Happy New Year 2024 ... and 2023's last teaser !

Threaded Mode | Linear Mode

31st December, 2023, 18:00 **Post: #1**

Valentin Albillo
Senior Member

Posts: 1,075
Joined: Feb 2015
Warning Level: 0%

Happy New Year 2024 ... and 2023's last teaser !

.

Hi, all,

Happy New Year 2024 to all of you

but before ***2023*** truly elapses, this fateful year which brought us the awesome *HP-15C CE*, let's *amazing-grace* it with a final ***teaser*** for those of you who would accept it. Very succinctly, concisely and even tersely:

> You all know the matrix ***Determinant***, *aka* **det**. Now, the matrix ***Permanent***, *aka* **per**, is defined likewise but with *all signs positive*, no negative terms, i.e. like this:
>
> $$\operatorname{per}\begin{pmatrix} a & b \\ c & d \end{pmatrix} = ad + bc$$
>
> $$\operatorname{per}\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} = aei + bfg + cdh + ceg + bdi + afh$$
>
> and the teaser is that you use the new *Christmas* toy calc you got last *Dec, 25th* (or any other, for that matter) to obtain the *permanent* of this nice little matrix I just lovingly concocted specially for you. The result is sure to ***surprise*** you ! :)

```
| 1 0 1 1 1 0 1 1 1 1 1 1 |
| 0 0 1 1 0 0 0 0 0 1 0 1 |
| 1 0 1 1 0 0 0 1 0 1 1 1 |
| 0 0 1 0 0 0 0 0 0 1 0 0 |
| 1 0 1 1 1 1 1 1 1 1 1 1 |
| 0 0 1 1 0 0 0 0 0 1 1 1 |
| 1 0 1 1 1 0 0 1 0 1 1 1 |
| 0 1 0 1 1 1 1 1 1 1 0 1 |
| 1 0 1 1 0 0 0 0 0 1 1 1 |
| 1 1 1 1 1 1 1 1 1 1 1 1 |
| 0 0 1 0 0 0 0 0 0 1 0 1 |
| 1 0 1 1 1 0 1 1 0 1 1 1 |
```

Admittedly this teaser is only for *power* calc users so don't feel too bad if you're not up to the task (I only needed 3 lines of code for my 4-decades old toy.) :)

Best regards and, again, ***Happy New Year 2024 !***
V.

PM | WWW | FIND   EDIT | QUOTE | REPORT

31st December, 2023, 20:24 **Post: #2**

Eddie W. Shore
Senior Member

Posts: 1,554
Joined: Dec 2013

RE: Happy New Year 2024 ... and 2023's last teaser !

Happy New Year, Valentin! And Happy New Year, MoHPC members!

EMAIL | PM | WWW | FIND   QUOTE | REPORT

31st December, 2023, 21:11 (This post was last modified: 31st December, 2023 21:16 by Jlouis.) **Post: #3**

Jlouis
Senior Member

Posts: 764
Joined: Nov 2014

RE: Happy New Year 2024 ... and 2023's last teaser !

Happy New Year to you all!

Let 2024 brings much more Collectors Editions!

Peace, love, Live long and Prosper!

All the best

JL

Attached File(s)

Thumbnail(s)

EMAIL PM FIND QUOTE REPORT

31st December, 2023, 21:54 (This post was last modified: 1st January, 2024 11:38 by Peter Klein.) **Post: #4**

Peter Klein
Junior Member

Posts: 46
Joined: Aug 2014

RE: Happy New Year 2024 ... and 2023's last teaser !

Happy New Year, all! May all your equations be solvable, whether in RPN, or in life.

EMAIL PM FIND QUOTE REPORT

1st January, 2024, 10:05 **Post: #5**

EdS2
Senior Member

Posts: 559
Joined: Apr 2014

RE: Happy New Year 2024 ... and 2023's last teaser !

Happy New Year!

Thanks for the teaser Valentin... from what I read, the permanent is expensive to compute... but then, you are not suggesting your program ran in a short time, only that it is a short program. I was almost tempted to compute this one by hand, but realised I'd misunderstood the definition.

EMAIL PM FIND QUOTE REPORT

1st January, 2024, 11:02 (This post was last modified: 5th January, 2024 09:18 by jthole.) **Post: #6**

jthole
Member

Posts: 80
Joined: Nov 2017

RE: Happy New Year 2024 ... and 2023's last teaser !

Happy new year as well! And that is a creative end to **2023** :-)

I am very interested in your solution, because I only could come up with the brute force approach (in python, on my computer).

And of course equally interested how you constructed this matrix.

EMAIL PM FIND QUOTE REPORT

1st January, 2024, 12:49 **Post: #7**

pier4r
Senior Member

Posts: 2,247
Joined: Nov 2014

RE: Happy New Year 2024 ... and 2023's last teaser !

Happy new year and thank you for the little challenge!

