HP Forum Archive 08

[Return to Index | Top of Index]

Short & Sweet Math Challenges for HP fans #2

*Message #1 Posted by **Ex-PPC member** on 1 June 2002, 9:27 p.m.*

Thanks to all of you who were interested in the very first S&SMC, posted to this forum a week or so ago. Its thread had a total of 17 messages posted (including 3 by myself), so I take this to mean you like the 'section'. Special thanks go to Mr. Andres C. Rodrigues for his enthusiastic support and highly encouraging words. To honour your request, here's a new challenge for all of you and your favourite HP handhelds.

Last week's challenge (the one about the tangents), was mostly theoretical in nature, with just a little programming being useful to empirically try and discover the underlying relationship (a+b+c=180). Once postulated, said relationship could be demonstrated using symbolic math, perhaps with the help of some able HP (such as an Hp48 or HP49), though unfortunately nobody commented on whether those advanced models could cope with it and how.

All in all, it was a rather theoretical (if interesting) challenge, so in order to compensate somewhat, this week's challenge, #2, is much more empirical and will have you and your HPs working hard in order to succeed. It goes like this:

- 1) Take your favourite HP calc which can do matrices, either using fast built-in capabilities or else suitable programs for the task at hand. Specifically, it must be capable of storing a 3x3 or 4x4 matrix and compute its determinant. (Models with that capability built-in in fast microcode include: HP-15C, HP-41C + Advantage ROM, HP-71 + Math ROM, HP-42S, and HP-48/49 models, among several others. Most models can also be easily programmed to compute determinants up to 3x3 at least, so you've got ample choice)

- 2) Now, you must define a 3x3 matrix called A, and fill it up with the integers from 1 to 9, in any order you like as long as none of them are repeated. For instance, you can have your matrix like this:

```
     |  6   1   8  |
A =  |  7   5   3  |
     |  2   9   4  |
```

  and its determinant is det(A) = 360

- 3) The challenge is: find the arrangements of the integers from 1 to 9 which:

1. make det(A) the minimum possible non-zero value.
2. make det(A) the maximum possible value.

We are only interested on absolute values of the determinant, regardless of the sign. The solutions are not unique, of course.

- 4) As you can see, there are fact(9) = 362880 possible arrangements of the integers 1,2,...,9 without repetitions, so you'll have to make good use of your ingenuity and your HP's capabilities if you intend to avoid very long computation times.

- 5) If you succeed in the 3x3 case, try your might with the 4x4 case, i.e: find the arrangements of the integers from 1 to 16 which make the determinant of the 4x4 matrix maximum or minimum (but non-zero). The number of possible arrangements is now fact(16), an impossibly large number to use brute force. Some finesse is indeed required. For instance, can you find an arrangement which makes det(A) = 1 ? Or = 4000 ? or = 4444 ? Or > 40000 ? Which is more, can you theoretically demonstrate what are the maximum/minimum values of the determinant for an NxN matrix ?

A hard nut to crack, but thanks to our HP's quality, it can be done ! Give it a chance, refrain from the temptation to use a pc, and let us all see how you and your HP cope with this challenge !

Re: Short & Sweet Math Challenges for HP fans #2

*Message #2 Posted by **John Ioannidis** on 2 June 2002, 5:24 p.m.,*
in response to message #1 by Ex-PPC member

Several observations:

1. The maximum and the mimimum solutions have the same absolute value (you get one by exchanging two rows or two columns of the other). Hence, we are only going to consider as solution the absolute value of the determinant.

2. The lowest solution is obviously 0, one of which is

```
| 1 2 3 |
| 4 5 6 |
| 7 8 9 |
```

3. Due to symmetry, the number of solutions for each possible value is a multiple of 72 (exchanging rows or exchanging columns does not change the absolute value of the determinant; there are 3 ways to arrange the rows, and another three to arrange the columns. Rotating or transposing does not change the value either, so there is another factor of 8 there; multiply and you get 72).

4. Again due to symmetry, the top left element can be considered to be 1; this leaves 8! instead of 9! permutations to consider, a much more tractable number on a calculator.

5. The really hard part is generating the permutations; it gets harder on HP calculators because you can't use a recursive algorithm and have to unwind the recursion yourself.

