HP Forum Archive 15

Short & Sweet Math Challenge #11: Who's Next ?

Message #1 Posted by ***Valentin Albillo*** *on 13 Oct 2005, 7:50 a.m.*

Hi all,

After several months' leave of absence, here you are, a new **S&SMC** challenge to brighten up your HP programming spirit. After challenges #1, #2, ..., #9, and #10, "who's next" ? #11, you say ? And how on earth did you came to that bold conclusion ?

Well, that's exactly the purpose of this S&SMC#11, to substantiate that very conclusion of yours with hard facts ... well, more like 'soft'(ware) facts, actually. Enter

The challenge

Try and write a program for any HP model that will accept a *sequence of numbers*, then produce its prediction for the *next* element of the sequence.

For instance:

- Given the sequence *1,2,3,4,5,6,7,8,9,10*, your program should produce the value ***11***.
- Given the sequence *11,13,15,17,19*, your program should produce the value ***21***.
- Given the sequence *-9,-16,-25,-36*, your program should produce the value ***-49***.

Your program must accept any number of elements (at least 2), then produce the simplest reasonable prediction for the next element. No provisions should be taken to predict further elements, just the next one.

Of course, mathematically speaking there's always a choice in selecting the next element of an arbitrary sequence, as an infinite number of sequences are possible initially having the very same elements. Thus, for instance, the sequence:

```
      0, 1
```

might continue like this:

```
      0, 1 ->  2, 3, 4, 5, ...
```

or like this:

```
      0, 1 ->  0, 1, 0, 1, ...
```

among infinite other possibilities. In this particular example, choosing 2 for the next element is simplest because in the given sequence, going from 0 to 1 entails a (+1) increment and lacking any more information it seems simpler to stay with it than unexpectedly reverse it to (-1) for no apparent reason. So, given the sequence above (0,1), your program is expected to predict **2** as the next element, not **0** or any other value.

Put your program to the test

Use your program to produce the next element for the following 11 test cases:

1. **:** 11, 13, 15, 17, 19, ?
2. **:** -9, -16, -25, -36, ?
3. **:** 8, 27, 64, 125, 216, ?
4. **:** 0, 0.01, 0.32, 2.43, 10.24, 31.25, ?
5. **:** 2, -3, -4, -1, 6, ?
6. **:** *(in radians)* sin(0), sin(0.1), sin(0.2), sin(0.3), sin(0.4), ?
7. **:** Ln(1), Ln(1.1), Ln(1.2), Ln(1.3), Ln(1.4), ?
8. **:** Pieces of pancake (i.e: flat, 2D) obtained by N cuts: 1, 2, 4, 7, 11, ?
9. **:** Maximum number of electrons in layer N: 0, 2, 8, 18, 32, 50, ?
10. **:** Maximum number of pieces of (3D) cake we get with N cuts: 1, 2, 4, 8, 16, ?
11. **:** 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, ?

Deadline

Midway next week I'll post my solution (plus comments) for the **HP-71B** (which is a simple, 3-line program) and I'll also provide an **HP-15C** version. You might want to try your hand at it this next weekend.

Caveat computat:

Absolutely refrain from using a PC, laptop, or PDA (unless running some HP calc emulator/simulator) to solve the challenge. A Mathematica, Visual Basic, C++, or Java solution is useless to the intended purposes of this challenge, in fact actually defeats them, and submitting one is to be considered unpolite behavior.

And who knows, you might find using your program a useful training for some outdated *'intelligence tests'* ... :-)

Best regards from V.

Edited: 13 Oct 2005, 8:27 a.m.

Re: Short & Sweet Math Challenge #11: Who's Next ?

Message #2 Posted by ***Howard Owen*** *on 13 Oct 2005, 1:45 p.m.,*
in response to message #1 by Valentin Albillo

```
>run
Next N? 1
Next N? 2
Next N? 3
Next N?
OK, what comes next? 4
Seq1=1
Seq2=2
Seq3=3
Next:4! Q.E.D. Baby!
>list
10 DIM N1(100) @ ON ERROR GOTO 20 @ INPUT "Next N? ";N @ C=C+1 @ N1(C)=N @ GOTO 10
20 IF C<2 THEN 10 ELSE INPUT "OK, what comes next? ";N
WRN:Line Too Long
30 FOR I=1 TO C @ DISP "Seq"&STR$(I)&"="&STR$(N1(I)) @ NEXT I @ DISP "Next:"&STR$(N)&"! Q.E.D. Baby!"
```

I knew guys like me in college computer lab, in fact, I *was* a guy like me, who would look for the loophole in every computing problem they saw. The problem statement for this challenge is :

Quote:

"Try and write a program for any HP model that will accept a *sequence of numbers*, then produce its prediction for the *next* element of the sequence.

Then follows a list of sequences to try, ranging from the trivial to the Pretty Tricky(tm). None of that really engaged my "inner prankster," but this line did:

Quote:

Midway next week I'll post my solution (plus comments) for the HP-71B (which is a simple, 3-line program)

Impossible! Well, not quite. 8)

To belabor the possibly obvious point, the problem statement could have said "Try and write a program for any HP model that will accept *only* a sequence of numbers, then produce its prediction for the next element of the sequence." I would then have had less fun with this example. Also, the word "prediction" in the problem statement is obviously a euphemism. Computers don't make predictions, people do, sometimes based on the output of a computer. Given that ambiguity, I feel my usage in the above program fits the stated problem. 8)

I'll bet I can predict what comes next. haha.