EMAIL PM FIND QUOTE REPORT

1st January, 2024, 14:13 **Post: #8**

3298
Member

Posts: 206
Joined: Oct 2014

RE: Happy New Year 2024 ... and 2023's last teaser !

Solved it without even bothering to fetch and unpack a calc. Hint: Boolean logic and tracing diagonals through the matrix with a finger was enough.

Happy new year anyway!

EMAIL PM FIND QUOTE REPORT

1st January, 2024, 14:27 (This post was last modified: 1st January, 2024 19:46 by brouhaha.) **Post: #9**

brouhaha
Senior Member

Posts: 487
Joined: Dec 2013

RE: Happy New Year 2024 ... and 2023's last teaser !

> **Valentin Albillo Wrote:** (31st December, 2023 18:00)
> The result is sure to ***surprise*** you ! 🙂

The answer didn't surprise me at all. Sorry!
:-)

> **Quote:**
> Admittedly this teaser is only for *power* calc users so don't feel too bad if you're not up to the task (I only needed 3 lines of code for my 4-decades old toy.) 🙂

I'm not nearly clever enough to see how to solve it in three lines of code, but I did solve it using a general purpose matrix-permanent algorithm which I wrote in C++, using integer arithmetic. (It could easily use floating point if necesary, and that might actually be faster.). It computed the result in approximately 4ms on my *power* calc, which has an AMD Ryzen 7 3800X, 3.9+ GHz CPU.

I briefly thought about also writing a 15C program, but the algorithm takes 2^(n-1) * n^2 operations. With the algorithm I used, I think it would be faster to compute it using a 16C.

Thanks for the fun challenge! I look forward to learning about your three-line solution.
Happy new year!

PM FIND QUOTE REPORT

1st January, 2024, 14:43 **Post: #10**

Maximilian Hohmann
Senior Member

Posts: 1,220
Joined: Dec 2013

RE: Happy New Year 2024 ... and 2023's last teaser !

Hello!

> **Valentin Albillo Wrote:** (31st December, 2023 18:00)
> The result is sure to ***surprise*** you ! 🙂

I think I know the answer without solving it :-) Happy **2024** for all of you! And no, please no collectors editions this year, last year was expensive enough...

Best regards
Max

EMAIL PM FIND QUOTE REPORT

1st January, 2024, 15:24 **Post: #11**

brouhaha
Senior Member

Posts: 487
Joined: Dec 2013

RE: Happy New Year 2024 ... and 2023's last teaser !

> **Maximilian Hohmann Wrote:** (1st January, 2024 14:43)
> Happy **2024** for all of you!

Should auld acquaintance be forgot...

PM FIND QUOTE REPORT

1st January, 2024, 19:34 **Post: #12**

Gerson W. Barbosa
Senior Member

Posts: 1,546
Joined: Dec 2013

RE: Happy New Year 2024 ... and 2023's last teaser !

Happy New Year You All!

This should work only for 2x2 and 3x3 matrices, so not quite relevant for that particular problem. Anyway, FWIW,

Code:

```
« DUP →DIAG NEG OVER RANK UNROT
PICK3 DIAG→ + LASTARG NIP + DUP DET
SWAP →DIAG AXL ΠLIST 2 * - -1 ROT 1 + ^ *
»
```

(HP49G, G+ and HP 50g)

EMAIL PM FIND QUOTE REPORT

1st January, 2024, 21:29 (This post was last modified: 1st January, 2024 21:47 by Maximilian Hohmann.) **Post: #13**

Maximilian Hohmann
Senior Member

Posts: 1,220
Joined: Dec 2013

RE: Happy New Year 2024 ... and 2023's last teaser !

Hello!

For me, the complexity of the task to construct a matrix whose permanent has a given value by far exceeds the task of computing this permanent. As I have no idea at all about how to do it - and have never heard about "permanent" with respect to matrices before either - I asked my friend ChatGPT for help:

You: "*Can you please compute a 12x12 matrix whose permanent is **xxxx** using only 0 and 1 as values*"

ChatGTP: "Generating a specific 12x12 matrix with a permanent of **xxxx** using only 0 and 1 values involves finding an arrangement of elements such that their product sums up to **xxxx** when considering all possible permutations. The direct calculation of such a matrix might require a lot of computational effort due to the factorial complexity involved.

Let's try a Python script to generate a 12x12 matrix with 0 and 1 values and check its permanent. Note that this approach might not guarantee a solution due to the complexity and exhaustive search required:

Code:

```
import itertools
import numpy as np

# Function to calculate the permanent of a matrix
def calc_permanent(matrix):
    n = len(matrix)
    perm_sum = 0

    for perm in itertools.permutations(range(n)):
        product = 1
```

This Python script attempts to generate random 12x12 matrices consisting of 0s and 1s and calculates their permanents. It continues the process until it finds a matrix whose permanent matches the specified value (**xxxx** in this case). However, due to the exhaustive search nature of this approach, it might take an extensive amount of time or might not find a suitable matrix within a reasonable timeframe.

