THE RECOGNITION PROBLEM FOR THE SET OF PERFECT SQUARES

Alan Cobham

International Business Machines Corp.
Thomas J. Watson Research Center
Yorktown Heights, New York

Summary

Lower bounds on the capacity and on the product of capacity and computation time are obtained for machines which recognize the set of squares. The bound on capacity is approached to within a factor of four by a specific machine which carries out a test based on the fact that every non-square is a quadratic non-residue of some rational prime. A machine which carries out a test based on the standard root-extraction algorithm is substantially less efficient in this respect. For neither machine is the bound on the capacity-time product closely approached.

I. Introduction

The familiar algorithm for extracting square roots provides one means for determining whether a given integer is or is not a perfect square. An alternative test can be based on the fact that every non-square integer is a quadratic non-residue of some rational prime. It is possible, independently of any hypothesis, to obtain a bound on the magnitude of the primes for which this test must be run as a function of the magnitude of the number tested. It appears, however, that to obtain a bound sufficiently small to make the test practical an additional assumption is needed, specifically, a mild form of the extended Riemann hypothesis. Accepting the latter, we find the test is economical in its storage requirements; it can, in fact, be implemented on an off-line Turing machine in such a way that, insofar as utilization of memory capacity goes, it is asymptotically at least within a factor of four of being the best possible square-recognizing device. In contrast, capacity requirements for a machine which executes the root-extraction test are exponentially greater. The residue test is also superior in average execution time, though in maximum execution times the root-extraction test has the edge.

In the next two sections we establish lower bounds on the capacity and on the capacity-time product of machines which recognize the set of perfect squares. The remaining sections are devoted to a brief analysis of a specific off-line Turing machine which implements the root-extraction test, and a more detailed analysis of a machine which implements the residue test. A few comments and conjectures are collected together in the final section.

Although the machines discussed in the later sections are of a special type, the preliminary bounding theorems apply more widely. In deriving these we employ the general notion of a recognition machine by which we mean any device consisting of a single, two-way infinite input tape scannable, one square at a time, in either direction and an associated black box capable of assuming any of a potentially infinite set of internal configurations. On the input tape is written initially the base b ($b \geq 2$) representation of a positive integer with all remaining tape squares blank. The machine is started in a designated one of its internal configurations with the scanning head positioned on the left-most symbol of the input number. At each step of operation a new symbol (from any, possibly infinite, alphabet) may be written on the square under scan, the scanning head moved one square in either direction and the internal configuration changed, the nature of these acts being determined solely by the current configuration and symbol under scan. We assume a configuration which causes a stop is always reached at which point this configuration alone determines whether the input is accepted or rejected. The set of inputs (tapes or the correlated numbers) accepted is the set recognized by the machine.

We shall sometimes refer to such machines as read/write machines in order to emphasize the distinction between these and a more restricted type, the read-only recognition machines, which do not print on, or otherwise alter, their input tape during the course of a computation, but which operate otherwise as the machines defined above.

With each recognition machine we associate two functions, C and M, each defined over the positive integers. C(N) has as value the base two logarithm of the cumulative number of distinct internal configurations reached during the course of all computations having input not exceeding N. M(N) has as value the total number

of moves made by the input scanning head during the course of the computation with N as input.

Under our general definition any set of integers is recognizable by a machine for which $C(N) \sim \log_2 N$, $M(N) \sim \log_b N$. Such a machine simply reads the input into the black box, different inputs leading to different final configurations, those corresponding to inputs belonging to the given set being the ones which cause acceptance. In specific cases, however, some restriction is placed on the permissible sets of acceptance configurations.

An off-line Turing machine is, for example, a recognition machine. In this case it is the final state-symbol combination rather than the configuration as a whole which determines acceptance or rejection. If, for such a machine T(N) denotes the number of steps in the computation with N as input then clearly $M(N) \leqslant T(N)$. Furthermore, each internal configuration of an off-line Turing machine is determined by a machine state, what is written on the non-blank portion of each internal work tape, and the relative location of each internal scanning head. If we denote by L(N) the number of distinct squares of internal work tape scanned during the computation with input N, then, provided L(N) is unbounded, we have the readily derived limiting relation,

$$\overline{\lim}\ L(N) / C(N) \geqslant 1 / \log_2 r,$$

where r is the number of symbols (blank included) of the internal alphabet of the machine. Thus M(N) and number of steps, and C(N) and size of work area are related. This relationship is not always a close one.

Many steps can intervene between successive moves of the input scanning head and T(N) may greatly exceed M(N). Similarly, if the work tapes are used inefficiently - this inefficiency may be unavoidable - L(N) may greatly exceed the bound indicated above. (Note: In contrast with the time and tape functions in general use [6, 9] our T and L have as arguments input numbers rather than their lengths.)

In terms of the functions C and T the main results presented in this paper are: section II - for any read-only machine which recognizes the set of squares $\underline{\lim}\ (C(N) / \log_2 \log N) \geqslant 1$; section III - for any read/write machine which recognizes the set of squares $\underline{\lim}\ (T(N)\, C(N) / \log^2 N) > 0$; section IV - there is a read-only, off-line Turing machine which uses the root-extraction algorithm to recognize squares for which $C(N) \sim \log_2 N$, $T(N) = O(\log^2 N)$; sections V and VI - there is a read-only, off-line Turing machine which uses the residue method to recognize squares for which $C(N) \sim 4 \log_2 \log N$, $T(N) = O(\log^3 N)$.

II. A Bound on Memory Capacity

In this section we establish the following result.

