1975

An Experiment Comparing Fortran Programming Times with the Software Physics Hypothesis

R. D. Gordon
M. H. Halstead

Report Number:
75-167

http://docs.lib.purdue.edu/cstech/113

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epubs@purdue.edu for additional information.
AN EXPERIMENT COMPARING FORTRAN PROGRAMMING TIMES
WITH THE SOFTWARE PHYSICS HYPOTHESIS

R. D. Gordon
M. H. Halstead

TR 167
October 17, 1975

INTRODUCTION

Recent discoveries in the area of Algorithm Structure or Software
Physics [1-25] have produced a number of hypotheses. One of these relates
the number of elementary mental discriminations required to implement an
algorithm to measurable properties of that algorithm, and the results of
one set of experiments confirming this relationship have been published
[16]. That publication, while significant, made no claim to finality,
suggesting instead that further experiments were warranted. This paper
will present the results of a second set of experiments, having the
advantages of being conducted in a single implementation language, Fortran,
from problem specifications readily available in computer textbooks.

Section I of this report presents the timing hypothesis, and the
elementary equations upon which it rests. Section II presents the details
of the experiment and the results which were obtained, and Section III
contains an analysis of the data.
SECTION I - TIMING HYPOTHESIS

Measurable properties of any implementation of any algorithm include:

\( n_1 \) = The count of distinct operators
\( n_2 \) = The count of distinct operands

(variables or constants)

\( N_1 \) = Total uses of operators
\( N_2 \) = Total uses of operands

The vocabulary, \( n \), is given by:

\[
\begin{align*}
n &= n_1 + n_2 \\
N &= N_1 + N_2
\end{align*}
\]

and the length, \( N \), is:

\[
N = N_1 + N_2
\]

From these properties, it is possible to obtain the volume, \( V \), in bits, as:

\[
V = N \log_2 n
\]

and the implementation level, \( L \), where \( L \leq 1 \), as:

\[
L = \frac{n_1^{\pm}}{n_1} \frac{n_2}{N_2}
\]

where \( n_1^{\pm} \), the minimum possible number of operators, will equal 2 for most algorithms. (One for the name of a function, plus one for a grouping symbol operator). It has been shown [4] that the product \( L \times V \) is invariant under translation from one language to another, and that for programs without impurities [3,6,8]:

\[
N = n_1 \log_2 n_1 + n_2 \log_2 n_2
\]

From this point, the following nine steps yield the timing equation:

1. A program consists of \( N \) selections from \( n \) elements.
2. A binary search of \( n \) elements requires \( \log_2 n \) comparisons.

3. A program is generated by making \( N \log_2 n \) comparisons.

4. Therefore, the volume, \( V \), is a count of the number of comparisons required.

5. The number of elementary mental discriminations required to complete one comparison measures the difficulty of the task.

6. The level, \( L \), is the reciprocal of the difficulty.

7. Therefore, \( E \), the count of elementary mental discriminations required to generate a program, is given by:

\[
E = \frac{V}{L}
\]  

(6)

8. \( S \), the speed with which the brain makes elementary mental discriminations can be obtained from psychology [26] as:

\[ 5 \leq S \leq 20 \text{ discriminations per second.} \]

9. Therefore, the time to generate a preconceived program, by a concentrating programmer, fluent in a language, is:

\[
\hat{T} = \frac{V}{SL}
\]  

(7)

Equation 7 may be expressed in more basic terms by substituting for \( V \) from equation 3, and for \( L \) from equation 4, with \( \eta_1^x = 2 \), giving:

\[
\hat{T} = \frac{n_1 n_2 N \log_2 n}{2 S \eta_2}
\]  

(8)

The effect of possible impurities [5] may be eliminated from equation 8 by substituting for \( N \) from equation 5. Letting \( S = 60 \times 18 = 1080 \) will then give, for time in minutes:

\[
\hat{T} = \frac{n_1 n_2 (n_1 \log_2 \eta_1 + n_2 \log_2 \eta_2) \log_2 n}{2160 \eta_2}
\]  

(9)

Each of the variables on the right hand side of equation 9 can be readily measured (or counted) in any computer program, and the experiment described in the next section was designed to compare results from that equation with observed programming times.
SECTION II. EXPERIMENTAL PROCEDURE

Eleven problems were arbitrarily selected from two published sources. In selecting candidates for the experiment, problems were sought which were stated in a non-procedural form. Further, the problem statement had to be complete. That is, in the course of solving a particular problem, specific laws of physics, mathematics, etc. would not have to be derived. The problems finally selected were taken from Knuth [27], and from Maurer and Williams [28], and cover a wide range of topics including character manipulation, list processing, simulation experiment, and mathematical analysis. The source of each problem statement is cited in Table 1.

On each of eleven days, one of these problems was implemented by the senior author. In order to maintain a consistent level of performance all work was conducted in a quiet room, free from distractions, during the same period of the day. The time required to fully implement the problem was obtained. This total time included the number of minutes spent reading the statement of the problem, preparing flowcharts and writing preliminary versions of the code, writing the final version of the code, desk checking, and the time spent working to correct errors in the program. Time to keypunch was not included.

For a number of reasons, including availability and fluency, all of the algorithms were implemented in Fortran. In the course of solving a problem the correctness of the implementation was checked by executing a sufficiently complex test case for which a correct answer was known. In some cases the solution to a problem was written as a subroutine and testing required that a main routine be written. In such a case only the preparation of the subroutine was considered for the experiment.
In addition, several implementations made use of subroutines previously written. Such routines were also not included. The complete text of each of the eleven programs is included in Appendix A.

After each program was completed, a careful count was made to determine values of $n_1$, $n_2$, $N_1$ and $N_2$. In obtaining these values all read, write, declarative statements and comments were ignored. The results are shown in Table I.
<table>
<thead>
<tr>
<th>No.</th>
<th>Program Specifications</th>
<th>Software Parameters</th>
<th>Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ref.*</td>
<td>Page</td>
<td>Problem</td>
</tr>
<tr>
<td>G1</td>
<td>K</td>
<td>158</td>
<td>21</td>
</tr>
<tr>
<td>G2</td>
<td>K</td>
<td>159</td>
<td>23</td>
</tr>
<tr>
<td>G3</td>
<td>K</td>
<td>196</td>
<td>7</td>
</tr>
<tr>
<td>G4</td>
<td>K</td>
<td>377</td>
<td>17</td>
</tr>
<tr>
<td>G5</td>
<td>K</td>
<td>158</td>
<td>22</td>
</tr>
<tr>
<td>G6</td>
<td>K</td>
<td>154</td>
<td>10</td>
</tr>
<tr>
<td>G7</td>
<td>M</td>
<td>32</td>
<td>3.2.21</td>
</tr>
<tr>
<td>G8</td>
<td>M</td>
<td>32</td>
<td>3.2.23</td>
</tr>
<tr>
<td>G9</td>
<td>M</td>
<td>88</td>
<td>8.3.2</td>
</tr>
<tr>
<td>G10</td>
<td>M</td>
<td>89</td>
<td>8.3.4</td>
</tr>
<tr>
<td>G11</td>
<td>M</td>
<td>27</td>
<td>3.2.4</td>
</tr>
</tbody>
</table>

*K = Knuth [27], M = Maurer and Williams [28].
SECTION III - ANALYSIS OF THE DATA

The programming time predicted by theory was obtained for each program by applying equation 9 to the data in Table I. This result, \( T \), can be compared with the observed value, \( T \), in Table 2. In addition, a count of the number of statements in each program was obtained, and the programs were ordered according to these values.

The average of the calculated values, 34 minutes, is fortuitously close to the observed value, 35 minutes. The coefficient of correlation is 0.934, only slightly smaller than the value of 0.952 reported in an earlier experiment [16]. In further agreement with that experiment, the correlation between length and observed times, 0.887, is lower than between observed and calculated times.