If you have specific constraints or additional information that can be used to narrow down the search space, it might help in devising a more targeted approach to finding such a matrix.

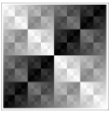
<-- end of ChatGPT answer

This must be by far the worst possible method to compute such a matrix and if this is the state of artificial intelligence on 1. January 2024 we will be safe for at least the remainder of the year from being taken over by machines :-)

Regads
May

EMAIL PM FIND QUOTE REPORT

1st January, 2024, 23:21 **Post: #14**



John Keith
Senior Member

Posts: 989
Joined: Dec 2013

RE: Happy New Year 2024 ... and 2023's last teaser !

Happy new year everyone!

My permanent program fails miserably for this challenge It will not work for any matrix larger than 9 x 9. I am eagerly awaiting Valentin's solution (and others!) to see what algorithm(s) can be used.

EMAIL PM FIND QUOTE REPORT

2nd January, 2024, 10:20 (This post was last modified: 2nd January, 2024 13:02 by EdS2.) **Post: #15**

EdS2
Senior Member

Posts: 559
Joined: Apr 2014

RE: Happy New Year 2024 ... and 2023's last teaser !

By erbeqrevat ebjf naq pbyhzaf jr pna frr this matrix unf n fcrpvny sbez. It wouldn't surprise me (having looked at it just a little) to find gung bar bs gur ebjf (be pbyhzaf) vf n ovanel rapbqvat bs gur qrfverq answer.

Edit: applied a bit of rot13 to avoid very mild spoilers

EMAIL PM FIND QUOTE REPORT

2nd January, 2024, 11:53 (This post was last modified: 4th January, 2024 10:47 by ramon_ea1gth.) **Post: #16**

ramon_ea1gth
Member

Posts: 72
Joined: Mar 2020

RE: Happy New Year 2024 ... and 2023's last teaser !

I'm having fun with the challenge. After programming a cute RPL recursive program in my HP 50g to manage matrices/permanents and realise, as said in other posts, that the required calculations grow insanely, now I'll try the binary way, base-2 approach, with shifting registers, but with the usual base-10 numbers. Let's see

Edit:
I'm testing this binary shifting approach to obtain the cofactors. A 5x5 matrix full of ones takes about 25 seconds to compute in my HP 50g. With zeroes, the code saves operations. Thus, I will let the calculator work overnigth and let's see if I get something in the morning (I started a TICKS command so I can measure the elapsed time). I know, I know: if I were serious I should have tried to make an estimation of the required time for the algorithm

Edit 2:
I got it! (after about 11 hours computing!). At least, after such a long calculation, the result is the expected I'll write my code later here, just as a curiosity, though it's not the most efficient solution.

Edit 3:
And this is my version of PERMANT. It takes two arguments from the stack:

- Level 2: a list with the matrix rows, considered as binary words, translated to base-10 reals.
- Level 1: the number of rows, i.e., 12.

Thus, the contents of the stack must be:

Code:

```
2: {3007, 773, 2839, 516, 3071, 775, 2967, 1533, 2823, 4095, 517, 2999}
1: 12
```

And the code of PERMANT:

Code:

```
« 0 0 0 → vec rows n a per   @ get from stack the list (vec) and nr of rows. Init n, a, per
  « vec 2 / 'vec' STO         @ binary shift right and store (fractional part included)
    rows 1 FOR n              @ go from last row to first
     vec n GET FP 2 * 'a' STO @ matrix element: get the LSB as the fractional part of row n times 2
     IF rows 1 ≠ a AND THEN   @ if rows larger than 1 and matrix element 'a' is nonzero
```

```
      vec OBJ→ 1 + n - ROLL DROP rows 1 - →LIST @ remove row n from vec
      IP rows 1 - PERMANT     @ get the integer part (like remove column) and recall PERMANT
      a * 'per' STO+          @ returned cofactor from PERMANT times 'a', and add to cumulative 'per'
     ELSE a 'per' STO+        @ just add 'a' to cumulative 'per'
```

Edit 4:
And the previous code even more optimised for binary matrices:

Code:

```
« 0 0 → vec rows n per
  « vec 2 / 'vec' STO
    rows 1 FOR n
     IF vec n GET FP THEN
      IF rows 1 ≠ THEN
      vec OBJ→ 1 + n - ROLL DROP rows 1 - →LIST
      IP rows 1 - PERMANT
      ELSE 1
      END
      'per' STO+
```

Edit 5:
With the optimised code, computing time is 9 hours in a physical HP 50g, instead of the previous 11 hours.

EMAIL PM WWW FIND QUOTE REPORT

2nd January, 2024, 13:53 (This post was last modified: 2nd January, 2024 14:05 by J-F Garnier.) **Post: #17**

J-F Garnier
Senior Member

Posts: 891
Joined: Dec 2013

RE: Happy New Year 2024 ... and 2023's last teaser !

