Scan A6877 etc

RG Wilson v
letter + attachments
8 sides

4 segments

√ f94

Robert G. Wilson → A6877

408 CENTURY PLAZA BUILDING
WICHITA, KANSAS 67202-3276

WILSON ESTATES
316-265-7957

23 January 1989

A6878
A6884
A6885

Neil James Alexander Sloane
% Mathematics Research Center
Bell Telephone Laboratories, Inc.
Murry Hill, New Jersey 07974

Subject: Ulam's Conjecture

Dear Sir,

Please find enclosed copies of the following:

A letter dated 21 July 83 from Prof. Wilbur J. Widmer
18 May 82
14 June 82 Dr. Henry Mullish

the "Titan file" by Fr. Joseph K. Horn, O. Praem CHHU v2n3p36-8
the "Table of "Wonderful Numbers" " from Fr. J.K. Horn.

This sequence will read: A6877

1, 2, 3, 6, 7, 9, 18, 25, 27, 54, 73, 97, 129, 171, 231,
313, 327, 649, 703, 871, 1161, 2223, 2363, 2919, 3711, 6171,
10971, 13255, 17647, 23529, 26623, 34239, 35655, 52527,
77031, 106239, 142587, 156159, 216367, 230631, 410011, ...

It will ~~go~~ to ∞ because 2^n requires n steps.

This sequence is analogous to your seq. 327.

Sequentially yours,

Robert G. Wilson

A6877

RECEIVED JUL 2 9 1983

The University of Connecticut

STORRS, CONNECTICUT 06268

THE SCHOOL OF ENGINEERING
Department of Civil Engineering

Box U-37

July 21, 1983

Mr. Peter Schorer
Hewlett-Packard Company
Computer Research Center
1501 Page Mill Road
Palo Alto, CA 94304

Dear Mr. Schorer:

Thank you for your letter of 24 May 1983 re The Syracuse Problem. Yes, I would be most interested in your results on this; if there is a charge for the manuscript, please let me know. I should have replied sooner to your letter, but could not due to heart surgery (five bypasses) followed by complications of blood clots in the right leg (from which bypass material had been taken).

I agree that mathematics manuscripts (that is, mathematical equations and related symbols) done by hand are much clearer to read than machine-executed symbols. Regarding computer usage, I have often regretted that the practice is to use the diagonal slash through the number zero rather than through the alpha letter "O" (I wonder how this is handled in Sweden where the slashed O is standard alphabet symbol similar in effect to the unlaut used in Germany!).

Re again, The Syracuse Problem: my literature ~~problems~~ 95 on this indicate that the "3n + 1 algorithm" was informally first posed in 1932 by Dr. Lothar Collatz (now retired from his professorship in mathematics at the University of Hamburg, Germany, but still actively involved in international seminars). Collatz wondered "what is the cycle structure of the diagraph on N generated by the number - theoretic function f(x) = (n/2), N even, or = (3N +1), N odd?" Unable to answer his own question himself, he began mentioning it to his colleagues as he visited various places. This information is from a letter which Dr. Collatz wrote several years ago to Dr. Lynn E. Garner of Brigham Young University's Mathematics Department. I have not seen the original letter, and I quote from a letter to me by Dr. Garner. Dr. Garner also has written a paper, "On the Collatz 3n + 1 Algorithm", which was published in the Proceedings of the American Mathematical Society, 82(1) May 1981:19-22.

Richard Guy (who discusses the problem in his 1981 book, "Some Unsolved Problems in Number Theory," Springer-Verlag, New York) wrote to Dr. Garner that "one of the first waves of popularity of the problem followed the 1950 IMV meeting at Harvard where Collatz (<u>again</u>) mentioned the problem to various mathematicians, presumably including Hasse and Kakutani." And Garner also cites (letter to me) Riho Terras as saying that the name "Syracuse Problem" was coined by Hasse during a visit to Syracuse (probably the city in New York). This is consistent with the literature citations (including one by Terras) given by Alf van der Pooten (PPC #3575) in the PPC Calculator Journal V9N6P23. H. Moller (also cited by A.v.d. Pooten) says that Frankel has verified the convergence conjecture for $N<2^{50}$ (Moller: Uber Hasses Verallgemeinerung des Syracuse-Algorithmos-Kakutanis Problem; Acta Arith.XXXIV, 1978, p. 220).