Now for some solutions:

1. The highest value is 412. Here is the first solution

```
| 1 4 8 |
| 5 9 3 |
| 7 2 6 |
```

   All the solutions giving 412 are the 72 permutations of this matrix.
2. There are 2736 zero solutions.
3. The most common solution is 45: there are 3456 of those.

Re: Short & Sweet Math Challenges for HP fans #2

*Message #3 Posted by **Ex-PPC member** on 5 June 2002, 10:15 p.m.,*
in response to message #2 by John Ioannidis

Hi Mr. Ioannidis,

It seems that this week's challenge hasn't catch the fancy of math-lover members of this forum, as the only message posted so far has been yours. Thanks for it, but I feel some comments are in order:

Mr. Ioannidis: *"The maximum and the mimimum solutions have the same absolute value (you get one by exchanging two rows or two columns of the other). Hence, we are only going to consider as solution the absolute value of the determinant."*

That's exactly what I said in my description of the challenge: "We are only interested on absolute values of the determinant, regardless of the sign."

Mr. Ioannidis: *"The lowest solution is obviously 0"*

Wrong. In my description of the challenge you can read: "find the arrangements of the integers from 1 to 9 which: 1.make det(A) the minimum possible non-zero value.". Observe that I explicitly requested *non-zero* value. With that condition in mind, the lowest solution cannot be 0, obviously.

Mr. Ioannidis: *"The really hard part is generating the permutations; it gets harder on HP calculators because you can't use a recursive algorithm and have to unwind the recursion yourself."*

Wrong again. At least one model does allow recursion: HP-71B BASIC language allows subprograms and user-defined functions (single-line and multi-line) to call themselves, so permitting recursion naturally.

Mr. Ioannidis: *"Now for some solutions: The highest value is 412."*

Correct. What about the minimum *non-zero* absolute value ?

Mr. Ioannidis: *"There are 2736 zero solutions. 3.The most common solution is 45: there are 3456 of those."*

While this is of course correct, remember I pleaded: " Give it [your HP calculator] a chance, refrain from the temptation to use a pc, and let us all see how you and your HP cope with this challenge !"

While I'm grateful to you for your interest in the challenge and your results, it seems obvious to me that you ignored my plea and did use your PC or computer, not your HP calculator. While this is fun and interesting, you should bear in mind that these challenges are tailored to levels of difficulty affordable for an HP calculator, and are not intended to be solved using a computer, far too easy for that, no challenge at all. No clever techniques to overcome the difficulties either, a brute-force approach suffices.

Let's hope that, before the week elapses, some other kind contributors of this forum will shed some light on the solutions for the minimum 3x3 determinant, and the maximum and minimum 4x4 determinant. Seeing some code for any HP calculator would be great ! And finding a theoretical expression for arbitrary NxN determinants would be eidetic.

So I repeat: put aside you fancy PC at 2 Ghz and let your 71B, your 42S or your 49 do the job, Ok ? :-)

Re: Short & Sweet Math Challenges for HP fans #2

*Message #4 Posted by **Thibaut.be** on 6 June 2002, 4:04 a.m.,*
in response to message #3 by Ex-PPC member

Well, I have to admit that in order to apply to this challenge I had to go back to my old maths books.

I love those quizzes, but I propose that you lower a bit the level of the questions, I guess you'd get more answers...

Re: Short & Sweet Math Challenges for HP fans #2

*Message #5 Posted by **Chris Randle (Lincoln - UK)** on 6 June 2002, 8:53 a.m.,*
in response to message #3 by Ex-PPC member

At least one model does allow recursion: HP-71B BASIC language allows subprograms and user-defined functions (single-line and multi-line) to call themselves, so permitting recursion naturally

All RPL machines can have programs that call themselves too. If you have a program called FAC to calculate a factorial, then:

```
<< -> n << IF n 1 == THEN 1 ELSE n 1 - FAC n * END >> >>
```

works quite nicely with FAC(n) calling FAC(n-1) * n, etc. Tried it on a 28S. I know the "IF n 1 ==" can be replaced by "IF n", but I thought the intent was clearer.

I'm going to have a go at your puzzle now, instead of doing the work I'm paid for ;-) I love puzzles like these for little machines. Thanks for posting them.

Recursion in HP calcs

*Message #6 Posted by **Andrés C. Rodríguez (Argentina)** on 6 June 2002, 11:38 a.m.,*
in response to message #3 by Ex-PPC member

I would just say that the HP 41 programming model (hence including the HP 41C/CV/CX and HP 42S calculators) allow for recursion at the user programs level.