> **brouhaha Wrote:** (1st January, 2024 14:27)
> ... I did solve it using a general purpose matrix-permanent algorithm which I wrote in C++, using integer arithmetic. (It could easily use floating point if necesary, and that might actually be faster.). It computed the result in approximately 4ms on my *power* calc, which has an AMD Ryzen 7 3800X, 3.9+ GHz CPU.

I solved it too, using a BASIC code based on the VA711 article with a few obvious optimizations, and some code simplifications.
I got the answer in 2.5 s with Emu71/Win on my Ryzen5 PC , i.e. estimated to about 1h on a real HP-71B so perfectly doable.

BASIC is still relevant in 2024 !

Code:

```
1 ! VA2023
2 ! last va2023 "challenge"
3 ! solution based on VA711 article
5 !
10 OPTION BASE 1
15 DIM A(12,12)
20 DATA 0,0,1,0,0,0,0,0,0,1,0,0
21 DATA 0,0,1,0,0,0,0,0,0,1,0,1
22 DATA 0,0,1,1,0,0,0,0,0,1,0,1
23 DATA 0,0,1,1,0,0,0,0,0,1,1,1
24 DATA 1,0,1,1,0,0,0,0,0,1,1,1
```

EMAIL PM WWW FIND QUOTE REPORT

3rd January, 2024, 01:56 (This post was last modified: 3rd January, 2024 03:23 by Valentin Albillo.) **Post: #18**

Valentin Albillo
Senior Member

Posts: 1,075
Joined: Feb 2015
Warning Level: 0%

RE: Happy New Year 2024 ... and 2023's last teaser !

.
Hi, all,

Thanks for your nice contributions to this ***New Year 2024*** thread, be they *best wishes* and/or *comments* and/or *solutions* on/to my little teaser. Some of my own comments/solution:

> **jthol Wrote:**

Happy new year as well! And **that is a creative start to 2024** :-) I am very interested in your solution, because I only could come up with the brute force approach (in python, on my computer). And of course equally interested how you constructed this matrix.

Thanks for your appreciation, indeed it is ! I'm giving my 3-line solution at the very end of this thread.

3298 Wrote:
Solved it without even bothering to fetch and unpack a calc. Hint: Boolean logic and tracing diagonals through the matrix with a finger was enough. Happy new year **anyway**!

Anyway ? You posted neither *code* nor the *logic* process nor the *value* of the permanent. That's *terseness* indeed. Is this the way you usually use to tackle teasers ?

brouhaha Wrote:
The answer **didn't surprise me at all**. Sorry!

How so ? Didn't you find it surprising to see a matrix whose permanent is such a value ?

brouhaha Wrote:
I did solve it using a general purpose matrix-permanent algorithm which **I wrote in C++**, using integer arithmetic [...] It computed the result in approximately 4ms on my power calc, which has an **AMD Ryzen 7 3800X, 3.9+ GHz CPU**.

Oh my, an *AMD such & such at 3.9+ Ghz in C++*. Exactly what I asked and the tool which I designed my teaser for. It seems that underlining calc wasn't enough of a hint. Sheesh ...

brouhaha Wrote:
Thanks for the fun challenge! I look forward to learning about **your three-line** solution. Happy new year!

Thanks. Find my *3-line code* below but your *C++* code is probably as short as mine.

Maximilian Hohmann Wrote:
I think **I know the answer** without solving it :-)

How perspicacious ! Though I think that there were many likely values, such as ***71***, ***41***, ***15***, etc.

Maximilian Hohmann Wrote:
Happy 2024 for all of you! And no, please **no collectors editions this year**, last year was expensive enough...

+1

Gerson W. Barbosa Wrote:
Happy New Year You All! **This should work only for 2x2 and 3x3** matrices, so not quite relevant for that particular problem [...]

Thanks for your wishes and for the code ... though I don't understand most of it, at all, but that's my bad, not yours ... (I think). Nice to see that *RPL*'s legendary unfathomability remains as strong in *2024* as ever !

Maximilian Hohmann Wrote:
For me, the complexity of the task **to construct a matrix whose permanent has a given value** by far exceeds the task of computing this permanent.

That's precisely the intended *"surprise"*, I see you fully got it !

Maximilian Hohmann Wrote:
As I have no idea at all about how to do it - and have **never heard about "permanent"** with respect to matrices before either - I asked my friend **ChatGPT** for help: [...]

I'm very glad that my humble teaser was useful for you to learn about new mathematical concepts. As for your *friend*, the *glorified* ***ELIZA***, the least I say, the better. I'm surprised that it didn't answer something like *"The permanent's value is -37.489002"* or include *"import VA_NYteaser as VA"* in the code or some other brain-dead hallucinations. ´

Note to all: please **don't discuss *ChatGPT* in this thread**, if you will, just create another. Thank you.

> **Maximilian Hohmann Wrote:**
> This must be by far **the worst possible method** to compute such a matrix and if this is the state of artificial intelligence on 1. January 2024 we will be safe for at least the remainder of the year from being taken over by machines :-)

Indeed I coincide, I concur and last but not least, I fully agree.

