Establishing lower bounds on algorithms—A survey

by E. M. REINGOLD
University of Illinois
Urbana, Illinois

INTRODUCTION

Algorithms for various computations have been known and studied for centuries, but it is only recently that much theoretical attention has been devoted to the analysis of algorithms. Turing machines and recursive functions were the first approaches, but these models, which provide much interesting mathematics, do not look at the problem from a practical standpoint. In “real” computing, no one uses Turing machines to evaluate polynomials or to multiply matrices, and little of practical significance is obtained from that approach. On the other hand, recent work has led to more realistic models and, correspondingly, to more practical results. Most of the results cannot be considered to be truly practical, but, all of them were motivated by practical considerations.

This survey is concerned with efforts to establish lower bounds on the number of operations required to solve various practically inspired problems; in particular we discuss the problems of sorting, searching, merging, root finding, polynomial evaluation, matrix multiplication, and many others. No theorems will be rigorously proved; for some the idea of the proof will be presented, and most will only be stated. The reader is urged to pursue in the literature the details of any topics which interest him.

In the establishment of lower bounds on algorithms we must consider the following questions:

- What function or class of functions is to be computed?
- What class of algorithms is allowed?
- With what are we measuring lower bounds?

The answers to the last two of these questions are inherently interwoven with the answer to the first question. In analyzing sorting we will consider different things important than in analyzing matrix multiplication, and so in each case we will allow different kinds of algorithms and we will measure their efficiency in different ways.

Even for a specific answer to the first question, how the efficiency of an algorithm should be measured is not obvious. Ideally, we would like to assign a realistic cost to every operation performed; such a model usually makes the establishment of lower bounds too difficult. To simplify the problem, we isolate the “key” operations and ignore all others. There are two ways to count the operations used by an algorithm: the number used on the worst case input or the expected number used on a random input, assuming some distribution of the inputs. An algorithm is minimax optimal or worst case optimal if no algorithm is more efficient in the worst case; an algorithm is minimean optimal or average case optimal if no algorithm is more efficient in the average case.

It should be noted that some of the results discussed here have never been formally published, but have become a part of the “folklore” of the area; in such cases the citation will be to the place they first found their way into print—usually a textbook. Moreover, this survey is not complete: to include every known result would give the paper undue length; in addition, many results undoubtedly remain unnoticed, buried in journals, technical reports, and unpublished manuscripts.

NOTATION

The floor and ceiling operations are defined as usual: \([x]\) is the greatest integer less than or equal to \(x\), and \([x]\) is the least integer greater than or equal to \(x\). We use the standard notation for the order of magnitude of a function: \(f(n) = \Theta(g(n))\) if there is a constant \(k > 0\) such that

\[
\limsup_{n \to \infty} \frac{f(n)}{g(n)} = k.
\]

If the limit

\[
\lim_{n \to \infty} \frac{f(n)}{g(n)} = k
\]
Lemma 1:

The minimum external path length of an extended binary tree with \( n \) external nodes is \( n \lceil \lg n \rceil + n - 2^\lceil \lg n \rceil \).

Lemma 2:

The minimum height of an extended binary tree with \( n \) external nodes is \( n \lceil \lg n \rceil \).

A complete discussion of most of the results discussed below can be found in Knuth.\(^{K72}\)

Sorting

Suppose we are given a set \( \{x_1, \ldots, x_n\} \); what is the minimum number of comparisons required to determine the permutation \( \pi \) so that \( x_{\pi(1)} < x_{\pi(2)} < \cdots < x_{\pi(n)} \)? Let \( S(n) \) denote the minimum number of comparisons needed to rank \( n \) distinct inputs according to some ordering, then we have

Theorem 1:

\[ \lceil \lg n! \rceil \leq S(n) \leq 1 + n \lceil \lg n \rceil \] and thus using Stirling's approximation we have that \( S(n) \sim n \lceil \lg n \rceil \). The upper bound follows from some of the better algorithms for sorting, for example, the binary insertion sort first observed by Steinhaus\(^{Stei50}\) before the advent of computer sorting. The lower bound is derived from Lemma 2 and the observation that each permutation of the \( n \) inputs must cause termination at a different external node of the tree. This is a "standard" information theoretic argument which says that given \( k \) outputs at least \( \lceil \lg k \rceil \) binary decisions are required to distinguish between them. This argument appears to have been discovered independently by several authors but first appeared in Steinhaus.\(^{Ste58}\)

There has been considerable work in refining the upper bound. Most notably, Ford and Johnson\(^{Fo59}\) have developed a method of sorting which Hadian\(^{Ha69a}\) showed required

\[ \sum_{k=1}^{n} \left\lfloor \frac{3}{4} \lg k \right\rfloor \]

comparisons to sort \( n \) inputs in the worst case. Knuth\(^{K72}\) has called this the merge-insertion sort. Comparing Hadian's result with Theorem 1 we find that the merge-insertion sort is optimal for \( n \leq 11 \) and \( n = 20, 21 \) but that it requires more than \( \lceil \lg n! \rceil \) comparisons for other values of \( n \). Wells\(^{We65}\) using a computer, has shown that it is also optimal when \( n = 12 \). Summarizing, we have

Theorem 2:

\[ S(n) = \sum_{k=1}^{n} \left\lfloor \frac{3}{4} \lg k \right\rfloor \] for \( n \leq 12 \) and \( n = 20, 21 \).