Theorem 1. For every read-only recognition machine which recognizes the set of perfect squares in base b notation ($b \geqslant 2$) we have

$$\underline{\lim} \frac{C(N)}{\log_2 \log_2 N} \geqslant 1.$$

Here, as in the following section, we limit attention to the case b = 2 — generalization to other values is straightforward. The proof depends on a crossing-sequence argument [7] in which use is made of the fact that one can specify, subject to some restriction, the digits of either the high order half or the low order half of a binary integer and fill in the remaining digits to obtain a square in only a limited number of ways.

Lemma 1. Let $n \geqslant 3$ be an integer and let u and v be odd integers in the range $2^n > u > v > 3 \cdot 2^{n-2}$. Then (a) u^2 and v^2 both have exactly 2n binary digits; (b) the last n digits of u^2 are distinct from the last n digits of v^2; (c) the first n digits of u^2 are distinct from the first n digits of v^2.

Proof. Since $2^{2n} > u^2 > v^2 > 9 \cdot 2^{2n-4} > 2^{2n-1}$, (a) holds.

If the last n digits of both u^2 and v^2 are the same we have $u^2 \equiv v^2 \pmod{2^n}$. Then $(u-v)(u+v) \equiv 0 \pmod{2^n}$, and thus, since $(u-v)+(u+v) = 2u$ is twice an odd number, either $2^{n-1} \mid u-v$ or $2^{n-1} \mid u+v$. But $2^{n-1} > u-v > 0$ and $4 \cdot 2^{n-1} > u+v > 3 \cdot 2^{n-1}$.

Consequently neither alternative can hold, which gives us (b).

Finally, suppose u^2 and v^2 have the same first n digits. Then $2^n > u^2 - v^2 > (v+2)^2 - v^2 > 4v > 2^n$; from this contradiction (c) follows.

Let $\mathcal{M}$ be a fixed, read/write machine which recognizes the set of perfect squares in binary notation. We focus attention on the way in which $\mathcal{M}$ operates with input from the set $W = W(n)$ composed of the 2^{n-3} squares of odd numbers u in the range $2^n > u > 3 \cdot 2^{n-2}$. To this end, for $1 \leqslant k < 2n$ and $s \in W(n)$, we let $S(s, k)$ be the crossing sequence for input s at the k^{th}

boundary; that is, S(s, k) is the finite sequence of configurations assumed by $\mathcal{M}$ as it crosses in either direction between the k^{th} and $(k+1)^{st}$ input tape square during the computation with input s.

Lemma 2. If $k \leqslant n$ ($k \geqslant n$) and s, $t \in W(n)$, and if s and t differ in their first k, (resp., last $2n-k$) digits then $S(s, k) \neq S(t, k)$.

Proof. Suppose that $k \leqslant n$, s, $t \in W$, and that s and t have the form $s = s_1 \cdot 2^{2n-k} + s_2$, $t = t_1 \cdot 2^{2n-k} + t_2$, where s_1, s_2, t_1, t_2 are non-negative integers with $2^{2n-k} > s_2, t_2$ and $s_1 \geqslant t_1$. If $S(s, k) = S(t, k)$ then the number $z = s_1 \cdot 2^{2n-k} + t_2$ agrees with s in its first k digits and with t in its last $2n-k$ digits; hence, by theorem 1 of ref. 7, since $\mathcal{M}$ accepts both s and t is also accepts z, which must therefore be a square. Clearly, z is odd and $2^{2n} > z > 9 \cdot 2^{2n-4}$, so z is a member of W. But z and t agree in their last n digits from which it follows from (b) of lemma 1 that $z = t$, hence that $s_1 = t_1$; that is s and t agree in their first k digits. By a similar argument, using (c) of lemma 1, we can show that if $k \geqslant n$ and $S(s, k) = S(t, k)$ then s and t agree in their final $2n-k$ digits. The lemma follows.

We turn now to proof of the theorem itself. For this we assume that $\mathcal{M}$ is a read-only machine. Let $Q = Q(n)$ be the number of distinct internal configurations of $\mathcal{M}$ reached during any of the computations with members of W(n) as input. Since $\mathcal{M}$ does not alter its input tape it cannot, during the course of any single computation, pass twice from the n^{th} to the $(n+1)^{st}$ symbol while in the same configuration, for this would imply infinite cycling in the operation of $\mathcal{M}$. Likewise, $\mathcal{M}$ cannot pass twice from the $(n+1)^{st}$ to the n^{th} symbol while in the same configuration. Thus, for each $s \in W$, S(s, n) is of length at most 2Q and consequently Q^{2Q} is an upper bound on the number of distinct sequences of this sort. On the other hand, W has 2^{n-3} members, each of which must, by the two lemmas, have a central crossing sequence distinct from that of each of the others. Thus, $Q^{2Q} \geqslant 2^{n-3}$, or $2Q \log_2 Q \geqslant n-3$. It follows that for each $\epsilon > 0$ there is an $n(\epsilon)$ such that $Q^{1+\epsilon}(n) \geqslant 2n$ whenever $n \geqslant n(\epsilon)$. For any such n, let N be the largest member of W(n). Observing that $2n > \log_2 N$, we have

$$(1+\epsilon) \log_2 Q \geqslant \log_2 \log_2 N .$$

Thus. since $C(N) \geqslant \log_2 Q(n)$,

$$(1+\epsilon) \frac{C(N)}{\log_2 \log_2 N} \geqslant 1 .$$

Passing to the limit for N and letting ϵ tend to zero we obtain the inequality stated in the theorem

This inequality, which we have proved only for the case $b = 2$, remains in unaltered form for larger b. In other words it represents a base independent property of square-recognition machines.