In conclusion, it may again be observed that one more set of experimental data do not contradict the simple hypothesis. As a result, further carefully controlled experiments by others would appear to be warranted.
<table>
<thead>
<tr>
<th>Program Number</th>
<th>Statement Count</th>
<th>Programming Time-Minutes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$T_{observed}$</td>
</tr>
<tr>
<td>G7</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>G8</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>G5</td>
<td>11</td>
<td>21</td>
</tr>
<tr>
<td>G6</td>
<td>15</td>
<td>30</td>
</tr>
<tr>
<td>G3</td>
<td>18</td>
<td>16</td>
</tr>
<tr>
<td>G1</td>
<td>18</td>
<td>19</td>
</tr>
<tr>
<td>G8</td>
<td>18</td>
<td>24</td>
</tr>
<tr>
<td>G4</td>
<td>32</td>
<td>39</td>
</tr>
<tr>
<td>G2</td>
<td>36</td>
<td>92</td>
</tr>
<tr>
<td>G9</td>
<td>38</td>
<td>43</td>
</tr>
<tr>
<td>G10</td>
<td>59</td>
<td>91</td>
</tr>
</tbody>
</table>

Means: $35.0$ $34.1$
REFERENCES:


Additional References:


C
C DIMENSION MAGIC(23,23)
DATA MAGIC /529*0/
C
N=1
IR=1
IC=23/2 + 1
C 100 MAGIC(IR,IC)=N
C IF (N.EQ.529) GO TO 900
N=N+1
C JR=IR-1
IF (JR.LT.1) JR=23
JC=IC-1
IF (JC.LT.1) JC=23
IF (MAGIC(JR,JC).EQ.0) GO TO 200
C JR=IR+1
IF (JR.GT.23) JR=1
JC=IC
C 200 IR=JR
IC=JC
GO TO 100
C C PRINT MAGIC SQUARE
C 900 DO 920 IR=1,23,1
920 WRITE (6,1000) (MAGIC(IR,IC), IC=1,23,1)
STOP
C 1000 FORMAT(23I4)
END
INTEGER DIGIT(10)
DATA DIGIT /1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9/
DIMENSION MAT(24,24)
DATA MAT /576*-1/
INTEGER WHITE(3,5)
DATA WHITE /1H,1H,1H+,
$ 1H,1H,1H+, 
$ 1H,1H,1H+, 
$ 1H,1H,1H+,
$ 1H,1H,1H+,
$ 1H+,1H+,1H+/
INTEGER EDGE(3,5,5)
DATA EDGE /1H,1H,1H+,
$ 1H,1H,1H+, 
$ 1H,1H,1H+, 
$ 1H,1H,1H+, 
$ 1H,1H,1H+, 
$ 1H,1H,1H+, 
$ 1H+,1H+,1H+, 
$ 1H+,1H+,1H+, 
$ 1H+,1H+,1H+, 
$ 1H+,1H+,1H+, 
$ 1H+,1H+,1H+/
INTEGER OUT(120,120)
COMMON OUT
READ INPUT MATRIX
DO 110 IR=2,23,1
110 READ (5,1000) (MAT(IR,IC), IC=2,23,1)
C
C DELETE +1 ADJACENT TO -1
C
C DO 220 IR=2,23,1
JR=25-IR
DO 210 IC=2,23,1
JC=25-IC
IF (MAT(IR,IC).NE.1) GO TO 200
IF (MAT(IR+1,IC).EQ.-1 .OR. MAT(IR+IC-1).EQ.-1)
$  MAT(IR,IC)=-1
200 IF (MAT(JR,JC).NE.1) GO TO 210
IF (MAT(JR+1,JC).EQ.-1 .OR. MAT(JR+JC+1).EQ.-1)
$  MAT(JR,JC)= -1
210 CONTINUE
220 CONTINUE
DEBUG PRINTOUT

DO 300 IR=1,24,1
    WRITE (6,90000) (MAT(IR,IC), IC=1,24,1)

MOVE TO OUTPUT ARRAY

N=0
DO 380 IR=1,23,1
    JR=(IR-1)*3
    DO 370 IC=1,23,1
        JC=(IC-1)*5
    END
    IF (MAT(IR,IC).NE.1) GO TO 325
    DO 310 I=1,3,1
        DO 310 J=1,5,1
            OUT(JR+I,JC+J)=1H+
        END
        GO TO 370
    END

BLACK SQUARE

DO 325 IF (MAT(IR,IC).EQ.0) GO TO 350
    MODE=1
    IF (MAT(IR,IC+1).EQ.0) MODE=MODE+1
    IF (MAT(IR+1,IC).EQ.0) MODE=MODE+2
    IF (MODE.EQ.1 .AND. MAT(IR+1,IC+1).EQ.0) MODE=5
    DO 330 I=1,3,1
    DO 330 J=1,5,1
    END
    OUT(JR+I,JC+J)=EDGE(I,J,MODE)
    GO TO 370

EDGE SQUARE

DO 350 I=1,3,1
    DO 350 J=1,5,1
    END
    OUT(JR+I,JC+J)=WHITE(I,J)

WHITE SQUARE

NUMBERED...

IF (.NOT. (MAT(IR+1,IC).EQ.0 .AND. MAT(IR-1,IC).NE.0 .OR. MAT(IR,IC-1).NE.0 .AND. MAT(IR,IC+1).EQ.0)) GO TO 370
    N=N+1
    NT=N/10
    NU=N-NT*10
    OUT(JR+1,JC+1)=DIGIT(NT+1)
    OUT(JR+1,JC+2)=DIGIT(NU+1)

CONTINUE

CONTINUE

OUTPUT PUZZLE

WRITE (6,3000)
DO 500 I=1,120,1
    WRITE (6,2000) (OUT(I,J), J=1,120,1)
    STOP

FOR MAT(23I1)
2000 FORMAT(1X,120A1)
3000 FORMAT(1H1,/,1H1)
9000 FORMAT(24I3)
END
TEST PROGRAM 3

INTEGER C, CO, PI, PO
INTEGER INPUT(8), OUTPUT(8), DIGIT(10)
DATA OUTPUT /8*1H/
DATA DIGIT /IH0, 1H1, 1H2, 1H3, 1H4, 1H5, 1H6, 1H7, 1H8, 1H9/
INTEGER GETCH
LOGICAL NUMERIC
NUMERIC(I)=I.GE.1H0 .AND. I.LE.1H9

READ (5,1000) INPUT
1000 FORMAT(8A10)
WRITE (6,2000) INPUT
2000 FORMAT(#0INPUT IS #,8A10)

PI=1
PO=1
CO=GETCH(INPUT,PI)
PI=PI+1

100 KOUNT=1
200 C=GETCH(INPUT,PI)
PI=PI+1

IF (C.EQ.1H ) GO TO 200
IF (C.NE.C0) GO TO 300
KOUNT=KOUNT+1
GO TO 200

300 IF (KOUNT.EQ.1 .AND. .NOT.NUMERIC(C0)) GO TO 400
CALL PUTCH (DIGIT(KOUNT), OUTPUT, PO)
PO=PO+1

400 CALL PUTCH (C0, OUTPUT, PO)
PO=PO+1
C0=C

IF (C0.NE.1H.) GO TO 100
CALL PUTCH (C0, OUTPUT, PO)
WRITE (6,3000) OUTPUT
3000 FORMAT(#0OUTPUT IS #,8A10)
STOP
END

THE ROUTINES GETCH AND PUTCH WERE NOT WRITTEN AS PART OF THIS EXPERIMENT.

INTEGER FUNCTION GETCH(IWORO, IPOS)

THE FUNCTION GETCH RETURNS A 6 BIT CHARACTER FROM A CHARACTER STRING OF POSSIBLY SEVERAL WORDS IN LENGTH. THE VALUE RETURNED IS LEFT JUSTIFIED, BLANK FILLED. CHARACTERS ARE NUMBERED LEFT TO RIGHT, 1, 2, ..., .