Re: Short & Sweet Math Challenge #11: Who's Next ?

Message #3 Posted by ***Thibaut.be*** *on 14 Oct 2005, 6:04 a.m.,*
in response to message #2 by Howard Owen

Valentin,

I have a pretty good idea how to do that. There's actually an in built function that solves such issues. May we use any function of the 71B or 15C, or should we concentrate on primary functions ?

May I also kindly suggest you to propsoe solution on, say, 41C or 42S modes, as it is the easiest to transpose to any HP non cumputer calculator ?

Re: Short & Sweet Math Challenge #11: Who's Next ?

Message #4 Posted by ***Valentin Albillo*** *on 14 Oct 2005, 6:37 a.m.,*
in response to message #3 by Thibaut.be

Hi, Thibaut:

Thibaut posted:

"May we use any function of the 71B or 15C, or should we concentrate on primary functions ?"

You may use whatever functions are available on any HP model, be they built-in or in ROMs or whatever. If there are different ways of doing it depending on the available functions you're welcome to list them as well. The more, the merrier.

"May I also kindly suggest you to propsoe solution on, say, 41C or 42S modes, as it is the easiest to transpose to any HP non cumputer calculator ?"

If by *"propsoe"* you mean "propose" and by *"modes"* you mean "models", my proposal is for each and every HP model and that of course includes both the 41C and 42S. Perhaps you were mislead by my explicitly mentioning the HP-71B and HP-15C, but I was merely saying that I'll provide solutions for these particular models, not that only these models could be used.

Obviously, I can't provide a solution tailored for each specific model, so I chose these particular ones, the HP-71B because routines written for it are easiest to understand and thus convert to other machines, and the HP-15C as a worthy representation of the RPN programming paradigm. But other contributors are both welcome and expected to provide their solutions for other HP models.

Waiting for your clever inputs,

Best regards from V.

Re: Short & Sweet Math Challenge #11: Who's Next ?

Message #5 Posted by **.** *on 14 Oct 2005, 7:08 a.m.,*
in response to message #1 by Valentin Albillo

Hi,

Doesn't this just involve implementing Lagrange Interpolation?

http://mathworld.wolfram.com/LagrangeInterpolatingPolynomial.html

.

Re: Short & Sweet Math Challenge #11: Who's Next ?

Message #6 Posted by ***Thibaut.be*** *on 15 Oct 2005, 12:30 p.m.,*
in response to message #5 by .

That was my idea.

Sorry for the typos Valentin.

Polynomial Extrapolation in RPL

Message #7 Posted by ***Marcus von Cube, Germany*** *on 14 Oct 2005, 5:28 p.m.,*
in response to message #1 by Valentin Albillo

Here is a solution in UserRPL, tested on a 49G+ but should work fine on a 28S, too:

I decided to arrange the values as a function f(x) where x are integers ranging from 0 to n-1 for the given values, with f(0) (or f0 for short) being the last given value and f(n-1) the first. The extrapolated value to be computed is therefore f(-1) or fp for short. This an arbitrary decision but it is workable, as you can see.

My solution fits a polynomial of degree n-1 fo n given values:

```
f(x) = a + b*x + c*x^2 + d*x^3 + ...
f(0) = a
f(1) = a +  b +  c +   d + ...
f(2) = a + 2b + 4c +  8d + ...
f(3) = a + 3b + 9c + 27d + ...
```

This has to solved for [(a = f0) b c d ...] and can be transformed into a matrix equation (here for degree 3, with 4 values given):

```
[ 1  1  1  0 ] [ b  ]   [ f1 - f0 ]
[ 2  4  8  0 ] [ c  ] = [ f2 - f0 ]
[ 3  9 27  0 ] [ d  ]   [ f3 - f0 ]
[ 1 -1  1  1 ] [ fp ]   [ f0      ]
```

This is for the sequence { f3, f2, f1, f0, ? } with ? = fp.

The solution comes in two parts:

MKMAT: Create the matrix

```
<< -> n
   << 1 n 1 -
      FOR r               row counter
        r
        IF n 2 > THEN
          2 n 1 -
          FOR c           column counter
            DUP r *
          NEXT
        END
        0
      NEXT
      1                    last row
      IF n 2 > THEN
        2 n 1 -
        FOR c
          DUP NEG
        NEXT
      END
      1                    very last element of matrix
      { n n } >ARRY        creates matrix
   >>
>>
'MKMAT' STO>
```

SSMC11: Compute the solution.

Input is a {list} of values, like {11 13 15 17 19}. The solution is appended to the list.

```
<< DUP                    make a copy of the list,
   OBJ> -> f0 n           expand to stack and save dimension n
                          and last element f0 in local variables
   << CASE n 2 <          Do nothing if list too short
        THEN n 1 - DROPN
        END
      n 1 - >LIST         convert back to list (without f0)
      -> f                save the list as local variable f
      << n MKMAT INV      create and invert the matrix
         n 1 - 1
         FOR i            backwards loop to reverse the list
           f i GET f0 -   compute f(k)-f0 and leave on stack
         -1 STEP
         f0               last element of right hand side
         { n 1 } >ARRY    create column vector
         *                solve the equation by matrix multiplication
         n GET            pick fp (the predicted value) from result
         +                append to original list (still on stack)
      >>
      END                 end of CASE statement
   >>
>>
'SSMC11' STO>
```

Solutions for test sequences #1 to #6 seem to be accurate.