Corollary. For a read-only, off-line Turing machine which recognizes the set of squares we have

$$\overline{\lim} \frac{L(N)}{\log \log N} > 0 .$$

This follows from the limiting relation between C and L mentioned in the preceding section.

III. A Bound on the Capacity-Time Product

The argument used in establishing theorem 1 actually makes use of crossing sets [9] rather than sequences, and depends essentially on the fact that $\mathcal{M}$ cannot write on its input tape. Returning to the general case with $\mathcal{M}$ a read/write machine, our next objective is that of obtaining a lower bound on the quantity $\Sigma_{s \in W} M(s)$.

Lemma 3. Let $B_k(r)$ be the number of members of W for which S(s, k) is of length r. Then $B_k(r) \leqslant 2^{|n-k|} Q^r$.

Proof. We assume $k \leqslant n$, the case $k \geqslant n$ being similar. From lemma 1, each member of W is distinct in its first n digits from all others; thus, by lemma 2, at most 2^{n-k} members can have any given sequence of k initial digits. Since there are Q^r possible crossing sequences of length r it follows that $2^{n-k} \cdot Q^r \geqslant B_k(r)$, as was to be shown.

Now,

$$\Sigma_{s \in W} M(s) \geqslant \Sigma_{k=1}^{2n-1} \Sigma_{r=0}^{\infty} r B_k(r),$$

and we wish to minimize the quantity on the right, subject to the obvious constraint, $\Sigma_{r=0}^{\infty} B_k(r) = 2^{n-3}$, and the constraint $B_k(r) \leqslant 2^{|n-k|} Q^r$, obtained in the preceding lemma. Clearly, this will be accomplished by making those $B_k(r)$, with r small, as large as possible. That is, the minimum will be obtained if $B_k(r) = 2^{|n-k|} Q^r$ for $r \leqslant p$, $B_k(p+1) = R$, and $B_k(r) = 0$, for $r \geqslant p+2$, where p and R are uniquely determined non-negative integers with $0 \leqslant R < 2^{|n-k|} Q^{p+1}$.

Lemma 4. For any real number $x \geq 1$ and integer $Q \geq 2$, let x be expressed in the form $x = \Sigma_{i=0}^{p} Q^i + R$, where p is an integer and $0 \leq R < Q^{p+1}$. Let $g(x) = \Sigma_{i=0}^{p} i\, Q^i + (p+1)R$. Then $g(x) \geq x \log_Q (x/4)$.

Proof. We introduce the function

$$h(x) = (x + \frac{1}{Q-1}) \log_Q (x(Q-1) + 1) - \frac{xQ}{Q-1} .$$

For x of the form $\Sigma_{i=0}^{p} Q^i$, direct substitution shows that $g(x) = h(x)$. For arbitrary $p \geq 0$, and x in the range $\Sigma_{i=0}^{p} Q^i \leq x \leq \Sigma_{i=0}^{p+1} Q^i$, the function $g(x) - h(x)$ is continuous and

$$g''(x) - h''(x) = - \frac{(Q-1) \log_Q e}{x(Q-1) + 1} < 0.$$

Since $g(x) - h(x) = 0$ at the endpoints of this interval we conclude that $g(x) \geq h(x)$ throughout the interval, and so for all $x \geq 1$. Now we have $h(x) \geq x(\log_Q x(Q-1) - \frac{Q}{Q-1}) \geq x \log_Q (x/4)$, where we have used the easily derived bound, $\log_Q 4(Q-1) \geq \frac{Q}{Q-1}$, for $Q \geq 2$. The lemma follows.

Theorem 2. For any read/write recognition machine which recognizes the set of perfect squares in base b notation we have

$$\overline{\lim} \frac{M(N)\, C(N)}{\log_2^2 N} \geq \frac{1}{4 \log_2 b}$$

Proof. We apply the result of lemma 4, with $x = 2^{n-3 - |n-k|}$, giving

$$\Sigma_{r=0}^{\infty} r B_k(r) \geq 2^{n-3} \log_Q 2^{n-5 - |n-k|}.$$

Thus, for $n \geq 5$,

$$\Sigma_{s \in W} M(s) \geq \Sigma_{k=5}^{2n-5} 2^{n-3} (n-5-|n-k|) \log_Q 2$$

$$= \frac{2^{n-3}}{\log_2 Q} \left((n-5) + 2 \Sigma_{k=5}^{n-1} (k-5) \right) = \frac{2^{n-3} (n-5)^2}{\log_2 Q}.$$

It follows that for at least one N in W(n), we have

$$M(N) \log_2 Q(n) \geq \frac{1}{4} (\log_2 N - 10)^2$$

If $\underline{\lim} (\log_2 Q(n) / \log_2 Q(n-1) > 1$ then $C(N) \geq \log_2 Q(n-1) \geq cN^d$, for suitable positive constants c and d.[1] In this case $\overline{\lim} (M(N)C(N)/\log_2^2 N)$ is infinite. Otherwise, for any $\epsilon > 0$ there exist arbitrarily large n such that $\log_2 Q(n) \leq (1+\epsilon) \log_2 Q(n-1) \leq (1+\epsilon) C(N)$. From the above relation we have

$$(1+\epsilon)\, M(N)C(N) \geq \frac{1}{4} (\log_2 N - 10)^2,$$

hence

$$(1+\epsilon) \overline{\lim} \frac{M(N)C(N)}{\log_2^2 N} \geq \frac{1}{4} .$$

Letting ϵ tend to zero yields the inequality of the theorem.

