tains some intuitive moments, but is nevertheless supposed to lead to the simplest possible equation system in all practical cases.

Undoubtedly, there are alternative approaches to the subject of this paper. For instance, the transformation property of canonical memories indicates that the feedback loops can be appropriately cut elsewhere than after the NOT-operators. Even so, this paper is supposed to have given the main limitations and rules for some new circuits, possible to apply whenever a memory function is required.

References

[1] H. Beckwith, "Flip-flop counter has expanded range," *Electronics*, vol. 28, pp. 149–151, January 1955.
[2] R. K. Richards, *Arithmetic Operations in Digital Computers*. New York: Van Nostrand, 1955, pp. 48–49.
[3] S. H. Unger, "A study of asynchronous logical feedback networks," Research Lab. of Electronics, Massachusetts Institute of Technology, Cambridge, Mass., Tech. Rept. 320, June 1957.

On-Line Turing Machine Computations

F. C. HENNIE

Abstract—**This paper investigates 1) the problem of finding lower bounds on the computation times of on-line Turing machines, and 2) the trade-off relationship between computation time and tape dimensionality. It considers problems in which a Turing machine is supplied with a sequence of inputs representing data to be stored on the machine's tape(s), followed by a sequence of inputs requesting the machine to find and examine various portions of the stored data. The approach taken is to assume that the machine has been designed to read in and store data in such a way as to minimize the time required to subsequently locate arbitrary portions of that data. This approach sometimes makes it possible to find good lower bounds on the computation time (number of machine steps) needed to process the portion of the input sequence that calls for the retrieval of data. It is shown that there are some problems in which an increase in tape dimensionality appreciably reduces the computation time needed. But it is already known that increasing the number of a machine's tapes (beyond two) does not appreciably decrease the computation time needed. Thus, tape dimensionality and tape multiplicity are parameters that affect computation time in basically different ways.**

I. Introduction

ONE MEASURE of the complexity of a function is the amount of time required to compute that function with a Turing machine. Since the particular Turing machine model used can have an important effect on this computation time, it is often convenient to distinguish between "off-line" and "on-line" computations. Briefly, in an off-line computation the entire (finite) string of input symbols must be written on one of the machine's tapes prior to the start of the computation, while in an on-line computation the symbols of the input string are presented to the machine sequentially. Certain aspects of off-line machines have been discussed in an earlier paper [1]; the present paper will treat only on-line machines.

An on-line machine starts its computation with all its tapes completely blank and receives its input symbols, one at a time, at a special input terminal. In response to each input symbol the machine is required to produce an appropriate output symbol at a special output terminal. In general, of course, the machine must go through a number of basic steps between the receipt of an input symbol and the production of the corresponding output symbol. However, the $i+1$st input symbol is to be supplied to the machine when, and only when, the machine produces the ith output symbol. Thus, an on-line machine may be thought of as realizing a transformation from input sequences to output sequences in which the ith symbol of the output sequence depends only upon the first i symbols of the input sequence. If the output alphabet of the machine is $\{0, 1\}$, the machine may alternatively be thought of as recognizing the set of input sequences for which the last symbol in the corresponding output sequence is a 1.

At each step in its computation, an on-line machine is able to change the symbols appearing under its reading heads, shift each of its tapes one square in either direction, and change its internal state. The total number of basic steps that a machine must execute in order to produce the output symbols associated with a given input sequence is called the computation time for that input sequence. If $T(n)$ is a function such that the computation time for every input sequence of length n is less than or equal to $T(n)$, then the machine is said to operate within the time bound $T(n)$. The input-output transformation realized by the machine will be said to

Manuscript received June 19, 1965; revised October 11, 1965. The work reported in this paper was supported in part by the Joint Services Electronic Program under Contract DA 36-039-AMC-03200(E), in part by the National Science Foundation under Grant GP-2495, and in part by the National Aeronautics and Space Administration under Grant NsG-496.

The author is with the Department of Electrical Engineering and the Research Laboratory of Electronics, Massachusetts Institute of Technology, Cambridge, Mass.

be "computable within the bound $T(n)$," and the set of sequences recognized by the machine will be said to be "recognizable within the bound $T(n)$."

This paper is concerned with the determination of lower bounds on the computation times of certain input-output transformations, and with the trade-off relation between computation time and the dimensionality of a Turing machine's tapes. Section II considers one-dimensional tapes and describes an input-output transformation for which a good lower bound on computation time can be found. The technique used to establish this bound is applied in Section III to show that increasing tape dimensionality from one to two can sometimes decrease considerably the computation times required by a transformation. This result is generalized in Section IV to show that, for multidimensional tapes, the Hartmanis-Stearns "square law" [2], [3] cannot be significantly improved upon, and that there is a basic difference between the dimensionality and the multiplicity of a Turing machine's tapes.

II. One-Dimensional Tapes

This section investigates the time required to perform a certain input-output transformation on an on-line Turing machine that uses one-dimensional tapes. Informally, this transformation can be described as follows. The Turing machine is first supplied with a sequence of inputs representing data to be stored on the machine's tape(s). This data is encoded in the form of equal-length blocks of 0's and 1's. After a specified number of data blocks have been read in, the machine is presented with a second sequence of inputs, also consisting of equal-length blocks of 0's and 1's. As each of these new blocks is read in, the machine must decide whether or not a matching block appeared in the first part of the input sequence, and to put out a 1 or a 0 accordingly.

Section II-A describes this transformation in more detail and indicates how it can be realized within an upper time bound proportional to n^2. Section II-B derives a lower bound on the computation time of any machine that realizes the transformation. These two bounds are compared and discussed in Section II-C.