AUTHOR: KEVIN KOLIS (23 JAN 75)

DIMENSION IWORO(1)
SUBROUTINE PUTCM (CHAR, STRING, POS)

SUBROUTINE PUTCM PLACES A GIVEN CHARACTER INTO A STRING AT THE SPECIFIED POSITION. CHARACTERS ARE NUMBERED LEFT TO RIGHT, 1, 2, ... .

AUTHOR: RONALD GORDON (24 JAN 75)

INTEGER CHAR, STRING(I); POS:

IW= (POS-1)/10+1
IC= (1-POS+IW*10) #6
M= SHIFT(00 77 77 77 77 77 77 77 77, IC)
STRING(IW) = OR( AND(STRING(IW); M) AND(SHIFT(CHAR, IC) COMPL(M)) )

RETURN

END
INTEGER PICK, CARD
LOGICAL DEBUG
INTEGER POP
INTEGER DECK(52)
INTEGER KOUNT(53)
DATA KOUNT /53*0/
INTEGER PILE(13,5)
COMMONPILE

DEBUG=.TRUE.
PC=RANF(13,0)

K=1
DO 100 I=1,4,1
DO 100 J=1,13,1
DECK(K)=I*100+J
100 K=K+1

DO 900 N=1,500,1
DO 150 I=1,13,1
150 PILE(I,1)=1
CALL SHUFFLE(DECK,52)

K=1
DO 200 I=1,4,1
DO 200 J=1,13,1
CALL PUSH(DECK(K),J)
200 K=K+1
IF (DEBUG) WRITE (6,1000) PILE
1000 FORMAT(18F12.5,/1X,13I5,/,)

L=53
PICK=1
300 CARD=POP(PICK)
IF (CARD.EQ.0) GO TO 400
L=L-1
IF (DEBUG) WRITE (6,2000) CARD, PICK
2000 FORMAT(14I4,# Picked FROM#,I3)
PICK=MOD(CARD,100)
GO TO 300

400 KOUNT(L)=KOUNT(L)+1
K=L-1
IF (DEBUG) WRITE (6,3000) K
3000 FORMAT(17I3,# CARDS LEFT#)
900 CONTINUE

WRITE (6,5000)
DO 920 I=0.52,1
PC=FLOAT(KOUNT(I+1))/500.0*100.0
920 WRITE (6,4000) I, KOUNT(I+1), PC
4000 FORMAT(17I2,F11.2)
5000 FORMAT(1X,CARDS LEFT # TIMES PERCENT#)
STOP
END
SUBROUTINE PUSH (ITEM, PICK)
INTEGER PICK
INTEGER PILE(13,5)
COMMON PILE
PILE(PICK,1)=PILE(PICK,1)+1
PILE(PICK, PILE(PICK,1)) = ITEM
RETURN
END
INTEGER FUNCTION POP (PICK)
INTEGER PICK
INTEGER PILE(13,5)
COMMON PILE
IF (PILE(PICK,1) .EQ. 1) GO TO 100
POP=PILE(PICK, PILE(PICK,1))
PILE(PICK,1)=PILE(PICK,1)-1
RETURN
C STACK EMPTY
100 POP=0
RETURN
END
C C
C SUBROUTINE SHUFFLE WAS NOT WRITTEN AS PART OF THIS EXPERIMENT.
C
C SUBROUTINE SHUFFLE (LIST, N)
DIMENSION LIST(N)
C
THIS ROUTINE WILL RANDOMLY SHUFFLE A LIST OF ITEMS.
C
REF: KNUTH, VOL. 2, P. 125, ALGORITHM P.
C
J=N
100 U=RANF(0.0)
K=FLOAT(J)*U+1.0
KEEP=LIST(K)
LIST(K)=LIST(J)
LIST(J)=KEEP
J=J-1
IF (J.GT.1) GO TO 100
RETURN
END
TEST PROGRAM 5

CREATE CIRCULAR LIST

KILL N-1 MEN

COUNT M MEN

KILL CURRENT MAN

KILL LAST MAN

KILL (L1) = N
WRITE (6, 1000) (KILL(I), I=1, N, 1)
1000 FORMAT (40I3)
STOP
END

*EOR
0, 4
THE MAIN PROGRAM WAS NOT WRITTEN AS PART OF THIS EXPERIMENT.

DIMENSION MATRIX (9,8)
READ (5,1000) MATRIX
WRITE (6,1001) MATRIX
CALL SADDLE (MATRIX, I, J)
PRINT, I, J
STOP
1000 FORMAT(9(I1,I1))
1001 FORMAT(9T2)
END

SUBROUTINE SADDLE (MAT, I, J)
DIMENSION MAT(9,8)
DIMENSION KEEPPR(9), KEEP(8)

160 IR=1,9,1
MIN=MAT(IR,1)
150 CONTINUE
160 KEEP(PR,IR)=MIN

260 IC=1,9,1
MAX=MAT(1,IC)
250 CONTINUE
260 KEEP(C,IC)=MAX

370 I=1,9,1
370 J=1,8,1
IF (KEEP(I),EQ,KEEP(J)) RETURN
370 CONTINUE
I=0
J=0
RETURN
END

142243011
246155222
256964032
356773343
356582055
351191445
459280075
465371586
INTEGER SUMDIV

SEARCH FOR SOME FRIENDLY NUMBERS

DO 100 N=1000,1500,1
M=SUMDIV(N)
IF (SUMDIV(M).NE.N) GO TO 100
WRITE (6,1000) N, M
100 CONTINUE
STOP
1000 FORMAT(I5,# AND#,I5,# ARE FRIENDLY. #)
END

INTEGER FUNCTION SUMDIV(N)

CALCULATE SUM OF DIVISORS OF N

SUMDIV=1
DO 100 I=2,N-1,1
IF (N/I*I.NE.N) GO TO 100
SUMDIV=SUMDIV+I
100 CONTINUE
RETURN
END
LOGICAL PRIME
COMMON LIST(100)

LIST(1)=2
LIST(2)=3
N=3
DO 110 I=3,100,1
100 N=N+1
IF (.NOT. PRIME(N)) GO TO 100
110 LIST(I)=N
WRITE (6,1000) LIST
N=1
DO 500 L=1,8
250 DO 300 I=0,L-1,1
IF (.NOT. PRIME(N+I)) GO TO 300
N=N+I+1
GO TO 250
300 CONTINUE
WRITE (6,2000) L, N
500 CONTINUE
STOP

1000 FORMAT(1X,20I6)
2000 FORMAT(1X,'SEQUENCE OF',I2,' BEGINS AT',I4)
END

LOGICAL FUNCTION PRIME(N)
COMMON LIST(100)

PRIME=.TRUE.
LIM=SQR( FLOAT(N ) ) + 0.5
DO 100 I=1,100,1
IF (LIST(I).GT.LIM) RETURN
IF (N/LIST(I).NE.N) GO TO 100
PRIME=.FALSE.
RETURN
100 CONTINUE
STOP 1
END
INTEGER GT
INTEGER WAIT, TOTAL
LOGICAL DEBUG
DO 100 GT=18,90,10
TOTAL=0
DEBUG=.TRUE.,
DO 110 I=1,3,1
CALL SIMUL (GT, WAIT, DEBUG)
TOTAL=TOTAL+WAIT
WRITE (6,1000) GT, WAIT
DEBUG=.FALSE.
110 CONTINUE
AVG=FLOAT(TOTAL)/3.0
WRITE (6,2000) AVG
100 CONTINUE
STOP
1000 FORMAT(I3, # SECOND GREEN LIGHT, WAIT =#,I6)
2000 FORMAT(# AVERAGE WAIT =#,F7.1,/) END