The appearance of the additional factor, $1/\log_2 b$, in the general statement of the theorem is not surprising; e.g., if the input base is squared the number of moves of the scanning head on the input tape is roughly halved, but the number of internal configurations need not change.

Theorem 3. For any read/write off-line Turing machine which recognizes the set of perfect squares we have

$$\overline{\lim} \frac{T(N)L(N)}{\log^2 N} > 0 .$$

($L(N) = 1$ if the machine has no internal tapes).

This does not appear to be a direct corollary of theorem 2, simply because, while we may freely replace M by T in that theorem, there is no obvious justification for replacing C by L. A proof of theorem 3, which I omit, follows the line of that of theorem 2, being based on an estimate for $\Sigma_{s \in W} M(s)L(s)$ rather than $\Sigma_{s \in W} M(s)$. It seems to depend on the fact that the storage structure of a Turing machine has polynomial-limited accessibility.[4] I don't know whether this is an essential feature of the proof.

Corollary. Under the hypotheses of the theorem,

$$\overline{\lim} \frac{\log(T(N)L(N))}{\log \log N} \geq 2.$$

IV. The Root-Extraction Method

I assume familiarity with the method for extracting square roots as presented in innumerable high school algebra texts, computation handbooks, and so on. A decimal example is worked out to the left of the double line in figure 1. The numbers appearing in the columns to the right suggest how this algorithm can, with reasonable economy, be carried out on a read-only,

off-line Turing machine with two internal work tapes. Assuming an input with an even number of digits - the case with an odd number is readily adjusted for - each successive k digit entry on tape A represents the current best integral approximation from below of the square root of the number composed of the first 2k digits of the number. The corresponding entry on tape B is the difference between the latter and the square of the number on tape A. It is not difficult to see that it is possible to pass from one pair of entries to the next without using additional tape space.

	29419861	Tape A	Tape B
0	00		29
1	+1	1	28
+1	+3	2	25
+1	+5	3	20
+1	+7	4	13
+1	+9	5	04
50	2500	50	441
+1	+101	51	340
+1	+103	52	237
+1	+105	53	132
+1	+107	54	025
540	291600	540	2598
+1	+1081	541	1517
+1	+1083	542	0434
5420	29376400	5420	43461
+1	+10841	5421	32620
+1	+10843	5422	21777
+1	+10845	5423	10932
+1	+10847	5424	00085
5424	29419776		85

FIGURE 1.

The number 29419861 used in the example is not a square since the entry on tape B is not zero when all digits of the input have been processed.

In order to minimize the amount of work tape used, the number on tape B can be shifted left one place whenever a zero appears in the first non-blank position. If this is done we have $L(N) \sim \log_b N$ (b is the input base), since the number of digits in each of the numbers on either work tape can exceed at most marginally half the number of input digits. It is also fairly obvious that $C(N) \sim \log_2 N$. Thus, $L(N)/C(N) \sim 1/\log_2 b$, the right term of which differs from the bound mentioned in section I by a small factor attributable to the fact that the blank is inefficiently employed by the machine in question.

As to time, we can get from one pair of entries to the next in a number of steps bounded by a multiple of $\log_b N$. The number of such pairs is also bounded by a multiple of $\log_b N$; hence, $T(N) = O(\log^2 N)$. I see no obvious way in which this order estimate can be improved. On the other hand, $M(N) \sim 3 \log_b N$, the constant three arising from the fact that aside from the pass across the input number during actual execution of the algorithm, the scanning head must make an additional pass back and forth in the preliminary test to determine whether the input has an odd or an even number of digits.

Comparing these results with the theorems of the preceding sections, we see that the root-extraction method yields a value of C(N) exponentially greater than the lower bound given in theorem 1. The bound of theorem 2 is approached within a factor of twelve. However, the more meaningful limit involved in theorem 3 is infinite. The "second level" limit involved in the corollary is three-halves the theoretical minimum.

V. The Residue Method

We assume the following theorem, discussed further in section VII.

Theorem (ERH). There exists an absolute constant κ such that every sufficiently large non-square integer N is a quadratic non-residue of some rational prime not exceeding $\kappa \log^2 N$.

In the next two sections we describe a read-only, off-line Turing machine which uses this result in testing for squares. We do this in two stages. In the first we construct a relatively simple machine, $\mathcal{M}_1$, which is less than optimal in both time and capacity characteristics. In the second stage we modify $\mathcal{M}_1$ to obtain an improved machine, $\mathcal{M}_2$.

The machine $\mathcal{M}_1$ has four work tapes, A, B, C, D. Let N be the input number and ν_A, ν_B, ν_C, ν_D, the numbers written on the work tapes at any designated moment - the notation for the latter being the same as that of the input. $\mathcal{M}_1$ operates as follows. (α) A close integral upper bound on $\kappa \log^2 N$ is computed and placed on tape A. (β) The number three is written on B. Subsequently, B is used to generate on request, and to store, the successive odd integers, except that if a request is made to generate a number greater than ν_A the machine halts and the

input is accepted. (γ) For each new ν_B the least non-negative residue of N modulo ν_B, is computed and placed on C. (δ) ν_B is then copied onto D and, using C and D only, the Jacobi symbol, $(\nu_C \mid \nu_D) = (\nu_C \mid \nu_B)$, is evaluated. If $(\nu_C \mid \nu_D) = -1$, the machine stops and the input is rejected; otherwise C and D are erased and the process returns to task (β) with a request for the next larger odd integer.

