Testing Bradley's Greatest Common Divisor Program on EXPER

Robert L. Hess and Frederick G. Sayward

Research Report #165

May 1979

This research was supported in part by the Army Institute for Research in Management Information and Computer Science and the Army Research Office under Grant DAAG 29-78-G-0121 to the Georgia Institute of Technology School of Information and Computer Science, subcontracted to Yale University.

Testing Bradley's Greatest Common Divisor Program on EXPER

Robert L. Hess
and
Frederick G. Sayward

Department of Computer Science
Yale University
New Haven, Connecticut 06520

May 1979

ABSTRACT

A program testing experiment run on the EXPER program testing system is described. The experiment involved taking a published generalization of Euclid's greatest common divisor algorithm which was known to be incorrect and seeing if trying to pass the mutation test would lead to discovering the errors. All errors were uncovered and in addition a slight improvement to the algorithm was found.

INTRODUCTION

The EXPER system [1] is an experimental program testing system designed by Timothy Budd, Richard DeMillo, Richard Lipton, and Frederick Sayward. In this report we will present experiences gained while analyzing a fast generalization of Euclid's greatest common divisor algorithm on EXPER. The algorithm was introduced and its complexity analyzed in [2,3]. A brief description follows:

Input: a natural number $n \geq 0$ and integers A[1],...,A[n].

Output: Igcd,Z[1],Z[2],...Z[n] where $Igcd \geq 0$ is the greatest common divisor of the elements of array A, and Z is an array of multipliers such that A[1]*Z[1]+A[2]*Z[2]+...+A[n]*Z[n]=Igcd.

Bradley's algorithm is of particular interest to us because a few years after its publication, during an attempt to formally prove it correct [4], several subtle errors were found to be present. Hence, if we start with the original program presented in [2,3] and run it through EXPER, we should expect to uncover at least the errors detected in [4].

METHOD

We began running EXPER on Bradley's Algorithm as it appeared in [3], called Algorithm 386. See appendix A for the initial program listing. Note that there is more than one correct answer to any input, since it is not guaranteed that the program will output a minimal set of multipliers. However, we decided to ignore this and let mutant correctness be determined by exact equality with the program. No mutant was allowed to run more than ten times longer than the original program.

The following is a history of our constructing test cases on EXPER which caused Bradley's Algorithm to fail and the corrective action we took. The line numbers refer to the corrected version of Bradley's Algorithm which is listed in appendix B.

The random test case 0 3 9 caused I to be undefined in line 52. The correction is the addition of line 48: I=N.

The cleanup loop (lines 62-67) is used when IGCD is found to be one. To test this portion of code we input -1 3. The result was K undefined in line 55; this was corrected by adding the condition statement at lines 50-51: IF(MP2.GT.I) GOTO 51.

On re-running with -1 3 the algorithm gave an incorrect answer. This was corrected by adding line 63-64: IF(IP1.GT.N) GOTO 40.

0 0 0 -3 was a special case which tested lines 1-11. The program gave an incorrect answer. This error was fixed by changing line 9 to IGCD=IABS(A(M)) and changing line 10 to Z(M)=A(M)/IGCD.

The test case 0 42 -6 15 exercised the gcd and multiplier loops (lines 20-61) with M equal to 2. This value of M caused K to be undefined in line 55. This was corrected by changing line 53 to K=I-J+MP1.

All of these changes were noted in the Certification of Algorithm 386 [4]. We found no additional errors.

ANALYSIS

Appendix B lists the final EXPER report on the corrected version of Bradley's Algorithm. There are two reasons for the high number of equivalent mutants. First, quite a few mutations are trivial: replacing Y1 by the absolute value of Y1 is equivalent because Y1 is always positive. Second, line 45-46: IF(C1.EQ.1) GOTO 60 is not an essential statement for the program's functinal correctness. It's only purpose is to reduce the number of calculations if the greatest common divisor is found to be 1.

Through this experiment, we found one improvement that can be made to the algorithm. Line 49: IGCD=A(M) can be replaced by IGCD=C1, which is slightly faster because it does not reference an array.

REFERENCES

[1] Tim Budd, Richard DeMillo, Richard Lipton, and Frederick Sayward, "Mutation Analysis", Yale University Department of Computer Science

Research Report 155, April 1979, pp. 28.

[2] Gordon H. Bradley, "Algorithm 386--Greatest Common Divisor of n Integers and Multipliers", _Communications of the ACM_, Vol. 13, No. 7 (July 1970), pp. 447-448.

[3] Gordon H. Bradley, "Algorithm and Bound for the Greatest Common Divisor of n Integers", _Communications of the ACM_, Vol. 13, No. 7 (July 1970), pp. 433-436.

