HP Forum Archive 08

Short & Sweet Math Challenges for HP fans #3

*Message #1 Posted by **Ex-PPC member** on 9 June 2002, 12:43 a.m.*

Last week's challenge involved an empirical search for extreme values of determinants, which required to either own an HP calc with built-in matrix operations, or else to remember just how a 3x3 or 4x4 determinant could be calculated, which made it necessary for some of you to reach for that dusty math book on the shelf.

This time we have a new S&S Math Challenge requiring an empirical search, but the underlying concepts are much simpler, no "Matrix Calculus for Dummies" needed. I'll introduce the challenge with an example of what we are going to find out with the help of our precious, museum-quality HP calcs [if you don't use them, they'll dissecate and die, you know]:

Have a look at the number 153. Does it have any interesting peculiarity ? Well, it does have several, the most remarkable one being that:

```
        153 = 1^3 + 5^3 + 3^3
```

i.e.: 153 is a 3-digit number equal to the sum of the 3rd powers of its own digits. Likewise, have a look at 1634, and you'll see that:

```
        1634 = 1^4 + 6^4 + 3^4 + 4^4
```

this is, 1634 is a 4-digit number equal to the sum of the 4th powers of its digits. Let's generalize this to N-digit numbers and so we have the following challenge:

1. Pick up your favorite HP calculator. Almost any will do for the task at hand, as it does not involve anything but arithmetic functions on integer numbers, and negligible memory requirements. Do *not* use any computer for this, the challenge is tailored for the speed and capabilities of HP calculators and using a computer just misses the point by a light-year or more.

2. Find *all* numbers of N digits that are equal to the sum of the N-th powers of their digits. You must find the numbers themselves, as well as just how many there are for each N. The digit 0 *won't* be accepted as first digit, so for instance 153 may be a solution for N=3 but 032 wouldn't even be tested.

3. You should determine all solutions for N=1,2,..., 10. The lower values of N, say up to N=4 are easy and feasible for any HP calculator, from the HP-25 upwards, even using a straightforward programming approach.

4. However, for N=5 to N=10, you'll need to refine your approach significantly if you intend to get results in a reasonable amount of time.

5. For test purposes, this is what you should get:

N	# Solutions	Solutions	Comments
1	9	1,2,3,4,5,6,7,8,9	0 is not acceptable
2	0	none	00 and 01 not acceptable
3	4	153, 370, 371, 407	easy
4	3	1634, ?, ?	easy
5	3	?,?,?	still easy
6	1	?	less easy
7	4	?,?,?,?	medium
8	3	?,?,?	hard
9	4	?,?,?,?	very hard
10	?	?	very, very hard

That's all. May I repeat once more: do *not* use your computer, use your HP calculator. It is much more difficult, but hey, that's where the challenge is. At least see if you can find the ONLY, unique solution for N=6 !

Of course, the real 48/49 nuts among you should try up to N=10, using Saturn Assembler if necessary.

Re: Short & Sweet Math Challenges for HP fans #3

Message #2 Posted by ***Thibaut.be*** *on 10 June 2002, 5:56 a.m.,*
in response to message #1 by Ex-PPC member

Hello,

I programmed a routine on my 41C. To be very honest I first tested it on my computer . I let it run all night and these are the results I found :

n = 4 : 1.634,8.208,9.474 n = 5 : 92.727,93.084 n = 6 = is till running.

Now, I guess there is something to deduct about the series. I'll be thinking about it, but it seems that if my 41C can find in looping the only n=6 solution, it is practically impossible to find the n>6 solutions with a 41...

Re: Short & Sweet Math Challenges for HP fans #3

Message #3 Posted by ***Thibaut.be*** *on 10 June 2002, 10:07 a.m.,*
in response to message #1 by Ex-PPC member

For n=6 : what about 548834 ?

Re: Short & Sweet Math Challenges for HP fans #3

*Message #4 Posted by **Andrés C. Rodríguez (Argentina)** on 10 June 2002, 6:02 p.m.,*
in response to message #1 by Ex-PPC member

This time I'm finding a little time to work in this stimulating challenge. Two questions:

1) Would you expect us to post here our code to solve the challenge, or just the numerical answers?

2) May the digits repeat in the number? (for instance, a number like 1232 would have been a valid answer if 1+16+81+16 equals 1232 (which of course does not!))

I already have a program running in my HP42S (using just 7 registers and less than 50 steps, even with a little error checking and sort of a "user interface")which works very well for numbers between 1 and 4 digits; while it also works for N=5 and greater, execution time will be more than excessive, so I am exploring a different approach...