> **John Keith Wrote:**
> I am **eagerly awaiting Valentin's solution** (and others!) to see what algorithm(s) can be used.

Thanks for trying, **John**, I hope you enjoyed it, see my solution below.

> **ramon_ea1gth Wrote:**
> I'm **having fun** with the challenge.

¡ Hola, paisano, ***Feliz Año Nuevo 2024*** !

Thanks for your valuable attempts to solve this teaser, much appreciated.

> **J-F Garnier Wrote:**
> I solved it too, **using a BASIC code based on the VA711 article** with a few obvious optimizations, and some code simplifications.

Long time no see, **J-F** ! And you have the nerve to use my own old program to solve my own teaser ! (*VA711* isn't an *article*, by the way ...)

In this particular case, the optimizations you applied work very well, but that's only because there are so many **0's** and the non-zero values are all **1's** which makes non-zero products evaluate to **1**, without performing any multiplications. For the *general* case of a *12x12* matrix with random elements, your program wouldn't work in feasible time, if at all, at best it would take just too long and the recursion would need a lot of memory to run.

But in this particular case your optimizations really do cope with the task as stated, so ***congratulations*** are in order, and thanks for your appreciation and your effort in creating a working solution.

> **J-F Garnier Wrote:**
> I got the answer in 2.5 s with Emu71/Win on my Ryzen5 PC , i.e. estimated to about **1h on a real HP-71B so perfectly doable**. **BASIC is still relevant in 2024 !**

Yes, **it's perfectly doable**, and yes, *HP-71B BASIC* still rules !

Now, my 3-line solution follows but first a little preamble:

Despite the extreme similarities in the respective definitions of ***Determinant*** and ***Permanent*** (just considering the *sign* of the permutations or not), their efficient computation and uses are totally different.

Determinants can be efficiently computed in polynomial time while there's no known procedure to compute *Permanents*, the naive algorithm (as used in my original program and in **J-F**'s optimization of it) runs in *superexponential* time and the very best still require *exponential* time, so even for matrices of even very moderate sizes it's really not feasible to compute the exact value of their permanent, though approximate and/or probabilistic methods are possible.

Enough of a preamble. I'll soon create and post an *Article* or *SRC#14* with all details, algorithms, practical uses, many worked examples and code for both the **HP-71B** and the **HP-15C** & clones. Meanwhile, this terse (as promised) exposition should be adequate:

My Solution

Although computing the permanent requires exponential running time there are exponents and there are exponents. Both *O(2^n)* and *O(10^n)* running times are exponential but the former is surely much faster than the latter. My subprogram below uses a *non-optimized* (this is a *teaser*, people, not a *formal* research paper !) implementation of *Ryser's formula*.

Program listing

This ***3****-line (156-byte)* subprogram for the **HP-71B** will compute the permanent of a real square *NxN* matrix passed by reference and return its value in a real variable, also passed by reference:

```
100 SUB PER(A(,),T) @ N=UBND(A,1) @ T=0 @ FOR K=1 TO 2^N-1 @ P=1 @ FOR I=1 TO N
110 S=0 @ B=1 @ FOR J=1 TO N @ IF BIT(K,J-1) THEN S=S+A(I,J) @ B=-B
120 NEXT J @ P=P*S @ NEXT I @ T=T+P*B @ NEXT K @ IF BIT(N,0) THEN T=-T
```

Let's use it to compute my teaser's *permanent*, directly from the command line:

```
>DESTROY ALL @ OPTION BASE 1 @ DIM A(12,12),P @ MAT INPUT A
```

(enter the matrix elements row by row, then)

```
>CALL PER(A,P) @ DISP P
```

2023 *(Emu71/Win: 22 sec., go71b: 2 min. 40 sec., Physical 71B: ~ 6 h)*

Comparing the timings, **J-F**'s optimization works faster than my subprogram but then it only works fine for matrices such as the one here, it wouldn't work for *arbitrary* matrices or it woul run much, much slower, while my subprogram is *general* and will work for arbitrary real square matrices, taking about the same time for any matrix.

V.

PM WWW FIND EDIT QUOTE REPORT

3rd January, 2024, 08:14 **Post: #19**

jthole
Member

Posts: 80
Joined: Nov 2017

RE: Happy New Year 2024 ... and 2023's last teaser !

Wait wait … so my algorithm gave the proper solution after all! (2023).

I assumed that I had overlooked a tiny detail, and it should have been 2024.

Thanks for posting your program; I don't have a 71B (with or without math module), but I will certainly study your short algorithm.

I am looking forward to your article :-)

EMAIL PM FIND QUOTE REPORT

3rd January, 2024, 11:59 **Post: #20**

Maximilian Hohmann
Senior Member

Posts: 1,220
Joined: Dec 2013

RE: Happy New Year 2024 ... and 2023's last teaser !