#6 reveals a value of SIN(0.499989) (I used ASIN to convert back from the numeric value).

I didn' check #7 yet.

#8 reveals 16, #9 reveals 72 but I can't comment on the accuracy.

#10 dosn't work out correctly, the result is 31 instead of 32. The polynomial cannot correctly approximate an exponential function.

#11 reveals 11; obviously the polynomial is heavily oscillating.

Marcus

Re: Polynomial Extrapolation in RPL

Message #8 Posted by ***Valentin Albillo*** *on 17 Oct 2005, 4:42 a.m.,*
in response to message #7 by Marcus von Cube, Germany

Hi, Marcus:

Marcus posted:

"Here is a solution in UserRPL, tested on a 49G+ but should work fine on a 28S, too [...] My solution fits a polynomial of degree n-1 fo n given values"

Thanks for your interesting contribution, but that's really overkill ! :-) The goal asked by the challenge, namely to produce the *next* element of the sequence, can be met by a simple 19-step RPN program for the HP-15C or a one-line program for the HP-71B, which, apart from being much shorter and faster than constructing and then solving a whole system of linear equations to find the coefficients of a polynomial which is then evaluated, they also do not use the large amounts of RAM that your approach requires.

"#10 dosn't work out correctly, the result is 31 instead of 32. The polynomial cannot correctly approximate an exponential function."

Here you are, a new challenge just for you: to try and get more than 31 pieces ouf of a 3D-cake with the next cut. You'll find it impossible, 31 is the correct solution and there's no exponential to be seen here.

Thanks again for contributing and

Best regards from V.

The original author edited out one minor typo

Edited: 17 Oct 2005, 8:55 a.m. after one or more responses were posted

Re: Polynomial Extrapolation in RPL

Message #9 Posted by ***Marcus von Cube, Germany*** *on 17 Oct 2005, 6:26 a.m.,*
in response to message #8 by Valentin Albillo

> Quote:
>
> Maximum number of pieces of (3D) cake we get with N cuts: 1, 2, 4, 8, 16, ?

> Quote:
>
> Here you are, a new challenge just for you: to try and get more than 31 pieces ouf of a 3D-cake with the next cut. You'll find it impossible, 31 is the correct solution and there's no exponential to be seen here.

Valentin,

I don't get the meaning of "cutting a 3D cake". If you simply look at the numbers, the next number is twice its predecessor. For me, this is just the exponential function 2^(n-1), n=1,2,... Computers don't cut cakes, they just deal with the numbers. For *me*, 31 is wrong and 32 should be the result.

Bram's solution seems to give the same results as mine. He must have looked up the resulting coefficients in a text book. Mathemtically, both solutions should be equivalent. But I must admit that, from a programmers standpoint, solving the equation each time is overkill...

What do you think is the best possible solution for case #11? I would say, a linear approximation (least squares) whould give a much more reasonable result than the polynomial.

Marcus

Re: Polynomial Extrapolation in RPL

Message #10 Posted by ***Valentin Albillo*** *on 17 Oct 2005, 7:25 a.m.,*
in response to message #9 by Marcus von Cube, Germany

Hi again, Marcus:

Marcus posted: *"I don't get the meaning of "cutting a 3D cake"*.

You get a typical cake, any 3D-shape as long as it's convex; a regular, cylindrical one will do. You then get some suitably planar cutting object, say a knife, and give the cake a planar cut along any plane that you like. You'll get 2 pieces. You then proceed to administer the poor cake yet another cut, trying to maximize the number of pieces you get. Lo and behold, you'll get four pieces. And so on ... by the time you've administered five such cuts, you'll be in posession of 31 pieces, not 32. Try it.

"If you simply look at the numbers, the next number is twice its predecessor."

That's a 'fortunate' coincidence and nobody told you to 'simply look at the numbers'. The exact wording of test case #10 is:

```
    10.: Maximum number of pieces of (3D) cake we get with N cuts: 1, 2, 4, 8, 16, ?
```

and it's very clearly stated from the start that said numbers represent maximum number of pieces of 3D cake we get with N cuts. I don't see why you decide to ignore the challenge's terms and decide that you'll simply 'look at the numbers' instead. Of course, you're entitled to do that, if you feel like it, but then you're not entitled to be surprised when you fail to get the correct result.

"For me, this is just the exponential function 2^(n-1), n=1,2,...

For me, as I clearly stated in the problem, this is a case of cake-cutting, which happens to return a polynomial number of pieces, not an exponential one. You ignore the stated conditions at your own peril.

"Computers don't cut cakes, they just deal with the numbers. "

Computers don't 'just deal with numbers', they deal with complex electronic interactions between particles and fields and such. That you decide to interpret them as numbers, text, images, or sounds, is up to you.

*"For *me*, 31 is wrong and 32 should be the result."*

Fine with me. I'm not going to get into a quarrel with you for this, couldn't care less, actually. Your problem is, IMHO, that you don't read and/or understand what I stated in the challenge: there is an infinite number of functions f(x) that happen to include any given sequence of datapoints. Among them, you were asked to select the simplest, and mathematically, polynomials are far simpler than exponential functions.