As a reference, it took some 25 minutes to find all solutions for N=3

Re: Short & Sweet Math Challenges for HP fans #3

*Message #5 Posted by **Ex-PPC member** on 10 June 2002, 7:13 p.m.,*
in response to message #4 by Andrés C. Rodríguez (Argentina)

Some quick answers to the questions raised:

Mr. Thibaut.be wrote:

"I programmed a routine on my 41C [...]I let it run all night and these are the results I found :

```
n = 4 : 1.634, 8.208, 9.474
n = 5 : 92.727, 93.084
n = 6 = [ ...] what about 548834 ? "
```

Your results are correct, except for the fact that it seems you missed the 3rd solution for N=5. An unwanted omission, or else a bug in your code ? As for the result for N=6, it would be interesting to know how long did your 41C took to find it. Will you try N=7 on the 41C or other faster machine ? (I would suggest a 42S, as the code should run unchanged, and it's 10x to 14x times faster. An HP32S or 32SII would be a good choice, too, as they are faster than the 42S, albeit less compatible with the 41C).

Mr. Andres C. Rodrigues wrote:

"Would you expect us to post here our code to solve the challenge, or just the numerical answers?"

As you like. I guess that posting code is Ok as long as it's not too large and it does use some interesting technique that other users may find worthwhile for learning purposes, or for discussing alternatives. Another possibility is to offer guidelines or hints that others may find useful to develop their own code. If code is too large or too straightforward, I guess that posting just the solutions would be preferable.

"May the digits repeat in the number?"

Yes, they may, and often do.

"I already have a program running in my HP42S [...] for N=5 and greater, execution time will be more than excessive, so I am exploring a different approach [...] As a reference, it took some 25 minutes to find all solutions for N=3"

I thank you for your interest in this challenge, Mr. Rodrigues, and please don't take me wrong, but 25 minutes for N=3 in a 42S is *far* too slow. You should definitely need to explore that different approach you mention :-) That is, unless it's a typo and you really meant 25 seconds.

Re: Short & Sweet Math Challenges for HP fans #3

Message #6 Posted by ***Andrés C. Rodríguez (Argentina)*** *on 10 June 2002, 8:09 p.m.,*
in response to message #5 by Ex-PPC member

Ok, thank you... I am even more engaged by the challenge. It was 25 minutes, and rather brute-force based programming, I admit...

While the "other" approach is promising, I still have to find a way to generate certain "patterns" of numbers, something very easy to do by hand, but not that easy so far to convert in HP code. It also may happen to be more elegant but slower than the first method!

Re: Short & Sweet Math Challenges for HP fans #3

Message #7 Posted by ***thibaut.be*** *on 11 June 2002, 8:35 a.m.,*
in response to message #5 by Ex-PPC member

While a loop instruction does not seem to be efficient for values aboce 10^5, I was wondering about expressing this problem in a mathematical way.

Actually I also recalled my linear programmation maths course and the SIMPLEXE algorithm. Have you heard about it ? It is used in economics to calculate maximums or minimums of a linear function under constraints. This is exactly what this is about excepted that the function is not linear.

for instance, for N=5, you have to solve the equation

10.000a + 1.000b + 100c + 10d + e = a^5 + b^5 + c^5 + d^5 + e^5

WHERE

1<=a<=9 0<=b<=9 0<=c<=9 0<=d<=9 0<=e<=9

and a;b;c;d;e are integers

Does this track is the correct one or is it time loosing searching this way ?

Clarification

Message #8 Posted by ***Andrés C. Rodríguez (Argentina)*** *on 11 June 2002, 1:26 p.m.,*
in response to message #4 by Andrés C. Rodríguez (Argentina)

It takes about seven minutes to find 153, 370, 371 and 407. It needs more time to verify there are no more solutions. Still improvable...

What do you think about that !!!!!!

Message #9 Posted by ***Frédéric ALBERT (from France)*** *on 11 June 2002, 6:47 a.m.,*
in response to message #1 by Ex-PPC member

1 : 1 2 3 4 5 6 7 8 9

2 :

3 : 153 370 371 407

4 : 1634 8208 9474

5 : 54748 92727 93084

6 : 548834

7 : 1741725 4210818 9800817 9926315

8 : 24678050 24678051 88593477

9 : 146511208 472335975 534494836 912985153

I have some difficulties for 10 but i am still working on .... I will post the code for hp49g when found !!!! If found ;-(

Fred

Re: What do you think about that !!!!!!