A. The Set A

The input sequences associated with the transformation considered in this section are based on the alphabet $\{0, 1, 2\}$, while the output sequences are based on the alphabet $\{0, 1\}$. The machine that performs the transformation is to produce an output of 1 only in response to an input of 2, and then only if the preceding input sequence satisfies each of the following conditions:

a) The preceding sequence consists of a number of identical-length blocks of 0's and 1's, consecutive blocks being separated by single 2's.
b) The total number of blocks received so far exceeds 2^k, where k is the number of 0's and 1's in each block.
c) The pattern of 0's and 1's that appears in the very last block to be received matches the pattern of at least one of the first 2^k blocks.

As an example, consider the input sequence

0 1 2 0 0 2 0 1 2 1 0 2 0 0 2 1 1 2 0 1 2

which consists of seven blocks of length $k=2$. To begin with, the output symbols corresponding to each of the 0's and 1's in this sequence must be 0. Since $2^k=4$, condition b) requires that outputs of 0 should also be produced for each of the first four 2's in the sequence. The fifth 2 terminates a block whose pattern matches the pattern of the second block; thus at this point conditions a), b), and c) are all satisfied and an output of 1 should be produced. On the other hand, the output corresponding to the sixth 2 should be 0, since this 2 terminates a block whose pattern does not appear anywhere within the first four blocks. Finally, an output of 1 should be produced in response to the last 2. The complete input and output sequences follow, along with several other pairs of input and output sequences.

input	01200201210200211201 2	input	0120112012112012
output	00000000000001000001	output	0000000000000000

input	10020012000210021102
output	00000000000000000000

input	1002001200020002011210021002000211120112110210020 1
output	0000000000000000000000000000000000000000000010000000100

While investigating the behavior of machines that realize this particular input-output transformation, we shall be primarily concerned with input sequences that consist of more than 2^k blocks of some common length k, separated by single 2's. The first 2^k blocks in such a sequence will be called the base of the sequence, and the remaining blocks will be called the tail of the sequence. The output symbol produced in response to the 2 that terminates a block in the tail of a sequence will be referred to as the output associated with that block. Finally, we shall denote the set of input sequences for which the corresponding output sequence ends in the symbol 1 as the set A. Thus, for example, the sequence 012002012102002112012 is a member of the set A. Its base consists of the four blocks 012002012102, while its tail consists of the three blocks 002112012. The outputs associated with the three tail blocks are 1, 0, and 1, in order of their occurrence.

In accordance with the usual terminology, a machine that performs the input-output transformation described above is one that recognizes the set A. Such a machine is not hard to design, at least conceptually. As it is presented with the first block of 0's and 1's in an input sequence, the machine marks off the length of this block on one of its tapes. This marked-off segment then serves as a yardstick against which the lengths of the succeeding blocks can be checked. As the blocks in the base portion of the input sequence are read in, the machine checks their length, records them on a second tape, and generates a corresponding sequence of output

0's. Of course, the machine must have some way of determining when all the base blocks have been read in; thus, it must compute the value of 2^k (k being determined by the length of the first block) and keep track of the number of blocks received.

As the 0's and 1's of a tail block are read in, the machine records them on a third tape and generates a corresponding sequence of output 0's. Once the block has been terminated by a 2, the machine starts to compare the newly recorded tail block with each of the previously recorded base blocks. If a match is found, a block output of 1 is produced; otherwise a block output of 0 is produced. Each tail block is treated in the same way.

Now consider the time that such a machine needs to generate the outputs associated with a given input sequence. Evidently the longest computation time will result if all the blocks have the same length, for as soon as two different length blocks have been received, the generation of all succeeding outputs becomes trivial. Next recall that 2^k is a real-time countable function [4]. Therefore, the blocks in the base of an input sequence can be checked for length, recorded, and counted in real time. That is, the outputs associated with the base of the sequence can be generated as fast as the inputs are supplied. Therefore, in determining the longest computation time, we need consider only input sequences containing more than 2^k blocks. Finally, note that the outputs associated with the 0's and 1's of the tail blocks can be produced as soon as the corresponding input symbols are received; the only "difficult" outputs are the block outputs.

In order to compare a new tail block with one of the recorded base blocks, the machine must scan through the two recorded patterns, matching them symbol by symbol. If the length of each block is k, the time required to do this is just k time units. However, a tail block may have to be compared with all of the recorded base blocks, and the reading head that scans the tail pattern must be returned to the beginning of that pattern between consecutive comparisons. Therefore, the time required to match one tail pattern against all the different base patterns might be as large as $2k2^k$. To this time we must add the time needed to return the appropriate reading head to the beginning of the list of base blocks, so that the machine will be in a position to check the next tail block. Thus, we should allow a total of $2k2^k+(k+1)2^k<3(k+1)2^k$ time units for the processing of each tail block.

If the number of tail blocks is x, the time required to record the base portion of an input sequence and to check each of the tail blocks is at most

$$T = (k + 1)2^k + 3x(k + 1)2^k.$$

On the other hand, the total number of symbols in the input sequence is

$$n = (k + 1)2^k + x(k + 1)$$

and

$$n^2 = (k + 1)^2 2^{2k} + 2x(k + 1)^2 2^k + x^2(k + 1)^2 > (k + 1)2^k + 3x(k + 1)2^k = T.$$

Therefore, the number of time units needed to process an input sequence of length n is always less than n^2, and the set A can be recognized within the time bound $T(n)=n^2$.