The convergence conjecture (i.e. that for any starting N, Collatz's algorithm always converges to the ever-repeating sequence 1, 4, 2, 1, ...) may have been posed as a separate problem, though it would seem that Collatz himself must have known this rather obvious feature of the question he posed. Since the original problem was not published formally at its inception in 1932, the documentation on its origin is initially by "word of mouth" and not fully definitive. It does appear, though, that Dr. Lothar Collatz is the author. Ironically, I learned of Collatz's relationship to the problem some five days after he departed from an international seminar the participated in here at the University of Connecticut in May 1982 (he even gave a separate seminar lecture to our Civil Engineering faculty during that visit)!

So, I shall be most interested in your paper; and I hope the above-given extended comments will be of some interest to you. You may possibly wish to contact Dr. Lynn Garner at Brigham Young University.

Sincerely yours,

Wilbur J. Widmer
Professor of Civil Engineering

WJW:jrb

cc: Robert G. Wilson (PPC 533)
408 Century Plaza Building
Wichita, Kansas 67202

John Kennedy (PPC 918)
Santa Monica College
1900 Pico Boulevard
Santa Monica, CA 90405

Box U-37

May 18, 1982

Dr. Henry Mullish
Senior Research Scientist
Courant Institute of Mathematical Sciences
New York University
Washington Square
New York, New York 10003

Dear Dr. Mullish:

Recently there was called to my attention a note "Ulam's Conjecture" on page 256 of your 1976 book The Complete Pocket Calculator Handbook (Collier Books, New York).

For several years I have been trying to pinpoint the originator of this conjecture. It is not by Stanislaw Ulam who has written to me, "The conjecture is beautiful. I wish it were mine!" There is some evidence that it may be due to a retired (1979) German mathematician, Lothar Collatz. I am about to write to Dr. Collatz.

I should appreciate it if you will let me know the source of your statement attributing this conjecture to Dr. Ulam. You write that Ulam "not long ago published an interesting paper ...." This is puzzling, since Dr. Ulam himself indicated to me that he is not the author. Possibly your source was a secondary one. In any event I am interested in finding out who actually first posed the problem.

I note also on the same page 256 of your book reference to the "6174" Problem. It is my understanding that this is Kaprekar's "constant"--Kaprekar is a frequent contributor to the Journal of Recreational Mathematics and has privately published a set of papers on "Kaprekar Numbers."

A stamped, self-addressed envelope is enclosed; I do hope you will let me know how you came by the name of Ulam as the author of the conjecture described.

Sincerely,

Wilbur J. Widmer
Professor of Civil Engineering

WJW:cr
Enclosure

Ovrd. again in Programmable Pocket Calculators by Henry Mullish & Stephen Kochan; Hayden Book Co., 1980.

New York University
A private university in the public service

Courant Institute of Mathematical Sciences
Department of Computer Science

251 Mercer Street
New York, N.Y. 10012

Interdepartmental Communication

June 14, 1982

Dear Roy Widner,

You are not the first to question my claim that Allan was responsible for that interesting conjecture. I have never seen it in print. I was told about it by a friend who very confidently spoke of it as Allan's conjecture. It would appear, however, that that is not the case. If the author is indeed lost, I would greatly appreciate hearing of it.

You are right about the "6174" problem. I have since established that it is Kaprekar's constant.

Thank you so much for bringing these matters to my attention.

Yours,

Roy Adler

```
250 IF ERRN=62 THEN USERMSG "File not found",ERRN @ RE
    TURN
260 USERMSG "ERROR "&STR$(ERRN),ERRN @ RETURN
270 DATA "HPILCMDS","DEMOLEX","DEMOLEX1","RIOWIO","INS
    TALL","MCOPY","AUTOLOOP"
```