Message #10 Posted by ***thibaut.be*** *on 11 June 2002, 8:24 a.m.,*
in response to message #9 by Frédéric ALBERT (from France)

Félicitations... et quel genre de routine

(Congratulations... and what kind of routine did you use ?)

Re: What do you think about that !!!!!!

Message #11 Posted by ***Jordi Hidalgo*** *on 11 June 2002, 12:16 p.m.,*
in response to message #10 by thibaut.be

Maybe he cheated ... The complete solution
Pickover's book "Wonders of Numbers" is really wonderful.

Regards,
Bye.

Jordi Hidalgo
HPCC member #1046

Re: What do you think about that !!!!!!

Message #12 Posted by ***thibaut.be*** *on 13 June 2002, 6:22 a.m.,*
in response to message #11 by Jordi Hidalgo

Well, no reaction from Mr ex-PPC Member ?

I really wonder if calculator program solutions rather than a loop exist. Though the link shown by Jori explains that what we were looking for are narcissitic numbers, no method to compute them was given.

For n=3 my CV took 14 minutes. So for n=10 it should take 14 minutes * 10^7, or a bit more than 266 years. So, Mr Ex-PPC Member, what is the solution ?

Re: What do you think about that !!!!!!

Message #13 Posted by ***Andrés C. Rodríguez (Argentina)*** *on 14 June 2002, 7:27 a.m.,*
in response to message #12 by thibaut.be

With a looping program with some "improvements", it took my 42S less than 4 minutes to find the 4 solutions for N=3, but it took almost 12 minutes to be sure there are no other solutions. While I have some ideas to try (not sure about them), I have had not enough free time during the week, I would like to have an extra couple of days to see what may happen before the answer is given by Mr. Ex-PPC.

Re: What do you think about that !!!!!!

Message #14 Posted by ***Fernando del Rey*** *on 14 June 2002, 5:27 p.m.,*
in response to message #13 by Andrés C. Rodríguez (Argentina)

I've tried on my HP-71B with a straightforward looping routine, also with some pretty trivial "improvements", and it takes about 320 minutes to find the solution for N=6.

But of course, there's no merit in doing it in BASIC and by "brute force". An this approach will not do the job for N>6 in a reasonable time. There has to be a much more clever way to solve the problem.

By the way, Mr. Hidalgo (a few posts earlier in this thread), please don't spoil the fun for the rest of us by giving the source of the solution away so quickly.

S&SMC #3: Final Remarks [LONG!]

Message #15 Posted by ***Ex-PPC member*** *on 14 June 2002, 11:17 p.m.,*
in response to message #1 by Ex-PPC member

Thanks to all of you who were interested in my S&SMC #3, the challenge is over and here are some final notes and remarks:

As has been mentioned in a post, the numbers who are equal to some mathematical function of their digits are generally referred to as *"Narcissistic Numbers"*. In Challenge #3, we were looking for numbers of N digits that are equal to the sum of the N-th powers of their digits. In the mathematical literature, these are known as *"PluPerfect Digital Invariants"*, i.e: PPDI. They can be defined for bases other than 10, but in what follows, base 10 will always be assumed, and we'll call Nth-order PPDI to a PPDI having exactly N digits. A lot is known about them, for example:

1. There is only a *finite* number of PPDIs, exactly 88. It is very easy to demonstrate that their number is finite, but finding that there are exactly 88 of them does require a lot of ingenuity and computer muscle.

2. There are *no* PPDIs of order N when N=2,12,13,15,18,22,26,28,30,36 nor for N>39, so 39 is the largest possible order.

3. There is *one and only one* PPDI when N=6,10,14,20,32,34,37,38

4. The *largest*, 88th PPDI of order N=39 is the 39-digit number:

   115132219018763992565095597973971522401

   which is equal to the sum of its digits raised to the 39th power.

5. The largest *prime* PPDi is the 23-digit numer:

   35452590104031691935943

   which is equal to the sum of its digits raised to the 23rd power.