(More detailed calculation shows that A can be recognized within the bound $n^2/\log_2 n$, but the n^2 bound is sufficient for present purposes.)

B. A Lower Bound on the Recognition Time of Set A

The purpose of this section is to derive a lower bound on the computation time required to recognize the set A. This will be done by showing that each machine that recognizes A must spend some minimum amount of time on the computations that it performs on certain input sequences. Of course, the exact amount of time required will depend upon the parameters of the machine—the number of tapes it has, the number of tape symbols it uses, and so on. On the other hand, we know from the preceding section that the set A can certainly be recognized within the time bound $T(n)=n^2$, so that it will be sufficient to consider only machines that operate within this bound. Therefore, let us suppose that we are given a certain on-line machine M that

a) recognizes the set A within the time bound $T(n)=n^2$, and
b) has t tapes, s tape symbols, and Q internal states.

We shall prove that the computations of this machine cannot all be performed within the time bound

$$\frac{Cn^2}{t \log s \log^2 n},$$

where C is an appropriate constant and n is the length of an input sequence.

In proving this result, it is convenient to use a generalization of the idea of "state." Let the *configuration* of a Turing machine at a given time denote

a) the current internal state of the machine,
b) the symbol patterns that currently appear on the machine's tapes, and
c) the positions of these patterns with respect to the machine's reading heads.

Thus, the configuration of a machine at a given time uniquely determines how that machine will react to a future input sequence. The *extent* of a configuration is the maximum distance between the left- and right-most nonblank symbols in any one of the tape patterns. Note that if the machine has been in operation for t time units, the extent of its configuration cannot exceed t.

In order to find a lower bound on the computation time of the given machine M, we shall consider only

the computations that M performs on certain restricted sets of input sequences, denoted A_1, A_2, A_3, etc. Each member of the set A_k contains exactly 2^{k+1} blocks of length k, separated by single 2's. That is, each member of A_k consists of a base segment of 2^k blocks, followed by a tail segment of 2^k blocks. No restrictions are placed on the patterns of 0's and 1's that appear in either the base blocks or the tail blocks. For example, the sequence 012002002102012102102012 belongs to the set A_2.

We now wish to find a lower bound on the maximum time that M might have to spend processing an individual member of the set A_k, in terms of k, t, s, and Q. Each computation that M performs on a member of A_k can be broken up into two parts: that spent processing the base of the input, and that spent processing the tail. We shall ignore the details of the first part and concentrate on the configuration in which it leaves the machine. Thus, we may think of the first part of M's computation as the act of mapping a base segment of a member of A_k into a machine configuration—that configuration in which M begins its computation on the tail portion of the input sequence.

The particular mapping from base segments to machine configurations that M uses strongly influences the time that M must spend on the tail portions of its computations. Thus, one way of minimizing the total computation time for the members of A_k is to "optimize" the mapping; i.e., select a mapping for which a suitably designed machine will have to spend as little time as possible on the tail portions of its computations. Of course, the mapping itself must be one that can be carried out relatively quickly, certainly within the bound $T(n)=n^2$. For present purposes, though, it is more convenient to consider mappings from a much broader class, completely ignoring the time that would be required for a Turing machine to carry out the mapping. Choosing an optimum mapping from the larger class will lead to a minimum computation time for the tail portions that is at least as small as that obtained by choosing a mapping from the smaller class of "quickly realizable" mappings.

Now consider the set of mappings from base segments of A_k to machine configurations such that:

1) Each configuration produced by the mapping is compatible with M; that is, it involves t tapes, the s tape symbols used by M, and the Q internal states used by M.
2) The extent of each configuration produced by the mapping is at most $4(k+1)^2\ 2^{2k}$, the maximum extent of a configuration that M might assume in the course of the computation that it performs on one of the members of A_k.
3) Each configuration produced by the mapping is one for which the given machine M would operate "correctly" on any single tail block. That is, suppose that M is placed in the configuration corresponding to some arbitrarily chosen base segment from A_k and supplied with a single arbitrarily chosen tail block of length k. M must then produce k outputs of 0, followed by a 1 or a 0 according as the given tail pattern does or does not appear within the given base segment.

The time that M needs to produce the $k+1$ outputs indicated in 3) depends upon the particular base segment and tail block chosen, as well as upon the particular mapping. The maximum such time for any base segment from A_k and any single tail block of length k will be called the *checking time* associated with a given mapping. From the set of mappings defined, select one having the smallest checking time; let this mapping be denoted F_k and the corresponding checking time be denoted w_k. Note that such a mapping F_k certainly exists. Only a finite number of mappings satisfy 1) and 2), and only a finite number of input sequences need be considered in testing 3) and in determining the minimum checking time.

The next step is to get a lower bound on w_k. This is given by

Lemma 1[1]

$$w_k > \frac{2^k - \log Q - t \log s - 1}{2t \log s}.$$

Proof: First note that the number of machine configurations that differ either in internal state or in the tape patterns that appear within x squares on either side of the machine's reading heads is

$$Qs^{t(2x+1)}$$

where the quantity $t(2x+1)$ represents the total number of tape squares involved. In particular, if we let

$$x_k = \frac{2^k - \log Q - t \log s - 1}{2t \log s}$$

then

$$Qs^{t(2x_k+1)} = Qs^{(2k-\log Q-1)/\log s} = 2^{2k-1} < 2^{2k} - 1.$$

Now arrange the base segments of A_k in classes according to the patterns of 0's and 1's that appear in their blocks. Since there are 2^k distinct patterns of length k, the number of such classes is $2^{2^k}-1$. Pick a representative member of each class and determine the machine configuration into which it is mapped by F_k. At least two of these representative base segments must map into configurations that are identical with respect to internal state and with respect to the tape patterns that appear within x_k squares of the reading heads, since the number of configurations that differ in these respects is less than $2^{2^k}-1$. Let these two base segments be denoted σ_1 and σ_2.