Hello!

> **Valentin Albillo Wrote:** (3rd January, 2024 01:56)
>
> .... implementation of Ryser's formula. ...

I have come across that one when I looked for "permanent" and even tried it out with python (which is really not my favorite programming language...) on my Mac. For me is interesting that something seemingly trivial as the permanent has not yet inspired some mathematical genius to come up with a computer-friendly method of calculation. Perhaps because there is so little practical use for the permanent?

Anyway, I still wonder how you constructed that matrix!

Regards
Max

EMAIL PM FIND QUOTE REPORT

3rd January, 2024, 13:08 **Post: #21**

3298
Member

Posts: 206
Joined: Oct 2014

RE: Happy New Year 2024 ... and 2023's last teaser !

> **Valentin Albillo Wrote:** (3rd January, 2024 01:56)
>
> > **3298 Wrote:**
> >
> > **Solved it without even bothering to fetch and unpack a calc**. Hint: Boolean logic and tracing diagonals through the matrix with a finger was enough. Happy new year **anyway**!
>
> Anyway ? You posted neither *code* nor the *logic* process nor the *value* of the permanent. That's *terseness* indeed. Is this the way you usually use to tackle teasers ? 🙂

It is when I'm afraid of spoiling the challenge for others. For instance, in the (online version of the) HHC contests I generally submit program size and checksum only (and maybe a cryptic hint), withholding the code and proper explanation until after

the deadline is over.
However, this time I misunderstood the challenge, so i "solved" something that was **far easier** than the actual challenge, enabling me to do so without a calculator. In my defense, I never needed to calculate the determinant of a matrix larger than 3x3 by hand, so I failed to realize that the rule of thumb I remembered from my school days will not work beyond this size. Based on that I ended up with a 0 result, which surprised me only in the way that your previous challenges usually resulted in the number naming the old or the new year. It seemed like the obvious answer otherwise ... again, due to my flawed understanding.

It kind of felt too easy for one of your challenges, almost boring actually. (That's what prompted the "anyway" part of my post...) It should have made me realize that I got something wrong. I feel stupid now.

On a slightly more serious note, the RPL code looks clearer to me than your Basic code. The names of the RPL commands called throughout are a better documentation than trying to trace what each Basic variable is subjected to. But I'm RPL-trained, which allows me to see stackrobatics as necessary but uninteresting fluff, much like variable assignments in a C-derived language. Readability is subjective.

EMAIL PM FIND QUOTE REPORT

3rd January, 2024, 13:17 **Post: #22**

Gerson W. Barbosa
Senior Member

Posts: 1,546
Joined: Dec 2013

RE: Happy New Year 2024 ... and 2023's last teaser !

> **Valentin Albillo Wrote:** (3rd January, 2024 01:56)
>
> > **Gerson W. Barbosa Wrote:**
> > Happy New Year You All! **This should work only for 2x2 and 3x3** matrices, so not quite relevant for that particular problem [...]
>
> Thanks for your wishes and for the code ... though I don't understand most of it, at all, but that's my bad, not yours ... (I think). Nice to see that *RPL*'s legendary unfathomability remains as strong in *2024* as ever !

Hi, Valentín,

There’s a mistake in my program. I’ve used RANK when all I need with the order of the matrix. As a result it was taking longer that it needed to for a 3x3 matrix (0.81 seconds instead of 0.23 seconds). This is better:

Code:

```
« DUP →DIAG NEG DUP AXL ΠLIST
UNROT DUP SIZE HEAD UNROT
PICK3 DIAG→ + LASTARG NIP +
DET ROT DUP + - -1 ROT OVER + ^ *
»
```

Not any more readable than the previous version though.
It’s based on a little formula I came up with.
Thanks for this teaser, even if I wasn’t up to the task. Until then I didn’t know there was something like the permanent of a matrix.

Best regards,

Gerson.

———————

Per(M) = Det(M’) - 2p

where

M’ is M with elements of the main diagonal multiplied by -1

and

p is the product of the elements of the main diagonal of M’.

This works for order 3 matrices. For order 2 matrices the sign of the result must be changed.

For example,

M:
[[-1 3 2]
[7 -4 5]
[6 8 9]]

M’:
[[1 3 2]
[7 4 5]
[6 8 -9]]

Det(M’) = 267

p = 1×4×(-9) = -36

Per(M) = 267 - 2×(-36) = 339

EMAIL PM FIND QUOTE REPORT

3rd January, 2024, 13:51 (This post was last modified: 3rd January, 2024 14:23 by Ajaja.) **Post: #23**

Ajaja
Junior Member

Posts: 33
Joined: Feb 2021

RE: Happy New Year 2024 ... and 2023's last teaser !