The solutions for S&SM Challenge #3 are all PPDI of orders N from 1 to 10, i.e:

```
Order N              Solutions
-----------------------------------------------------
   1       1, 2, 3, 4, 5, 6, 7, 8, 9
   2       none
   3       153, 370, 371, 407
   4       1634, 8208, 9474
   5       54748, 92727, 93084
   6       548834
   7       1741725, 4210818, 9800817, 9926315
   8       24678050, 24678051, 88593477
   9       146511208, 472335975, 534494836, 912985153
  10       4679307774
```

Can our beloved HP calculators find these solutions ? Yes, of course. As I said, almost any model from the HP-25 onwards can find all solutions up to N=4 with relative ease, even using a straightforward brute-force approach. Finding solutions for orders N=5 to N=10 in a reasonable amount of time does require a combination of faster models, such as the HP-71B, HP-32S/SII, HP-42S or HP-48/49, as well as much more refined, optimized programming, and even a completely different approach.

Let's see an example of how can you proceed in order to achieve the goal. In this example, we'll compute all 3 solutions for order N=4, using an HP-71B, as its BASIC language helps to make the programming easier to understand. We'll start with a vey simple brute-force approach, then we will refine it stepwise, creating better and faster versions which will allow us to reach higher orders. Then, once we reach the limits of our direct approach, we will hint at a newer, much faster approach which will deliver what we want. Then, a direct conversion of the HP-71B BASIC code to the HP-41C's native RPN will be shown.

Caveat emptor: You should bear in mind that, in all cases, the programs shown have been produced strictly for demonstration purposes and are not meant to be the ultimate programming solution for this challenge, or even highly optimized, so I don't claim or intend them to be state-of-the-art programming at all.

Program P1-71B:

As we are looking for all 4-digit numbers equal to the sum of the 4th powers of its digits, this means we are looking for numbers which are solutions of this Diophantine equation:

[ABCD] = 1000*A+100*B+10*C+D = A^4+B^4+C^4+D^4

where A goes from 1 to 9, and B,C,D all go from 0 to 9. A very simple program, made essentially of four nested FOR-NEXT loops is quickly produced:

```
10 DESTROY ALL @ DIM A,B,C,D @ STD @ DELAY 0,0
20 FOR A=1 TO 9 @ FOR B=0 TO 9 @ FOR C=0 TO 9 @ FOR D=0 TO 9
30 IF A^4+B^4+C^4+D^4=1000*A+100*B+10*C+D THEN DISP A;B;C;D
40 NEXT D @ NEXT C @ NEXT B @ NEXT A @ DISP "DONE"
```

When run, this program displays all three solutions [1634, 8208, 9474], then terminates displaying "DONE" after verifying there are no more.

Running time: T = 2050 sec.

Program P2-71B:

One of the easiest ways to reduce the running time of a program is to simplify the computations within the innermost loop. In this case, program P1 is continually evaluating the expression:

1000*A+100*B+10*C+D

inside the innermost loop. But a bit of thinking or a quick hand simulation will convince you that this is just the value of the number itself, which is being incremented by one in each iterarion. So we can simply initialize this value to 1000 (A=1, B=0, C=0, D=0), and then increment it after each iteration, thus saving a lot of computation:

```
10 DESTROY ALL @ DIM A,B,C,D,N @ STD @ DELAY 0,0
20 N=1000 @ FOR A=1 TO 9 @ FOR B=0 TO 9 @ FOR C=0 TO 9 @ FOR D=0 TO 9
30 IF A^4+B^4+C^4+D^4=N THEN DISP N
40 N=N+1 @ NEXT D @ NEXT C @ NEXT B @ NEXT A @ DISP "DONE"
```

Running time: T = 1854 sec. (1.11 times faster than P1)

Program P3-71B:

A further refinement comes when we realize that we are continually computing the 4th powers of the digits of N inside the innermost loop. But there are only 10 distinct values, namely 0^4, 1^4, ..., 9^4, so we can save a lot of time pre-calculating them outside of all loops and storing them on a suitably dimensioned array, then simply recalling the values of the powers when needed, instead of computing them anew each time. As it is much faster to recall an array element than to raise a number to a power, great time savings are to be expected:

```
10 DESTROY ALL @ OPTION BASE 0 @ DIM A,B,C,D,N,P(9) @ STD @ DELAY 0,0
15 FOR A=0 TO 9 @ P(A)=A^4 @ NEXT A
20 N=1000 @ FOR A=1 TO 9 @ FOR B=0 TO 9 @ FOR C=0 TO 9 @ FOR D=0 TO 9
30 IF P(A)+P(B)+P(C)+P(D)=N THEN DISP N
40 N=N+1 @ NEXT D @ NEXT C @ NEXT B @ NEXT A @ DISP "DONE"
```

Running time: T = 460 sec. (4.46 times faster than P1)

Program P4-71B:

Is that all ? Can't we still do more optimization ? Yes. Look at the IF at line 30. We are constantly recalling and adding up the powers of the digits to compare them against the value of N. But it's clear that inside the FOR D=... loop, the value of P(A)+P(B)+P(C) is invariant, as it does not depend on the loop variable, D. Same applies to P(A)+P(B) inside the FOR C loop, and P(A) inside the FOR B loop. Keeping this is mind, we avoid recomputing sums and invariant values repeatedly, but we'll simply compute them once outside of the respective loops, and store them in intermediate variables, like this:

```
10 DESTROY ALL @ OPTION BASE 0 @ DIM A,B,C,D,N,S,T,U,P(9) @ STD @ DELAY 0,0
15 FOR A=0 TO 9 @ P(A)=A^4 @ NEXT A @ N=1000
20 FOR A=1 TO 9 @ S=P(A) @ FOR B=0 TO 9 @ T=S+P(B)
30 FOR C=0 TO 9 @ U=T+P(C) @ FOR D=0 TO 9 @ IF U+P(D)=N THEN DISP N
40 N=N+1 @ NEXT D @ NEXT C @ NEXT B @ NEXT A @ DISP "DONE"
```

Running time: T = 320 sec. (6.41 times faster than P1)

Program P5-71B:

There's still one trick out of our sleeve. Just notice what happens when, for instance, we reach the inner FOR C and FOR D loops when A=9 and B=8. At this point, the sum is already 9^4+8^4 = 10657, which exceeds the maximum possible value for N, namely N=9999. So, further adding the 4th powers of C and D is no use, since the resulting sum can never equal N. This means we can skip altogether the full FOR C and FOR D inner loops in this case, so saving 100 full iterations. Applying this simple idea gives us the program:

```
10 DESTROY ALL @ OPTION BASE 0 @ DIM A,B,C,D,N,S,T,U,V,P(9) @ STD @ DELAY 0,0
15 FOR A=0 TO 9 @ P(A)=A^4 @ NEXT A @ N=1000
20 FOR A=1 TO 9 @ S=P(A)
```

```
25 FOR B=0 TO 9 @ T=S+P(B) @ IF T>N THEN N=N+(10-B)*100 @ GOTO 80
30 FOR C=O TO 9 @ U=T+P(C) @ IF U>N THEN N=N+(10-C)*10 @ GOTO 70
40 FOR D=0 TO 9 @ V=U+P(D) @ IF V>N THEN N=N+10-D @ GOTO 60 ELSE IF V=N THEN PRINT N
50 N=N+1 @ NEXT D
60 NEXT C
70 NEXT B
80 NEXT A @ DISP "DONE"
```

Running time: T = 242 sec. (8.47 times faster than P1)

[By the way, just in case you are wondering, declaring the variables with INTEGER precision instead of floating-point DIM is actually *slower*! HP-71B BASIC always works with floating-point variables internally, so declaring them INTEGER just adds a lot of conversions to floating-point precision and back to integer]

So you see, applying just a few, simple common sense ideas to our initial try has resulted in a program that, while still being very simple and not much larger, nevertheless runs nearly an order of magnitude faster. Which is better, the savings are even greater for larger orders of N, so that this approach allows us to compute all solutions (and to certify there are no more) in these running times:

```
Order N      P1      P2      P3      P4      P5
-------------------------------------------------
   3       02:32   02:21   00:40   00:32   00:25
   4       34:10   30:54   07:40   05:20   04:02
   5         -       -       -       -     37:20
   6         -       -       -       -   5:07:13
```

Remember, these times are measured from start to until the program stops by itself, not merely until the last solution is displayed. Times for an HP-32S/SII, HP-42S or HP-48/49 would be even better, as their CPUs are much faster than the old 640 Khz Saturn CPU of the 71B.

However, for N=7 to N=10 our current approach does not work in reasonable times, and we find we have two possible options:

- a) stick to our current approach, but change to a faster language, using for instance FORTH instead or BASIC, or better still, using Saturn Assembler language (using the 71B Forth/Assembler ROM, for instance).

  Converting our program P5 to Saturn assembler is not difficult at all, and we can take advantage of fast custom-made all-integer routines, much faster than the floating-point ones we're using in BASIC. The resulting *binary subprogram* (not BASIC keyword) is faster than our BASIC version by two full orders of magnitude, and so allows us to go up to N=7 in under 30 min., N=8 in some 4 hours, and N=9 in roughly a day and a half continuous running time. The final summit, N=10, would take more than 2 weeks, so this approach's usefulness ends at N=9.