To further eliminate ambiguities, I specifically stated that said values represented the number of pieces you get when administering planar cuts to a 3D cake. This narrows it all to polynomials, once again.

Of course, it goes without saying that I selected this example to make people intuitively fall for 32, an thus be annoyed that their programs were producing 31 instead. They would search for a bug, find none, search the web or experiment a little, and eventually be pleasantly surprised to discover there was no bug after all and their program was smarter than themselves because in real life, contrary to intuition, you do get 31 pieces of cake, not 32. Seems you haven't reached to that point yet.

(your reply to Bram) It's one way to compute the next number, but who decides what is really meant with the given sequence?"

That would be *me*, because I am the one issuing the original challenge, and I clearly state that these numbers are pieces of cake, etc. So that's the meaning of the given sequence for this particular sequence in this particular challenge that I, in particular, issued. :-)

Stop the weeping at having miserably fallen for #10 and have a look at these URLs: this one and this one, too

"What do you think is the best possible solution for case #11?"

Under the conditions given and clearly stated in the challenge, the best possible solution for case #11 is 11.

Thanks again for your interest and

Best regards from V.

Edited: 17 Oct 2005, 7:41 a.m.

Re: Polynomial Extrapolation in RPL

Message #11 Posted by ***J-F Garnier*** *on 17 Oct 2005, 9:36 a.m.,*
in response to message #10 by Valentin Albillo

Hi Valentin

I appreciated your challenge, as usual!

One question: *To further eliminate ambiguities, I specifically stated that said values represented the number of pieces you get when administering planar cuts to a 3D cake. This narrows it all to polynomials, once again.*

How do you know (or demonstrate) that the number of pieces is given by a polynomial?

J-F

Re: Polynomial Extrapolation in RPL

Message #12 Posted by ***Valentin Albillo*** *on 17 Oct 2005, 9:52 a.m.,*
in response to message #11 by J-F Garnier

Hi, Jean-François:

Jean-François posted:

"How do you know (or demonstrate) that the number of pieces is given by a polynomial?"

It's a simple problem of combinatorics. I first read about it decades ago in some Martin Gardner's book about recreational mathematics.

You can find a lot of good references here:

Sequence A000127

Best regards from V.

Re: Polynomial Extrapolation in RPL

Message #13 Posted by ***Marcus von Cube, Germany*** *on 17 Oct 2005, 9:48 a.m.,*
in response to message #10 by Valentin Albillo

Hi Valentin!

> Quote:
>
> They would search for a bug, find none, search the web or experiment a little, and eventually be pleasantly surprised to discover there was no bug after all and their program was smarter than themselves because in real life, contrary to intuition, you do get 31 pieces of cake, not 32. Seems you haven't reached to that point yet.

:-)

In the meantime I wasn't searching the internet but busily cutting a spherical "cake" with an imaginary knife in my head and I started to discover what the callenge meant. I had just been "looking at the numbers" before...

All your test cases were deliberately choosen to be represented by polynomials either excatly or with reasonable accuracy (ln, sin); therefore a polynomial fit would always produce the desired result.

Why is my program so overly complicated? I simply didn't remember any suitable formula, so I started from scratch and tried to figure one out for myself. On my 49g+ it just worked the way it was intended. Bram's solution (and yours will be too) is much more elegant, programming wise. From a mathematical standpoint, they are equivalent.

Re: Short & Sweet Math Challenge #11: Who's Next ?

Message #14 Posted by ***Bram*** *on 17 Oct 2005, 4:06 a.m.,*
in response to message #1 by Valentin Albillo

Hi Valentin,

I must admit that I didn't find the mathematical approach myself, but no problem: the challenge is in the programming.
So I wrote a program for my HP-48G:

```
<< DUP OBJ-> 0 -> N S
  << 1 N
    FOR I
      -1 I 1 - ^ N I COMB * * 'S' STO+
    NEXT
    S + >>
>>
```

It reads a list with the series in it from the stack and adds the next element to the tail of this list. So {8 27 64 125 216} becomes {8 27 64 125 216 343}, ready for the next run (that will add 512).

```
The program gives the following answers:
1.      21
2.      -49 (-(7^2))
3.      343 (7^3)
4.      77.76
5.      17
6.      0.479.. (=SIN(0.5))
7.      0.405.. (=LN(1.5))
8.      16
9.      72
10.     31
11.     11
and {0 1} => {0 1 2} as required
```

I wonder. Does #4 have a special "meaning"? I notice that the second decimal is increasing and that after 5 runs the prediction comes out (exactly?) 1000. That's kind of special, isn't it?

By the way, is it a coincidence that you came up with 11 test cases for this S&SMC#11 and that the answer of the 11th case is 11? I guess you did this on purpose ...

groeten,
Bram

Re: Short & Sweet Math Challenge #11: Who's Next ?

Message #15 Posted by ***Marcus von Cube, Germany*** *on 17 Oct 2005, 4:38 a.m.,*
in response to message #14 by Bram

Bram,

your results are the same as mine but your solution is way shorter! You admit that you didn't work out the math yourself but can you at least explain what your FOR loop is doing?

Re: Short & Sweet Math Challenge #11: Who's Next ?