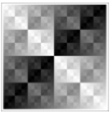
There are many rows and columns with zeros. I think, the fastest way to calculate permanent of the matrix is to sort it first, then transpose, sort again and after that use J-F Garnier optimized subprogram. It turns this 12x12 permanent into sum of 8 9x9 permanents if I'm not mistaken.
And result should be the same:

Code:

```
    |\^/|     Maple 7 (IBM INTEL NT)
._|\|   |/|_. Copyright (c) 2001 by Waterloo Maple Inc.
 \  MAPLE  /  All rights reserved. Maple is a registered trademark of
 <____ ____>  Waterloo Maple Inc.
      |       Type ? for help.
> with(LinearAlgebra):
> Permanent(<
> <0,0,0,0,0,1,1,1,1,1,1,1>|
> <0,0,0,0,1,0,0,0,0,0,0,1>|
> <0,0,0,0,1,0,0,0,0,0,1,1>|
```

PM FIND QUOTE REPORT

3rd January, 2024, 15:59 (This post was last modified: 3rd January, 2024 16:00 by John Keith.) **Post: #24**



John Keith
Senior Member

Posts: 989
Joined: Dec 2013

RE: Happy New Year 2024 ... and 2023's last teaser !

> **Gerson W. Barbosa Wrote:** (3rd January, 2024 13:17)
>
> Per(M) = Det(M’) - 2p
>
> where
>
> M’ is M with elements of the main diagonal multiplied by -1
>
> and
>
> p is the product of the elements of the main diagonal of M’.
>
> This works for order 3 matrices. For order 2 matrices the sign of the result must be changed.
>
> For example,
>
> M:
> [[-1 3 2]
> [7 -4 5]
> [6 8 9]]
>
> M’:
> [[1 3 2]
> [7 4 5]

[6 8 -9]]

Det(M’) = 267

p = 1×4×(-9) = -36

Per(M) = 267 - 2×(-36) = 339

That is quite interesting. In my (obsolete) program, I compute the 3 x 3 permanent with inline code as follows, requiring 9 multiplications and 5 additions.

Code:

```
        OBJ\-> DROP                             @ Explode matrix onto stack
        5. PICK OVER * PICK3 6. PICK * +
        9. ROLLD 6. PICK * PICK3 5. ROLL * +
        6. ROLLD 4. ROLL * UNROT * + *
        UNROT * + UNROT * +                     @ 3 * 3 permanent
```

This may be faster than using **DET**.

EMAIL PM FIND QUOTE REPORT

3rd January, 2024, 18:02 **Post: #25**

Gerson W. Barbosa
Senior Member

Posts: 1,546
Joined: Dec 2013

RE: Happy New Year 2024 ... and 2023's last teaser !

John Keith Wrote: (3rd January, 2024 15:59)

Gerson W. Barbosa Wrote: (3rd January, 2024 13:17)

Per(M) = Det(M’) - 2p

where

M’ is M with elements of the main diagonal multiplied by -1

and

p is the product of the elements of the main diagonal of M’.

This works for order 3 matrices. For order 2 matrices the sign of the result must be changed.

For example,

M:
[[-1 3 2]
[7 -4 5]
[6 8 9]]

M’:
[[1 3 2]
[7 4 5]
[6 8 -9]]

Det(M’) = 267

p = 1×4×(-9) = -36

Per(M) = 267 - 2×(-36) = 339

That is quite interesting. In my (obsolete) program, I compute the 3 x 3 permanent with inline code as follows, requiring 9 multiplications and 5 additions.

Code:

```
        OBJ\-> DROP                             @ Explode matrix onto stack
        5. PICK OVER * PICK3 6. PICK * +
```

```
        9. ROLLD 6. PICK * PICK3 5. ROLL * +
        6. ROLLD 4. ROLL * UNROT * + *
        UNROT * + UNROT * +                  @ 3 * 3 permanent
```

This may be faster than using **DET**.

You're definitely right! Actually about 4.5 times as fast. Size is the only advantage here (67 bytes versus 100 bytes).

Code:

```
« DUP →DIAG NEG DUP
AXL ΠLIST UNROT 3.
DIAG→ + LASTARG NIP +
DET SWAP DUP + -
»
```

EMAIL PM FIND QUOTE REPORT

3rd January, 2024, 19:35 **Post: #26**

ramon_ea1gth
Member

Posts: 72
Joined: Mar 2020

RE: Happy New Year 2024 ... and 2023's last teaser !

> **Valentin Albillo Wrote:** (3rd January, 2024 01:56)
> ¡ Hola, paisano, ***Feliz Año Nuevo 2024*** !
> Thanks for your valuable attempts to solve this teaser, much appreciated.

¡Hola Valentín!
I have to refrain tackling your challenges. They are addictive! Even after finishing I was back revising my, anyway, suboptimal code, realising that I can save more operations with binary matrix values.
It was a good excuse to think and read further from your hints: now I also find challenging how to prepare the matrix with the predefined result.
¡Feliz Año Nuevo también!