- b) change our current approach for a completely different one. To that effect, we notice that (for our example, N=4) we are exhaustively searching all arrangements from 1000 to 9999, using 4 nested loops, so a total of 9000 cases are tested in all. Our trick in P5 avoids some of these, but only to the point were we have an exponential increase of roughly 8.5x per additional digit instead of 10x. Useful but insufficient.

However, *we are testing far more cases than we actually need !*. For instance, for N=6754, we are computing the sum 6^4+7^4+5^4+4^4 and comparing it versus 6754. But the sum itself is invariant whatever the order of the digits may be: we have Sum4(6754) = Sum4(6745) = Sum4(4567) = ...

So, the magic word is *lists*: you just need to test all *different* lists, *order doesn't matter !!* This tremendously reduces the number of cases we need to test, to the point where, with clever list-generating programming, we can succeed even for N=10 ! And, with the help of a fast PC, we can even find all 88 PPDIs, up to order 39, in a reasonable amount of time.

Of course, I won't give here the solution using this lists approach, that's for you to take over. But for the sake of all of you who tried this challenge on your HP-41Cs or your HP-42S, I'm including a straight conversion of P4 to HP-41C/CV/CX's RPN, for the case N=3:

Program P4-41C:

```
01  LBL "N3"       16  DSE 16          31  RCL 14
02    9            17  LBL 01          32    +
03  STO 00         18  RCL IND 10      33  RCL 15
04  LBL 00         19  STO 13          34   X=Y?
05  RCL 00         20  RCL 16          35  VIEW X
06    3            21  STO 11          36  ISG 15
07   Y^X           22  LBL 02          37  LBL 04
08  STO IND 00     23  RCL IND 11      38  ISG 12
09  DSE 00         24  RCL 13          39  GTO 03
10  GTO 00         25    +             40  ISG 11
11   100           26  STO 14          41  GTO 02
12  STO 15         27  RCL 16          42  ISG 10
13  1.009          28  STO 12          43  GTO 01
14  STO 10         29  LBL 03          44  "DONE"
15  STO 16         30  RCL IND 12      45  PROMPT
```

This 45-line program will compute all four solutions for N=3, in these times:

```
  Time to find and display 1st solution (153) ....        24 sec.
                           2nd solution (370) ....  1 min 37 sec.
                           3rd solution (371) ....  1 min 37 sec.
                           4th solution (407) ....  1 min 49 sec.
                           "DONE" ................  5 min  6 sec.
```

This program will also run unchanged on an HP-42S, only much faster.

That's all. Next S&SMC #4 soon, stay tuned !

Re: S&SMC #3: Final Remarks

Message #16 Posted by ***Andrés C. Rodríguez (Argentina)*** *on 15 June 2002, 2:49 p.m.,*
in response to message #15 by Ex-PPC member

Thank you, Mr. Ex-PPC Member for a nice challenge, I hope next time I will be able to do it better. My best program incorporated some of the techinques you suggested, but I haven't had enough time during the week to fully polish it. For N=3, I needed about 4 minutes to reach 407, and 12 minutes to exhaust all possibilities. Your example is about four times as fast, taking 3 minutes 12 seconds in my HP42S. By the way, the use of registers and steps count was similar in my case.

I was trying a shortcut by means of aborting the inner loop as soon as the sum of the powers exceeded the number under test, since the function is monotonic inside the inner loop. I also stored the differences between the powers in an array, instead of the powers themselves; so a simple RCL + IND 10 updated the X register with the next value to try. At the loop exit, I substracted 9**n from the sum-of-powers last value, and went to the next loop.

A binary search inside the inner loop, halving the range on each iteration may reduce the tests from 10 to 4, but I would lose my job (and-or my family) if I spend that much time and dedication from Monday to Friday. If I find reasonable way to do that in an HP42S, I will let you know.

Thank you again, and keep the challenges coming, please!

Re: S&SMC #3: Final Remarks [LONG!]

Message #17 Posted by ***Andrés C. Rodríguez (Argentina)*** *on 16 June 2002, 8:32 a.m.,*
in response to message #15 by Ex-PPC member

While I know the challenge is over, I finished my HP 42S program, using recursive loops. It asks for M (the order of the problem), between 3 and 10, and then proceeds to find all solutions and to exhaust the possibilities.

For M=3, it took 1 minute 46 seconds to finish.

For M=4, it took 17 minutes.

For M=5, it took 3 hours.

The program uses some 420 bytes of program memory, and runs with 24 registers.