SUBROUTINE SIMUL (GT, WAIT, DEBUG)
INTEGER TIME, Q1, Q2, WAIT1, WAIT2, GT, RANDOM
LOGICAL DEBUG
INTEGER WAIT
INTEGER ON, OFF

Q1=0
Q2=0
WAIT1=0
WAIT2=0
TIME=0
LIGHT=1
ON=0
OFF=0
IF (DEBUG) WRITE (6,3000) GT

100 Q1=Q1+RANDOM(5,15)
Q2=Q2+RANDOM(6,24)
IF (DEBUG) WRITE (6,2000) TIME, LIGHT, Q1, WAIT1, Q2, WAIT2

C ADD TO QUEUES

C REMOVE FROM QUEUES IF GREEN
IF (LIGHT.EQ.1) GO TO 200
Q2=Q2-MIND(Q2,20)
GO TO 250

200 Q1=Q1-MIND(Q1,36)

C ACCUMULATE WAITING TIME

250 CONTINUE
WAIT1=WAIT1+Q1*10
WAIT2=WAIT2+Q2*10
IF (DEBUG) WRITE (6,2000) TIME, LIGHT, Q1, WAIT1, Q2, WAIT2
T
IM=TIME+10
IF (LIGHT.EQ.0) GO TO 300
ON=ON+10
IF (ON.EQ.GT) LIGHT=0
GO TO 400

300 OFF=OFF+10
IF (OFF+ON.NE.100) GO TO 400
C
400 IF (TIME.LT.300) GO TO 100
IF (DEBUG) WRITE (6,1000) Q1, WAIT1, Q2, WAIT2
WAIT=WAIT1+WAIT2
RETURN
C
1000 FORMAT(*GCARS LEFT IN Q1 =*,I3,*, WAITING TIME =*,I5,/, + *,CARS LEFT IN Q2 =*,I3,*, WAITING TIME =*,I5)
2000 FORMAT(6I10)
3000 FORMAT(*ISIMULATION OF*,I3,*, SECOND GREEN LIGHT*,// + *, TIME LIGHT Q1 WAIT1 Q2 WAIT2*)
END
INTEGER FUNCTION RANDOM (I, J)
X=J-I
RANDOM=X*RANF(0,0)
RANDOM=RANDOM+I
RETURN
END
DO 100 I=1,2,1
WRITE (6,1000)
CALL SIMUL (250)
100 CONTINUE
STOP
1000 FORMAT(#BEGIN SIMULATION#)
END

SUBROUTINE SIMUL (LIMIT)

INTEGER TIME, RANDOM, CO, WAIT, TOTAL
INTEGER Q(3,600), QL(3)
COMMON Q

DO 100 I=1,3
100 QL(I)=0
KOUNT=0
TIME=0
TOTAL=0
NA=0

175 IF (TIME.NE.NA) GO TO 200
KOUNT=KOUNT+1

MIN=1
IF (QL(MIN).GT.QL(2)) MIN=2
IF (QL(MIN).GT.QL(3)) MIN=3
CO=RANDOM(100,350)
QL(MIN)=QL(MIN)+1
QL(MIN,QL(MIN))=KOUNT*100000000 + CO
NA=TIME+RANDOM(0,160)
IF (KOUNT.EQ.LIMIT) NA=-1
WRITE (6,1000) TIME, KOUNT, MIN, CO, QL(MIN), NA
GO TO 175

TIME=TIME+1
DO 250 I=1,3,1
IF (QL(I).EQ.0) GO TO 250

DO 210 J=1,QL(I),1
210 Q(I,J)=Q(I,J)+1000

MAN=Q(I,1)/100000000
WAIT=Q(I,1)-MAN*100000000/1000
CO=MOD(Q(I,1),1000)-1
Q(I,1)=MAN*100000000 + WAIT*1000 + CO
IF (CO.GT.0) GO TO 250

QL(I)=QL(I)-1
DO 220 J=1,QL(I),1
220 Q(I,J)=Q(I,J)+1
WRITE (6,3000) TIME, MAN, I, WAIT, QL(I)
CALL MAIHAit (O, WAIT, MAN)
250 CALL LINELEN (O, I, QL(I))
SUBROUTINE HAXWAIT (MODE, L, N)
C
C GATHERS STATS ON WAITING LINES
C
DIMENSION MAX(10), NUM(10)
INTEGER AVG, TOTAL
DATA MAX, NUM, TOTAL, KNT /22*0/
C
IF (MODE.NE.0) GO TO 200
TOTAL=TOTAL+N
KNT=KNT+1
DO 100 I=1,10,1
IF (MAX(I).GT.L) GO TO 100
MAX(I)=L
NUM(I)=N
RETURN
100 CONTINUE
RETURN

SUBROUTINE LINELEN (MODE, Q, L)
C
C GATHERS LINE LENGTH STATS
C
INTEGER Q, AVG
INTEGER LEN(3), MAX(3), KNT(3)
DATA LEN, MAX, KNT /9*0/
C
IF (MODE.NE.0) GO TO 100
LEN(Q)=LEN(Q)+L
KNT(Q)=KNT(Q)+1
IF (MAX(Q).LT.L) MAX(Q)=L
RETURN

100 WRITE (6,2000)
DO 200 I=1,3,1
AVG = FLOAT(LEN(I))/FLOAT(KNT(I)) + 0.5
WRITE (6,1000) I, MAX(I), AVG
LEN(I)=0
MAX(I)=0
200 KNT(I)=0
RETURN
1000 FORMAT(# Q=##,I1, # MAX=#,I3, # AVERAGE=#,I3)
2000 FORMAT(#0LINE LENGTH STATISTICS!#)
END
INTEGER SUNDIG3

THIS PROGRAM HUNTS FOR SOME NUMBERS SUCH THAT THE SUM OF THE CUBES OF THE DIGITS OF THE NUMBER EQUALS THE NUMBER.

DO 100 I=2,500,1
IF (I.NE.SUNDIG3(I)) GO TO 100
WRITE (6,1000) I, I
100 CONTINUE
STOP

1000 FORMAT(*THE SUM OF THE CUBES OF THE DIGITS OF I4,N4 EQUALS I4,N4*)
END

INTEGER FUNCTION SUNDIG3(N)

COMPUTE THE SUM OF THE CUBES OF THE DIGITS OF N

INTEGER WORKER
SUNDIG3=0
WORKER=IABS(N)
100 IF (WORKER.EQ.0) RETURN
SUNDIG3=SUNDIG3 + MOD(WORKER,10) ** 3
WORKER=WORKER/10
GO TO 100
END
## APPENDIX B

**SOFTWARE PARAMETER DATA FOR PROGRAMS**

**G1 THROUGH G11.**

<table>
<thead>
<tr>
<th>i</th>
<th>OPERATOR</th>
<th>( f_{1,i} )</th>
<th>i</th>
<th>OPERAND</th>
<th>( f_{2,i} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>EOL</td>
<td>18</td>
<td>1</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>2</td>
<td>=</td>
<td>15</td>
<td>2</td>
<td>JR</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>IF( )</td>
<td>5</td>
<td>3</td>
<td>JC</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>,</td>
<td>4</td>
<td>4</td>
<td>IR</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>+</td>
<td>3</td>
<td>5</td>
<td>N</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>.EQ.</td>
<td>2</td>
<td>6</td>
<td>23</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>.LT.</td>
<td>2</td>
<td>7</td>
<td>IC</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>( )</td>
<td>2</td>
<td>8</td>
<td>MAGIC</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>-</td>
<td>2</td>
<td>9</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>/</td>
<td>1</td>
<td>10</td>
<td>529</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>Go To 100</td>
<td>1</td>
<td>11</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>Go To 200</td>
<td>1</td>
<td>13</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Go To 900</td>
<td>1</td>
<td>( n_2 = 11 )</td>
<td>( N_2 = 51 )</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>DO</td>
<td>1</td>
<td>15</td>
<td>.GT.</td>
<td>1</td>
</tr>
</tbody>
</table>