Message #16 Posted by ***Bram*** *on 17 Oct 2005, 6:21 a.m.,*
in response to message #15 by Marcus von Cube, Germany

With pleasure, Marcus,

```
1.      At first I didn't have a clue how to compute the next element.
        It has something to do with curve fitting was all I could come up with.
2.      Someone posted a link
3.      I read and tried to understand the article
4.      I took formula (4), expanded it to a 4 point situation and
        filled in all x's and y's for the "function" (1,a), (2,b), (3,c) and (4,d)
        that I defined for my list of {a b c d} in order to compute (5,e)
5.      it appears that P(5)=e=-1.a+4.b-6.c+4.d
6.      I happen to spot Pascal's triangle in the "coefficients"
7.      I concluded that the general formula is the sum of (sign . binomial . element)
8.      sign = (-1)^(I-1), binomial = COMB(N,I), element = element from stack after OBJ->
9.      I tried the sigma-function on the HP-48G, but I couldn't figure out
        how to get an indexed array or list element into the formula.
        Fortunately the OBJ-> command converts a list into about the ideal format to process.
        I need to know the length first to enter the FOR loop and
        then one by one the elements are fed into the algorithm.
```

I was very surprised to achieve such a short program.

Re: Short & Sweet Math Challenge #11: Who's Next ?

Message #17 Posted by ***Marcus von Cube, Germany*** *on 17 Oct 2005, 6:36 a.m.,*
in response to message #16 by Bram

Bram,

your answer has just crossed my response to Valentin where I assumed you had the solution looked up in a text book. But you obviously did a lot of thinking yourself :-)

What I think can be easily proven (haven't it tried yet) is that both solutions are simply the same. Solving the matrix equation for the next element (named fp or f(-1) in my post) symbolically should reveal the very same coefficients.

Your solution is quite elegant! But I'm still not convinced that the polynomial approach is always the best solution. It's one way to compute the next number, but who decides what is really meant with the given sequence? (See me argument with Valentin about cutting a cake.)

Marcus

Re: Short & Sweet Math Challenge #11: Who's Next ?

Message #18 Posted by ***Bram*** *on 17 Oct 2005, 10:41 a.m.,*
in response to message #17 by Marcus von Cube, Germany

> Quote:
>
> (...) But I'm still not convinced that the polynomial approach is always the best solution. It's one way to compute the next number, but who decides what is really meant with the given sequence?

Initially I thought that my program would stock 19 as the next number into the list {21 22 25 27 29}, but polynomically you'll have to opt for 36. Apparently real life doesn't always chooses the simplest solution .... ;-)

Re: Short & Sweet Math Challenge #11: Who's Next ?

Message #19 Posted by ***Valentin Albillo*** *on 17 Oct 2005, 6:46 a.m.,*
in response to message #14 by Bram

Hi, Bram:

Bram posted: *"I must admit that I didn't find the mathematical approach myself, but no problem: the challenge is in the programming."*

First of all, thanks for your continued interest in my S&SMCs and your kind contributions. I did find the mathematical approach myself, it's actually quite easy, but once you do, the programming is simple and the challenge, at least for me, is to optimize it as much as possible for program size, memory requirements and speed.

"So I wrote a program for my HP-48G:"

My original solutions (a 19-step RPN program for the HP-15C and a one-line program for the HP-71B) are quite similar to yours, except they're even simpler, making no use of the COMB functionality (which the HP-15C has but the HP-71B lacks), and thus can be easily adapted to most any HP model, including the HP-25C, HP-10C, or HP-12C, in 25 steps or so, say, with absolutely minimal memory requirements as no datapoints need be stored.

"The program gives the following answers:"

```
1.      21
2.      -49 (-(7^2))
3.      343 (7^3)
4.      77.76
5.      17
6.      0.479.. (=SIN(0.5))
7.      0.405.. (=LN(1.5))
8.      16
9.      72
10.     31
11.     11
and {0 1} => {0 1 2} as required
```

All of them absolutely correct, of course, except #6 and #7 return approximations to sin(0.5) and ln(1.5), not the exact values. Most people would think that #10 should be 32, but it isn't, you can't get more than 31 pieces by giving that number of planar cuts to a convex 3D-cake, here human intuition guesses 32 and flunks out.

"I wonder. Does #4 have a special "meaning"? I notice that the second decimal is increasing and that after 5 runs the prediction comes out (exactly?) 1000. That's kind of special, isn't it?"

Yes, the values for #4 were generated by y = x^5/100, for x from 0 to 5, so the next value is 6^5/100 = 7776/100 = 77.76, and you eventually get to 10^5/100 = 100000/100 = 1000, exactly. It was an attempt to obfuscate things a little by adding decimals, probably quite successful. Without the decimals (i.e.: 0, 1, 32, 243, 1024, 3125) the generating function is probably easier to guess (y = x^5).

"By the way, is it a coincidence that you came up with 11 test cases for this S&SMC#11 and that the answer of the 11th case is 11? I guess you did this on purpose ..."

Yes, that's no coincidence at all. :-) I thought it would be fun for smart readers like yourself to see S&SMC#11 include a test case #11 which features an 11-element sequence whose next element is precisely 11. :-) The nice touch is to make sure that this 11-element sequence looks interesting and its next element is far from obvious, so it comes as a nice surprise when you realize its value is exactly 11.

Best regards from V.