Re: Recursion in HP calcs

*Message #7 Posted by **John K. (US)** on 6 June 2002, 8:19 p.m.,*
in response to message #6 by Andrés C. Rodríguez (Argentina)

[T]the HP 41 programming model [...] allow for recursion at the user programs level.

Unfortunately, due to the limitations one calling chain depth, this ability is somewhat limited. One of the programming projects I was working on many, many moons ago was a routine to copy out the return address registers in the 41 to the main storage registers and back again to seemlessly allow for longer calling chains. I don't think I ever got it working quite right and, since it was (obviously) filled with synthetic commands, it won't work on a 42S. Still, it might be interesting to see if I still have a copy of it somewhere...

Re: Recursion in HP calcs

*Message #8 Posted by **Andrés C. Rodríguez (Argentina)** on 7 June 2002, 7:23 a.m.,*
in response to message #7 by John K. (US)

You are right, I should have said that "...the 41 programming model allow for limited recursion"

Re: Recursion in HP calcs -- Pseudocode

*Message #9 Posted by **Paul Brogger** on 7 June 2002, 6:48 p.m.,*
in response to message #7 by John K. (US)

Here's how I think it would work.

.

.

Scenario:

. Main process named “MP”

. . calls recursive subprocess to figure something.

,

Recursive subprocess named “RSP”

. Figures something in two blocks of code (“RSP_a” & “RSP_b”),

. . with possible recursive self-call in between them.

. Needs to maintain two local variables, “x” & “y” across recursive calls.

,

Logic:

. process MP

. . allocate recursive call/return stack

. . initialize stack pointer to 0

. . [do some work]

. . initialize x = 0

```
. . initialize y = 0
. . set caller = “MP” [could be a numeric code, whatever]
. . GSB RSP_a
. . [ do some more work ]
. . clean up & exit.
. . endprocess MP
.
process RSP_a
. [use x & y]
. push x
. push y
. push caller
. IF recursive descent indicated THEN
. . set caller = “RSP_a”
. . GTO RSP_a
. ELSE
. . GTO RSP_b
. endprocess RSP_a
.
process RSP_b
```

. pop caller

. pop y

. pop x

. [use x & y]

. IF caller = “RSP_a” THEN

. . GTO RSP_b

. ELSE

. . RTN

. endprocess RSP_b

.

"push" & "pop", of course, would utilize an array as a stack to save values and caller codes.

I'll have to try to implement a general solution to, say, the "Tower of Hanoi problem" on my 42s this weekend . . .

Re: Recursion in HP calcs

*Message #10 Posted by **Paul Brogger** on 7 June 2002, 12:56 p.m.,*
in response to message #6 by Andrés C. Rodríguez (Argentina)

If one kept one's own "call/return stack" in user memory in the form of codes indicating what was called and where to return, couldn't one simply use GTO rather than XEQ/RTN (or equivalents)? That should extend the depth of the call/return stack greatly. (Especially in, say, a 32K 42s or a 48/49.)

Re: Recursion in HP calcs

*Message #11 Posted by **thibaut.be** on 7 June 2002, 2:09 p.m.,*
in response to message #10 by Paul Brogger

XEQ <lbl> or GTO <lbl> are exactly the same, aren't they ?

Re: Recursion in HP Calcs

*Message #12 Posted by **Paul Brogger** on 7 June 2002, 2:26 p.m.,*
in response to message #11 by thibaut.be

Details, details!

Actually, that may betray how little I actually am engaged now in *programming* the things.

On my -32s (and presumably the -32sii) I believe that XEQ means "GOSUB" (call with return) and GTO means "GOTO" (go there without expectation of return). If I've got it right, the first enters a return address on the internal return stack, while the latter doesn't.

My HP-48G doesn't seem to have a GOTO, but perhaps I haven't looked closely enough. Regardless, one or more recursive routines could be broken into segments and sequenced through codes passed to special "Call" and "Return" subroutines. (Without a GOTO, it would just be a little more work.)

Of course, any variables whose values need to be maintained across such a recursive call would need to be maintained by the programmer on a stack as well. As with *any* actual implementation, there would be an real (if not a practical) limitation on just how far the process could recursively descend before the end of memory is reached.