\( n_1 = 15 \), \( N_1 = 59 \)
<table>
<thead>
<tr>
<th>i</th>
<th>OPERATOR</th>
<th>( f_{1,i} )</th>
<th>i</th>
<th>OPERAND</th>
<th>( f_{2,i} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>,</td>
<td>45</td>
<td>1.</td>
<td>1</td>
<td>53</td>
</tr>
<tr>
<td>2.</td>
<td>EOL</td>
<td>36</td>
<td>2.</td>
<td>IR</td>
<td>17</td>
</tr>
<tr>
<td>3.</td>
<td>=</td>
<td>29</td>
<td>3.</td>
<td>IC</td>
<td>17</td>
</tr>
<tr>
<td>4.</td>
<td>(</td>
<td>29</td>
<td>4.</td>
<td>MAT</td>
<td>17</td>
</tr>
<tr>
<td>5.</td>
<td>+</td>
<td>23</td>
<td>5.</td>
<td>JR</td>
<td>11</td>
</tr>
<tr>
<td>6.</td>
<td>-</td>
<td>15</td>
<td>6.</td>
<td>JC</td>
<td>11</td>
</tr>
<tr>
<td>7.</td>
<td>.EQ.</td>
<td>11</td>
<td>7.</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>8.</td>
<td>DO</td>
<td>10</td>
<td>8.</td>
<td>J</td>
<td>8</td>
</tr>
<tr>
<td>9.</td>
<td>IF(</td>
<td>10</td>
<td>9.</td>
<td>MODE</td>
<td>8</td>
</tr>
<tr>
<td>10.</td>
<td>.NE.</td>
<td>5</td>
<td>10.</td>
<td>N</td>
<td>5</td>
</tr>
<tr>
<td>11.</td>
<td>.OR.</td>
<td>3</td>
<td>11.</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>12.</td>
<td>*</td>
<td>3</td>
<td>12.</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>13.</td>
<td>.AND.</td>
<td>3</td>
<td>13.</td>
<td>OUT</td>
<td>5</td>
</tr>
<tr>
<td>14.</td>
<td>Go To 370</td>
<td>3</td>
<td>14.</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>15.</td>
<td>.NOT.</td>
<td>1</td>
<td>15.</td>
<td>23</td>
<td>4</td>
</tr>
<tr>
<td>16.</td>
<td>/</td>
<td>1</td>
<td>16.</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>17.</td>
<td>Go To 200</td>
<td>1</td>
<td>17.</td>
<td>NT</td>
<td>3</td>
</tr>
<tr>
<td>18.</td>
<td>Go To 210</td>
<td>1</td>
<td>18.</td>
<td>25</td>
<td>2</td>
</tr>
<tr>
<td>19.</td>
<td>Go To 325</td>
<td>1</td>
<td>19.</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>20.</td>
<td>Go To 350</td>
<td>1</td>
<td>20.</td>
<td>NU</td>
<td>2</td>
</tr>
<tr>
<td>21.</td>
<td>DIGIT</td>
<td>2</td>
<td>22.</td>
<td>IH+</td>
<td>1</td>
</tr>
<tr>
<td>23.</td>
<td>EDGE</td>
<td>1</td>
<td>24.</td>
<td>WHITE</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ n_1 = 20, \quad N_1 = 231 \]

\[ n_2 = 24, \quad N_2 = 197 \]
<table>
<thead>
<tr>
<th>i</th>
<th>OPERATOR</th>
<th>( f_{1,i} )</th>
<th>i</th>
<th>OPERAND</th>
<th>( f_{2,i} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>EOL</td>
<td>18</td>
<td>1</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>=</td>
<td>11</td>
<td>2</td>
<td>PO</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>,</td>
<td>8</td>
<td>3</td>
<td>PI</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>+</td>
<td>5</td>
<td>4</td>
<td>CO</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>IF( )</td>
<td>4</td>
<td>5</td>
<td>KOUNT</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>CALL PUTCH( )</td>
<td>3</td>
<td>6</td>
<td>C</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>GETCH( )</td>
<td>2</td>
<td>7</td>
<td>OUTPUT</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>.EQ.</td>
<td>2</td>
<td>8</td>
<td>INPUT</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>Go To 200</td>
<td>2</td>
<td>9</td>
<td>IHb</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>.NE.</td>
<td>2</td>
<td>10</td>
<td>NUMERIC</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>( )</td>
<td>2</td>
<td>11</td>
<td>DIGIT</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>Go To 300</td>
<td>1</td>
<td>12</td>
<td>IH+</td>
<td>1</td>
</tr>
</tbody>
</table>

\( \eta_2 = 12, \ N_2 = 49 \)

\( \eta_1 = 16, \ N_1 = 64 \)
<table>
<thead>
<tr>
<th>i</th>
<th>OPERATOR</th>
<th>f_{1,i}</th>
<th>i</th>
<th>OPERAND</th>
<th>f_{2,i}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>EDL</td>
<td>32</td>
<td>1</td>
<td>1</td>
<td>33</td>
</tr>
<tr>
<td>2</td>
<td>,</td>
<td>27</td>
<td>2</td>
<td>PICK</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>=</td>
<td>26</td>
<td>3</td>
<td>PILE</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>( )</td>
<td>15</td>
<td>4</td>
<td>K</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>DO</td>
<td>7</td>
<td>5</td>
<td>I</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>+</td>
<td>6</td>
<td>6</td>
<td>L</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>-</td>
<td>3</td>
<td>7</td>
<td>13.0</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>*</td>
<td>.2</td>
<td>8</td>
<td>J</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>IF( )</td>
<td>2</td>
<td>9</td>
<td>DECK</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td>.EQ.</td>
<td>'2</td>
<td>10</td>
<td>100</td>
<td>3</td>
</tr>
<tr>
<td>11</td>
<td>RANF( )</td>
<td>1</td>
<td>11</td>
<td>CARD</td>
<td>3</td>
</tr>
<tr>
<td>12</td>
<td>CALL SHUFFLE( )</td>
<td>1</td>
<td>12</td>
<td>D</td>
<td>3</td>
</tr>
<tr>
<td>13</td>
<td>CALL PUSH( )</td>
<td>1</td>
<td>13</td>
<td>KOUNT</td>
<td>3</td>
</tr>
<tr>
<td>14</td>
<td>POP( )</td>
<td>1</td>
<td>14</td>
<td>PC</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>Go To 400</td>
<td>1</td>
<td>15</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>16</td>
<td>Go To 300</td>
<td>1</td>
<td>16</td>
<td>500</td>
<td>2</td>
</tr>
<tr>
<td>17</td>
<td>FLOAT( )</td>
<td>1</td>
<td>17</td>
<td>52</td>
<td>2</td>
</tr>
<tr>
<td>18</td>
<td>/</td>
<td>1</td>
<td>18</td>
<td>POP</td>
<td>2</td>
</tr>
<tr>
<td>19</td>
<td>Go To 100</td>
<td>1</td>
<td>19</td>
<td>N</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>20</td>
<td>53</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>21</td>
<td>ITEM</td>
<td>1</td>
</tr>
</tbody>
</table>

\( n_1 = 19, \quad N_1 = 131 \)

\( n_2 = 21, \quad N_2 = 113 \)
<table>
<thead>
<tr>
<th>i</th>
<th>OPERATOR</th>
<th>$f_{1,i}$</th>
<th>i</th>
<th>OPERAND</th>
<th>$f_{2,i}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>EOL</td>
<td>11</td>
<td>1.</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>2.</td>
<td>=</td>
<td>11</td>
<td>2.</td>
<td>L2</td>
<td>6</td>
</tr>
<tr>
<td>3.</td>
<td>( )</td>
<td>7</td>
<td>3.</td>
<td>N</td>
<td>5</td>
</tr>
<tr>
<td>4.</td>
<td>DO</td>
<td>3</td>
<td>4.</td>
<td>MAN</td>
<td>5</td>
</tr>
<tr>
<td>5.</td>
<td>,</td>
<td>3</td>
<td>5.</td>
<td>I</td>
<td>3</td>
</tr>
<tr>
<td>7.</td>
<td>+</td>
<td>1</td>
<td>7.</td>
<td>K</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>8.</td>
<td>KILL</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>9.</td>
<td>L</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10.</td>
<td>M</td>
<td>1</td>
</tr>
</tbody>
</table>