S&SMC#11: My solutions & comments [LONG]

Message #20 Posted by ***Valentin Albillo*** *on 18 Oct 2005, 4:31 a.m.,*
in response to message #1 by Valentin Albillo

Hi, all:

Thanks to all of you interested in my S&SMC#11, most specially to those who contributed their own solutions and comments.

S&SMC#11 asked for a program to compute the *next* element of a given sequence. As some of you observed, this could be considered a case of *polynomial fitting*, where we first compute the coefficients of a polynomial which includes all values in the sequence, then we evaluate the polynomial to get the next value.

However, though workable, this is simply *overkill* as we're not interested in making further predictions or interpolations, just getting the *very next element*, so we can do without the complexity, time, and memory resources necessary for a full-fledged polynomial fit. We'll use a much simpler mechanism, based in **finite differences**.

The details are better understood with an example. Imagine we're asked to predict the next element of the sequence which begins (8, 27, 64, 125). We then form the following disposition, where the first row contains the sequence's elements and each number in the rows below is computed as the difference of the two above it, like this:

```
                     n1    n2    n3   n4
                     ---------------------
  original sequence:  8    27    64   125
    1st differences:    19    37    61       (19 = 27- 8, 37 = 64-27, 61 = 125-64)
    2nd differences:       18    24          (18 = 37-19, 24 = 61-37)
    3rd differences:          6              ( 6 = 24-18)
```

where we stop as soon as there's a *single value* in the last row. Now, to predict the next element, lacking futher information we simply assume that the last, Nth differences are constant, and proceed to compute the next entries *adding upwards*, row by row, like this:

```
                     n1     n2    n3   n4      n5
                    -------------------------------
  original sequence:  8     27    64   125 ..  216   (predicted next element = 125 + 91)
    1st differences:    19    37    61 ..   91       (91 = 61 + 30
    2nd differences:       18    24 ..  30           (30 = 24 + 6)
    3rd differences:          6  ..  6               (3rd differences are constant = 6)
```

and so we get **216** as the predicted value for the next element. In this particular case, the original elements were 2^3, 3^3, 4^3 and 5^3, so we would expect 6^3 to be the next element, which it is (6^3 = 216).

Given this simple, finite-difference algorithm, we could write a short program to input the original sequence, then compute the finite differences to form a triangular disposition as seen above, and use it to obtain the predicted next value. But we can proceed more economically by having a closer look at how the predicted value is computed. In the example above, the predicted value (216) is computed like this:

```
   216 = 125 + 91
       = 125 + 61 + 30
       = 125 + 61 + 24 + 6
       = 125 + (125-64) + 24 + 6
       = 125 + (125-64) + (125-64)-(64-27) + 6
       = 125 + (125-64) + (125-64)-(64-27) + [(125-64)-(64-27)]-[(64-27)-(27-8)]
       = 125 + 125 - 64 + 125 - 64 - 64 + 27 + 125 - 64 - 64 + 27 - 64 + 27 + 27 -8
       = 125*4 - 64*6 + 27*4 - 8
```

this is, in the notation above:

```
    n5 = n4*4 - n3*6 + n2*4 - n1
```

Doing likewise with 5- and 6-point sequences, we similarly get:

```
    n6 = n5*5 - n4*10 + n3*10 - n2*5 + n1
```

and

```
    n7 = n6*6 - n5*15 + n4*20 - n3*15 + n2*6 - n1
```

and the coefficients which multiply the n's immediately reminds us of **Pascal's triangle**:

```
              1
            1   1
          1   2   1
        1   3   3   1
      1   4   6   4   1
    1   5   10  10  5   1
   1  6   15  20  15  6   1
```

where each number is the *sum* of the two above it. But every element in each row of Pascal's triangle is given by a *binomial coefficient* **C(m,n)** (*number of combinations of m elements taken n at a time*), so, for the last row above we have:

```
    n7 = n6*C(6,5) - n5*C(6,4) + n4*C(6,3) - n3*C(6,2) + n2*C(6,1) - n1*C(6,0)
```

and similarly for the general case. This can be readily implemented in any HP model, no matter how small (say HP-10C, HP-12C, HP-25), because:

- no elements need to be stored: each is used only *once*, so they can be input one at a time, interactively, to be immediately used.
- no need to compute each C(m,n) = m!/n!/(m-n)!, because each binomial coefficient can be computed *from the preceding one* by performing just one integer multiplication and one integer division.

Here's the resulting 19-step RPN program for the HP-15C:

```
   LBL A
   STO I     store the number of elements
     1       initialize an auxiliar counter
    CHS
   STO 0
   STO 1     initialize running binomial coefficient
    CLX      initialize running sum
  *LBL 0
    R/S      accept input from the user
   RCL 1     update binomial coefficient
   RCL* I
   RCL/ 0
   STO 1
     *       update running sum
     +
   DSE 0     update auxiliar counter
  *LBL 0
   DSE I     update primary counter
   GTO 0     if not done, go for another input
             done, stop with next element in display
```

which uses no allocatable registers (just the permanent registers R0, R1, RI) and works very fast for any number of elements. The sequence's elements are introduced one at a time, in reverse order (this saves a few extra steps at no cost since you press exactly the same keystrokes as you would if entering them in order). Here's a couple of trial runs:

Sequence 8, 27, 64, 125, ?

```
FIX 2
    4  [A]  125 [R/S]  64 [R/S]  27 [R/S]  8 [R/S] -> 216.00
```