[1] In this paper all logarithms will be taken to the base 2.

There must be at least one pattern of 0's and 1's that appears in one of the segments σ_1 and σ_2 but not in the other. Consider what will happen if M is placed in the configuration corresponding to either σ_1 or σ_2 and supplied with a single tail block containing the critical pattern. In order for M to produce the correct output for this tail block, it must determine whether its configuration corresponds to σ_1 or to σ_2. But such a determination necessarily entails examining at least one tape symbol that is located more than x_k squares away from a reading head. Since the time required for M to bring such a symbol under the appropriate reading head exceeds x_k, we have established that $w_k > x_k$.

Q.E.D.

Remember that the mapping of base segments into machine configurations that M itself performs is included among the mappings from which F_k was selected. But by definition, F_k is a mapping for which the maximum amount of time that M must spend on its first tail block is as small as possible. Thus, when M uses its own mapping to prepare itself for examining tail blocks, it cannot have a smaller "worst-case" computation time for the first tail block than that associated with F_k. In other words, there must be some input sequence in the set A_k which, when applied to M, causes M to spend at least w_k time units on the first tail block.

Intuitively, it seems that the process of checking one block in the tail of a sequence is essentially independent of the process of checking the next block, and that the worst-case computation time should be the same regardless of the position of the block within the tail. In order to justify this feeling, however, we must prove that in the process of checking the blocks near the beginning of the tail, M cannot rearrange its records in such a way as to speed up the checking of later blocks. The basis of the argument will be that if such a rearrangement were possible, it would be possible to find a mapping that is "better" than F_k. (Readers who prefer to trust their intuition on this point may simply note the statement of Lemma 2 which follows and then skip to Theorem 1.)

The proof will be facilitated if the members of A_k are displayed in the form of a rooted tree, with one path through the tree for each member of A_k. Since every sequence in the set A_k consists of 2^{k+1} blocks, each of which may contain any one of 2^k patterns, the appropriate tree consists of $2^{k+1}+1$ levels of nodes with 2^k branches emanating from each nonterminal node. Figure 1 shows the tree for the case in which $k=1$, when there are exactly two possible block patterns (0 and 1), hence two branches out of every nonterminal node. It is convenient to separate those branches in the tree that correspond to base blocks from those that correspond to tail blocks, as is done by the broken vertical line in Fig. 1. The subtrees that stem from nodes on this separating line will be called the *clusters* of the tree; each cluster corresponds to exactly one of the 2^k possible base segments.

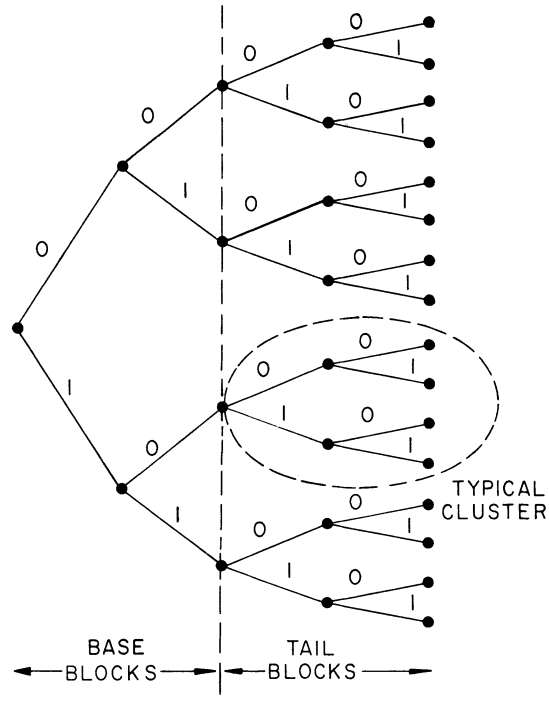

Fig. 1. The tree for the set A_1.

Lemma 2

There is at least one input sequence in the set A_k which, when applied to the given machine M, causes M to spend at least $2^k w_k$ time units on the tail portion of the sequence.

Proof: Consider each member of A_k in turn, apply it to M, and note how much time M spends on each of the blocks in the tail of the sequence. If M spends less than w_k time units on a given block, label the corresponding branch in the tree with a minus sign; otherwise label the corresponding branch with a plus sign. If all the branches leaving a particular node are labeled with a minus sign, call that node a *minus node;* otherwise call it a *plus node.*

We must first show that it is not possible for every cluster to contain a minus node. Note that each node in a cluster corresponds to a configuration of M; in particular, a configuration in which M is about to receive the symbols of a tail block and to compute the appropriate block output. If all the branches leading out of some node in a cluster are labeled with a minus sign, the configuration corresponding to that node must be one for which M will spend less than w_k time units on the next tail block, regardless of the pattern contained in that block.

Thus, if every cluster contains a minus node, it must be possible to define a new mapping F_k' that has a smaller checking time than F_k', as follows. Consider each base segment in turn, note the cluster in the tree to which it corresponds, and find a sequence of tail blocks that leads to a minus node in that cluster. Then apply the given base segment to M, follow it by the appropriate sequence of tail blocks, and note the configuration

in which M is left. Then let this configuration be the one assigned to the given base segment by the new mapping F_k'.

