0_1 & 1 & 1 & 1 & 1 & \\
 & & & & 1 & 1_2 & 1 & 1 & 1 & \\
 & & & & 1 & 1 & 1_2 & 1 & 1 & \\
 & & & & 1 & 1 & 1 & 1_2 & 1 & \\
 & & & & 1 & 1 & 1 & 1 & 1_2 & \\
 & & & & 1 & 1 & 1 & 1 & 1 & 0_2 \\
 & & & & 1 & 1 & 1 & 1 & 1_3 & \\
 & & & & 1 & 1 & 1 & 1_3 & 1 & \\
 & & & & 1 & 1 & 1_3 & 1 & 1 & \\
 & & & & 1 & 1_3 & 1 & 1 & 1 & \\
 & & & & 1_3 & 1 & 1 & 1 & 1 & \\
 & & & 0_3 & 1 & 1 & 1 & 1 & 1 & \\
 & & 0_1 & 1 & 1 & 1 & 1 & 1 & 1 & \\
 & & 1 & 1_2 & 1 & 1 & 1 & 1 & 1 & \\
 & & 1 & 1 & 1_2 & 1 & 1 & 1 & 1 & \\
 & & 1 & 1 & 1 & 1_2 & 1 & 1 & 1 & \\
 & & 1 & 1 & 1 & 1 & 1_2 & 1 & 1 & \\
 & & 1 & 1 & 1 & 1 & 1 & 1_2 & 1 & \\
 & & 1 & 1 & 1 & 1 & 1 & 1 & 1_2 & \\
 & & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0_2 \\
 & & 1 & 1 & 1 & 1 & 1 & 1 & 1_3 & \\
 & & 1 & 1 & 1 & 1 & 1 & 1_3 & 1 & \\
 & & 1 & 1 & 1 & 1 & 1_3 & 1 & 1 & \\
 & & 1 & 1 & 1 & 1_3 & 1 & 1 & 1 & \\
 & & 1 & 1 & 1_3 & 1 & 1 & 1 & 1 & \\
 & & 1 & 1_3 & 1 & 1 & 1 & 1 & 1 & \\
 & & 1_3 & 1 & 1 & 1 & 1 & 1 & 1 & \\
 & 0_3 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \\
0_1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 &
\end{array}$$

CARD 1	
0	1 1 2
1	1 1 0

CARD 2	
0	0 0 3
1	1 1 2

CARD 3	
0	1 0 1
1	1 0 3

FIG. 10. Operating record of the Turing machine whose serial number is 73037233$_8$

Now let us start this machine, with its card 1, on an all-0 tape. After the second shift, the tape situation 1_3 arises where the length of the string is one. From this point on the sequence of events is shown by the above flowchart and it is clear that this machine is a never-stopper.

As our second illustration, we consider the holdout with the serial number 73033132$_8$ and the following card description.

CARD 1	
0	1 1 2
1	1 1 0

CARD 2	
0	0 0 3
1	0 1 2

CARD 3	
0	0 1 1
1	1 0 2

To come to a stop, this machine must get a tape situation where card 1 scans a 1, $\begin{array}{c|c|c|c|c|c}\hline & & & 1_1 & & \\ \hline\end{array}$. Now card 1 is called only if card 3 scans a 0; and in this case, to get the stop situation, we should have a 1 to the right, $\begin{array}{c|c|c|c|c|c}\hline & 0_3 & 1 & & & \\ \hline\end{array}$. Now this situation cannot occur. Indeed, card 3 is called only if card 2 scans a 0 and as the 0-line of card 2 shows, it overprints the square by a 0 and shifts to the left. Hence card 3 will always scan a square with a 0 to the right. Thus we see that the stop situation is never reached by this machine.

Let us note that the approaches used in these two illustrations involve important ideas ("flowchart" and "back-tracking") of general use in various fields.

IV. *SH*(3) *and Miscellaneous Comments*

In the course of our work described above, several unexpected features turned up. Originally, 3-card Turing machines showing the induction patterns of the "holdouts" were discovered and programs were devised to eliminate them. This approach proved to be difficult and of little use, since only a few could be so eliminated. Hence, if one should attempt to settle the *BB*-4 or the *BB*-5 problem, efficient programs to eliminate the Turing machines showing these patterns must be devised since they will be necessarily too numerous to check by hand. Also, new patterns must show up for increasing card numbers, since we know that $\sum(n)$ is noncomputable [2].