[4] Larry C. Ragland and Donald I. Good, "Certification of Algorithm 386", _Communications of the ACM_, Vol. 16, No. 4, April 1973, p. 257.

APPENDIX A

The following is a listing of Bradley's original algorithm for calculating the greatest common divisor of n integers and multipliers as found in [3].

```
      SUBROUTINE GCDN(N,A,Z,IGCD)
C
C    N   NUMBER OF INTEGERS
C    A() INPUT ARRAY OF N INTEGERS.  INPUT IS DESTROYED.
C    Z() OUTPUT ARRAY OF N MULTIPLIERS
C   IGCD OUTPUT GREATEST COMMON DIVISOR
C
      INPUT A
      RDONLY N
      OUTPUT Z,IGCD
      DIMENSION A(N),Z(N)
      INTEGER A,Z,C1,C2,Y1,Y2,Q
C
C  FIND THE FIRST NON-ZERO INTEGER
      DO 1 M=1,N,1
      IF(A(M).NE.0) GOTO 3
 1    Z(M)=0
C  ALL ZERO INPUTS RESULTS IN ZERO GCD AND Z
      IGCD=0
      RETURN
C IF LAST NUMBER IS THE ONLY NON-ZERO NUMBER, EXIT IMMEDIATELY
 3    IF(M.NE.N) GOTO 4
      IGCD=A(M)
      Z(M)=1
      RETURN
 4    MP1=M+1
      MP2=M+2
C CHECK THE SIGN OF A(M)
      ISIGN=0
      IF(A(M).GE.0) GOTO 5
      ISIGN=1
      A(M)=-A(M)
C CALCULATE GCD VIA N-1 APPLICATIONS OF THE GCD ALGORITHM FOR TWO INTEGERS.
C SAVE THE MULTIPLIERS.
 5    C1=A(M)
      DO 30 I=MP1,N,1
      IF(A(I).NE.0) GOTO 7
      A(I)=1
      Z(I)=0
      GOTO 25
 7    Y1=1
      Y2=0
      C2=IABS(A(I))
 10   Q=C2/C1
      C2=C2-Q*C1
C TESTING BEFORE COMPUTING Y2 AND BEFORE COMPUTING Y1 BELOW SAVES N-1
C ADDITIONS AND N-1 MULTIPLICATIONS
```

```
      IF(C2.EQ.0) GOTO 20                                  31  32
      Y2=Y2-Q*Y1                                               33
      Q=C1/C2                                                  34
      C1=C1-Q*C2                                               35
      IF(C1.EQ.0) GOTO 15                                  36  37
      Y1=Y1-Q*Y2                                               38
      GOTO 10                                                  39
 15   C1=C2                                                    40
      Y1=Y2                                                    41
 20   Z(I)=(C1-Y1*A(M))/A(I)                                   42
      A(I)=Y1                                                  43
      A(M)=C1                                                  44
C TERMINATE GCD CALCULATIONS IF GCD EQUALS 1
 25   IF(C1.EQ.1) GOTO 60                                  45  46
 30   CONTINUE                                                 47
 40   IGCD=A(M)                                                49
C CALCULATE MULTIPLIERS
      DO 50 J=MP2,I,1                                          52
      K=I-J+2                                                  53
      KK=K+1                                                   54
      Z(K)=Z(K)*A(KK)                                          55
 50   A(K)=A(K)*A(KK)                                          56
 51   Z(M)=A(MP1)                                              57
      IF(ISIGN.EQ.0) GOTO 100                              58  59
      Z(M)=-Z(M)                                               60
 100  RETURN                                                   61
C GCD FOUND, SET REMAINDER OF THE MULTIPLIERS EQUAL TO ZERO.
 60   IP1=I+1                                                  62
      DO 65 J=IP1,N,1                                          65
 65   Z(J)=0                                                   66
      GOTO 40                                                  67
      END
```