HP-75 OPENSAFE GAME PROGRAM

Hans E Trixer 150

```
  1  ! HANS E TRIXER [150]
  2  ! 1272 BYTES
  3  ! A GAME PRGM TO GUESS THREE TWO-DIGIT
  4  ! NUMBERS
  5  ! DIFFICULTY FROM 1 TO 10
  6  ! 10 IS EASIEST, 6 IS AVERAGE
  7  !
  8  ! HAPPY SAFE (VAULT) CRACKING!
  9  !
 10  DELAY 2
      @ DISP 'LEADING "0" MUST BE KEYED-IN!'
      @ A=0
 20  DISP "To open the safe, dial xx.xx.xx"
 30  INPUT "How difficult (1-10)? ","5"; D
      @ IF D>10 THEN 30
 40  RANDOMIZE
      @ X=RND*1000000
      @ X$=STR$(INT(X))
      @ F=0
 50  IF LEN(X$)#6 THEN 40
 60  A$=X$[1,2]
      @ B$=X$[3,4]
      @ C$=X$[5,6]
 70  DISP D;"tries for the first pair:"
      @ DELAY 1
 80  FOR I=1 TO D
      @ A=A+1
 90  DISP I;
      @ INPUT ") Number? ";Z$
100  IF Z$>A$ THEN DISP "Too high"
110  IF Z$<A$ THEN DISP "Too low"
120  IF Z$=A$ THEN 140
130  NEXT I
      @ IF I=D+1 THEN DELAY 2
      @ GOTO 300
140  DISP "Correct....now the second pair!"
      @ WAIT 1
150  FOR I=1 TO D
      @ A=A+1
160  DISP I;
      @ INPUT ") Number? ";Z$
170  IF Z$>B$ THEN DISP "Too high"
180  IF Z$<B$ THEN DISP "Too low"
190  IF Z$=B$ THEN 210
200  NEXT I
      @ IF I=D+1 THEN DELAY 2
      @ GOTO 310
210  DISP "Very good-now for the last pair!"
      @ WAIT 1
220  FOR I=1 TO D
      @ A=A+1
230  DISP I;
      @ INPUT ") Number? ";Z$
240  IF Z$>C$ THEN DISP "Too high"
250  IF Z$<C$ THEN DISP "Too low"
260  IF Z$=C$ THEN 280
270  NEXT I
      @ IF I=D+1 THEN DELAY 2
      @ GOTO 320
280  DELAY 2
      @ DISP "BRAVO, you got the safe open!"
      @ F=1
290  DISP "Take the money and"
      @ DISP TAB(15);"R U N"
      @ GOTO 330
300  DISP "Sorry, you missed it!"
      @ GOTO 330
310  DISP "You were doing ok, but missed it"
      @ GOTO 330
320  DISP "Shame, you nearly got it right!"
      @ GOTO 330
330  DISP "The number was: ";A$&"-"&B$&"-"&C$&"..."
      @ WAIT 2
340  IF F=1 THEN DISP "and you took";A;"tries out
     of";D*3
      @ WAIT 2
     DISP "Again Y/N?"
      @ W$=WKEY$
      @ IF W$="Y" THEN 30
360  END
```


Hans E. Trixer [150]
P.O. Box A 140
Avondale
ZIMBABWE

TITAN FILE
Joseph K Horn 13

ULAM IN ASSEMBLY!

[This is not part III of the HP-71 file type series. We have covered SDATA files in Part I (V2 N1 P21), and DATA files in Part II with an example program called "PHONE" (V2 N2 P?). Part III will cover TEXT files. But it will have to wait till next month because something irresistably wonderful is pre-empting it. -jkh]

Editors note. The Phone program example of using data files was supposed to have been in our last issue with part II of Joseph's file type series. Space did not permit its inclusion. It is included with this issue and follows this months column.

Stanislaw Ulam didn't invent it. In fact, nobody knows who did! But it is my favorite unsolved Number Theory conjecture. It is so simple to understand and play with that it is the programmer's perfect exercise.

Here 'tis: Pick any number (not too big). Now do this. If it's an odd number, multiply it by 3, then add 1; but if it's an even number, then simply divide it by two. This gives you a new number. Take this new number and go back to where I said "Now do this." Repeat this process UNTIL YOU REACH 1.

Let's try it with 3:

1) 3 is ODD, so I multiply it by 3 and add 1, which gives 10.
2) 10 is EVEN, so I divide it by 2, which gives 5.
3) 5 is ODD, so I multiply it by 3 and add 1, which gives 16.
4) 16 is EVEN, so I divide it by 2, which gives 8.
5) 8 is EVEN, so I divide it by 2, which gives 4.
6) 4 is EVEN, so I divide it by 2, which gives 2.
7) 2 is EVEN, so I divide it by 2, which gives 1; stop!

Notice that it took seven steps. We started at 3 and got all the way up to 16 before we got down to 1! Some numbers go very high, and take a long time to get down to 1. (Try 27 some time).

Matter of fact, there is no reason at all (that I can think of) that there might not be some starting number that never reaches 1 at all! It might just keep getting bigger and bigger, with a few dramatic backslidings here and there just to keep our hopes up. Or if it's really nasty, it might get caught in an endless loop, going through the same numbers over and over!

Ulam's Conjecture (its popular name) says that all numbers will eventually reach 1. But it's just a conjecture, because nobody's proven it yet. If you can prove it, you will earn eternal fame in the history of Number Theory!

Ulam's Conjecture is the perfect programming exercise. Whenever I am learning a new machine, I program Ulam. My first HP-25, MicroSoft BASIC, HP-67, HP-41 and HP-71 BASIC programs were all Ulam's Conjecture! And now that John Baker is kind enough to explain '71 Assembly Language to us in his column Exploring the 71 IDS, it is time to attack Ulam in Machine Language!

What we want, of course, is to program the Ulam process described above, which the Germans call the "Syracuse Algorithm". Let's call S(x) the Syracuse function. Using this notation, the above example boils down to this:

S(3)=10; S(10)=5; S(5)=16; S(16)=8; S(8)=4; S(4)=2; S(2)=1.

Notice that it takes seven steps for 3 to get to 1, using the Syracuse algorithm. Programming a S(x) function is easy in BASIC, but what we really need is a program that takes S(x) over and over until it hits 1, and keeps count! I want an ULAM(x) function that tells me how many steps it takes x to get to 1! Using our above example, it would boil down to this:

ULAM(3)=7.

Doing it in BASIC is easy:

Joseph K. Horn [13]
1042 Star Route
Orange CA 92667 USA
(714) 633-2041

```
10 DEF FNU(X) @ C=0
20 IF X<=1 THEN 40
30 IF RMD(X,2) THEN X=X+X DIV 2+1 @ C=C+2 @ GOTO 30 ELSE X=X DIV
     2 @ C=C+1 @ GOTO 20
40 FNU=C @ END DEF
```

Enter this program, then type FNU(3) and see 7, because 3 takes 7 steps to get to 1.

I am "cheating" in the algorithm here, taking advantage of the fact that 3x+1 must be an even number if x is odd, so I just divide by 2 right away without testing (line 30). This is to optimize for speed. The algorithm is: (3x+1)/2=x+x\2+1 (x odd).

But try FNU(27). You'll see the right answer, 111, but it takes almost 3 seconds to find it (not counting the .12 seconds of FNU overhead). After all, 27 gets above 9000 before it finally gets down to 1! Try FNU(537099606). It takes almost half a minute to get the answer of 965. That's faster than by hand, but let's do it in Assembly!

The following Assembly Language program creates a LEX file that adds a new BASIC keyword called ULAM. ULAM is a function that takes a number and tells how many steps it takes it to get to 1 by repeated application of the Syracuse algorithm. This lets you type, for example, ULAM(3) and see 7. What is startling, of course, is the speed. ULAM(27) takes about 0.02 seconds (not counting the .04 seconds of ULAM overhead), which is over 100 times faster than BASIC! And ULAM(537099606) returns the answer in 0.12 seconds, which is over 200 times faster than BASIC!

ULAM is useable in CALC mode and in programs. It will accept any input (except NaN), but it will return valid results only for positive integers less than 1E12. It carries its internal calculations out to 20 digits (=16 hex digits), so there is no danger of internal overflow as your number gets bigger and smaller in its journey to 1 (which is better than BASIC!).

To use, either type the mnemonics as shown (with 2 leading spaces where indicated), leaving out the comments, and then assemble; or run MAKELEX (see V2 N1 P20) with the following data:

```
ULAMLEX   ID#99   80 bytes

     0123 4567 89AB CDEF ck

000: 55C4 14D4 C454 8502 A9
001: 802E 0013 0231 5058 4E
002: 0A00 0991 0100 0000 29
003: F710 0000 0000 0000 A7
004: 0710 00F7 55C4 14D4 74
005: 101F F811 8FC1 DB08 0A
006: F2EA 21AF 68FC 0CE0 F9
007: 822A F1AF 3B67 AF69 D1
008: FB02 B758 1C83 2FEA 85
009: 72B7 6AFA B758 2265 E2
00A: EFAF 98F4 BCE0 AFA8 3A
00B: F223 B1AF 68D8 32F0 82
```

NOTE: Don't enter the spaces; they are only visual aids.

One reason that Ulam's Conjecture is so hard to prove is that everybody's been looking at the ULAM function. But ULAM(x) is a useless function, mathematically! What ULAM(x) is tells us nothing about x. It is useless information, as useless as the answer to the question "What is the nth prime number?"

But if you modify the Syracuse algorithm just a tad, you get a useful function! Just two changes, and we have a function that gives useful information (and may be the key to proving Ulam's Conjecture!) First change: don't stop when you reach 1; stop as soon as you fall below the number you started with. Second change: Count as one "step" either the division by two (if even), or multiplying by 3, adding one, and dividing by 2 (if odd). Since this is a Modified Syracuse Algorithm, let's call the function MSA(x). MSA(x) is the number of steps it takes x to fall below itself by repeatedly performing x=x DIV 2 (if x is even), and x=x+(x DIV 2)+1 (if x is odd).

Here's a BASIC routine that does it:

```
40 DEF FNM(X) @ C=0 @ M=X
50 C=C+1 @ IF RMD(X,2) THEN X=X+X DIV 2+1 @ GOTO 50 ELSE X=X DIV
     2 @ IF X>M THEN 50
60 FNM=C @ END DEF
```

The MSA(x) function is "useful" because it tells us something interesting about x. For example, the MSA of any even number is 1, because in 1 step all even numbers fall below themselves (they immediately get divided by 2). MSA(5) is 2, because it gets to 4 in two steps, and 4 is less than the number we started with (5). Notice that this is true for every 4th number starting with 5; i.e. all numbers of the form 5+n2^MSA(5).

MSA(3)=4. This means that every 16th number starting with 3 takes exactly 4 steps to fall below itself; i.e. all numbers of the form 3+n2^MSA(3).

In general, any number x will fall below itself in MSA(x) steps, and besides that x, every number of the form x+n2^MSA(x) will fall below itself in the same number of steps too! If we can prove that all numbers fall below themselves, it follows that all numbers will reach 1, and the Conjecture is proved! But how does one generalize MSA for all numbers?

The Assembly code for ULAM need be changed in only a few places to change it to MSA. Can you do it? Here are a few values of both functions so that you can test your code:

X	ULAM(x)	MSA(x)
27	111	59
703	170	81
1537	153	2
3711	237	37
34239	310	92
35655	323	135
626331	508	176
63728127	949	376
268549803	964	5
999999999999	296	21

Thanks to John Kennedy for first bringing this problem to my attention in the PPC Journal, V6 N1 P9, in which he gave it its immortal name, "Ulam's Conjecture." Douglas Hofstadter spoke of it at length in his book Gödel, Escher, Bach (p. 401), in which he called it the "Wondrous Property" of numbers. A. K. Dewdney described it in his column "Computer Recreations" in Scientific American as "Hailstone Numbers", and Martin Gardner preceded him in his "Mathematical Games" column, same magazine (June 1972 p. 115), calling it a "transcendental problem easy to state but not at all easy to solve." Richard Guy describes it in his book Unsolved problems in Number Theory as the "Collatz Sequence". Heppner, Möller, and Steiner all called it the "Syracuse Algorithm". Riho Terras called it "A stopping time problem

```
          +-------------+
          |   ULAMLEX   |
          +-------------+

+-------------------------------------------------------------+
|   LEX 'ULAMLEX'       * Create a LEX file called ULAMLEX.   |
|   ID #99              * Use LEX ID number 153 (hex 99).     |
|   MSG 0               * No table of special error messages. |
|   POLL 0              * No special poll handlers.           |
|                       * Begin system entry point labels:    |
| POP1N EQU #0BD1C      * Pop argument from stack into CPU regs.|
| RJUST EQU #12AE2      * Convert floating-point to integer.  |
| DCHXW EQU #0ECDC      * Convert decimal integer to hex.     |
| HXDCW EQU #0ECB4      * Convert hex back to decimal integer.|
| FLOAT EQU #1B322      * Convert integer back to floating-point|
| FNRTN4 EQU #0F238     * Put answer on stack & return to BASIC.|
|                       * Begin function text table:          |
|   ENTRY ULAM          * Runtime code starts at label ULAM.  |
|   CHAR #F             * It's a function (not a statement).  |
|   KEY 'ULAM'          * Call it ULAM(x).                    |
|   TOKEN 1             * Make it ULAMLEX's first keyword.    |
|   ENDTXT              * End of function text table.         |
|                       * Begin ULAM's actual assembly code.  |
|   NIBHEX 811          * First & only argument is numeric.   |
| ULAM GOSUBVL POP1N    * Fetch argument from math stack into |
|                       *   CPU register A.                   |
|   GOSBVL RJUST        * Convert argument in floating-point  |
|                       *   form into decimal integer.        |
|   C=A W               * Convert decimal integer into a hex  |
|   GOSBVL DCHXW        *   integer and leave in hex mode.    |
|   SB=0                * Clear Sticky Bit for even/odd test. |
|   B=0 W               * Clear the loop counter.             |
|   D=0 W               * To see if we've reached 1 yet, we need|
|   D=D+1 B             *   a 1; clear D and add 1 to set D=1.|
| TEST C=A W            * C holds our growing & shrinking #.  |
|   ?C<=D W             * Is C<=1 yet?                        |
|   GOYES DONE          * If so, we're done; otherwise:       |
| LOOP B=B+1 W          * Add 1 to the counter.               |
|   ASRB                * A=A\2.  This is the even/odd test;  |
|   ?SB=0               *   if no bit falls off the right end,|
|   GOYES TEST          *   it was even.  Since we're supposed|
|                       *   to divide it by 2, but already did|
|                       *   by the ASRB, just set C=A & repeat.|
|   C=C+A W             * If odd, we can multiply by 3, add 1, |
|   C=C+1 W             *   and divide by two by C=C+A+1, since|
|                       *   (3C+1)/2=C+C\2+1, and A=C\2.      |
|   A=C W               * Get A ready for next even/odd test. |
|   B=B+1 W             * Since we "skipped" a step by dividing|
|                       *   by 2 right away, we must increment|
|                       *   the counter for the skipped step. |
|   SB=0                * Clear Sticky Bit for next test.     |
|   GOTO LOOP           * Repeat Syracuse algorithm.          |
| DONE C=B W            * We're done.  Fetch hex counter into C.|
|   GOSBVL HXDCW        * Convert it to a decimal integer.    |
|   A=C W               *                                     |
|   GOSBVL FLOAT        * Convert it to floating-point form.  |
|   C=A W               *                                     |
|   GOVLNG FNRTN4       * Return it to BASIC.                 |
|   END                 * by Joseph K. Horn [13] 05/13/1985   |
+-------------------------------------------------------------+
```

A6877

*** Table of "Wonderful Numbers" ***

Ulam(2)=1	Ulam(649)=144	Ulam(52527)=339	Ulam(3542887)=583
Ulam(3)=7	Ulam(703)=170	Ulam(77031)=350	Ulam(3732423)=596
Ulam(6)=8	Ulam(871)=178	Ulam(106239)=353	Ulam(5649499)=612
Ulam(7)=16	Ulam(1161)=181	Ulam(142587)=374	Ulam(6649279)=664
Ulam(9)=19	Ulam(2223)=182	Ulam(156159)=382	Ulam(8400511)=685
Ulam(18)=20	Ulam(2463)=208	Ulam(216367)=385	Ulam(11200681)=688
Ulam(25)=23	Ulam(2919)=216	Ulam(230631)=442	Ulam(14934241)=691
Ulam(27)=111	Ulam(3711)=237	Ulam(410011)=448	Ulam(15733191)=704
Ulam(54)=112	Ulam(6171)=261	Ulam(511935)=469	Ulam(31466382)=705
Ulam(73)=115	Ulam(10971)=267	Ulam(626331)=508	Ulam(36791535)=744
Ulam(97)=118	Ulam(13255)=275	Ulam(837799)=524	Ulam(63728127)=949
Ulam(129)=121	Ulam(17647)=278	Ulam(1117065)=527	Ulam(127456254)=950
Ulam(171)=124	Ulam(23529)=281	Ulam(1501353)=530	Ulam(169941673)=953
Ulam(231)=127	Ulam(26623)=307	Ulam(1723519)=556	Ulam(226588897)=956
Ulam(313)=130	Ulam(34239)=310	Ulam(2298025)=559	Ulam(268549803)=964
Ulam(327)=143	Ulam(35655)=323	Ulam(3064033)=562	Ulam(537099606)=965

This table is copied from a letter to Joseph K. Horn from his brother Jim, 17 December 1979. Darrell Kirk wrote a TI-59 program that got up to 10,971 in one week. Jim wrote an HP-67 program that got to 77,031 in one week. Jim's KIM-1, programmed in machine code, finished the table in two weeks (it got to 35,655 in half an hour).

Jim's original table has one erroneous entry: Ulam(127460351)=950. But the Ulam of that number is only 177. I am not sure that the number which I substituted for Jim's wrong entry is the smallest number with an Ulam of 950; this ought to be verified. The other entries are reliable.

All numbers that have a larger Ulam value than do any smaller numbers are called "Wonderful Numbers".