I also find that the generality of a program (opposed to write it just to solve a particular value of M) conspires against its speed.

As I am not familiar enough with the Forum formatting techniques for such a long listing (6 pages), I offer to send a RTF text file with Introduction, Code and Comments to anyone interested, who sends me his-hers email address. Or, after some extra time, I may be able to publish it here.

My program (very long)

Message #18 Posted by ***Andrés C. Rodríguez (Argentina)*** *on 16 June 2002, 8:46 a.m.,*
in response to message #17 by Andrés C. Rodríguez (Argentina)

Well, this is my first attempt at formatting, not 100% readable,

Author: Andrés C. Rodríguez Date: June 16, 2002 Calculator: HP 42S Size=25

Principle of operation:

Accepts "M", which is the number of digits searched for. There are recursive loops for each digits position.

The Units loop contains the test statements, to check if the Number Under test (NUT) equals the Sum of Powers (SOP) of its digits.

Since the Units loop is monotonic, an extra test is used to abort the loop when there are no more solutions in the decade under test.

To obtain greater speed, this program uses an array (R0-R9), which contains the differences between the powers to be added. The last value of such array is negative, allowing for an easy loop-back.

A Diagnostics routine is provided, which slows the program but helps in debugging or understanding, showing the progression of the values under test.

This program fully uses the 8-level pending subroutine return addresses of the HP42S, hence it needs to be adapted for other calculators like the HP 41 family. The Units routine is called via GTO and has fixed return addresses because of this limitation.

In some cases, a value for NUT already incremented is decremented upon the loop exit, since the calling, outer loop would increment it again (like the carry in addition) causing an erroneous condition.

Based upon the "M" value, the program starts at the proper looping routine. The outermost routine loops between 1 and 9, all others loop between 0 and 9.

While it uses recursive loops, a difference array, a shortcut for Units loop abort and a lot of inline code for speed, I still consider it a "Brute Force" approach.

Special symbols:

The | symbol means "line feed" for nicer Alpha displays The words "Roll Down" are used for the so-called function The symbol is the Alpha Append character

```
LBL “PPC3R”             ; Initialize
CLRG
CLST
FIX 0
“Enter M (3 10)|”       ; Get M (| = line feed)
PROMPT
CLLCD
IP                              ; M must be an integer between 3 and 10
3
x>y?
GTO 26
Roll Down                       ; Just for clarity, I write it this way...
10
x<y?
GTO 26
Roll Down
STO 20
0.009
STO 23                  ; Constant used for loop initialization
9
STO 22                  ; R22 will be reused later on
LBL 22                  ; Initialize array of powers differences
RCL 22
1
-
RCL 22
RCL 20
y^x
STO - IND 22
STO + IND Y
DSE 22
GTO 22                  ; Differences are in R0 – R9


; Now, in R(i) we have  [(i+1)**m – i**m]
                ; For instance, for m=3, we should have:


; R0 = (1-0) = 1        R1 = (8-1) = 7
; R2 = (27-8) = 19      R3 = (64-27) = 37
; R4 = (125-64) = 61    R5 = (216-125) = 91
; R6 = (343-216) = 127  R7 = (512-343) = 169
                        ; R8 = (729-512) = 217  R9 = (0-729) = -729 (!)


10                              ; Continue initialization
RCL 20
```

```
1
-
y^x                             ; Calculate 10^(m-1) as initial value, to
STO 21                  ; initialize R21 with Number-Under-Test (NUT)
1                               ; Use 1 to
STO 22                  ; initialize R22 with Sum-of-Powers (SOP)


RCL 20                  ; ***** S T A R T *****
9
+
1.009                           ; The outermost loop starts with 1
XEQ IND Y                       ; Entry point of nested routines based on M.
“DONE |”                        ; When back, prepare final message,
CLST                            ; be polite by clearing the stack,
BEEP                            ; and salute
PROMPT
GTO “PPC3”                      ; Start over
LBL 26                  ; Improper M values
“Bad M |”
TONE 0
PROMPT                  ; Warn the user
GTO “PPC3”                      ; Start over



LBL 19                          ; The following pages contain very
STO 19                          ; similar, nested routines, which
LBL 09                  ; are used to cycle thru each digit
RCL 23                  ; position.
XEQ 18                  ; See comments on LBL 12, the one which
RCL IND(19)             ; takes care of the hundreds digit.
STO + 22                        ; LBL 19 to LBL 12 are essentially the
1       ; same.
2       ; The calls are sort of recursive, from
STO + 21                        ; the most significant digits to the
ISG 19                  ; lesser ones.
GTO 09                  ; The internal labels and addresses are
STO –21                 ; shifted from one routine to the next.
RTN


LBL 18
STO 18
LBL 08
RCL 23
XEQ 17
RCL IND(18)
```

```
STO + 22
        1
STO + 21
ISG 18
GTO 08
STO - 21
RTN


LBL 17
STO 17
LBL 07
RCL 23
XEQ 16
RCL IND(17)
STO + 22
        1
STO + 21
ISG 17
GTO 07
STO -21
RTN






LBL 16
STO 16
LBL 06
RCL 23
XEQ 15
RCL IND(16)
STO + 22
        1
STO + 21
ISG 16
GTO 06
STO - 21
RTN


LBL 15
STO 15
LBL 05
RCL 23
XEQ 14
RCL IND(15)
```

```
STO + 22
        1
STO + 21
ISG 15
GTO 05
STO -21
RTN


LBL 14
STO 14
LBL 04
RCL 23
XEQ 13
RCL IND(14)
STO + 22
        1
STO + 21
ISG 14
GTO 04
STO - 21
RTN



LBL 13
STO 13
LBL 03
RCL 23
XEQ 12
RCL IND(13)
STO + 22
        1
STO + 21
ISG 13
GTO 03
STO -21
RTN


LBL 12                  ; Hundreds loop
STO 12                  ; Get initial value from calling routine
LBL 02          ; Inner hundreds loop starts here
RCL 23          ; Use 0.009 parameter
XEQ 11          ; to calls the decades routine
RCL IND(12)     ; Then, increase the hundreds digit
STO + 22                ; so SOP is properly incremented,
        1                       ; and also increment the NUT by one
```

```
STO + 21
ISG 12          ; Try again with the next hundreds digit
GTO 02
STO – 21                ; The thousands routine will increment it (!)
RTN                             ; After exhausting, return


LBL 11                  ; Decades loop (somehow different)
STO 11                  ; Get initial value from calling routine
LBL 01          ; Inner decades loop starts here
GTO 10          ; Branches to Units routine,
LBL 21          ; providing a return address (!)
RCL IND(11)     ; Then, increases the decades digit
STO + 22                ; so SOP is properly incremented,
        1                       ; and also increment the NUT by one
STO + 21
ISG 11          ; Try again with the next decades digit
GTO 01
STO –21         ; The hundreds routine will increment it (!)
RTN                     ; After exhausting, return



LBL 10                  ; Units routine
RCL 23                  ; Initialize index with 0.009
STO 10
RCL 21                  ; Recall NUT
RCL 22                  ; Recall SOP
        LBL 00          ; internal units loop
        x>y?                    ; If SOP > NUT, go to next decade
        GTO 25          ; (shortcut based on monotonicity)
        x=y?                    ; If SOP =      NUT, This is an answer!!
        VIEW X          ; Show answer without frills, and
RCL + IND (10)  ; properly increase SOP in the stack            ,
        1                       ; and also increment the NUT by one
STO + Z         ; Just by now, we keep the next SOP in X,
Roll Down               ; and the next NUT in Y
ISG 10          ; Try again with the next units digit
GTO 00
STO 22          ; Only before returning, we update SOP
Roll Down               ; and NUT from the stack
STO 21          ; to the proper registers
GTO 21                  ; and go back to the decades routine
LBL 25          ; Remaining of this decade has no solutions,
9       ; increment NUT in R21 (!) by 9
STO + 21                ; Note: is not needed to increment SOP (!)
GTO 21                  ; Go back to the digits routine
```

```
LBL 27                    ; Diagnostics Routine, just for debugging.
CLA                               ; It would be called by XEQ 27, inserted just
ARCL Y                    ; after LBL 00. It slows the program.
" "                               ; Appends a space
ARCL X
" |"                              ; Appends a line feed
AVIEW                             ; Shows running values of NUT and SOP
RTN                               ; (A TONE 7 may replace VIEW X for tests)
```

Yes, I wrote it on a PC...

Message #19 Posted by ***Andrés C. Rodríguez (Argentina)*** *on 16 June 2002, 8:52 a.m.,*
in response to message #18 by Andrés C. Rodríguez (Argentina)

... just as a text file. But I didn't run anything in the PC!

I just used the PC as a text editor, and then keyed the program to the HP42S. How nice a bidirectional interface would have been !!

[Return to Index | Top of Index]

GTO Go back to the main exhibit hall