The mapping F_k' certainly satisfies conditions 1) and 3). It also satisfies condition 2), since the configurations that it produces are configurations that M reaches in the course of its computation on an input sequence of length $2(k+1)2^k$ or less. Because M operates within the time bound n^2, the extent of such a configuration cannot exceed $4(k+1)^2 2^{2k}$. Thus, F_k' belongs to the set of mappings from which F_k was chosen. The checking time of F_k' is that associated with a minus node, and hence is by definition less than w_k, the checking time of F_k. But this contradicts the fact that F_k was chosen to have the smallest possible checking time. Therefore, the assumption that every cluster can have a minus node is untenable, and in fact some cluster must have only plus nodes.

But if a given cluster has only plus nodes, every node in the cluster must have at least one outgoing branch that is labeled with a plus sign. Therefore, there must be at least one path through the cluster that involves only plus-labeled branches. Select the member of A_k whose base segment corresponds to the path from the root of the tree to the cluster in question, and whose tail segment corresponds to the path through the cluster that contains only plus-labeled branches. This member of A_k is one for which M spends at least w_k time units on each tail block and, hence, a total of at least $2^k w_k$ time units on the entire tail segment.

Q.E.D.

The results of Lemmas 1 and 2 can now be combined to give a lower bound on the computation time required to recognize the set A.

Theorem 1

Let M be an on-line machine that has t tapes and s tape symbols and recognizes the set A. Let $T(n)$ denote the maximum amount of time that M spends on any input sequence of length n. Then $T(n)$ exceeds

$$\frac{n^2}{48t \log s \log^2 n}$$

for sufficiently large values of n.

Proof: We need only consider the case in which M operates within the bound n^2, for otherwise the statement of the Theorem would certainly be true. Let n_k denote the common length of the sequences in the set A_k, and let T_k denote the maximum time that M must spend on a sequence from A_k. Then for any given value of n, choose k so that $n_k \leq n < n_{k+1}$. Since the computation time required by M is certainly monotonic in n, $T(n) \geq T_k$, while from Lemmas 2 and 1 we have that

$$T_k \geq 2^k w_k > \frac{2^k(2^k - \log Q - t \log s - 1)}{2t \log s}.$$

Thus, if n, and hence k, is sufficiently large,

$$T_k > \frac{2^{2k}}{3t \log s}.$$

Now note that $n_{k+1} = 2(k+2)2^{k+1}$ and, thus, $\log n_{k+1} > k+2$. Therefore,

$$T(n) \geq T_k > \left\lceil \frac{4(k+2)^2 2^2}{4(k+2)^2 2^2} \right\rceil \frac{2^{2k}}{3t \log s}$$

$$> \frac{{n_{k+1}}^2}{48t \log s \log^2 n_{k+1}}.$$

As long as $n_{k+1} > n > e$, the ratio $n_{k+1}/\log n_{k+1}$ will exceed the ratio $n/\log n$. Therefore,

$$T(n) > \frac{n^2}{48t \log s \log^2 n}.$$

Q.E.D.

C. Discussion

Theorem 1 gives a lower bound on the rate at which the recognition time for the set A must grow as a function of n, t, and s. To see whether this growth rate is reasonable, recall that the number b of essentially different base portions of the members of set A_k is $2^{2^k} - 1$, while the total length n of each member of A_k is $(k+1)2^{k+1}$. The number of bits needed to specify which of the b essentially different base segments has occurred is approximately 2^k, which in turn is approximately $n/\log n$. But if the machine in question uses t tapes and s tape symbols, the maximum number of squares per tape needed to record this specification is approximately $n/t \log s \log n$. This number of squares represents the maximum number of tape shifts that might have to be made in order to check a single tail block.

The total number of tail blocks in a member of A_k is 2^k, or approximately $n/\log n$. Theorem 1 states that the total computation time must, in the worst case, be proportional to the product of the number of tail blocks to be checked and the maximum number of squares per tape needed to record all the essential information about the base segment. Thus, it is not possible to improve very much upon the rather obvious approach of recording, as compactly as possible, all the patterns that appear in the base segment, and then searching through this record for each new tail block that is received. The statement of Theorem 1 is perhaps not so surprising as the fact that it can be proved.

The growth rate indicated in Theorem 1 is less than the growth rate obtained in Section II-A by a factor of $1/\log^2 n$. This discrepancy arises because the machine described in Section II-A uses a somewhat inefficient method of recording base patterns. Instead of simply listing the patterns one after another, it is more economical to record them in "factored" form, extracting common initial segments from two or more patterns

whenever possible. Thus, the pair of patterns 0000 and 0001 might be recorded as 000(0+1), and the set {0000, 0001, 0101, 1000, 1010} might be recorded as 0(00(0+1)+101)+10(00+10). Using such a scheme, it is possible to design a machine that recognizes the set A and operates within a time proportional to $n^2/t \log s \log^2 n$, for sufficiently large values of t and s.[2] Thus, the "optimum" computation time of Theorem 1 is within a multiplicative constant of being the best attainable time.

The fact that the "search time" needed to check a single tail block has such a strong influence on the recognition time for the set A suggests consideration of other computing models in which this search time can be reduced. One possibility, to be investigated in Section III, is to provide the Turing machine with two-dimensional tapes. Since the storage of a given number of bits of data can be accomplished within a much smaller "diameter" on a two-dimensional tape than on a one-dimensional tape, the search time can be made much smaller by using a two-dimensional tape. Decreasing the search time should in turn reduce the rate of growth of total computation time.