We want to share an experience with the programmers among our readers. In our experiments with the recurrence patterns, a bound of 18 was first set for the shift-number and we were left with 46 holdouts. While operating each of the 46 machines by hand we found some to show recurrence patterns. This caused a brief period of apprehension, well-known to programmers, about the possible presence of some basic error in our program. However, a check showed that all these contrary holdouts showed the recurrence patterns after 18 shifts, with three showing right after 19 shifts! We were gratified that all these were eliminated by increasing the shift bound in the final version of our program to 50.

Concerning the conjecture that $SH(3) = 21$, we note that $SH(3) \leqq SH^*(3) + 2$, where $SH^*(3)$ is the maximum of $s(M)$ for valid *BB*-3 entries M normalized in the manner discussed in Section II. We find from our work that $SH^*(3) = 21$, so that $SH(3) \leqq 23$. However, if there is a valid *BB*-3 entry M with $s(M) \geqq 22$, then upon renumbering the cards of M (readjusting the calling indices and considering a mirror image if necessary), we must have a normalized valid *BB*-3 entry M^* in our four lots with either (i) $s(M^*) = 21$ and at least one of the entries p_{10}, p_{20}, p_{30} equal to zero; or (ii) $s(M^*) = 20$ and at least two of the entries p_{10}, p_{20}, p_{30} equal to zero. An inspection of the print-outs for the 20 and 21 shifters shows that this does not happen (Figure 2), and so $SH(3) = 21$.

A question was raised by some *BB-n* enthusiasts as to whether a maximum scorer in the *BB-n* game (a valid entry in the *BB-n* classification with a score of $\sum(n)$) must always have an unbroken string of 1's in its output tape when it stops; the conjecture being that it must. An inspection of the print-outs for the five 6-scorers shows that this need not be the case (Figure 1), and this question is therefore also settled.

V. *Conclusion*

The reader will surely realize that if one attempts to apply the method described above to the problem *BB*-1963, for example, then difficulties of prohibitive character are bound to arise. In the first place, the number of cases becomes astronomical, and the storage and execution for the computer programs involved will defeat any efforts to use existing computers. Even if we assume that somehow we manage to squeeze through the computer the portion of our approach involving partial recurrence patterns, the number of holdouts may be expected to be enormous. Over and beyond such "physical" difficulties, there is the basic fact of the noncomputability of $\sum(n)$, which implies that no single finite computer program exists that will furnish the value of $\sum(n)$ for every n.

In the absence (at present) of a formal concept of "noncalculability" for individual well-defined integers, it is of course not possible to state in precise form the conjecture that there exist values of n for which $\sum(n)$ is *not* effectively calculable. Several colleagues (including Dr. C. Y. Lee of the Bell Telephone Laboratories) pointed out to us the wide significance of these issues; we hope to return to some of these in a paper now under preparation.

Acknowledgments. The authors wish to thank Dr. C. Y. Lee of the Bell Telephone Laboratories, Incorporated, for many stimulating comments, and Dr. Roy F. Reeves, Director of the Numerical Computation Laboratory, The Ohio State University, for the use of the computing facilities.

RECEIVED MARCH, 1963; REVISED AUGUST, 1964

ADDENDUM

A family of *BB-n* machines that achieve fantastically enormous scores for $n \geqq 8$ has been constructed by Mr. Milton W. Green of Stanford Research Institute [*Proceedings of the 5th Annual Symposium on Switching Circuit Theory and Logical Design*, October 1964, pages 91–94]. Further developments along the lines suggested by the absolutely overwhelming results of Mr. Green will be presented in a forthcoming paper by T. Rado and J. Randels of The Ohio State University.

REFERENCES

1. KLEENE, S. C. *Introduction to Metamathematics*. D. Van Nostrand, Princeton, 1952.
2. RADO, T. On non-computable functions. *Bell Sys. Tech. J. 41*, 3 (May, 1962).
3. ——. *On Non-Computable Functions*. The Bell Telephone System Monograph 4199, 1962.