Sequence 0, 0.01, 0.32, 2.43, 10.24, 31.25, ?

```
    6  [A]  31.25 [R/S] 10.24 [R/S]  2.43 [R/S]
             0.32 [R/S]  0.01 [R/S]     0 [R/S] ->  77.76
```

This program can be very easily translated to most any HP model, an HP-25 version or HP-12C version are about 25 steps long. It can also be converted straight away to an **HP-71B** version, namely the following *one-liner* (78 bytes):

```
1 INPUT "N=";I @ A=I+1 @ B=1 @ S=0 @ FOR I=-I TO -1 @ INPUT "X=";X @ B=B*I/(A+I) @ S=S-X*B @ NEXT I @ DISP S
```

Trial run, test 4: 0, 0.01, 0.32,2.43, 10.24, 31.25, ?

```
>RUN


N=6      [ENTER]


X=31.25  [ENTER]
X=10.24  [ENTER]
X=2.43   [ENTER]
X=0.32   [ENTER]
X=0.01   [ENTER]
X=0      [ENTER]


 77.76
```

That's all. Thanks again for your interest and

Best regards from V.

Re: S&SMC#11: My solutions & comments (edited)

Message #21 Posted by ***Arnaud Amiel*** *on 18 Oct 2005, 9:13 a.m.,*
in response to message #20 by Valentin Albillo

Thanks, and as usual it all looks so simple now. I will try to find time to get it working on my 55.

However, I have always been reluctant to use interpoling beyond the interval where I have data as it usually produces rubbish (unless you know which "physics" law to follow. Anyone knows of rules validating the use of interpolation beyond the interval?

Thanks,

Arnaud

Edited: 18 Oct 2005, 10:52 a.m.

Re: S&SMC#11: My solutions & comments

Message #22 Posted by ***Bram*** *on 19 Oct 2005, 5:06 a.m.,*
in response to message #20 by Valentin Albillo

Hi Valentin,

Thank you for your explanations and your programs, which are very compact indeed. The first one runs on my HP-32SII apart from the duplicate label. The 32SII doesn't allow this. As it is a dummy line to ignore a skip of the DSE 0, it can of course be replaced with another label.

All in all it was nice exercising.

S&SMC#11: Flubbing Fibonacci?

Message #23 Posted by ***Karl Schneider*** *on 19 Oct 2005, 11:53 p.m.,*
in response to message #20 by Valentin Albillo

Valentin --

Thanks for posting a clever and interesting exercise. The method and HP-15C program you provided seems fairly clever; I'll try the HP-71B version when I get more proficient with it.

Curiously, while the program (adapted just slightly for the HP-41C and HP-42S) works as advertised for the examples you provided, it does not give the correct results for the simple Fibonacci sequence:

$a_{n+1} = a_n + a_{n-1}$

0, 1, 1, 2, 3, 5, 8, 13, 21, ...

I get either 0 or twice the correct value as the next term (depending upon how many terms were specified).

I have not analyzed the technique to explain why this is so. Would you have some mathematical insights about this?

Thanks,

-- KS

Re: S&SMC#11: Flubbing Fibonacci?

Message #24 Posted by ***Crawl*** *on 20 Oct 2005, 12:14 a.m.,*
in response to message #23 by Karl Schneider

The Fibonacci sequence isn't a polynomial sequence.

The nth term can be given in closed form

Fn = ((1 + sqr(5))^n - (1 - sqr(5))^n)/(sqr(5)*2^n)

S&SMC#11: Followup to "Flubbing Fibonacci"

*Message #25 Posted by **Karl Schneider** on 20 Oct 2005, 3:19 a.m.,*
in response to message #23 by Karl Schneider

I posted,

> Quote:
>
> Curiously, while the program (adapted just slightly for the HP-41C and HP-42S) works as advertised for the examples you provided, it does not give the correct results for the simple Fibonacci sequence:
>
> $a_{n+1} = a_n + a_{n-1}$
>
> 0, 1, 1, 2, 3, 5, 8, 13, 21, ...
>
> I get either 0 or twice the correct value as the next term a_{n+1}(depending upon how many terms were specified).

The program gives 0 as the next term a_{n+1} for odd $n > 1$; it gives double the correct value of the next term for even $n > 1$. So, if a_{n+1} represents the correct value of the next term,

$a_{n+1} * [1 + (-1)^n]$ is returned.

It's somewhat correct... ;-)

-- KS

Re: S&SMC#11: Flubbing Fibonacci?

*Message #26 Posted by **Arnaud Amiel** on 20 Oct 2005, 3:30 a.m.,*
in response to message #23 by Karl Schneider

That was my concern, see message 21 above. However I didn't have time to find an example. I didn't have time to think about the challenge either but I wouldn't have found the answer, exactly because of this.

Arnaud

Re: S&SMC#11: Flubbing Fibonacci?

*Message #27 Posted by **Valentin Albillo** on 20 Oct 2005, 5:44 a.m.,*
in response to message #23 by Karl Schneider

Hi, Karl:

Karl posted:

"Would you have some mathematical insights about this?"

Yes, the finite difference algorithm my solution implements will produce the 'correct' next element (in the sense given) for any sequence generated by evaluating polynomials of any degree at equally spaced data points.