APPENDIX B

In this appendix we list the corrected version of Bradley's algorithm and the final EXPER report which contains the test cases and an accounting of the algorithm's mutants. Asterisks denote corrections.

```
      SUBROUTINE GCDN(N,A,Z,IGCD)
C
C     N   NUMBER OF INTEGERS
C     A() INPUT ARRAY OF N INTEGERS.  INPUT IS DESTROYED.
C     Z() OUTPUT ARRAY OF N MULTIPLIERS
C   IGCD OUTPUT GREATEST COMMON DIVISOR
C
      INPUT A
      RDONLY N
      OUTPUT Z,IGCD
      DIMENSION A(N),Z(N)
      INTEGER A,Z,C1,C2,Y1,Y2,Q
C
C  FIND THE FIRST NON-ZERO INTEGER
      DO 1 M=1,N,1                                                 1
      IF(A(M).NE.0) GOTO 3                                      2  3
 1    Z(M)=0                                                       4
C  ALL ZERO INPUTS RESULTS IN ZERO GCD AND Z
      IGCD=0                                                       5
      RETURN                                                       6
C IF LAST NUMBER IS THE ONLY NON-ZERO NUMBER, EXIT IMMEDIATELY
 3    IF(M.NE.N) GOTO 4                                         7  8
      IGCD=IABS(A(M))                 ********************         9
      Z(M)=A(M)/IGCD                  ********************        10
      RETURN                                                      11
 4    MP1=M+1                                                     12
      MP2=M+2                                                     13
C CHECK THE SIGN OF A(M)
      ISIGN=0                                                     14
      IF(A(M).GE.0) GOTO 5                                     15 16
      ISIGN=1                                                     17
      A(M)=-A(M)                                                  18
C CALCULATE GCD VIA N-1 APPLICATIONS OF THE GCD ALGORITHM FOR TWO INTEGERS.
C SAVE THE MULTIPLIERS.
 5    C1=A(M)                                                     19
      DO 30 I=MP1,N,1                                             20
      IF(A(I).NE.0) GOTO 7                                     21 22
      A(I)=1                                                      23
      Z(I)=0                                                      24
      GOTO 25                                                     25
 7    Y1=1                                                        26
      Y2=0                                                        27
      C2=IABS(A(I))                                               28
 10   Q=C2/C1                                                     29
      C2=C2-Q*C1                                                  30
```

```
C TESTING BEFORE COMPUTING Y2 AND BEFORE COMPUTING Y1 BELOW SAVES N-1
C ADDITIONS AND N-1 MULTIPLICATIONS
      IF(C2.EQ.0) GOTO 20
      Y2=Y2-Q*Y1
      Q=C1/C2
      C1=C1-Q*C2
      IF(C1.EQ.0) GOTO 15
      Y1=Y1-Q*Y2
      GOTO 10
 15   C1=C2
      Y1=Y2
 20   Z(I)=(C1-Y1*A(M))/A(I)
      A(I)=Y1
      A(M)=C1
C TERMINATE GCD CALCULATIONS IF GCD EQUALS 1
 25   IF(C1.EQ.1) GOTO 60
 30   CONTINUE
      I=N                                 ********************
 40   IGCD=A(M)
C CALCULATE MULTIPLIERS
      IF(MP2.GT.I) GOTO 51                ********************
      DO 50 J=MP2,I,1
      K=I-J+MP1                           ********************
      KK=K+1
      Z(K)=Z(K)*A(KK)
 50   A(K)=A(K)*A(KK)
 51   Z(M)=A(MP1)
      IF(ISIGN.EQ.0) GOTO 100
      Z(M)=-Z(M)
 100  RETURN
C GCD FOUND, SET REMAINDER OF THE MULTIPLIERS EQUAL TO ZERO.
 60   IP1=I+1
      IF(IP1.GT.N) GOTO 40                ********************
      DO 65 J=IP1,N,1
 65   Z(J)=0
      GOTO 40
      END
```

The following 34 test cases were developed to kill mutants.

	IN:	N	A[1]	A[2]	. . . .		A[N]	
	OUT:	IGCD	Z[1]	Z[2]	. . . .		Z[N]	
1	IN:	3	0	0	0			
	OUT:	0	0	0	0			
2	IN:	3	0	3	9			
	OUT:	3	0	1	0			
3	IN:	4	-3	-3	-3	-3		
	OUT:	3	-1	0	0	0		
4	IN:	3	2	0	0			
	OUT:	2	1	0	0			
5	IN:	2	-1	3				
	OUT:	1	-1	0				
6	IN:	4	0	0	0	-3		
	OUT:	3	0	0	0	-1		
7	IN:	4	2	2	2	1		
	OUT:	1	0	0	0	1		
8	IN:	2	36	7				
	OUT:	1	1	-5				
9	IN:	4	36	54	12	4		
	OUT:	2	-1	1	-1	-1		
10	IN:	2	36	6				
	OUT:	6	0	1				
11	IN:	3	1	2	3			
	OUT:	1	1	0	0			
12	IN:	4	36	90	5000	30		
	OUT:	2	2222	-1111	4	0		
13	IN:	3	0	-3	6			
	OUT:	3	0	-1	0			
14	IN:	3	-5	0	10			
	OUT:	5	-1	0	0			
15	IN:	3	-8	4	3			
	OUT:	1	0	1	-1			
16	IN:	6	3	9	3	4	4	4
	OUT:	1	-1	0	0	1	0	0
17	IN:	4	0	3	14	3		
	OUT:	1	0	5	-1	0		