Let L_i, i = A, B, C, D, denote the number of squares of tape i used in the computation with input N. Let T_i, $i = \alpha, \beta, \gamma, \delta$, denote the number of steps required to make a single pass through task (i). We now estimate these quantities.

(α) An approximation of $\kappa \log^2 N$ can be obtained by counting the number of digits in N, squaring the result, and multiplying by an appropriate integral constant. If $\omega(N)$ is this approximation then $\omega(N) \geq \kappa \log^2 N$ and $\log_b \omega(N) \sim 2 \log_b \log N$. Time to execute this task is dominated by the time to count the digits of N: $T_\alpha = O(\log N)$. The space-all on tape A - needed is just that required to write the final result:
$L_A \sim \log_b \omega(N) \sim 2 \log_b \log N$.

(β) Comparison of ν_B with ν_A and the addition of two to ν_B can be done in time $T_\beta = O(\log \nu_A) = O(\log \log N)$. Since $\nu_B \leq \nu_A$, we have $L_B \lesssim 2 \log_b \log N$.

(γ) Suppose ν_B has k digits. Copy the first k digits of N onto tape C and by successive subtractions of ν_B from ν_C reduce ν_C until $0 \leq \nu_C < \nu_B$. Copy the $(k+1)^{st}$ digit of N on the right end of ν_C and repeat the subtraction process. Do the same for each successive digit of N until these are exhausted. When the task terminates ν_C is the residue of N modulo ν_B. If before copying a digit of N we erase the initial zeros on C and shift ν_C left to fill in the squares left vacant then only marginally more than k squares of C need be scanned during this procedure. Furthermore, for task (δ) no additional squares of C are needed. Thus $L_C \sim \log_b \nu_B \lesssim 2 \log_b \log N$. The time required is $T_\gamma = O(\log N \log \nu_B) = O((\log N)(\log \log N))$.

Before analyzing task (δ) we recall the following properties of the Jacobi symbol. Here s and t are positive integers with t odd.

(i) $(0 \mid t) = 0 \qquad (1 \mid t) = 1.$

(ii) $(2s \mid t) = \begin{cases} +(s \mid t) & \text{if } t \equiv 1, 7 \pmod 8 \\ -(s \mid t) & \text{if } t \equiv 3, 5 \pmod 8 \end{cases}$

(iii) $(s \mid t) = \begin{cases} +(t \mid s) & \text{if } s \equiv 1 \pmod 4 \text{ or } t \equiv 1 \pmod 4 \\ -(t \mid s) & \text{if } s \equiv t \equiv 3 \pmod 4 \end{cases}$

(iv) If $s_1 \equiv s_2 \pmod t$ then $(s_1 \mid t) = (s_2 \mid t)$.

(v) If $(s \mid t) = -1$ then s is a quadratic non-residue of t; if t is prime and s is a quadratic non-residue of t then $(s \mid t) = -1$.

Basically, it is property (v) which assures us that $\mathcal{M}_1$ gives only correct answers.

The procedure followed on $\mathcal{M}_1$ for evaluating a Jacobi symbol $(s \mid t)$, with t odd and $0 \leq s < t$, is as follows.

(1) If s = 0 or 1 use (i), otherwise go on to (2).

(2) If $s > 0$ is even divide s by two and use (ii). Repeat (2) (new s = 1/2 old s) until an odd s is obtained. If s = 1 return to (1), otherwise continue to (3).

(3) Using (iii) interchange the roles of s and t and go on to (4).

(4) Using (iv), replace $(t \mid s)$ by $(t' \mid s)$, where $t' \equiv t \pmod s$ and $0 \leq t' < s$, and return to (1) (with new s = old t', new t = old s).

The procedure must eventually stop, and can stop only at phase (1). To establish a bound on the number of times the various phases of the procedure can be executed we show that if one pass through phase (2) is initiated with symbol $(s \mid t)$ and the next with symbol $(u \mid v)$ then $uv \leq 1/2\ st$. If s is even then u = 1/2s, v = t and this is clearly the case. If s is odd we must pass through both (3) and (4) which results in the replacement of $(s \mid t)$ by $(u \mid s)$, where $0 \leq u < s < t$ and $t \equiv u \pmod s$. Then $t = u + ks$, for some positive integer k, hence $t \geq 2u$. Thus, $u \leq 1/2t$, $v = s$ which establishes the inequality. Analysis of the flow structure of the procedure shows that this inequality implies that in evaluating $(\nu_C \mid \nu_D)$ no phase is entered more than $O(\log \nu_C \nu_D) = O(\log \log N)$ times.

(δ) In carrying out task (δ) on $\mathcal{M}_1$ we need keep at any time only two numbers, one on C and one on D. The sign changes introduced in (2) and (3), and the interchanges in role of the two numbers, introduced in (3), can be kept track of by the finite state control mechanism. Since the numbers involved get successively smaller, and since getting from one pair to the next can in all cases be done without using extra tape, we have

$L_D \sim \log_b \nu_D \lesssim 2 \log_b \log N.$

As to execution time, the simple checks and computations required for a single pass through (1), (2) or (3) can be done in O(log log N) steps while the division required in executing (4) takes $O(\log^2 \log N)$ steps. Thus, recalling our bound on the number of times any phase of the procedure can be entered, we have

$T_\delta = O(\log^3 \log N).$

From this analysis of $\mathcal{M}_1$ we have

$L(N) = L_A + L_B + L_C + L_D \lesssim 8 \log_b \log N.$

In the main routine, task (α) is executed only once, the remaining tasks at most once each for each odd integer not exceeding ω(N). Thus,

$T(N) = T_\alpha + O(\omega(N))(T_\beta + T_\gamma + T_\delta) =$
$O(\log N) + O(\log^2 N)(O(\log \log N) +$
$O((\log N) \log \log N) + O(\log^3 \log N)) =$
$O((\log^3 N) \log \log N).$

Finally, we have $C(N) \lesssim 7 \log_2 \log N$, and I suspect seven is not the smallest coefficient for which a relation of this sort holds. In any case, that it holds for coefficient seven follows from the fact that ω(N), as computed by $\mathcal{M}_1$, depends only on the length of N. Thus, for inputs not exceeding N, only O (log N) distinct numbers appear on tape A whereas the allotted tape is capable of holding $O(\log^2 N)$ numbers. In other words, the number of different configurations reached by $\mathcal{M}_1$ is substantially less than would be the case if all tapes were used "efficiently."

VI. The Residue Method, cont.

Machine $\mathcal{M}_2$ is obtained by modifying $\mathcal{M}_1$ in three ways: combining the function of tape C with that of tape A, and of that of tape D with that of tape B, and applying a "leaky sieve" to the odd numbers prior to computing the residue and Jacobi symbol, thus bypassing these tasks for many composite numbers. To see that the functions of A and C can be combined we need only observe that no use is made of the information on tape A when $\mathcal{M}_1$ is computing on C and D conversely. Thus, where $\mathcal{M}_1$ executes tasks (γ) and (δ), $\mathcal{M}_2$ first erases A and then executes these tasks using A in place of C. Then, where $\mathcal{M}_1$ executes (β), $\mathcal{M}_2$ erases A, recomputes ω(N) placing it on A, and proceeds to (β). In effect, at the cost of computing ω(N) $O(\log^2 N)$ times, rather than just once, tape C can be dispensed with.

C:	89	89	89	89	11	11	11
D:	319	52	26	13	13	2	1

FIGURE 2.

A:	0,89	3,89	3,89	3,89	3,11	25,11	25,11
B:	1,319	1,52	2,26	4,13	22,13	22,2	44,1

FIGURE 3.

It is less obvious that D and B can be combined. In $\mathcal{M}_1$ tape D is needed so that ν_B can be preserved while the Jacobi symbol is being evaluated. We have observed that as the evaluation of a Jacobi symbol progresses the two numbers retained decrease in magnitude. Thus, as the computation progresses, space becomes available on the two tapes containing these numbers which, it turns out, can be used to retain sufficient information about the course of the computation so that ν_B can be reconstructed after the evaluation is completed. I hope a simple example will be sufficient to show how this can be done.

Suppose (89 | 319) is to be evaluated. On $\mathcal{M}_1$ the numbers appearing on tapes C and D at successive stages of this evaluation might be as in figure 2. These are obtained as indicated in the following equations.

$$\begin{aligned} 319 &= 3 \cdot 89 + 52 \\ 52 &= 2 \cdot 26 \\ 26 &= 2 \cdot 13 \\ 89 &= 6 \cdot 13 + 11 \\ 13 &= 1 \cdot 11 + 2 \\ 2 &= 2 \cdot 1 \end{aligned}$$

From these the following system can be derived.

$$\begin{aligned} 319 &= 0 \cdot 89 + 1 \cdot 319 \\ &= 3 \cdot 89 + 1 \cdot 52 \\ &= 3 \cdot 89 + 2 \cdot 26 \\ &= 3 \cdot 89 + 4 \cdot 13 \\ &= 3 \cdot 11 + 22 \cdot 13 \\ &= 25 \cdot 11 + 22 \cdot 2 \\ &= 25 \cdot 11 + 44 \cdot 1 \end{aligned}$$

At the stages corresponding to those indicated in figure 2 tapes A and B of $\mathcal{M}_2$ contain the information shown in figure 3. Clearly $\nu_B = 319$ can be reconstructed from the information available at the final stage, and this can be done without using additional tape space.

In general, in evaluating (s | t) on $\mathcal{M}_2$, we

have, at each stage two numbers, a_1 and a_2, on tape A and two, β_1 and β_2, on tape B, and for these, $t = a_1 \cdot a_2 + \beta_1 \cdot \beta_2$. Since the logarithm of a number equals roughly the number of its digits, and since $\log_b t \geq \log_b a_1 + \log_b a_2$, $\log_b t \geq \log_b \beta_1 + \log_b \beta_2$, each pair can be written (with a blank separating the two members of a pair) on an amount of tape at most marginally greater than that required to accommodate t. Verification that it is possible to get from the pairs at one stage to those at the next without using additional tape is straightforward. Thus tape D can be dispensed with. As for the time $\mathcal{M}_2$ takes to evaluate a Jacobi symbol, it is clearly greater than $\mathcal{M}_1$ takes. $O(\log^5 \log N)$ appears to be an adequate estimate -- in any case the time for carrying out this task is obviously negligible compared with that for other phases of the test.

The factor $\log \log N$ in the order estimate for T(N) for $\mathcal{M}_1$ is intuitively attributable to the fact that $\mathcal{M}_1$ obtains the residue of N modulo all $O(\log^2 N)$ odd integers less than $\omega(N)$, rather than just the $O(\log^2 N / \log \log N)$ primes in this range. An initial test for primality seems, according to my analysis, to consume additional time which offsets exactly the time saved in calculating residues, thus leaving the order estimate of T(N) unaltered. Nevertheless, a compromise does lead to a reduction in running time. On $\mathcal{M}_2$ each new value of ν_B is first tested to determine if it has any proper factors not exceeding $\log^{3/4} N$. If not the remaining tasks for this value of ν_B are executed. If so $\mathcal{M}_2$ simply skips these tasks and goes on to the next larger ν_B.

Operating in this manner, $\mathcal{M}_2$ computes residues only for those ν_B which are either prime, or are composite with all prime factors greater than $\log^{3/4} N$. For N large these composites can have only two prime factors, since a number with more factors must exceed $\log^{9/4} N > \omega(N)$. From the prime number theorem [5] the number of primes not exceeding $\omega(N)$ is $O(\log^2 N / \log \log N)$, while it is a corollary of that theorem that the number of composites not exceeding $\omega(N)$ which are products of two primes, both greater than $\log^{3/4} N$, is also $O(\log^2 N / \log \log N)$. Thus, since the time to compute the residue of N modulo any single ν_B is $O((\log N) \log \log N)$, the total time $\mathcal{M}_2$ spends in computing residues is $O(\log^3 N)$.

I will omit a detailed description of how $\mathcal{M}_2$ performs this preliminary sifting. It can be done using only tapes A and B and without using more than roughly $2 \log_b \log N$ squares of either. The time required to test a single ν_B is $O((\log^{3/4} N)(\log^4 \log N))$. Thus, the time to test all ν_B not exceeding $\omega(N)$ is $O((\log^{11/4} N)(\log^4 \log N)) = O(\log^3 N)$.

From the above discussion, we see that there are three tasks, each of which may require a total of $O(\log^3 N)$ steps to execute during the test of a single input, but no task requires more time than this. Therefore, for $\mathcal{M}_2$, $T(N) = O(\log^3 N)$. Since two tapes have been got ride of without increasing the space used on the remaining two, $L(N) \lessapprox 4 \log_b \log N$.

For $\mathcal{M}_2$ we have $C(N) \sim 4 \log_2 \log N$. To see this, we observe that if N is large enough, if β is an odd number between three and $1/2\,\omega(N)$, and if α, $0 \leq \alpha < \beta$, is a quadratic residue of β, then there is a square M, $1/2\,N < M \leq N$ such that $M \equiv \alpha \pmod{\beta}$ and $\beta \leq \omega(M)$. Thus, at some point in the computation with M as input, $\nu_A = \alpha$, $\nu_B = \beta$. In other words, both tapes A and B are utilized with near maximum efficiency - except that the blank is rather underemployed. (No tape of $\mathcal{M}_2$ need ever contain more than two blanks between non-blank squares.)

For this machine the limit occurring in theorem 3 is infinite but, just as for the square root method, the limit occurring in its corollary 3/2 the bound given.

We conclude this section with some remarks on the averages of T and L. First, we have $\bar{T}(N) = \frac{1}{N} \sum_{n=1}^{N} T(n) = O(\log N)$. A simple heuristic argument suggests the plausibility of this relation. That is, <u>most</u> integers are quadratic non-residues of very small primes, e.g., one-third are quadratic non-residues of three, three-fifths of either three or five, and 27/35 of three, five or seven. Thus, in most cases the residue test terminates quickly. A precise proof, which I omit, can be based on the following lemma.

<u>Lemma.</u> Let R(N) be the number of positive integers not exceeding N which are quadratic residues of all primes not exceeding $\log_e N$. Then there is a constant c such that $R(N) = O(N^{1 - c/\log \log N})$.

It is perhaps no more than an oddity, but a minor modification of $\mathcal{M}_2$ yields a machine for which the average of L(N) is also small. On this machine each recomputation of $\omega(N)$ is carried only so far as to make sure that the result of the complete computation would be larger than ν_B. Then $\bar{L}(N) = \frac{1}{N} \sum_{n=1}^{N} L(n) = O(1)$. The implied constant can be made as small as desired.

VII. Quadratic Non-Residues and the ERH

The correctness of the residue test for squares depends on the theorem stated at the beginning of section V. A form of the extended Riemann hypothesis of sufficient strength to yield this theorem is the following.

ERH: For every real, non-principal character χ, the roots of the function $L(s,\chi) = \Sigma_{n=1}^{\infty} \chi(n)\, n^{-s}$ having positive real part have real part 1/2.

I have not seen the theorem explicitly stated in the literature. A related result, due to Ankeny, concerning the smallest quadratic non-residue of a given prime does appear. [1] Examination of Ankeny's proof reveals that more is proved than stated, and a minor modification yields a proof of the theorem stated here. An alternative proof, based on methods of Wang, [10] is perhaps more illuminating in that it shows explicitly how the exponent on the logarithm is related to the location of the zeros of $L(s,\chi)$.

The exponent two is not known to be the best possible; it might be as small as one. The magnitude of this exponent affects our estimates for both C(N) and T(N). Specifically, if a is the best exponent, either larger or smaller than two, then $C(N) \sim 2a \log_2 \log N$ and $T(N) = O(\log^{1+a} N)$, providing, that is, that a is rational or in some sense easy to compute, and that appropriate and obvious modifications are made in the machine executing the test. I have done some computational testing of the theorem, the results of which tentatively suggest that two is, in fact, the smallest possible exponent.