III. Two-Dimensional Tapes

In this section we shall consider on-line Turing machines that are supplied with one or more infinite, two-dimensional "tapes." For simplicity, such a machine will be called a two-dimensional machine. At each step in its computation, a *two-dimensional machine* can write new symbols in the currently scanned squares, shift each of its tapes left, right, up, or down (or not at all), and change its internal state. As with all on-line machines, a two-dimensional on-line machine starts its computations with its tapes completely blank. Input sequences are then applied, and output sequences generated, exactly as in the case of one-dimensional tapes. In particular, the definition of computation time remains unchanged.

As suggested in the preceding section, the use of two-dimensional tapes can decrease the amount of time needed to recognize the set A. In fact, it is possible to design a two-dimensional machine that recognizes A within a time proportional to $n^{3/2}$, as opposed to the n^2 growth rate required by a one-dimensional machine. Thus, the time required by the two-dimensional realization is proportional to the $\frac{3}{4}$ power of the best time attainable by a one-dimensional realization. Rather than pursuing the two-dimensional realization of A further, let us turn instead to a recognition problem for which the disparity between the one- and two-dimensional computation times is even greater than that for set A.

The set of input sequences to be investigated in this section is best described in terms of the behavior of a two-dimensional machine that recognizes it. This machine, called M_2, operates on the input symbols U, D, L, R, W, and E, which stand for the commands Up, Down, Left, Right, Write, and Examine, respectively. The machine has a single two-dimensional tape, and uses the tape symbols 0 and 1. At the beginning of a computation every tape square contains a 0.

Upon receiving one of the input symbols U, D, L, or R, machine M_2 shifts its reading head one square in the direction indicated—that is, it shifts its tape one square in the opposite direction—and produces an output of 0. Upon receiving the input symbol W, the machine does not move its tape, but instead writes the symbol "1" in the currently scanned square and produces an output of 0. Upon receiving the input symbol E, the machine does nothing to its tape, but produces an output of 0 or 1 according to whether the currently scanned square contains a 0 or a 1. For example, if M_2 is started with a 0 in every tape square and is presented with the input sequence $R\ W\ U\ R\ W\ D\ W\ L\ E\ D\ E\ R\ U\ E\ R$, it will produce the output sequence 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0.

The set of input sequences that is recognized by M_2 will be called the set B. By virtue of its definition, B can be recognized by a two-dimensional machine within the time bound $T(n)=n$. According to Hartmanis and Stearns [2], [3], going from a two-dimensional realization to a one-dimensional realization requires at most a squaring of the computation time. Thus, it is certainly possible to design a one-dimensional machine that recognizes the set B within the time bound $T(n)=n^2$. We now wish to find a lower bound on the computation time of a one-dimensional machine that recognizes B.

For this purpose, it is convenient to consider only certain sets of input sequences. These sets, denoted B_1, B_2, B_3, etc., are defined in terms of the effect of their members on the two-dimensional machine M_2. Each member of the set B_k has two parts: a *writing segment* followed by a *reading segment*. The writing segment is a sequence consisting of U's, D's, L's, R's, and W's. It causes M_2 to move its reading head along a "spiral" path, like that shown in Fig. 2, writing 1's in some arbitrarily chosen subset of the squares it visits. As soon as a total of k^2 squares have been visited (but not necessarily written in) the reading head is returned via as direct a route as possible to the starting point, or *home square*.

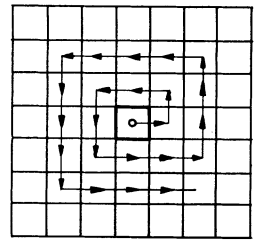

Fig. 2. Path followed during writing segment.

[2] Rough estimates show that the constant of proportionality need not be greater than about 20 or 30, although it is undoubtedly possible to do much better.

The reading segment of each member of B_k is divided into k consecutive portions, called "interrogations." Each interrogation in turn consists of three parts:

1) A sequence of U's, D's, L's, and R's that causes M_2 to move its reading head as directly as possible from the home square to one of the squares visited by the writing segment;
2) A single E, which causes M_2 to produce as an output the symbol written in the square in question;
3) A sequence of U's, D's, L's, and R's that causes M_2 to move its reading head as directly as possible back to the home square.

Thus, the interrogating sequence $U\ U\ U\ R\ R\ R\ R\ E\ D\ D\ D\ L\ L\ L\ L$ causes M_2 to examine the contents of the square that is located three squares above and four squares to the right of the home square. The precise path followed to and from the interrogated square is unimportant. Any one of the k^2 allowable squares may be examined by any given interrogation.

The number of input symbols needed to move the reading head of M_2 along the spiral path is k^2-1, the number needed to write 1's in some subset of the squares visited is at most another k^2, and the number needed to return the reading head to the home square is $k-1$. Thus, the total number of symbols in the writing segment of a member of B_k is less than $3k^2$. The number of input symbols needed for a single interrogation is at most $2k-1$, since any square that was visited in the writing portion can be reached from the home square in $k-1$ or fewer moves. Thus, the number of symbols in the reading portion of a member of B_k is less than $2k^2$, and the total number of symbols in a member of B_k is less than $5k^2$.

Now suppose that M_1 is a one-dimensional machine that recognizes the set B. How much computation time does M_1 require, assuming that it has t tapes, s tape symbols, and Q internal states? Since it is known to be possible to design a one-dimensional machine that recognizes B within the time bound n^2, we may as well assume that M_1 operates within this bound. Then an argument very close to that of Section II-B can be used to show that there is some member of set B_k upon which M_1 must spend on the order of k^3 time units. This implies that M_1 requires a computation time proportional to $n^{3/2}$.