18 IN:	2	318	336							
OUT:	6	-19	18							
19 IN:	2	300	207							
OUT:	3	-20	29							
20 IN:	3	0	0	2						
OUT:	2	0	0	1						
21 IN:	3	-3	0	0						
OUT:	3	-1	0	0						
22 IN:	2	0	4							
OUT:	4	0	1							
23 IN:	6	102	30	66	42	44	46			
OUT:	2	14	-49	0	0	1	0			
24 IN:	4	0	42	-6	15					
OUT:	3	0	0	2	1					
25 IN:	5	0	3	0	6	0				
OUT:	3	0	1	0	0	0				
26 IN:	9	0	0	0	0	0	4	4	3	3
OUT:	1	0	0	0	0	0	1	0	-1	0
27 IN:	3	4	0	5						
OUT:	1	-1	0	1						
28 IN:	2	11	19							
OUT:	1	7	-4							
29 IN:	3	-3	-1	-1						
OUT:	1	0	-1	0						
30 IN:	6	0	0	0	4	8	2			
OUT:	2	0	0	0	0	0	1			
31 IN:	6	-2	4	5	6	7	8			
OUT:	1	2	0	1	0	0	0			
32 IN:	5	5	1	2	1	5				
OUT:	1	0	1	0	0	0				
33 IN:	3	8	14	7						
OUT:	1	-6	3	1						
34 IN:	7	128	64	32	16	8	4	2		
OUT:	2	0	0	0	0	0	0	1		

RESULTS:

NUMBER OF TEST CASES = 34

NUMBER OF MUTANTS = 5121
NUMBER OF DEAD MUTANTS = 4956 (96.8%)
NUMBER OF LIVE MUTANTS = 2 (0.0%)
NUMBER OF EQUIV MUTANTS = 163 (3.2%)

NUMBER OF MUTATABLE STATEMENTS = 67
GIVING A MUTANTS/STATEMENT RATIO OF 76.43

MUTANT TYPE	TOTAL	DEAD		LIVE		EQUIV	
CONSTANT REPLACEMENT	42	39	92.9%	0	0.0%	3	7.1%
SCALAR VARIABLE REPLACEME	1575	1562	99.2%	0	0.0%	13	0.8%
SCALAR FOR CONSTANT REP.	336	314	93.5%	0	0.0%	22	6.5%
CONSTANT FOR SCALAR REP.	222	220	99.1%	0	0.0%	2	0.9%
SOURCE CONSTANT REPLACEME	13	13	100.0%	0	0.0%	0	0.0%
ARRAY REF. FOR CONSTANT R	180	176	97.8%	1	0.6%	3	1.7%
ARRAY REF. FOR SCALAR REP	900	899	99.9%	0	0.0%	1	0.1%
COMPARIABLE ARRAY NAME RE	30	30	100.0%	0	0.0%	0	0.0%
CONSTANT FOR ARRAY REF RE	51	51	100.0%	0	0.0%	0	0.0%
SCALAR FOR ARRAY REF REP.	480	479	99.8%	0	0.0%	1	0.2%
ARRAY REF. FOR ARRAY REF.	240	239	99.6%	0	0.0%	1	0.4%
UNARY OPERATOR INSERION	463	359	77.5%	1	0.2%	103	22.2%
ARITHMETIC OPERATOR REPLA	154	154	100.0%	0	0.0%	0	0.0%
RELATIONAL OPERATOR REPLA	50	43	86.0%	0	0.0%	7	14.0%
UNARY OPERATOR REMOVAL	2	2	100.0%	0	0.0%	0	0.0%
STATEMENT ANALYSIS	29	29	100.0%	0	0.0%	0	0.0%
STATEMENT DELETION	66	65	98.5%	0	0.0%	1	1.5%
RETURN STATEMENT REPLACEM	63	63	100.0%	0	0.0%	0	0.0%
GOTO STATEMENT REPLACEMEN	195	191	97.9%	0	0.0%	4	2.1%
DO STATEMENT END REPLACEM	30	28	93.3%	0	0.0%	2	6.7%

The following mutants remain live since we did not find a test case to kill them and it is not obvious that they are equivalent.

```
   STATEMENT 45 CHANGED FROM
25     IF(C1.EQ.1) GOTO 60
   TO
25     IF(C1.EQ.Z(I)) GOTO 60


   STATEMENT 42 CHANGED FROM
20     Z(I)=(C1-Y1*A(M))/A(I)
   TO
20     Z(I)=(C1-Y1*++A(M))/A(I)
```