Up to this point the availability of an explicit value for the constant κ has been implicitly assumed. Actually, and unfortunately, not only do I not know such a value, but I don't even know whether one is in principle computable. Obviously, the test cannot be implemented with confidence lacking knowledge of this constant.

VIII. Comments and Conjectures

The coefficient in the asymptotic relation for C(N) derived in section VI is a factor of four greater than the theoretical minimum obtained in section II. This gap may result from the fact that our machine is too big, our bound too small, or from both these causes. Short of an improvement resulting from a reduction in exponent size discussed in the last section, I am unable to envisage on what principles a smaller square-recognizing device might be based. However, it seems entirely plausible that the bound given in theorem 1 can be increased. This probably cannot be done simply by refining the analysis of the argument as given, for essentially the same argument can be used to obtain an analogous theorem for machines which recognize binary palindromes, and in this case the lower bound is attainable. [3] However, the digital structure of squares is considerably more complex than that of palindromes, and it seems reasonable to suppose, or hope, that a stronger argument yielding a higher bound can be found. It would, incidentally, be extremely interesting should such an argument yield a coefficient greater than two, for this in turn would imply a closing of the gap in the exponent size mentioned in section VII.

For the machines described in sections IV and VI, L(N) and C(N) are, an asymptotic sense, in optimal relationship, except that both use the blank with nearly maximal inefficiency. In either case we could dispense with one symbol of the internal alphabet, and by recoding of the information stored internally come arbitrarily close to the absolute optimal relationship; i.e., for any $\epsilon > 0$ there is a machine with b internal symbols, including the blank, executing essentially the same algorithm such that $\lim L(N)/C(N) < 1/\log_{b+\epsilon} 2$.

For the machine described in section V the optimal relation between L(N) and C(N) does not hold. The question is, could we use this observation alone to justify the assertion that there exists a machine performing the same task which uses the same internal alphabet but less tape? In general, if we know a given set is recognizable by an off-line TM with a given C(N) what can we say concerning an upper bound on L(N) for some machine-possibly different from the first-which recognizes this set?

For the root extraction method we have $T(N)\, C(N) = O(\log^3 N)$ and for the residue method $T(N)\, C(N) = O(\log^{3+\epsilon} N)$, for any $\epsilon > 0$. On the the other hand, theorem 2 suggests the possibility of a method for which $T(N)\, C(N) = O(\log^2 N)$. Whether such a method exists is an open question. However, it has been observed by S. Cook that the Toom multiplication algorithm can be applied to the problem of root extraction, [4] yielding a method for recognizing squares which can be carried out on an off-line Turing machine for which $C(N) = O(\log N)$, $T(N) = O(\log^{1+\delta} N)$, where δ approaches zero as N increases. Thus, for this method, $T(N)\, C(N) = O(\log^{2+\delta} N)$, hence the exponent two in the limits in theorems 2 and 3 is the best

possible, and the bound in the corollary of theorem 3 is attainable. I have been unable to find any way of reducing the exponent three in the order relation for T for machines for which $C(N) = O(\log \log N)$, however, many possibilities remain unexplored.

Finally, I would like to touch on the subject of small capacity machines in general. Stearns, Hartmanis and Lewis[9] have shown that the capacity of a machine which recognizes any set not recognizable by a finite state machine must be at least $O(\log \log \log N)$. Apparently there is no known example of a set, interesting in its own right, recognizable by such a small capacity machine. On the other hand there is a wealth of interesting sets recognizable by machines for which $C(N) = O(\log \log N)$; e.g., the set of powers of two (any base), the set of Fibonacci numbers and, assuming the ERH, the set of squares. It seems a plausible speculation that the set of cubes, the set of fourth powers and, in general, the set of values of any polynomial with integer coefficients should also have this property, but I have been able to find no proof of this. Trial computations for a few simple examples suggest that an analogue of the theorem stated in section V may hold in each case provided that prime powers as well as primes are considered.

It would be of interest to have a characterization of the sets having this property of the sort which characterizes the sets recognizable by linear bounded automata as being the ξ_2^* - sets. The best I have been able to do in this direction is to show that these sets are all strongly rudimentary in the sense introduced by Bennett.[2] I doubt whether the converse is true.

References

1. Ankeny, N. C., "The Least Quadratic Non Residue," Annals of Math. (55) 1952, 65-72.

2. Bennett, J. H., "On Spectra," Doctoral Dissertation, Princeton Univ., 1962.

3. Cobham, A., "Time and Memory Capacity Bounds for Machines which Recognize Squares or Palindromes," IBM Research Report RC-1621, 1966.

4. Cook, S. A., "On the Minimum Computation Time of Functions," Doctoral Dissertation, Harvard Univ., 1966.

5. Hardy, G. H., and Wright, E. M., "An Introduction to the Theory of Numbers," Clarendon Press, 1954.

6. Hartmanis, J., and Stearns, R. E., "On the Computational Complexity of Algorithms," Trans. Amer. Math. Soc. (117) 1965, 285-306.

7. Hennie, F. C., "Crossing Sequences and Off-Line Turing Machine Computations," 1965 IEEE Conference Record on Switching Circuit Theory and Logical Design, 168-172.

8. LeVeque, W. J., "Topics in Number Theory," 2 vols., Addison-Wesley, 1958.

9. Stearns, R. E., Hartmanis, J., and Lewis, P. M., "heirarchies of Memory Limited Computations," 1965 IEEE Conference Record on Switching Circuit Theory and Logical Design, 179-190.

10. Wang, Y., "On the Least Primitive Root of a Prime," Scientia Sinica (10) 1961, 1-14.