First consider mappings from the writing segment of the members of B_k to one-dimensional machine configurations. In particular, consider those mappings that meet each of the following constraints:

4) Each machine configuration produced by the mapping is compatible with M_1. That is, it involves t tapes, the s tape symbols used by M_1, and the Q internal states of M_1.
5) The extent of each configuration is at most $25k^4$, the maximum extent of a configuration that M_1 might assume in the course of the computations that it performs on the members of B_k.
6) Each configuration is one for which M_1 would operate "correctly" for any single interrogating sequence. That is, suppose that M_1 is placed in the configuration assigned to a particular writing segment from B_k and supplied with one arbitrarily chosen interrogating sequence. In response to the "E" of this sequence, M_1 must produce an output of 1 or 0 according as a 1 would or would not have been written in the examined square by the given writing segment.

From these mappings, select one for which the maximum time that M_1 might have to spend on the arbitrary interrogation described in 6) is as small as possible. Call this "best" mapping G_k, and call the corresponding maximum "interrogation time" v_k.

Paralleling the argument of Lemma 1, we can now show that

$$v_k > y_k = \frac{k^2 - t \log s - \log Q - 1}{2t \log s}.$$

To do this, note that the number of configurations that differ with respect either to internal state or to the symbols that appear within y_k squares on either side of M_1's various reading heads is

$$Qs^{t(2y_k+1)} = 2^{k^2-1}.$$

On the other hand, there are 2^{k^2} different patterns of 0's and 1's that can be left on M_2's tape by a writing segment from the set B_k. Thus, G_k must map some two writing segments that produce different patterns on M_2's tape into configurations of M_1 that agree with respect to internal state and with respect to the tape symbols that appear within y_k squares of the reading heads. Therefore, it must be possible to choose a particular writing segment and a first interrogating segment for which M_1 must examine some tape symbol that is, at the beginning of the interrogation, more than y_k squares away from a reading head. Since M_1 must spend at least y_k+1 time units to reach such a symbol, the maximum interrogation time for the mapping G_k does exceed y_k.

The next step is to show, by paralleling the argument of Lemma 2, that there is some member of B_k for which M_1 must spend at least kv_k time units generating the output symbols associated with the reading segment of the sequence. Again it is convenient to represent the members of B_k in the form of a tree. This tree has 2^{k^2} "main branches," one for each of the different writing segments of the members of B_k. At the end of each main branch is a subtree, or "cluster," containing k levels of branches. Emanating from each node in each cluster are k^2 branches, one for each of the different interrogations that might be made in each part of the reading segment of a member of B_k.

Following the argument of Lemma 2, apply each member of B_k to the machine M_1 and determine the amount of time that M_1 spends on each interrogating

sequence in the reading segment. Then label each branch in each cluster of the tree with a plus or minus sign according as M_1 does or does not take at least v_k time units for the corresponding interrogation. If in each cluster there were at least one node from which only minus branches emanated, it would be possible to find a mapping G_k' which satisfied conditions 4), 5), and 6) and whose associated maximum interrogation time was less than v_k. But since the choice of G_k precludes the existence of such a G_k', there must be at least one cluster that contains only "plus" nodes. Therefore, there must be at least one member of B_k for which M_1 spends at least kv_k time units on the reading segment.

Finally, paralleling the argument of Theorem 1 gives a lower bound $T_1(n)$ on the computation time required by M_1. Recall that the number of symbols in each member of B_k is less than $5k^2$. Then, given some arbitrary value of n, choose k so that

$$5k^2 \leq n < 5(k+1)^2$$

and hence

$$n^{3/2} < 100k^3.$$

Since $T_1(n)$ is certainly monotone increasing, $T_1(n)$ is at least as large as the maximum time that M_1 must spend on the reading segments of the members of B_k. Thus,

$$T_1(n) \geq kv_k > \frac{k(k^2 - t\log s - \log Q - 1)}{2t\log s}.$$

If n, and hence k, is sufficiently large,

$$T_1(n) > \frac{k^3}{3t\log s} > \frac{n^{3/2}}{300t\log s}.$$

Thus, $T_1(n)$ must grow in proportion to $n^{3/2}$. On the other hand, the time required to recognize B with a two-dimensional machine, denoted by $T_2(n)$, need grow no faster than n. Expressing $T_1(n)$ in terms of $T_2(n)$ gives

$$T_1(n) > \alpha(T_2(n))^{3/2}$$

where α is a constant for any particular values of t and s. In other words, at least for the problem of recognizing the set B, reducing the tape dimensionality from two to one requires raising the growth rate of the computation time to at least the 3/2 power.

IV. Dimensionality vs. Speed

A. Generalizations of the Set B

The techniques of the preceding sections can also be applied to machines with three- or higher-dimensional tapes. In particular, the definition of set B can be extended to larger numbers of dimensions. Paralleling the definition of the machine M_2, consider an m-dimensional machine M_m that is capable of moving its tapes in all of the $2m$ possible directions, writing 1's in the "squares" that it visits, and examining the symbols that it scans. Let this machine be supplied with input sequences from the alphabet $\{d_{1+}, d_{1-}, \cdots, d_{m+}, d_{m-}, W, E\}$. The symbols d_{i+} and d_{i-} cause the machine to shift its tape parallel to the i-axis, while W and E, respectively, cause it to write a 1 or to examine the square that it currently scans for the presence of a 1. As before, the machine is to produce an output of 1 only in response to an input of E, and then only if the square currently scanned contains a 1.

Let the set of input sequences recognized by M_m be denoted W^m. (Thus the set B is identical to the set W^2.) The set W^m can certainly be recognized in real time by an m-dimensional machine, but how much time is required to recognize it with a machine having tapes of fewer dimensions? Let $T_j{}^m(n)$ denote the smallest computation time with which W^m can be recognized by a j-dimensional machine. Then the methods of the preceding section can be used to show that

$$7)\quad T_j{}^m(n) \geq C(T_m{}^m(n))^{2-j/m}$$

where C is a function only of the number of tapes and tape symbols used by the j-dimensional machine.

The argument leading to this result can be summarized as follows. First define a collection of sets $W_1{}^m, W_2{}^m, \cdots, W_k{}^m, \cdots$, to correspond to the sets B_k of Section III. Each member of $W_k{}^m$ begins with a writing segment that causes M_m to visit each square in an m-dimensional hypercube of side k, and to write 1's in various of these squares. This writing segment is followed by a reading segment in which exactly k^{m-1} of the squares in the hypercube are interrogated.

Next consider the application of the sequences in $W_k{}^m$ to some j-dimensional machine M_j that recognizes W^m. The arguments of Section III imply that the worst-case computation time for a single interrogation is at least proportional to k^{m-j+1}, and that the worst-case computation time for all k^{m-1} interrogations is at least proportional to k^{2m-j}. On the other hand, the time required to process the members of $W_k{}^m$ on M_m is at most proportional to k^m. Thus, the time required by the j-dimensional machine is at least proportional to the $(2m-j)/m$ power of the time required by the m-dimensional machine.

Modifications of the machines M_m lead to definitions of sets for which the lower time bounds $T_j{}^m$ given by 7) are attainable. Thus, there are recognition problems for which necessary and sufficient computation times in various dimensions can be determined within multiplicative constants. Furthermore, the growth rate of the computation time of the modified form of set W^m increases with each decrease in tape dimensionality from m to 1.

B. Conclusions

This paper serves two purposes. The first is to present a technique for finding lower bounds on the computation times of certain on-line recognition problems. The problems for which this technique seems to be most use-

ful are those in which data are presented to the machine by one part of the input sequence and later "requested" of the machine by another part of the input sequence. Thus, the machine must store, or record, the incoming data in such a way that it can be found quickly upon demand.

The technique considered here largely ignores the problem of recording input data and concentrates instead on the problem of retrieving previously stored data. In doing this, it is assumed that the data has been recorded in the best possible way, i.e., so as to minimize the time required to retrieve arbitrary pieces of data later on. Retrieval times derived using this assumption will certainly not exceed the retrieval times that can be attained when other recording schemes are used. Thus, we are willing to allow as much time as necessary to be spent on "preprocessing" the stored data if we can thereby bound the subsequent retrieval time.

Note that the assumption of optimum preprocessing really enables us to do two things. First, it leads to a lower bound on the times required to retrieve one item of stored information (e.g., one pattern in the case of set A, or the contents of one tape square in the case of set B). This bound is based on the number of distinct past histories that must be "remembered" and on the number of tape squares needed to represent them. Second, it leads to the conclusion that the retrieval of successive items of information really constitutes independent problems, and that the worst-case computation times are the same for each successive inquiry.

Thus, in addition to providing lower bounds on the computation times needed to recognize the sets A and B, the technique of Sections II and III may prove useful for other problems with the same general characteristics. In summary, these characteristics are: 1) on-line computation, 2) the necessity of retrieving arbitrary items of data from an arbitrarily long list of items previously read in, and 3) independence between successive requested items

The second purpose of this paper has been to investigate the relationship between tape dimensionality and computation time. The work of Hartmanis and Stearns [2], [3] has shown that reducing the dimensionality of a Turing machine's tape need never require more than squaring the original computation time. It has not been clear, however, whether such a decrease in dimensionality really requires as much as a squaring of the computation time. This question is especially pertinent since it has been shown that reducing the *number* of a Turing machine's tapes does *not* require an appreciable increase in computation time, provided the tapes are one dimensional and provided the number of tapes is not decreased below two [5].

It is still not clear whether the squaring is necessary. However, the results of Sections III and IV-A do show that an appreciable increase in computation time is necessary, at least for some problems, when the number of dimensions is reduced. In particular, if the number of dimensions is reduced from m to j, the computation time may have to be raised to the $(2m-j)/m$ power. Thus, if the number of dimensions is greatly reduced, the computation time must be raised to a power that is close to two.

Looking at the matter from a different point of view, the addition of each new dimension buys something in terms of reduced computation time, at least for some problems. For the examples discussed here, this gain decreases with each additional dimension, but it is not clear whether this is necessarily the case. What is clear is that tape dimensionality and tape multiplicity affect computation time in different ways. Whereas increasing the number of dimensions can appreciably reduce the rate at which the computation time must grow with n, increasing the number of (one-dimensional) tapes beyond two does not appreciably change the rate of growth of computation time.

References

[1] F. C. Hennie, "One-tape, off-line Turing machine computations," *Information and Control*, vol. 8, December 1965.

[2] J. Hartmanis and R. E. Stearns, "On the computational complexity of algorithms," *Trans. Amer. Math. Soc.*, vol. 117, pp. 285–306, May 1965.

[3] J. Hartmanis and R. E. Stearns, "Computational complexity of recursive sequences," *1964 Proc. of the Fifth Annual Symp. on Switching Circuit Theory and Logical Design*, p. 82.

[4] H. Yamada, "Real-time computation and recursive functions not real-time computable," *IRE Trans. on Electronic Computers*, vol. EC-11, pp. 753–760, December 1962.

[5] F. Hennie and R. E. Stearns, "Two-tape simulation of multitape Turing machines," to be published.