Define \( \bar{S}(n) \) to be the minimum average number of comparisons required to sort \( n \) items. As before, we

![Figure 1—An extended binary tree representing the computation of the median of three numbers \( x, y, \) and \( z \)](image)
must have \( n! \) external nodes and so by Lemma 1
\[
\tilde{S}(n) \geq \frac{1}{n} (n \lceil \log n \rceil + n - 2^{\log_2(n)}).
\]
Letting \( \lceil \log n \rceil = \log n + \theta, 0 \leq \theta < 1 \) this becomes
\[
\tilde{S}(n) \geq n \log n + 1 + \theta - 2^{\log_2(n)}, \quad 0 \leq \theta < 1.
\]
Since in \([0, 1]\) the function \( 1 + \theta - 2^\theta \) has a range of \([0, 0.88]\) we have
Theorem 3:
\[
\tilde{S}(n) \geq \log n! + O(1) = n \log n - n \log 2 + O(\log n).
\]
Theorem 3 was first observed by Gieason Gl56 and first published by Kislicyn. Kis62, Kis63

Searching

If we are given a sorted set \( x_1 < x_2 < \cdots < x_n \), how difficult is it to determine in which of the \( n+1 \) ranges \( y \) lies? Let \( s(n) \) be the minimum number of comparisons to do the searching. The binary search algorithm, first noted by Steinhaus56 and \( \log n \) comparisons. Applying Lemma 2, we see that \( \lceil \log (n+1) \rceil \) comparisons is a lower bound; since \( \lceil \log (n+1) \rceil = \lceil \log n \rceil + 1 \), we know that binary search is optimal in the worst case.

Sandelius61 noted that binary search is also optimal in the average case: applying Lemma 1 as before we see that the minimum possible average is \( \log(n+1) + 0(1) \) and binary search achieves that bound.

A somewhat related problem is the discovery of the single counterfeit coin, either heavier or lighter, in a group of \( n \) coins; this problem is well known in the literature of recreational mathematics. One is usually allowed to use a balance scale and hence the comparisons are of linear functions, over \( \{-1, 0, 1\} \), of the inputs rather than just pairwise comparisons. One optimality result for this problem is due to Smith and he has shown that when \( n = (3^k - 1)/2 \), \( k \) such comparisons are necessary and sufficient.

Merging

Given two sorted sets \( A_1 < A_2 < \cdots < A_m \) and \( B_1 < B_2 < \cdots < B_n \) (all distinct) what is the best way to merge these into a single sorted set \( x_1 < x_2 < \cdots < x_{m+n} \)? Since the \( n+m \) elements are all distinct, there are \( \binom{n+m}{m} \) ways the A's may appear among the B's. Thus by Lemma 2, if \( M(n, m) \) is the minimum number of comparisons required to do the merging,
\[
M(n, m) \geq \lg \binom{n+m}{m}.
\]
A simple upper bound on \( M(n, m) \) is \( m+n-1 \) since the "usual" merging algorithm outputs at least one element for each comparison; the last element requires no comparisons. Hwang and Lin have developed a much better merging algorithm which requires
\[
m + \left\lfloor \frac{n}{2^t} \right\rfloor - 1 + tm \quad \text{where} \quad m \leq n \quad \text{and} \quad t = \left\lfloor \frac{\log n}{m} \right\rfloor.
\]
Since this is less than \( \lceil \log \binom{n+m}{m} \rceil + \min(m, n) \) we have
Theorem 4:
\[
\lceil \log \binom{n+m}{m} \rceil + \min(m, n) \geq M(n, m) \geq \lceil \log \binom{n+m}{m} \rceil.
\]
When \( m = 1 \), merging becomes a simple search and so
\[
M(1, n) = \lceil \log (n+1) \rceil.
\]
When \( m = 2 \), the analysis of merging is quite difficult; Graham and Hwang and Lin have independently shown that
\[
M(2, n) = \lceil \log (n+1) \rceil.
\]
By the construction of what Knuth calls an oracle, Karp and Graham independently showed that
\[
M(n, n) = 2n - 1.
\]
An oracle is a hypothetical device which constructs a worst case for any possible algorithm; in other words, it acts as an adversary to the algorithm, forcing the algorithm to do the maximum possible work. The oracle in this case is \( A_i < B_j \) if and only if \( i < j \).

Selection

Suppose that there are \( n \) inputs and we want to know which is the \( k \)th best in a ranking; Hadian and Sobel have termed this the "selection problem." On the other hand, if we wish to know the \( k \) best and their ranking, they call this the "ordering problem." When \( k = 1 \) and \( k = 2 \), the ordering and selection problems coincide and a minimal solution for one is also a minimal solution for the other.

The case \( k = 1 \) is trivial and a simple induction argument shows that \( n-1 \) comparisons are required. Rabin has shown that in the case \( k = 1 \), that is, computing the maximum, \( n-1 \) comparisons are necessary even if comparisons may be made between any analytic functions of the inputs. For \( k = 2 \), the problem of determining the minimal number of comparisons was first posed by Steinhaus in 1929. Schreier first stated the solution to this problem, but with an incorrect proof. Slupecki gave another incorrect proof. The first correct proof was given by Kislicyn by means of an oracle; he proved that
n−2+⌈lg n⌉ comparisons were necessary and sufficient to find the second largest element in a set.

Recently, Blum, Floyd, Pratt, Rivest and Tarjan (personal communication) have developed a remarkable algorithm which finds the $k^{th}$ largest element in a set of $n$ elements in $O(n)$ comparisons in the worst case, regardless of $k$. Prior to the development of this algorithm it was commonly conjectured that the median of a set of $n$ elements could not be computed in fewer than $O(n \log n)$ comparisons in the worst case. It was known\cite{Van70} that it could be done by an algorithm in which the expected number of comparisons was $O(n)$; but in the worst case this algorithm requires more than $O(n)$ comparisons.

Summarizing these results, let $V_k(n)$ be the smallest number of comparisons required to find the $k$th largest element of a set of $n$ elements, then we have

**Theorem 5:**

(a) $V_1(n) = n-1$

(b) $V_2(n) = n-2+⌈lg n⌉$

(c) $V_k(n) = O(n)$ for all $k$.

Pohl\cite{Pohl69} approached the selection problem in an entirely different way, showing that at least $\min\{k, n-k+1\}$ storage locations are required to determine the $k$th largest element of a set of $n$ elements.

**ALGEBRAIC PROBLEMS**

In this section we will discuss the minimum number of arithmetic operations required to compute various functions. The arithmetic operations we will allow are addition, subtraction, multiplication, and division; no other operations (comparisons, exponentiation, etc.) will be allowed. To make an analysis, we need a precise definition of what algorithms are allowed.

Let $*$ denote any of the arithmetic operations addition, subtraction, multiplication or division. A scheme is defined as a sequence of operations

$$P_i = Q_i R_i \quad i = 1, 2, \ldots, m$$

where each $Q_i$ and $R_i$ is either a constant, an input value, or a $P_j$ where $j < i$.

**Polynomial evaluation**

Suppose we are given a number $x$ and asked to compute $x^n$ for a fixed $n$, by a scheme as described above; what is the minimum number of steps required? Starting with a sufficiently large $x$, we can prove by induction that after $k$ steps of a scheme the largest number obtainable is $x^k$ which is computed by squaring $x$, squaring the result, and so on. Thus we must have $x^k \geq x^n$ and hence

$$k \geq \lceil \lg n \rceil.$$

Thus at least $\lceil \lg n \rceil$ multiplications are required to compute $x^n$ from $x$. Proving that this minimum is asymptotically achievable is more difficult. The following theorem is due to Brauer;\cite{Brauer39, Kao69} Val'skii\cite{Val'skii59} arrived independently at the same result.

**Theorem 6:**

Let $m(n)$ be the least number of multiplications required to compute $x^n$ for given values of $x$, then $m(n) \sim \lceil \lg n \rceil$.

This problem readily generalizes to: What is the smallest number of arithmetic operations needed to evaluate the $n^{th}$ degree polynomial

$$f(x) = a_n + a_{n-1}x + \cdots + a_1x + a_0,$$

for given values of $x$? Ostrowski\cite{Ostrowski54} was the first to suggest that this problem be analyzed, and he gave results for quadratic and cubic polynomials. When it is known a priori that the values of $x$ given will be equally spaced, a method using finite differences might be most convenient; however, we will assume that the values of $x$ will be arbitrary. The method usually used is Horner's method:

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0,$$

which requires $n$ additions and $n$ multiplications. Can this be improved? One can easily find specific polynomials for which there is a better method; for example

$$f(x) = 1 + x + 2x^2 + 3x^3 + x^4,$$

which can be evaluated in two multiplications and two additions by proceeding as follows:

$$x^2, x^2+1, (x^2+1)(x^2+1)+x.$$

However, we want a more general method, one which will work for all polynomials and for all values of $x$. Clearly Horner's method is a valid scheme for polynomial evaluation, and in fact, Horner's method is optimal, in the sense that it requires the fewest operations necessary in the type of scheme allowed, for we have:

**Theorem 7:**

Any scheme which can evaluate an arbitrary $n^{th}$ degree polynomial has at least $n$ additions/subtractions and $n$ multiplications/divisions.
In this theorem, the necessity of the $n$ additions/subtractions was shown by Belaga [56, 61, 69] while the necessity of the $n$ multiplications/divisions is due to Pan. Not only is Horner's method optimal, but it is uniquely optimal: Borodin [61] has shown that any scheme using only those $2n$ operations is essentially Horner's method.

A different approach can be taken if a large number of values $P(x)$ are required. Consider the following example, due to Todd. We want to evaluate the polynomial

$$P = x^4 + Ax^3 + Bx^2 + Cx + Dx + E + F.$$ 

Define the following polynomials

$$P_1 = x^2 + ax = x(a + x)$$

$$P_2 = (P_1 + x + b)(P_1 + c)$$

$$P_3 = (P_2 + d)(P_1 + e)$$

and determine $a, b, c, d, e,$ and $f$ such that $P = P_3 + f$. This can be done by the solution of linear equations and a single quadratic equation. Once these equations have been solved, $P$ can be evaluated using only three multiplications and seven additions using the sequence

$$P_1, P_2, P_3, P = P_3 + f,$$

a savings of three multiplications at the expense of one addition and some “preconditioning” of the coefficients. Since multiplication is usually much slower than addition and since we sometimes want the same polynomial evaluated at many arbitrarily spaced points, the above method can represent a significant improvement over Horner’s method.

A scheme with preconditioning is formally defined as a scheme in which the $Q_i$ and $R_i$ are, in addition, allowed to be any real functions of the coefficients of the polynomial to be evaluated. The scheme due to Todd above, is an example of such preconditioning. The idea of preconditioning is due to Motzkin and he showed that if a scheme with preconditioning computes all $n^{th}$ degree polynomials then it contains at least $\lfloor (n+1)/2 \rfloor$ multiplications. Combining this result with the full strength of the result of Belaga used in Theorem 7, we have

**Theorem 8:**

Any scheme with preconditioning which can evaluate an arbitrary $n^{th}$ degree polynomial has at least $\lfloor (n+1)/2 \rfloor$ multiplications and at least $n$ additions/subtractions.

Much effort has been spent to find a scheme with preconditioning which attains the lower bounds of Theorem 8. Early papers by Motzkin and Knuth gave methods which evaluate polynomials of degrees four, five, and six in $\lfloor (n+1)/2 \rfloor + 1$ multiplications and $n + 1$ additions; Pan [61, 66] has given similar methods for $n \leq 12$. In each of these cases, the methods are applicable only for a particular value of $n$. Pan [61, 66] gives a method valid for $n \geq 2$ which requires $\lfloor (n+1)/2 \rfloor + 1$ multiplications and $n + 2$ additions/subtractions and in Reference Pan 59 he gives a method for $n \geq 5$ for which $\lfloor n/2 \rfloor + 2$ multiplications and $n + 1$ additions/subtractions are needed. For $n \geq 3$, Knuth gives a method using $n + 1$ additions/subtractions and the number of multiplications varies between $\lfloor (n+1)/2 \rfloor + 1$ and approximately $3n/4$. Belaga [61, 62] has proved that $\lfloor (n+1)/2 \rfloor + 1$ multiplications and $n + 1$ additions suffice to evaluate any $n^{th}$ degree polynomial, but these operations may involve complex numbers. Finally, Eve’s modified Knuth’s method to give a method requiring $\lfloor n/2 \rfloor + 2$ multiplications and $n$ additions/subtractions, all of which involve only real numbers. The preconditioning in Eve’s algorithm is, unfortunately, irrational; Rabin and Winograd (personal communication) have developed a method in which the preconditioning involves only rational operations, but his method then requires about $\lceil n/2 \rceil$ multiplications to evaluate a polynomial.

The best general algorithm for polynomial evaluation, Eve’s, requires only the minimum number of additions/subtractions, however, it unfortunately requires one more than the minimum number of multiplications. It is known that when $n$ is odd both of these lower bounds cannot be simultaneously achieved, and a similar result holds when $n = 4$ and $n = 6$. There is no known general algorithm using $\lfloor n/2 \rfloor + 1$ multiplications when $n \leq 8$, although such methods are known for $n = 4, 6, 8$, where the algorithms require one or two extra addition/subtraction operations.

The above optimality theorems show that no one method will work for all polynomials of degree $n$ unless it has a certain minimum number of operations, but there are some “special” polynomials which can be evaluated far more rapidly, for example, a $x^4$ requires only five multiplications and no additions, instead of the minimums given by Theorems 7 and 8. There are “few” such polynomials for Belaga [61] has shown

**Theorem 9:**

The set of $n^{th}$ degree polynomials which can be evaluated by schemes with preconditioning in fewer operations than specified in Theorem 7 has Lebesgue measure zero in the space of all $n^{th}$ degree polynomials. Pan [62] has proved a similar result for schemes without preconditioning.

Most of these results have been generalized to polynomials of many variables and to rational func-
taneously evaluating several polynomials in the same
ations. In particular, such results are given in References
Be58, Mot55b, Os54, and Pan62. Some of the results
have also been obtained for the problem of simulta­
neously evaluating a polynomial and its first
derivative. Kn69, Mu71a
variablePan66 and for the specific case of the simul­
taneous evaluation of a polynomial when the coeffi­
cients are rational; Paterson and Stockmeyer Pat71 have
shown that 0 (<\sqrt{n}) operations are necessary and suf­
cient for the evaluation of an nth degree polynomial.

Linear algebra

By generalizing the notions in the above results to
arbitrary fields, Winograd Wi70 proved some very elegant
theorems. Let F be a field and let x1, x2, . . . , xn be a set
of variables. The question then becomes, what is the
minimum number of field operations needed to compute
the m field elements

\[ \Psi_i : F(x_1, \ldots, x_n) \rightarrow F \]

Winograd gave a very general definition of a scheme
without preconditioning, and considered only the
number of multiplications/divisions. He showed:

**Theorem 10:**

Let \( \Phi \) be an \( m \times n \) matrix whose elements are in
the field \( F \), let \( \phi \) be an \( m \) vector of elements in \( F \), and let \( x \)
denote the \( n \) column vector \((x_1, \ldots, x_n)\) so that

\[ \Phi x + \phi \in F(x_1, \ldots, x_n)^m. \]

If there are \( u \) column vectors in \( \Phi \) such that no non­
trivial linear combination of them (over \( F \)) is in \( F^n \),
then any scheme, without preconditioning, computing
\( \Phi x + \phi \) requires at least \( u \) multiplications/divisions.

Pan's result on the number of multiplications needed
for polynomial evaluation without preconditioning
(part of Theorem 7) follows from this theorem as a
corollary, for here \( \Phi \) is the \( 1 \) by \( n+1 \) matrix

\[ (1, x, x^2, \ldots, x^n) \]

and the columns of \( \Phi \) are all linearly independent so
that \( u = n \).

We also have another corollary:

**Theorem 11:**

Let \( X \) be a \( p \) by \( g \) matrix and let \( y \) be a \( q \) column
vector. Then to compute \( Xy \) requires at least \( pq \)
multiplications/divisions and so the ordinary method
of computing \( Xy \) minimizes the number of multiplica­
tions/divisions.

This follows from Theorem 10 by defining

\[ \Phi_{ij} = \begin{cases} y_k & \text{if } j = iq + k \quad 1 \leq k \leq q \\ 0 & \text{otherwise} \end{cases} \]

and letting \( z = (x_{11}, \ldots, x_{1q}, x_{21}, \ldots, x_{2q}, \ldots, x_{pq}) \).

Fiduccia Fid71 proved a theorem similar to Winograd's,
but involving submatrices rather than columns:

**Theorem 12:**

Let \( \Phi, \phi, x \) and \( F \) be as in Theorem 10. If \( \Phi \) has a
\( u \) by \( v \) submatrix \( S \) such that there are no nontrivial
vectors \( \alpha \) and \( \beta \) such that \( \alpha \beta \) is in zero, then at least
\( u+v-1 \) multiplications/divisions are required to com­
pute \( \Phi x + \phi \).

Immediate corollaries to this theorem are that at
least three real multiplications are required to compute
the product of two complex numbers (also proved by
Munro Mu71a) and that at least seven real multiplications
are required to compute the product of two quaternions.

Winograd Win70 similarly generalized Motzkin's result
(part of Theorem 8) on the number of multiplications
when preconditioning is allowed:

**Theorem 13:**

Let \( \Phi, \phi, x \) and \( F \) be as in Theorem 10. If there are
\( u \) column vectors in \( \Phi \) such that no nontrivial linear
combination of them (over \( F \)) is in \( F^n \), then any scheme
with preconditioning computing \( \Phi x + \phi \) requires at least
\( \lceil (u+1)/2 \rceil \) multiplications/divisions.

Motzkin's result follows from this exactly as Pan's
followed from Theorem 10, and we have a corollary similar
to Theorem 11:

**Theorem 14:**

Let \( X \) and \( y \) be as in Theorem 11, then every algorithm
for computing \( Xy \) requires at least \( pq/2 \) multiplica­
tions/divisions which do not depend only on the entries of \( X \)
or only on the entries of \( y \).

Winograd Win70 showed the possibility of approaching
the lower bound given in Theorem 14, by giving an
algorithm, which uses preconditioning, to compute \( Xy \)
in \( \lceil g/2 \rceil + \lceil q/2 \rceil \) multiplications; this algorithm then
leads to an algorithm to multiply two \( n \) by \( n \) matrices
in \( \lceil n^2/2 \rceil + 2n \lceil n/2 \rceil \) or approximately \( n^2/2 \) multi­
plications; Winograd's result is somewhat surprising.
since the usual method of matrix multiplication, that is by the definition, requires \( n^3 \) multiplications, and it had not been thought that this could be diminished.

This work was soon followed by an astonishing result of Strassen, who showed that two \( n \) by \( n \) matrices could be multiplied using only \( 4.7n^{\log_25} \) (about \( 4.7n^{2.81} \)) arithmetic operations. Strassen's method is based on a clever trick by which 2 by 2 matrices are multiplied using only seven scalar multiplications (instead of eight) and eighteen scalar additions:

\[
\begin{pmatrix}
  a & b \\
  c & d
\end{pmatrix}
\begin{pmatrix}
  w & x \\
  y & z
\end{pmatrix} =
\begin{pmatrix}
  (a+d)(w+z) + (b-d)(y+z) + d(y-w) - (a+b)z + a(x-z) \\
  (c+d)w + d(y-w) + a(x-z) - (c-a)(w+x)
\end{pmatrix}
\]

Since this trick does not make use of commutivity of multiplication, it follows that the method generalizes to higher order matrices by decomposing them into blocks. Strassen goes on to apply his methods to matrix inversion, computing the determinant, and solving linear systems of equations and he shows that each of these can be done in \( O(n^{\log_25}) \) arithmetic operations, provided certain submatrices are nonsingular.

It is not, in general, known whether or not Strassen's method is optimal. Hopcroft and Kerr have worked on this problem and they give a generalization of Strassen's method for multiplying \( m \) by 2 times 2 by \( n \) matrices which requires \( \lceil (3m+1)n/2 \rceil \) multiplications. They then show that this number of multiplications is minimal for the cases \( n = m = 3 \) and \( m = 2, n \) arbitrary; the optimality of Strassen's method for 2 by 2 matrices follows immediately from their results.

Several years prior to the work of Strassen and Winograd, Kljuev and Kokovkin-Scheherbak had approached the problem of the solution of an \( n \) by \( n \) linear system in a different manner. Using a detailed examination of the number and placement of zeroes in the matrix, they proved

**Theorem 15:**

If only operations on entire rows are permitted then \( \frac{1}{2}n(n-1)(2n-1) + n^2 \) additions/subtractions and \( \frac{1}{2}n(n^2+3n-1) \) multiplications/divisions are required to solve an \( n \) by \( n \) system of linear equations.

Since these are exactly the numbers of operations required by Gaussian elimination, we have as a corollary that Gaussian elimination is optimal, when one is restricted to operating on entire rows. Strassen's method is faster, but it uses operations on submatrices rather than on rows.

MISCELLANEOUS PROBLEMS

This section is devoted to a potpourri of results, which stand more or less alone, without a general frame of reference.

**Nonlinear equations**

Given a nonlinear equation, the problem is to approximate the solution to within a prescribed accuracy using arithmetic operations. Vashakmadze has established some lower bounds on the minimum number of operations necessary to approximate solutions to certain differential equations, and Emel'yanov and Il'in studied the same question for certain integral equations.

Various results have been discovered concerning the optimality of iterative root finding methods such as the secant method or Newton's method. For example, Rissanen has shown that the secant method is, in a sense, optimal among all algorithms which use the "same amount of information" and which also satisfy a certain "smoothness" condition. Much work has also been done to find the best starting values for Newton's method applied to square roots; for example, References Mou67, Kin69, and Ster69. Recently, Paterson (personal communication), using the "standard" definition of the efficiency of an iterative scheme, has shown Newton's method to be optimal, for the calculation of square roots (in the sense that no rational scheme can have greater efficiency).

**Scalar arithmetic**

So far, we have been concerned only with how many operations need to be performed, not considering the time required for individual operations. The usual method for adding/subtracting two \( n \) digit numbers requires time proportional to \( n \) and the usual method of multiplication requires time proportional to \( n^2 \). Can
either of these methods be improved upon? For addition/subtraction no substantial improvement is possible since the usual time is about \( n \) "cycles," and there are \( 2n \) inputs (digits) while on each cycle one can use at most two of the inputs. Multiplication, however, can be done more quickly than the usual method. Karatsuba and Ofman developed a method which requires time proportional to \( n \log n \). Toom generalized this algorithm and proved that for all \( \epsilon > 0 \) there is a multiplication algorithm such that the time required to multiply two \( n \) digit numbers is \( O(n^{1+\epsilon}) \). Schonhage and Strassen devised a different algorithm which requires at most time \( O(n \log n \log \log n) \); this is the most efficient algorithm known, but it has not been proved optimal.

The only non-trivial optimality result is due to Cook and Aanderaa where they proved that on a bounded activity machine, an on-line "super" Turing machine, multiplication cannot be performed in less than time \( O(n \log n / \log \log n) \).

Maximization of unimodal functions

A unimodal function of one variable is function \( f \) which has a unique maximum \( z \); it is characterized by

\[
\begin{align*}
    x < y < z &\Rightarrow f(x) < f(y) < f(z) \\
    x > y > z &\Rightarrow f(x) > f(y) > f(z)
\end{align*}
\]

Suppose we want to locate, to within a unit interval, this unique maximum; what is the smallest number of function evaluations required? This question was first studied by Kiefer. He showed that if we are to locate the maximum over an interval \([0, L]\) and \( F_i \leq L < F_{i+1} \), where \( F_i \) is the \( i \)th Fibonacci number, then \( n+1 \) function evaluations are necessary and sufficient. Karp and Miranker characterized optimal strategies when parallel function evaluations are allowed.

REFERENCES

Arb69 M A ARBIB
Theories of abstract automata
Prentice-Hall Inc Englewood Cliffs NJ 1969
section 3.2

Arl70 V L AR LazAROV E A DINIC
M A KRONROD I A FARADZHEV
On economical construction of the transitive closure of an oriented graph
Dokl Akad Nauk SSSR 194 1970 pp 487-488 (Russian)
English translation in Soviet Math Dokl 11 1970 pp 1209-1210
Be58  E G BELAGA  
Some problems involved in the calculation of polynomials  
Dokl Akad Nauk SSSR 123 1958 pp 775-777 (Russian)  
See Math Reviews 21 1960 review number 3935

Be61  
On computing polynomials in one variable with initial conditioning of the coefficients  

Bo71  A BORODIN  
Horner's rule is uniquely optimal  
Theory of Machines and Computations  
Edited by Z Kohavi and A Paz  
Computations Haifa Israel 1971

Br39  A BRAUER  
On addition chains  
Bull Amer Math Soc 45 1939 pp 736-739

Co69  S A COOK S O AANDERAA  
On the minimum computation time of functions  
Trans Amer Math Soc 142 1969 pp 219-314

Em67  K V EMEL'YANOV A M IL'IN  
Number of arithmetical operations necessary for the approximate solution of Fredholm integral equations of the second kind  

Ev64  J EVE  
The evaluation of polynomials  
Numer Math 6 1964 pp 17-21

Fid71  C M FIDUCCIA  
Fast matrix multiplication  
Proc of Third Annual ACM Symp on Theory of Computing 1971 pp 45-49

Fis71  M J FISCHER A R MEYER  
Boolean matrix multiplication and transitive closure  
IEEE Conf Record of the Twelfth Annual Symps on Switching and Automata Theory 1971 pp 129-131

Fo59  L R FORD JR S M JOHNSON  
A tournament problem  
Amer Math Monthly 66 1959 pp 387-389

Fu70  M E FURMAN  
Application of a method of fast multiplication of matrices in the problem of finding the transitive closure of a graph  

G156  A GLEASON  
Unpublished internal IBM memorandum 1956

Gr71  R L GRAHAM  
Sorting by comparisons  

Ha69a  A HADIAN  
Optimality properties of various procedures for ranking n different numbers using only binary comparisons  
Technical Report Number 117 Department of Statistics University Minnesota Minneapolis Minnesota 1969

Ha69b  M SOBEL  
Selecting the tth largest using binary errorless comparisons  
Colloquia Mathematica Societatis Janos Bolyai  
Budapest Hungary 1969 pp 585-589

Ha70  
Ordering the t largest of n items using binary comparisons  
Proc Second Chapel Hill Conf on Combinatorial Math and its Application 1970

Ho170  R C HOLT E M REINGOLD  
On the time required to detect cycles and connectivity in directed graph  
To appear in Math Systems Theory 1972

Hop71a  J HOPCROFT L KERR  
On minimizing the number of multiplications necessary for matrix multiplication  
SIAM J Appl Math 20 1971 pp 30-36

Hop71b  R TARJAN  
Planarity testing in VlogV steps  
Information Processing 1971 Proc of IFIP Congress 71 1971

Hop71e  R M KARP  
A n^{5/2} algorithm for maximum matchings in bipartite graphs  
J Combinatorial Theory 4 1968 pp 19-35

Kie53  J KIEFER  
Sequential minimax search for a maximum  
J Soc Indust Appl Math 5 1957 pp 105-137

Kie57  
Optimum sequential search and approximation methods under minimum regularity assumptions  
J Soc Indust Appl Math 5 1957 pp 105-137

Kin69  R F KING D L PHILLIPS  
The logarithmic error and Newton's method for the square root  
Comm ACM 12 1969 pp 87-88

Kis62  S S KISLICYN  
On a bound for the least mean number of pairwise comparisons needed for a complete ordering of n objects with different weights  
Vestnik Leningrad Univ Ser Mat Meh Astronom 17 1962 pp 162-163 (Russian with English summary)

Kis63  
An improved bound for the least mean number of comparisons needed for the complete ordering of a finite set  
Vestnik Leningrad Univ Ser Mat Meh Astronom 18 1963 pp 143-145 (Russian with English summary) English translation in Document Number AD68523 available from the Clearinghouse Department of Commerce (this is an edited machine translation by the USAF 1967) pp 178-181

From the collection of the Computer History Museum (www.computerhistory.org)
Kis64 ____________ On the selection of the kth element of an ordered set by pairwise comparisons
Sibirsk Mat Zh 5 1964 pp 557-564 (Russian) See Math Reviews 29 1965 review number 2198

Kl65 V V KILUEV N I KOKOVIN-SCHERBAK
On the minimization of the number of arithmetic operations for the solution of linear algebraic systems of equations

Kl66 ____________ Minimization of the number of arithmetical operations in a transformation of matrices
Ukrain Mat Zh 18 1966 pp 122-128 (Russian) See Math Reviews 34 1967 review number 3767

Kl67 ____________ Minimization of computational algorithms in certain transformations of matrices

Kn62 D E KNUTH
Evaluation of polynomials by computers
Comm ACM 6 1962 pp 585-590

Kn69 ____________ The art of computer programming, Volume 2
Addison-Wesley Reading Massachusetts 1969 sections 4.3.3 4.6.3 and 4.6.4

Kn72 ____________ The art of computer programming, Volume 3
Addison-Wesley Reading Massachusetts to appear 1972 section 5.3

Ko68 N I KOKOVIN-SCHERBAK
Minimization of computational algorithms in the solution of the elimination problem

Ko70 ____________ On minimization of calculation algorithms in solutions of arbitrary systems of linear equations
Ukrain Mat Zh 22 1970 pp 494-502 (Russian)

Ma71 K MARUYAMA
Parallel methods and bounds of evaluating polynomials
Report Number 437 Department of Computer Science University of Illinois Urbana Illinois 1971

Mot55a T S MOTZKIN
Evaluation of polynomials
Bull Amer Math Soc 61 1955 p 163 abstract only

Mot55b ____________ Evaluation of rational functions
Bull Amer Math Soc 61 1955 p 163 abstract only

Mou67 D G MOURSUND
Optimal starting values for Newton-Raphson calculation of $e^x$
Comm ACM 10 1967 pp 430-432

Mu71a I MUNRO
Some results concerning efficient and optimal algorithms
Proc of Third Annual ACM Symp on Theory of Computing 1971 pp 40-44

Mu71b M S PATERSON
Optimal algorithms for parallel polynomial evaluation
IEEE Conf Record of the Twelfth Annual Symps on Switching and Automata Theory 1971 pp 132-139

Mu71c ____________ Efficient determination of the transitive closure of a directed graph
Information Processing Letters 1 1971 pp 56-58

Of62 YU OFMAN
On the algorithmic complexity of discrete functions

Os54 A M OSTROWSKI
On two problems in abstract algebra connected with Horner’s rule

Pan59 V YA PAN
Schemes for computing polynomials with real coefficients

Pan62 ____________ On several ways of computing values of polynomials

Pan65 ____________ The computation of polynomials of fifth and seventh degree with real coefficients
Zh Vychisl Mat i Mat Fiz 5 1965 pp 116-118 (Russian). English translation in USSR Comput Math and Math Phys 5 1965 pp 159-1611

Pan66 ____________ Methods of computing values of polynomials

Pat71 M S PATERSON L STOCKMEYER
Bounds on the evaluation time for rational polynomials
IEEE Conf Record of the Twelfth Annual Symps on Switching and Automata Theory 1971 pp 140-143

Po69 I POHL
A minimum storage algorithm for computing the median
Report number RC2701 IBM Yorktown Heights New York 1969

Ra71 M O RABIN
Proving simultaneous positivity of linear forms
Third Annual ACM Symp on Theory of Computing 1971 Invited address
Establishing Lower Bounds on Algorithms

Ri71 J Rissanen
On optimal root-finding algorithms
J Math Anal Appl 36 1971 pp 220-225

Sa61 M Sandelius
On an optimal search procedure
Amer Math Monthly 68 1961 pp 133-134

Scho71 A Schonhage V Strassen
Fast multiplication of large numbers
Computing 7 1971 pp 281-292 (German with English summary)

Schre72 J Schreier
On tournament eliminations systems
Matheesis Polska 7 1932 pp 154-100 (Polish)

Sl51 J Slupecki
On the system S of tournaments
Collog Math 2 1951 pp 286-290

Sm47 C A B Smith
The counterfeit coin problem
Math Gaz 31 1947 pp 31-39

Sp69 P M Spiro
The time required for group multiplication

Stein50 H Steinhaus
Mathematical snapshots
Oxford University Press New York first edition 1950
second edition 1960

Stein58
Some remarks about tournaments
Calcutta Mathematical Society Golden Jubilee
Commemoration Vol 1958-1959 Part II pp 323-327

Ster69 P H Sterbenz C T Fike
Optimal starting approximations for Newton's method
Math Comp 23 1969 pp 313-318

Str69 V Strassen
Gaussian elimination is not optimal

Ts71 R Tarjan
Ph D Thesis Stanford 1971

Todd55 J Todd
Motivation for working in numerical analysis
Comm Pure Appl Math 8 1955 pp 97-116

Too63 A L Toom
The complexity of a scheme of functional elements
realizing the multiplication of integers
English translation in Soviet Math Dokl 4 1963 pp 714-716

Val59 R E Val'skii
The smallest number of multiplications necessary to
raise a number to a given power
Problemy Kibernet 2 1959 pp 73-74 (Russian).
English translation in Problems of Cybernetics 2
1961 pp 395-397

Van60 M H Van Emden
Increasing the efficiency of quicksort
Comm ACM 13 1970 pp 563-567

Vas69 T S Vashakmadze
On some optimal algorithms
Tbilisi Sahelmc Univ Gamoqeneb Math Inst Srom
Math Gaz 31 1947 pp 31-39 1 1969 pp 7-16 (Russian with Georgian summary)

Stein58 65 1965 pp 497-498

Win65 S Winograd
Applications of a language for computing in
combinatorics
Information Processing 1965 Proc of IFIP Congress
65 1965 pp 497-498

Wi67 S Winograd
On the time required to perform addition
J Assoc Comput Mach 12 1965 pp 277-285

Wi68 S Winograd
On the time required to perform multiplication
J Assoc Comput Mach 14 1967 pp 793-802

Wi70 S Winograd
How fast can computers add?
Scientific American October 1968 pp 93-100

From the collection of the Computer History Museum (www.computerhistory.org)