I imagine that general-purpose "Recursive Call" and "Recursive Return" subroutines could be written for each calculator, with some documentation detailing how to use them, and what other requirements there may be of the programming and storage for the actual routines being called. (I wouldn't bother with the -32s -- not enough memory. But the -41, -42S & beyond should be capable of supporting such.)

In fact, I don't know what the limit on subroutine descent is in the 49/49 models. I doubt it's anywhere near as limited as it is on the earlier machines. But I don't have my manual handy . . .

Re: Recursion in HP Calcs

*Message #13 Posted by **thibaut.be** on 7 June 2002, 4:56 p.m.,*
in response to message #12 by Paul Brogger

Well, on the 41C the GSB function is well present. Now I have a doubt. The difference between GTO and GSB is, as most of us know I guess, that the RTN function steps to the next instruction after GSB.

I will check that on my 41, but I think the XEQ function works like the GSB in that sense...

Re: Recursion in HP Calcs

*Message #14 Posted by **Andrés C. Rodríguez (Argentina)** on 7 June 2002, 5:05 p.m.,*
in response to message #13 by thibaut.be

In fact, XEQ as used in the HP41/42 replaced the GSB mnemonic used in previous models. The discussion about recursion has to do with the "pending returns" depth of an internal "return addresses" stack (nothing to do with the RPN stack, of course). The HP41 had 6 levels of pending returns, the HP42 increased it to 8; in any case this limits the "recursionability" of these models.

So your statement can be enhanced as "the RTN function steps to the next instruction after GSB" [as long as there are no more than "n" pending returns].

Re: GSB Considered Harmful

*Message #15 Posted by **Paul Brogger** on 7 June 2002, 5:52 p.m.,*
in response to message #14 by Andrés C. Rodríguez (Argentina)

Right. We're not talking about the RPN stack at all.

But, my point is, if you *AVOID* using GSB/RTN (or whatever) to call subroutines, you aren't bound by the limited subroutine call depth supported by the calculator's firmware.

To avoid GSB/RTN, one would have to *simulate* GSB/RTN functionality using application logic that is based on GTO (and probably utilizing arrays). This logic would have to save return "addresses" and local variable values on a stack, pushing the values at the time of pseudo-GSB, and popping those values as part of a pseudo-RTN.

So, given some clever programming and enough memory, you needn't be limited by the depth calculator's internal return stack.

Return addresses

*Message #16 Posted by **Andrés C. Rodríguez (Argentina)** on 7 June 2002, 6:11 p.m.,*
in response to message #15 by Paul Brogger

While we may think about keeping our own management of return addresses, it should be noted that you cannot return to an arbitrary program line. You can simulate a return by means of a GTO to an existent label (somehow identified by your "return handling" extensions), or go to a PHYSICAL address (which is most harmful, indeed!) using synthetic programming and a lot of discipline.

While I was the one who stated that HP41/42 could operate recursively, I am also among the first to admit such feature is very limited for general recursion.

Re: Return addresses

*Message #17 Posted by **John K. (US)** on 7 June 2002, 8:58 p.m.,*
in response to message #16 by Andrés C. Rodríguez (Argentina)

The method I used was actually a little simpler (in theory, anyway) than that mentioned by Mr. Brogger. It involved copying the two status registers ("a" & "b" as I recall), which contain all six return pointers and the program counter, to main memory, starting at an address that the user specified in an initialization routine. Two other registers were also used to keep track of the number of return frames (set of six return pointers) that had been used and the current reference count within each frame (so that the routine knew when it was time to copy the current frame and start a new one), and a third register to keep track of the starting point. I tried using one of the alpha registers for the last, but every once in a while something would happen and a program would munge the value. Lossage would then ensue.

It worked -- to a point. One of the problems I ran into involved the program counter, and another problem was (as is often the case with reference-counting systems) with calls to routines that didn't play along. The details are a little fuzzy in my head since I haven't really thought about this for 15 years or so.

(Wow. High weirdness in the forums.)