The Fibonacci sequence you give is usually defined as a linear recurrence, in which the Nth term depends on a linear combination of preceding terms. These kind of recurrences usually result in sequences generated by either exponentials or trigonometrics (which are nothing more than exponentials for complex arguments), but not polynomials except in degenerate cases.

Your sequence is 1,1,2,3,5,8,13,21,..., where each term is computed from the preceding ones like this:

$$a_{n+2} = a_{n+1} + a_n$$

i.e., each term is the sum of the two preceding ones. Thus, its nth-term can be expressed as a function of n in this way:

$$a(n) = C_1.x_1^n + C_2.x_2^n$$

where C_1,C_2 are suitable constants and x_1,x_2 are the two roots of the equation associated to the recurrence, namely (notice how closely it mimics the form of the recurrence itself):

$$x^2 = x + 1$$

so x_1 and x_2 are, respectively, (1+sqr(5))/2 and (1-sqr(5))/2, and you can determine C_1,C_2 by making use of the initial conditions:

```
a(1) = 1
a(2) = 1
```

resulting in the values:

```
     C1 = 1/sqr(5)
     C2 = -1/sqr(5)
```

so you finally get:

```
     a(n) = (((1+SQR(5))/2)^n-((1-SQR(5))/2)^n)/SQR(5)
```

which, certainly, it's *not* a polynomial.

If the roots of the associated equation turned out to be complex, then the general term would involve trigonometrics, but the details are fairly similar. Which is more, you can leave everything the same, with exponentials of imaginary values, and if your HP model allows you to compute them, it would work perfectly Ok.

After all, who needs a sin(x) function when you can compute it as (e^(i*x) - e^(-i*x))/(2*i) ? :-)

Best regards from V.

Thank you, V [Re: S&SMC#11]

Message #28 Posted by ***Karl Schneider*** *on 20 Oct 2005, 10:43 p.m.,*
in response to message #27 by Valentin Albillo

Thank you, Valentin -- an excellent and informative explanation, as usual.

The test case 4 sequence:

0, 0.01, 0.32, 2.43, 10.24, 31.25

I could ask what its definition is, but I expect that I could find out by running your recently-posted HP-71B polynomial curve-fitting routine. Could the HP-41C Stat Pac PC-F routine also find it (maximum order = 3)?

-- KS

Re: Thank you, V [Re: S&SMC#11]

Message #29 Posted by ***Valentin Albillo*** *on 21 Oct 2005, 4:23 a.m.,*
in response to message #28 by Karl Schneider

Hi, Karl:

Karl posted:

"Thank you, Valentin -- an excellent and informative explanation, as usual."

You're welcome, I'm really fortunate to have such kind, keen, and appreciative 'readers'. :-)

"The test case 4 sequence: 0, 0.01, 0.32, 2.43, 10.24, 31.25 I could ask what its definition is, but I expect that I could find out by running your recently-posted HP-71B polynomial curve-fitting routine. Could the HP-41C Stat Pac PC-F routine also find it (maximum order = 3)?".

It is:

```
    P(x) = (x^5)/100
```

for x= 0,1,...,5 and yes, any curve-fitting routine worth its salt can find it, provided it copes with degrees up to 5, so perhaps the PC-F routine will be unable to find this one (the best a 3rd-degree polynomial can achieve with these data is to have an absolute maximum error H=0.52, see below). Next element is, of course, (6^5)/100 = 7776/100 = 77.76.

My HP-71B Minimax Polynomial program that you mention can (of course!!) find this, but it's absolutely *overkill* ! We have an idiom in Spanish for this, namely *"Matar moscas a cañonazos"* which literally means *" To kill flies with a canon"* but properly translates in English as *"To crack a nut with a sledgehammer"*. Nevertheless, a sample run of my program goes as follows:

```
  >CAT


     MINIMAXP   BASIC  3018 07/09/04 06:14


  >RUN


       Verbose ? (Y/N):  N
    [K]bd,[D]ef,[F]ile=  K
              # Points=  6
               #1: X,Y=  0,0
               #2: X,Y=  1,0.01
               #3: X,Y=  2,0.32
               #4: X,Y=  3,2.43
               #5: X,Y=  4,10.24
               #6: X,Y=  5,31.25
               Store in  *
```

```
          Degree=
      Max. error=  0


Degree 1 ... H= 8.16000000000   (max. error for a 1st-degree polynomial)
Degree 2 ... H= 2.76666666690   (max. error for a 2nd-degree polynomial)
Degree 3 ... H= 0.52000000000   (max. error for a 3rd-degree polynomial)
Degree 4 ... H= 0.03750000000   (max. error for a 4th-degree polynomial)
Degree 5 ... H= 0.00000000000   (max. error for a 5th-degree polynomial)


A0= 0.00000000000
A1= 0.00000000000
A2= 0.00000000000
A3= 0.00000000000
A4= 0.00000000000
A5= 0.01000000000
```

Thus, it has found that these datapoints are perfectly represented (absolute maximum error H = 0.00000000000) by the 5th-degree polynomial:

```
P(x) = A5*x^5 = 0.01*x^5 = (x^5)/100
```

Thanks again for your interest and kind words and

Best regards from V.

Edited: 21 Oct 2005, 4:34 a.m.

[Return to Index | Top of Index]

GTO Go back to the main exhibit hall