$n_1 = 7, \quad N_1 = 38$

$n_2 = 10, \quad N_2 = 35$
<table>
<thead>
<tr>
<th>i</th>
<th>OPERATOR</th>
<th>$f_{1,i}$</th>
<th>i</th>
<th>OPERAND</th>
<th>$f_{2,i}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>18</td>
<td>1</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>2</td>
<td>EOL</td>
<td>15</td>
<td>2</td>
<td>IR</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>=</td>
<td>14</td>
<td>3</td>
<td>IC</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>( )</td>
<td>10</td>
<td>4</td>
<td>MAT</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>DO</td>
<td>6</td>
<td>5</td>
<td>MIN</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>IF( )</td>
<td>3</td>
<td>6</td>
<td>MAX</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>.GT.</td>
<td>1</td>
<td>7</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>.LT.</td>
<td>1</td>
<td>8</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>.EQ.</td>
<td>1</td>
<td>9</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td>10</td>
<td>J</td>
<td>3</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td>11</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td>12</td>
<td>KEEP</td>
<td>2</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td>13</td>
<td>KEEP</td>
<td>2</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td>14</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

$n_1 = 9, \quad N_1 = 69$

$n_2 = 14, \quad N_2 = 62$
G7.

<table>
<thead>
<tr>
<th>OPERATOR</th>
<th>f_{1,i}</th>
<th>OPERAND</th>
<th>f_{2,i}</th>
</tr>
</thead>
<tbody>
<tr>
<td>EOL</td>
<td>7</td>
<td>N</td>
<td>6</td>
</tr>
<tr>
<td>=</td>
<td>5</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>,</td>
<td>4</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>DO</td>
<td>2</td>
<td>SUMDIV</td>
<td>3</td>
</tr>
<tr>
<td>SUMDIV( )</td>
<td>2</td>
<td>1000</td>
<td>2</td>
</tr>
<tr>
<td>IF( )</td>
<td>2</td>
<td>M</td>
<td>2</td>
</tr>
<tr>
<td>.NE.</td>
<td>2</td>
<td>1500</td>
<td>1</td>
</tr>
<tr>
<td>Go To 100</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ n_2 = 8, \quad N_2 = 23 \]

\[ n_1 = 12, \quad N_1 = 30 \]
<table>
<thead>
<tr>
<th>( f_{1,1} )</th>
<th>( \text{OPERATOR} )</th>
<th>( \text{OPERAND} )</th>
<th>( f_{2,1} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>EOL</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>14</td>
<td>=</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>( )</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td>( )</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>+</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>DO</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>IF( )</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>.NOT.</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>PRIME</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>Go To 100</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>Go To 300</td>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>Go To 250</td>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>SQRT( )</td>
<td>14</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>FLOAT( )</td>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>.GT.</td>
<td>16</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>/</td>
<td>17</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>*</td>
<td>18</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>.NE.</td>
<td>19</td>
<td>1</td>
</tr>
</tbody>
</table>

\( \eta_1 = 19, \ N_1 = 73 \)

\( \eta_2 = 15, \ N_2 = 55 \)
<table>
<thead>
<tr>
<th>i</th>
<th>OPERATOR</th>
<th>$f_{1,i}$</th>
<th>1</th>
<th>OPERAND</th>
<th>$f_{2,1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>EOL</td>
<td>38</td>
<td>1.</td>
<td>O</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>=</td>
<td>32</td>
<td>2.</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>11</td>
<td>3.</td>
<td>Q1</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>+</td>
<td>11</td>
<td>4.</td>
<td>Q2</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>IF( )</td>
<td>5</td>
<td>5.</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>.EQ.</td>
<td>3</td>
<td>6.</td>
<td>ON</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>-</td>
<td>3</td>
<td>7.</td>
<td>LIGHT</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>*</td>
<td>3</td>
<td>8.</td>
<td>OFF</td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td>DO</td>
<td>2</td>
<td>9.</td>
<td>TOTAL</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>RANDOM( )</td>
<td>2</td>
<td>10.</td>
<td>WAIT 1</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>MINO( )</td>
<td>2</td>
<td>11.</td>
<td>WAIT 2</td>
<td>4</td>
</tr>
<tr>
<td>12</td>
<td>Go To 400</td>
<td>2</td>
<td>12.</td>
<td>TIME</td>
<td>4</td>
</tr>
<tr>
<td>13</td>
<td>CALL SIMULA( )</td>
<td>1</td>
<td>13.</td>
<td>GT</td>
<td>3</td>
</tr>
<tr>
<td>14</td>
<td>FLOAT( )</td>
<td>1</td>
<td>14.</td>
<td>DEBUG</td>
<td>3</td>
</tr>
<tr>
<td>15</td>
<td>/</td>
<td>1</td>
<td>15.</td>
<td>WAIT</td>
<td>3</td>
</tr>
<tr>
<td>16</td>
<td>Go To 200</td>
<td>1</td>
<td>16.</td>
<td>RANDOM</td>
<td>3</td>
</tr>
<tr>
<td>17</td>
<td>Go To 250</td>
<td>1</td>
<td>17.</td>
<td>I</td>
<td>2</td>
</tr>
<tr>
<td>18</td>
<td>Go To 300</td>
<td>1</td>
<td>18.</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>19</td>
<td>.NE.</td>
<td>1</td>
<td>19.</td>
<td>X</td>
<td>2</td>
</tr>
<tr>
<td>20</td>
<td>.LT.</td>
<td>1</td>
<td>20.</td>
<td>90</td>
<td>1</td>
</tr>
<tr>
<td>21</td>
<td>Go To 100</td>
<td>1</td>
<td>21.</td>
<td>.TRUE.</td>
<td>1</td>
</tr>
<tr>
<td>22</td>
<td>RANF( )</td>
<td>1</td>
<td>22.</td>
<td>.FALSE.</td>
<td>1</td>
</tr>
<tr>
<td>23</td>
<td>AVG</td>
<td>1</td>
<td>23.</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>5</td>
<td>1</td>
<td>24.</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>15</td>
<td>1</td>
<td>25.</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>6</td>
<td>1</td>
<td>26.</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>24</td>
<td>1</td>
<td>27.</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>20</td>
<td>1</td>
<td>28.</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>36</td>
<td>1</td>
<td>29.</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>100</td>
<td>1</td>
<td>30.</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>300</td>
<td>1</td>
<td>31.</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>J</td>
<td>1</td>
<td>32.</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