Final Remarks

*Message #18 Posted by **Ex-PPC member** on 8 June 2002, 11:35 p.m.,*
in response to message #1 by Ex-PPC member

Thanks to all those people that posted and/or were interested in this thread. This time there were fewer solutions given and no HP-calc code was produced, but I hope that at least you enjoyed the challenged. Now, just some final notes:

- Mr. Ioannidis was able to find the maximum value for the 3x3 determinant, 412. This is indeed correct.
- The minimum *non-zero* value es 1. One such arrangement is:

```
|  1  2  3  |
|  7  5  8  |  =  1
|  4  6  9  |
```

- The minimum *non-zero* value for the 4x4 determinant is also 1. One solution is:

```
|  1  2  3  4  |
|  5  6  7  9  |
```

```
| 8 10 11 12 | = 1
| 16 15 13 14 |
```

A more economic notation for the above solution is:

```
det(1,2,3,4,5,6,7,9,8,10,11,12,16,15,13,14) = 1
```

- As for the additional, miscellaneous questions, here are two solutions:

```
det(1,2,3,4,5,6,16,8,12,15,10,9,14,7,11,13) = 4000
det(1,2,3,4,5,8,14,6,11,16,9,10,15,7,12,13) = 4444
```

- Finally, the solution to the maximum value for the 4x4 determinant, I'll leave that for you. For instance, the following arrangement gives a suitably high value, but ... is it the maximum possible value ? That's for you to find out:

```
det(16,9,7,4,8,1,13,11,3,12,15,5,6,10,2,14) = 39030
```

- Also, many of you were interested in recursion. Actually, it has little to do with this challenge, as the most efficient procedure does not entail using recursion at all, but it is an interesting topic in itself.

  As was stated, some HP calculators do support recursive procedures natively, such as the 71B BASIC, which supports recursive subprograms and user defined functions, and also the RPL models, which by the very nature of their programming model being based on a stack do also permit recursion very easily.

  What to do when a programming language does not support recursion ? Simple, you fake it. For instance, in the 41C normal recursion would be limited to 6 levels deep if using XEQ. So the answer lies in not using XEQ (or GOSUB, GSB, etc. in other HP models), which uses the internal, limited return stack, but simply to use that much-dreaded instruction, GOTO (GTO, etc), together with a fake "return stack", created and maintained by ourselves, typically in some structure like an array, or some suitable registers put aside for this.

  Does this sound difficult, or does it require complex, lengthy programming to achieve it ? Far from it, it's extremely simple, almost trivial. It has been mentioned here that the "Towers of Hanoi" program would be a suitable test case, and it is. As an example, suppose we were to implement this using a version of BASIC that does not support recursion. As the only HP calculator that includes BASIC is the HP-71B and it does support recursion, let's use some other version, say the most simple, limited BASIC of them all, the one used in a famous contemporary of the HP-41C, the original Tandy Radio Shack TRS-80 Pocket Computer (exact clone of the Sharp PC-1211 Pocket Computer). This is a perfect machine for the experiment, as its BASIC is as simple as it gets, it has only 1.4 Kb of memory, and very few commands, not even multiple arrays or two-dimensional arrays.

  Well, in such limited machine/language you can use the simulated-return technique to program the entire "Towers of Hanoi", simulated recursion and all, in just *six* (6) lines of BASIC. That qualifies as absolutely trivial. You can try the technique in your favorite HP calc as well, see how many lines/steps/bytes does it take you to program ToH simulating recursion. You can test your program running this example, which on the TRS-80 PC-1 goes like this (N is the number of disks, limited only by available memory, i.e: more than 50 disks possible)

```
RUN
N=3
FROM 1 TO 3
FROM 1 TO 2
FORM 3 TO 2
FROM 1 TO 3
FROM 2 TO 1
FROM 2 TO 3
FROM 1 TO 3
DONE
```

Re: Final Remarks

*Message #19 Posted by **Chris Randle (Lincoln - UK)** on 9 June 2002, 6:00 a.m.,*
in response to message #18 by Ex-PPC member

I'm still thinking about this puzzle! My brain is slower than most!!

I hope to have some code soon, so could you give me another 24 hours to come up with something (like code that can find the answers) before posting any solution? I mean algorithmic solution - not numeric (which has already been found).

Re: Final Remarks

*Message #20 Posted by **Ex-PPC member** on 9 June 2002, 6:52 p.m.,*
in response to message #19 by Chris Randle (Lincoln - UK)

Chris Randle wrote:

"I'm still thinking about this puzzle! [...] could you give me another 24 hours to come up with something (like code that can find the answers) before posting any solution?"

Thanks for your interest, and don't worry for any deadline to develop your solution, as I rarely post my 'code' solutions, if ever.

Normally I'll post a "Final Remarks" message, which may contain some numeric solutions and/or interesting related notes, hints and references, and that's it.

[Return to Index | Top of Index]