EMAIL PM WWW FIND QUOTE REPORT

4th January, 2024, 14:01 (This post was last modified: 4th January, 2024 14:24 by J-F Garnier.) **Post: #27**

J-F Garnier
Senior Member

Posts: 891
Joined: Dec 2013

RE: Happy New Year 2024 ... and 2023's last teaser !

> **Ajaja Wrote:** (3rd January, 2024 13:51)
> There are many rows and columns with zeros. I think, the fastest way to calculate permanent of the matrix is to sort it first, then transpose, sort again and after that use J-F Garnier optimized subprogram.

If you look closely at my solution, you will see that I sorted the array rows by the number of 1 in each row, not by the value of the binary representation:

```
20 DATA 0,0,1,0,0,0,0,0,0,1,0,0
21 DATA 0,0,1,0,0,0,0,0,0,1,0,1
22 DATA 0,0,1,1,0,0,0,0,0,1,0,1
23 DATA 0,0,1,1,0,0,0,0,0,1,1,1
24 DATA 1,0,1,1,0,0,0,0,0,1,1,1
25 DATA 1,0,1,1,0,0,0,1,0,1,1,1
26 DATA 1,0,1,1,1,0,0,1,0,1,1,1
27 DATA 0,1,0,1,1,1,1,1,1,1,0,1
28 DATA 1,0,1,1,1,0,1,1,0,1,1,1
29 DATA 1,0,1,1,1,0,1,1,1,1,1,1
30 DATA 1,0,1,1,1,1,1,1,1,1,1,1
31 DATA 1,1,1,1,1,1,1,1,1,1,1,1
```

This was the most important part of my 'optimization'. The other one was to consequently skip the minor calculation for the zero elements:

```
135 IF A(1,K)=0 THEN 145 ! little optimization
```

In that way, the recursive tree was reduced to 2 paths at each level instead of 11 (up to row 8), mostly important for the first recursive steps to cut calculation time.

I did notice the structure of the sorted array and the exception of row 8 (of the sorted array), but without paying too much attention to it.
I was quite happy to get a solution in HP BASIC in a sensible time, even on a machine as slow as the 71B.

Of course, as Valentin pointed out, this solution is only effective for this case with this special array structure.

J-F

EMAIL PM WWW FIND QUOTE REPORT

5th January, 2024, 11:11 (This post was last modified: 5th January, 2024 12:47 by Gjermund Skailand.) **Post: #28**

Gjermund Skailand
Member

Posts: 73
Joined: Dec 2013

RE: Happy New Year 2024 ... and 2023's last teaser !

Update/Small speed improvement
Here is a sys-RPL implementation of Valentin's code for a general matrix computation of permanent :
Compiles to 255.5 bytes, chksum #917Dh
On my HP50g it solves the 12x12 matrix in slightly less than 13,5 minutes.

calculations can be aborted by pushing ON/cancel

Code:

```
::
  CH1NOLASTWD
  CK&DISPATCH2
  #4
  ::
    FPTR2 ^CKNUMARRY                          ( Ensure numeric array )
    DUP FPTR2 ^MDIMS ?SKIP #0           ( 2 dimensions square matrix )
    OVER #<>case FPTR2 ^ErrBadDim      ( give error message if not )
    #0 #0 %0 %0 %0 %0
    {{ P B S T IN K N }}                        ( declare local variables )
```

Updated formatting
best regards
Gjermund

EMAIL PM FIND QUOTE REPORT

5th January, 2024, 11:12 **Post: #29**

Gjermund Skailand
Member

Posts: 73
Joined: Dec 2013

RE: Happy New Year 2024 ... and 2023's last teaser !

How do I avoid that space indenting is not stripped away??)

EMAIL PM FIND QUOTE REPORT

5th January, 2024, 11:48 (This post was last modified: 5th January, 2024 12:53 by Didier Lachieze.) **Post: #30**

Didier Lachieze
Senior Member

Posts: 1,614
Joined: Dec 2013

RE: Happy New Year 2024 ... and 2023's last teaser !

> **Gjermund Skailand Wrote:** (5th January, 2024 11:12)
> How do I avoid that space indenting is not stripped away??)

By using the code tags (the **#** in the message editor toolbar).

EMAIL PM FIND QUOTE REPORT

5th January, 2024, 11:56 (This post was last modified: 6th January, 2024 15:56 by Gjermund Skailand.) **Post: #31**

Gjermund Skailand
Member

Posts: 73
Joined: Dec 2013

RE: Happy New Year 2024 ... and 2023's last teaser !

Thanks, Didier

I was curious to test the real speed of hp50g, running hpgcc3 code at 120MHz, so I ported to a hpgcc3 program, 1084bytes, program and code, enclosed.
This program calculates the 12x12 matrix in less than 1 sec.
It calculates permanents up size {15,15} in about 4 sec.

It requires a modified rom to run this, but if anyone is interested, let me know.
PS anyone got timings on other calculators?

best regards
Gjermund

Attached File(s)

permnt.zip (Size: 45.21 KB / Downloads: 2)

User(s) browsing this thread: Valentin Albillo*