$\eta_1 = 22, \ N_1 = 124$

$\eta_1 = 32, \ N_2 = 104$
<table>
<thead>
<tr>
<th>i</th>
<th>OPERATOR</th>
<th>f_{1,i}</th>
<th>i</th>
<th>OPERAND</th>
<th>f_{2,i}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>EOL</td>
<td>59</td>
<td>1</td>
<td>1</td>
<td>33</td>
</tr>
<tr>
<td>2</td>
<td>=</td>
<td>45</td>
<td>2</td>
<td>1</td>
<td>30</td>
</tr>
<tr>
<td>3</td>
<td>( )</td>
<td>43</td>
<td>3</td>
<td>0</td>
<td>21</td>
</tr>
<tr>
<td>4</td>
<td>,</td>
<td>33</td>
<td>4</td>
<td>QL</td>
<td>17</td>
</tr>
<tr>
<td>5</td>
<td>+</td>
<td>17</td>
<td>5</td>
<td>Q</td>
<td>15</td>
</tr>
<tr>
<td>6</td>
<td>IF( )</td>
<td>12</td>
<td>6</td>
<td>MIN</td>
<td>9</td>
</tr>
<tr>
<td>7</td>
<td>DO</td>
<td>7</td>
<td>7</td>
<td>KNT</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>.GT.</td>
<td>5</td>
<td>8</td>
<td>J</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>*</td>
<td>4</td>
<td>9</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>10</td>
<td>-</td>
<td>4</td>
<td>10</td>
<td>MAX</td>
<td>6</td>
</tr>
<tr>
<td>11</td>
<td>/</td>
<td>4</td>
<td>11</td>
<td>NA</td>
<td>5</td>
</tr>
<tr>
<td>12</td>
<td>FLOAT( )</td>
<td>4</td>
<td>12</td>
<td>KOUNT</td>
<td>5</td>
</tr>
<tr>
<td>13</td>
<td>.NE.</td>
<td>3</td>
<td>13</td>
<td>TIME</td>
<td>5</td>
</tr>
<tr>
<td>14</td>
<td>Go To 175</td>
<td>3</td>
<td>14</td>
<td>TOTAL</td>
<td>5</td>
</tr>
<tr>
<td>15</td>
<td>Go To 200</td>
<td>2</td>
<td>15</td>
<td>CO</td>
<td>5</td>
</tr>
<tr>
<td>16</td>
<td>RANDOM( )</td>
<td>2</td>
<td>16</td>
<td>L</td>
<td>5</td>
</tr>
<tr>
<td>17</td>
<td>.EQ.</td>
<td>2</td>
<td>17</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>18</td>
<td>Go To 250</td>
<td>2</td>
<td>18</td>
<td>1000</td>
<td>4</td>
</tr>
<tr>
<td>19</td>
<td>CALL MAXWAIT( )</td>
<td>2</td>
<td>19</td>
<td>MAN</td>
<td>4</td>
</tr>
<tr>
<td>20</td>
<td>CALL LINELEN( )</td>
<td>2</td>
<td>20</td>
<td>LEN</td>
<td>4</td>
</tr>
<tr>
<td>21</td>
<td>Go To 100</td>
<td>2</td>
<td>21</td>
<td>WAIT</td>
<td>3</td>
</tr>
<tr>
<td>22</td>
<td>CALL SIMULA( )</td>
<td>1</td>
<td>22</td>
<td>10^8</td>
<td>2</td>
</tr>
<tr>
<td>23</td>
<td>MOD( )</td>
<td>1</td>
<td>23</td>
<td>10^7</td>
<td>2</td>
</tr>
<tr>
<td>24</td>
<td>.GE.</td>
<td>1</td>
<td>24</td>
<td>MODE</td>
<td>2</td>
</tr>
<tr>
<td>25</td>
<td>.LT.</td>
<td>1</td>
<td>25</td>
<td>N</td>
<td>2</td>
</tr>
<tr>
<td>26</td>
<td>NUM</td>
<td>2</td>
<td>26</td>
<td>AVG</td>
<td>2</td>
</tr>
<tr>
<td>27</td>
<td>AVG</td>
<td>2</td>
<td>27</td>
<td>0.5</td>
<td>2</td>
</tr>
<tr>
<td>28</td>
<td>0.5</td>
<td>2</td>
<td>28</td>
<td>250</td>
<td>1</td>
</tr>
<tr>
<td>29</td>
<td>250</td>
<td>1</td>
<td>29</td>
<td>100</td>
<td>1</td>
</tr>
<tr>
<td>30</td>
<td>100</td>
<td>1</td>
<td>30</td>
<td>360</td>
<td>1</td>
</tr>
<tr>
<td>31</td>
<td>360</td>
<td>1</td>
<td>31</td>
<td>160</td>
<td>1</td>
</tr>
<tr>
<td>32</td>
<td>160</td>
<td>1</td>
<td>32</td>
<td>LIMIT</td>
<td>1</td>
</tr>
<tr>
<td>33</td>
<td>LIMIT</td>
<td>1</td>
<td>33</td>
<td>10</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ n_1 = 25, \ n_2 = 261 \]

\[ n_2 = 34, \ n_2 = 222 \]
<table>
<thead>
<tr>
<th></th>
<th>OPERATOR</th>
<th>$f_{1,1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>EOL</td>
<td>8</td>
</tr>
<tr>
<td>2.</td>
<td>=</td>
<td>5</td>
</tr>
<tr>
<td>3.</td>
<td>,</td>
<td>3</td>
</tr>
<tr>
<td>4.</td>
<td>IF( )</td>
<td>2</td>
</tr>
<tr>
<td>5.</td>
<td>Go To 100</td>
<td>2</td>
</tr>
<tr>
<td>6.</td>
<td>DO</td>
<td>1</td>
</tr>
<tr>
<td>7.</td>
<td>.NE.</td>
<td>1</td>
</tr>
<tr>
<td>8.</td>
<td>( )</td>
<td>1</td>
</tr>
<tr>
<td>9.</td>
<td>IABS( )</td>
<td>1</td>
</tr>
<tr>
<td>10.</td>
<td>.EQ.</td>
<td>1</td>
</tr>
<tr>
<td>11.</td>
<td>+</td>
<td>1</td>
</tr>
<tr>
<td>12.</td>
<td>MOD( )</td>
<td>1</td>
</tr>
<tr>
<td>13.</td>
<td>**</td>
<td>1</td>
</tr>
<tr>
<td>14.</td>
<td>/</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>OPERAND</th>
<th>$f_{2,1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>WORKER</td>
<td>5</td>
</tr>
<tr>
<td>2.</td>
<td>SUMDIG3</td>
<td>4</td>
</tr>
<tr>
<td>3.</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>4.</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>5.</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>6.</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>7.</td>
<td>500</td>
<td>1</td>
</tr>
<tr>
<td>8.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>9.</td>
<td>N</td>
<td>1</td>
</tr>
<tr>
<td>10.</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

$n_1 = 14, N_1 = 29$

$n_2 = 10, N_2 = 21$
An experiment comparing Fortran programming times with the software physics hypothesis

by R. D. GORDON and M. H. HALSTEAD
Purdue University
Lafayette, Indiana

ABSTRACT
Recent discoveries in the area of Algorithm Structure or Software Physics have produced a number of hypotheses. One of these relates the number of elementary mental discriminations required to implement an algorithm to measurable properties of that algorithm, and the results of one set of experiments confirming this relationship have been published. That publication, while significant, made no claim to finality, suggesting instead that further experiments were warranted. This paper will present the results of a second set of experiments, having the advantages of being conducted in a single implementation language, Fortran, from problem specifications readily available in computer textbooks.

The first section of this paper presents the timing hypothesis, and the elementary equations upon which it rests. The second section presents the details of the experiment and the results which were obtained, and the third section contains an analysis of the data.

TIMING HYPOTHESIS
Measurable properties of any implementation of any algorithm include:

\( \eta_1 = \) The count of distinct operators
\( \eta_2 = \) The count of distinct operands (variables or constants)
\( N_1 = \) Total uses of operators
\( N_2 = \) Total uses of operands

The vocabulary, \( \eta \), is given by:

\[ \eta = \eta_1 + \eta_2 \]  
(1)

and the length, \( N \), is:

\[ N = N_1 + N_2 \]  
(2)

From these properties, it is possible to obtain the volume, \( V \), in bits, as:

\[ V = N \log_2 \eta \]  
(3)

and the implementation level, \( L \), where \( L \leq 1 \), as:

\[ L = \frac{\eta_1^*}{\eta_1} \frac{\eta_2}{N_2} \]  
(4)

where \( \eta_1^* \), the minimum possible number of operators, will equal 2 for most algorithms. (One for the name of a function, plus one for a grouping symbol operator).

It has been shown that the product \( L \times V \) is invariant under translation from one language to another, and that for programs without impurities:

\[ N = \eta_1 \log_2 \eta_1 + \eta_2 \log_2 \eta_2 \]  
(5)

From this point, the following nine steps yield the timing equation:

1. A program consists of \( N \) selections from \( \eta \) elements.
2. A binary search of \( \eta \) elements requires \( \log_2 \eta \) comparisons.
3. A program is generated by making \( N \log_2 \eta \) comparisons.
4. Therefore, the volume, \( V \), is a count of the number of comparisons required.
5. The number of elementary mental discriminations required to complete one comparison measures the difficulty of the task.
6. The level, \( L \), is the reciprocal of the difficulty.
7. Therefore, \( E \), the count of elementary mental discriminations required to generate a program, is given by:

\[ E = \frac{V}{L} \]  
(6)

8. \( S \), the speed with which the brain makes elementary mental discriminations can be obtained from psychology as:

\[ 5 \leq S \leq 20 \text{ discriminations per second.} \]

9. Therefore, the time to generate a preconceived program, by a concentrating programmer, fluent in a language, is:

\[ T = \frac{V}{SL} \]  
(7)

Equation 7 may be expressed in more basic terms by substituting for \( V \) from equation 8, and for \( L \) from equation 4, with \( \eta_1^* = 2 \), giving:

\[ T = \frac{3N_1N_2 \log_2 \eta_2}{2S \eta_1} \]  
(8)
The effect of possible impurities may be eliminated from equation 8 by substituting for \( N \) from equation 5. Letting \( S = 60 \times 18 = 1080 \) will then give, for time in minutes:

\[
\hat{T} = \frac{\eta N_2 (\log \tau_1 + \log \tau_2)}{2160} \eta \tag{9}
\]

Each of the variables on the right hand side of equation 9 can be readily measured (or counted) in any computer program, and the experiment described in the next section was designed to compare results from that equation with observed programming times.

**EXPERIMENTAL PROCEDURE**

Eleven problems were arbitrarily selected from two published sources. In selecting candidates for the experiment, problems were sought which were stated in a non-procedural form. Further, the problem statement had to be complete. That is, in the course of solving a particular problem, specific laws of physics, mathematics, etc. would not have to be derived. The problems finally selected were taken from Knuth, and from Maurer and Williams, and cover a wide range of topics including character manipulation, list processing, simulation experiments and mathematical analysis. The source of each problem statement is cited in Table I.

On each of eleven days, one of these problems was implemented by the senior author. In order to maintain a consistent level of performance all work was conducted in a quiet room, free from distractions, during the same period of the day. The time required to fully implement the problem was obtained. This total time included the number of minutes spent reading the statement of the problem, preparing flowcharts and writing preliminary versions of the code, writing the final version of the code, desk checking, and the time spent working to correct errors in the program. Time to keypunch was not included.

**TABLE I—Experimental Data**

<table>
<thead>
<tr>
<th>Program No.</th>
<th>Ref.</th>
<th>Page</th>
<th>Problem</th>
<th>( \tau_1 )</th>
<th>( \tau_2 )</th>
<th>( N_1 )</th>
<th>( N_2 )</th>
<th>Implementation Time-Minutes</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>K</td>
<td>158</td>
<td>21</td>
<td>15</td>
<td>11</td>
<td>59</td>
<td>61</td>
<td>19</td>
</tr>
<tr>
<td>G2</td>
<td>K</td>
<td>159</td>
<td>23</td>
<td>20</td>
<td>24</td>
<td>231</td>
<td>197</td>
<td>22</td>
</tr>
<tr>
<td>G3</td>
<td>K</td>
<td>198</td>
<td>7</td>
<td>16</td>
<td>12</td>
<td>64</td>
<td>49</td>
<td>16</td>
</tr>
<tr>
<td>G4</td>
<td>K</td>
<td>377</td>
<td>17</td>
<td>10</td>
<td>21</td>
<td>131</td>
<td>113</td>
<td>39</td>
</tr>
<tr>
<td>G5</td>
<td>K</td>
<td>158</td>
<td>22</td>
<td>7</td>
<td>10</td>
<td>38</td>
<td>36</td>
<td>21</td>
</tr>
<tr>
<td>G6</td>
<td>K</td>
<td>154</td>
<td>10</td>
<td>9</td>
<td>14</td>
<td>69</td>
<td>62</td>
<td>30</td>
</tr>
<tr>
<td>G7</td>
<td>M</td>
<td>32</td>
<td>3.2.21</td>
<td>12</td>
<td>8</td>
<td>50</td>
<td>23</td>
<td>5</td>
</tr>
<tr>
<td>G8</td>
<td>M</td>
<td>32</td>
<td>3.2.23</td>
<td>19</td>
<td>15</td>
<td>73</td>
<td>56</td>
<td>24</td>
</tr>
<tr>
<td>G9</td>
<td>M</td>
<td>88</td>
<td>8.3.2</td>
<td>22</td>
<td>32</td>
<td>124</td>
<td>104</td>
<td>42</td>
</tr>
<tr>
<td>G10</td>
<td>M</td>
<td>89</td>
<td>8.3.4</td>
<td>25</td>
<td>34</td>
<td>261</td>
<td>222</td>
<td>91</td>
</tr>
<tr>
<td>G11</td>
<td>M</td>
<td>27</td>
<td>3.2.4</td>
<td>14</td>
<td>10</td>
<td>29</td>
<td>21</td>
<td>6</td>
</tr>
</tbody>
</table>

For a number of reasons, including availability and fluency, all of the algorithms were implemented in Fortran. In the course of solving a problem the correctness of the implementation was checked by executing a sufficiently complex test case for which a correct answer was known. In some cases the solution to a problem was written as a subroutine and testing required that a main routine be written. In such a case only the preparation of the subroutine was considered for the experiment. In addition, several implementations made use of subroutines previously written. Such routines were also not included.

After each program was completed, a careful count was made to determine values of \( \eta, \tau_1, N_1, N_2 \). In obtaining these values all read, write, declarative statements and comments were ignored. The results are shown in Table I.

**ANALYSIS OF THE DATA**

The programming time predicted by theory was obtained for each program by applying equation 9 to the data in Table I. This result, \( T \), can be compared with the observed value, \( \hat{T} \), in Table II. In addition, a count of the number of statements in each program was obtained, and the programs were ordered according to these values.

The average of the calculated values, 34 minutes, is fortuitously close to the observed value, 35 minutes. The coefficient of correlation is 0.984, only slightly smaller than the value of 0.952 reported in an earlier experiment. In further agreement with that experiment, the correlation between length and observed times, 0.887, is lower than between observed and calculated times.

In conclusion, it may again be observed that one more set of experimental data does not contradict the simple hypothesis. As a result, further carefully controlled experiments by others would appear to be warranted.

* Additional details available from the author.

**TABLE II—Experimental Results**

<table>
<thead>
<tr>
<th>Program Number</th>
<th>Statement Count</th>
<th>Programming Time-Minutes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( T_\text{observed} )</td>
<td>( T \text{ Eq. 9} )</td>
</tr>
<tr>
<td>G7</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>G11</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>G8</td>
<td>11</td>
<td>21</td>
</tr>
<tr>
<td>G6</td>
<td>16</td>
<td>30</td>
</tr>
<tr>
<td>G3</td>
<td>18</td>
<td>36</td>
</tr>
<tr>
<td>G1</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>G8</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>G4</td>
<td>32</td>
<td>39</td>
</tr>
<tr>
<td>G2</td>
<td>36</td>
<td>52</td>
</tr>
<tr>
<td>G9</td>
<td>39</td>
<td>48</td>
</tr>
<tr>
<td>G10</td>
<td>50</td>
<td>81</td>
</tr>
</tbody>
</table>

Mean: 38.8, 34.1
REFERENCES


Additional References:

