[VA] SRC #013 - Pi Day 2023 Special

14th March, 2023, 20:22 (This post was last modified: 18th March, 2023 04:16 by Valentin Albillo.) **Post: #1**

Valentin Albillo
Senior Member

Posts: 958
Joined: Feb 2015
Warning Level: 0%

[VA] SRC #013 - Pi Day 2023 Special

Hi, all,

Just in case you hadn't noticed, today it's ***March, 14*** *aka* π ***Day***, so

Happy Pi Day 2023 and Welcome to my *SRC #13 - Pi Day 2023 Special*

(We just enjoyed this nice *Pi Day's Oreo cake* for lunch, courtesy of my daughter Laura's *haute cuisine* abilities)

This ***SRC #13*** is intended to commemorate once more this most ubiquitous constant, π. There are many other threads about *Pi Day 2023* but this one is mine. After posting a number of threads over the years about π *Day*, such as these,

SRC #010 - Pi Day 2022 Special
SRC #009 - Pi Day 2021 Special
SRC #006 - Pi Day 2020 Special: A New Fast Way to Compute Pi

it would seem problematic to find *new*, intriguing appearances of the little critter but far from it, π is well-nigh *inexhaustible* and to prove the point let me introduce a couple' additional appearances for your enjoyment, which will appear *one after another* so that you can focus on just one at a time. Let's begin with ***#1*** ...

> ***Note****: No hard rules so no need for a parallel thread, post here whatever you want as long as it's on topic and* ***NO CODE PANELS****, but I'd appreciate it if you'd use vintage HP calcs (physical/virtual), otherwise I'll consider you to have failed the challenge whatever your results/timings.*

1. Let's count ...

π's value can be obtained by evaluating a plethora of transcendental functions, infinite summations and products, definite integrals, stochastic processes, etc., but if you don't remember any of them you can still get a nice approximation to the value of π *(exact as **N** goes to infinity)* by following these simple steps:

> **1.** Choose a positive integer ***N***
>
> **2.** Tally up how many integers in the range ***1***...***N*** have no repeated prime factors
>
> **3.** Output $\sqrt{\dfrac{6N}{Count}}$

For example, for ***N*** = ***10*** we find that the seven integers ***1***, ***2***, ***3***, ***5***, ***6***, ***7*** and ***10*** have no repeated prime factors, so ***Count*** = ***7*** and you get ~ ***2.9277*** as an approximation to π *(err ~ 6.8%)*.

Now write your very own program and try ***N*** = ***12,345***, ***100,000***, ***567,890*** and ***1,000,000*** to see if you get the following results, which I obtained using this little witty ***4****-line (217-byte)* **HP-71B** program I wrote for the occasion *(uses **Math** and **JPC** ROMs; 179 bytes without **USING** "image")*:

```
DESTROY ALL @ INPUT T @ SETTIME 0 @ ...
...
...
DISP USING "2(3DC3DC3DC3D,2X),2(Z.8D,X),5DZ.2D";T,S,SQR(6*T/S),ABS(PI-RES),TIME

 >RUN ->  ? 12345  -> ... , etc.

     N       Count     π approx     |Error|      go71b   Emu71/Win  Physical
                                                 @128x     @976x     HP-71B
-----------------------------------------------------------------------
    12,345    7,503  3.14198205  0.00038939    0.10"    0.01"        13"
   100,000   60,794  3.14155933  0.00003333    0.28"    0.04"        36"
   567,890  345,237  3.14158684  0.00000582    0.69"    0.09"    1' 28"
 1,000,000  607,926  3.14159550  0.00000285    0.93"    0.12"    1' 59"
```

However, as the procedure is so simple, the difficulty here lies not so much in the programming as in the *efficiency*, thus the challenge consists mainly in achieving correct results in times ***as good*** or ***better*** than the ones given above, using exclusively *vintage HP calcs*, physical or virtual *(indicate emulation's speed and timings for the virtual/physical calcs and try to avoid prematurely spoiling it all for other people.)*

Well, see if you can deliver and, if feeling venturous and your calc is up to it, post also the results and timings for ***N*** = ***10*** *million*, ***25*** *million* and ***33*** *million* (which gives an approximation to π correct to ***8*** *digits*.)

If I see interest, I'll post my original solution & comments in a few days and part ***#2*** next ***April, 1st***.

That's all. Any and all *constructive* and *on-topic* comments will be most welcome and appreciated.

V.
Edit: some errors corrected.

PM WWW FIND EDIT QUOTE REPORT

16th March, 2023, 12:16 (This post was last modified: 20th March, 2023 07:19 by 2old2randr.) **Post: #2**

2old2randr
Junior Member

Posts: 42
Joined: Jan 2018

RE: [VA] SRC #013 - Pi Day 2023 Special

Hi Valentin,

Edit: I have corrected the formatting (thanks, Valentin for the explanation) and added a column for run times using the (much faster) Sys RPL version of the Möbius function written by Gerald H. This is more than twice as fast as the HP-71B times you have listed.

I attempted this because it is simpler than your usual problems and you were complaining about the lack of RPL solutions for your last challenge Using a brute force solution on a physical HP 50g, I get decent run times (13' 20'' for 33 million [**5' 21" for the Sys RPL version**]) as shown below. Although I am not sure if the 50g qualifies as a vintage calculator, of course.

```
                                         Runtime  (seconds)
       Number        Count   Approximation  User RPL   Sys RPL
-------------  -----------  -------------  --------  -------
```

10	7	2.92770021885	0.44	0.29
12,345	7,503	3.14198204634	15.52	6.02
100,000	60,794	3.14155932716	45.08	16.93
567,890	345,237	3.14158683822	110.40	40.39
1,000,000	607,926	3.14159550063	147.82	54.99
10,000,000	6,079,291	3.14158749068	469.88	139.11
25,000,000	15,198,180	3.14159239999	692.35	277.43
33,000,000	20,061,593	3.14159276017	800.43	321.11
100,000,000	60,792,694	3.14159307180	n/a	448.08
1,000,000,000	607,927,124	3.14159259637	n/a	1453.24

I used the equation S(n) = Sum(i from 1 to sqrt(n); mu(i) * int(n / i * i)) to get the count of square free numbers less than or equal to 'n'. In the equation, mu is the Möbius function.

The code (VA is the problem solution, MOB is the Möbius function):
```
VA
« → n
  « 0. 1. n √ IP
    FOR i
      i MOB n i SQ / IP * +
    NEXT
    DUP 6. n * SWAP / √
  »
»

MOB (User RPL version)
« R→I
  IF DUP 1 > THEN
    FACTOR
    IF DUP TYPE 9. SAME THEN
      DUP →STR
      IF "^" POS THEN
        DROP 0
      ELSE
        SIZE 1. + 2. / 1 SWAP 2. MOD { NEG } IFT
      END
    ELSE
      DROP -1
    END
  END
»

MOB (System RPL version)
::
  CK1&Dispatch
  # FF
  ::
    {
      ROTDROPSWAP
      %1
      EQUALcase
      FPTR2 ^RNEGext
      FPTR2 ^DROPZ0
    }
    1LAMBIND
    ::
      FPTR2 ^ZAbs
      FPTR2 ^DupQIsZero?
      caseSIZEERR
      FPTR2 ^DupZIsOne?
      ?SEMI
      FPTR2 ^MZSQFF
      #2/
      ZINT 1
      SWAP
      ZERO_DO
      1GETLAM
      COMPEVAL
      LOOP
    ;
    ABND
```

```
   ;
 ;
```

Sudhir

EMAIL PM FIND QUOTE REPORT

16th March, 2023, 21:23 **Post: #3**

John Keith
Senior Member

Posts: 883
Joined: Dec 2013

RE: [VA] SRC #013 - Pi Day 2023 Special

In case you are not aware of it, our fellow member Gerald H wrote a very fast version of MOB which is in this thread.

I'm not sure whether external programs are allowed in these challenges but I do recall several HP-71 programs using LEX files, so I would think that they are fair game for other HP's as well.

EMAIL PM FIND QUOTE REPORT

16th March, 2023, 21:49 **Post: #4**

Valentin Albillo
Senior Member

Posts: 958
Joined: Feb 2015
Warning Level: 0%

RE: [VA] SRC #013 - Pi Day 2023 Special

.
Hi, **John**,

> **John Keith Wrote:** (16th March, 2023 21:23)
> I'm not sure whether external programs are allowed in these challenges but I do recall several HP-71 programs using LEX files, so **I would think that they are fair game** for other HP's as well.

Absolutely. There are no rules other than no code panels, and using *vintage HP models* (such as the *HP 50g*) is highly encouraged (but not mandatory,) after all this is the ***Museum of HP Calculators*** ... 🙂

In short, external programs/libraries/LEX files/binaries/etc. are allowed and welcome.

Thanks for your interest and Regards.
V.

PM WWW FIND EDIT ✗ QUOTE REPORT

17th March, 2023, 15:08 **Post: #5**

vaklaff
Member

Posts: 110
Joined: Dec 2019

RE: [VA] SRC #013 - Pi Day 2023 Special

I'm not really contributing, I just want to say a public thank-you to Valentin from me and my wife. This topic was completely new to us. She (high school teacher of mathematics and programming) and her students had great fun playing with the approximation!

EMAIL PM FIND QUOTE REPORT

18th March, 2023, 12:21 **Post: #6**

J-F Garnier
Senior Member

Posts: 790
Joined: Dec 2013

RE: [VA] SRC #013 - Pi Day 2023 Special

My initial attempt was, instead of counting the square-free numbers, to count the numbers that have a square prime factor, and to subtract this amount from N.
The count of numbers <=N having the 2^2 factor is IP(N/2^2), those having the factor 3^2 is IP(N/3^2), and so on, so the count of square-free numbers should be S = N - IP(N/2^2) - IP(N/3^2) - IP(N/5^2) ...
In that way, I only had to explore primes up to SQR(N), so it was pretty fast.
Unfortunately, numbers that are multiple of several square primes such as (2*3)^2 are counted several times, and the result is underestimated:

```
20 N=12345 ! test case
```

```
30 S=N @ P=1
40 P=FPRIM(P+1) @ S=S-IP(N/P^2) @ IF P<SQR(N) THEN 40
50 DISP N;S

>RUN
12345 6793 (true result=7503)
```

I didn't find a way to easily manage the numbers with multiple square prime factors.
An improvement was to use the current sum S instead of N: S = N ; S=S-IP(S/2^2); S=S-IP(S/3^2) ... but it's still an approximation at most:

```
40 P=FPRIM(P+1) @ S=S-IP(S/P^2) @ IF P<SQR(N) THEN 40

>RUN
12345 7529 (true result=7503)
```

So, I resorted to the formula using the Möbius function as disclosed in the post #2 above.
Matter of fact, this formula starts with the same terms as my first attempt:
S = N - IP(N/2^2) - IP(N/3^2) - IP(N/5^2) ...
but "magically" manages the numbers with multiple square prime factors with terms such as +IP(N/(2*3)^2) !

Here is my implementation, (correct) results and timings on my Emu71/DOSBox, at about 150x speed:

```
10 ! SRC13
20 INPUT N
25 T=TIME
30 S=N @ FOR I=2 TO SQR(N) @ S=S+FNM(I)*IP(N/I^2) @ NEXT I
40 T=TIME-T
50 DISP N;S;T
80 !
90 DEF FNM(N) ! Moebius function
110 C=-1 @ Q=1
120 P=PRIM(N) @ IF P=0 THEN P=N
130 IF P=Q THEN FNM=0 @ END
140 IF P<N THEN C=-C @ N=N/P @ Q=P @ GOTO 120
150 FNM=C @ END DEF

>RUN
12345 7503 .23
10000 60794 .65
567890 345237 1.37
1000000 607926 1.8
```

However, I'm not fully satisfied by applying a formula without understanding how and why it works.

Now, I'm curious to read Valentin's solution and explanations !

J-F

EMAIL PM WWW FIND QUOTE REPORT

18th March, 2023, 23:48 (This post was last modified: 18th March, 2023 23:52 by 2old2randr.) **Post: #7**

2old2randr
Junior Member

Posts: 42
Joined: Jan 2018

RE: [VA] SRC #013 - Pi Day 2023 Special

> **J-F Garnier Wrote:** (18th March, 2023 12:21)
> I didn't find a way to easily manage the numbers with multiple square prime factors.

I was trying the same approach using the inclusion-exclusion principle, i.e., that the true count would be given by:

```
count = n/2^2 + n/3^2 + n/5^2 + ...
- n/(2^2*3^2) - n/(2^2*5^2) - n(3^2*5^2) - ...
+ n/(2^2*3^2*5^2) + ...
- ...
```
or (rearranging)
```
count = n/2^2 - n/(2^2*3^2) - n(2^2*5^2) - ... + n/(2^2*3^2*5^2) + ... - n/(2^2*3^2*5^2*7^2) - ...
```

This involves generating all combinations of the squares of the primes but what makes it feasible is that the terms rapidly go to zero since the product of squares grows so rapidly and the search tree can be pruned whenever the product of the squares is greater than the input number. In fact, I do have an implementation that computes the count in a couple of seconds for 1e6 (if given a list of primes up to 1000) on a physical HP 50g. Unfortunately, there is bug in my backtracking process after pruning that I have not been able to resolve and the program does not generate the correct count after 1763 (1764=2^2*3^2*7^2 is the first number where the backtracking is needed).

Sudhir

19th March, 2023, 01:37 **Post: #8**

Valentin Albillo
Senior Member

Posts: 958
Joined: Feb 2015
Warning Level: 0%

RE: [VA] SRC #013 - Pi Day 2023 Special

.
Hi, **2old2randr**, **vaklaff** and **J-F Garnier**,

> **2old2randr Wrote:** (16th March, 2023 12:16)
>
> [Sorry the tables and code have messed up formatting - **I couldn't figure out how to get that right without using code blocks**.

Here's how: specifying **font 'Courier'** and replacing spaces by the non-breaking character " ", like this:

```
     Number      Count    Approximation Runtime (Seconds)
-------------------------------------------------------
         10          7   2.92770021885    0.44
     12,345      7,503   3.14198204634   15.52
    100,000     60,794   3.14155932716   45.08
    567,890    345,237   3.14158683822  110.40
  1,000,000    607,926   3.14159550063  147.82
 10,000,000  6,079,291   3.14158749068  469.88
 25,000,000 15,198,180   3.14159239999  692.35
 33,000,000 20,061,593   3.14159276017  800.43
```

> **2old2randr Wrote:**
>
> I attempted this because **it is simpler than your usual problems** and you were complaining about **the lack of RPL solutions** for your last challenge 🙂

Certainly. Glad to see some brave *RPL*-user actually using his vintage *RPL* calc to solve my challenge. Much appreciated.

> **2old2randr Wrote:**
>
> Using a brute force solution on a physical HP 50g, I get decent run times (max. of 13' 20'') as shown below. Although **I am not sure if the 50g qualifies as a vintage calculator,** of course.

As for your second statement, **yes**, the **HP 50g** fully qualifies as a *vintage HP calculator*, thus no problem. And *BTW*, all your results are correct.

> **vaklaff Wrote:**
>
> **I'm not really contributing**, I just want to say a public thank-you to Valentin from me and my wife. **This topic was completely new to us**. She (high school teacher of mathematics and programming) and her students had great fun playing with the approximation!

Of course you're contributing ! In particular, to boost my morale, which helps me keep on creating and posting these challenges. Much appreciated, and glad to know that you and your wife *(and her students !)* enjoyed the topic and learned something new. Please give my best regards to your charming wife.

> **J-F Garnier Wrote:**
>
> Here is my implementation, (correct) results and timings on my Emu71/DOSBox, at about 150x speed:[...]

Thanks for your continued interest and results, **J-F**. Please post your best estimates for the runtimes when using a physical **HP-71B**, for comparison purposes. Also, try and include the results/timings for ***N*** = ***10****,* ***25*** *and* ***33*** *million*, if you can.

> **J-F Garnier Wrote:**
>
> However, **I'm not fully satisfied by applying a formula without understanding how and why it works**. Now, I'm curious to read Valentin's solution and explanations !

However, in this post of yours in my recent ***SRC #012e - Then and Now: Roots*** thread, you *did* use *Bornemann*'s formula, about which you posted, I quote:

> *"**I didn't fully understand the underlying math, but was able to decipher the formula and translate it into 71B code**"*.

Same here, right ? 🙂

As for reading my *solution and explanations*, I'll provide both but you know well that I don't usually give formal proofs, references or lengthy math lectures, so keep your expectations accordingly, Ok ?

I'll post my solution (actually *two* of them, optimized for different purposes) either next *Sunday* or *Monday*, around *10 PM GMT+1*. Thanks to all of you for your interest and contributions.

Best regards.
V.

19th March, 2023, 06:31 (This post was last modified: 19th March, 2023 08:45 by 2old2randr.) **Post: #9**

2old2randr
Junior Member

Posts: 42
Joined: Jan 2018

RE: [VA] SRC #013 - Pi Day 2023 Special

I could only find a recursive implementation for the inclusion/exclusion method but this turns out to be much slower than the earlier brute force solution (even given the list of primes a priori). Just for giggles, here are the run times for the first two numbers using this approach - I did not bother running for the others.

```
     Number      Count    Approximation Runtime (Seconds)
  ---------------------------------------------------------
      12,345       7,503  3.14198204634   140.36
     100,000      60,794  3.14155932716   902.63
```

The code - in case someone wants to try converting it to an iterative solution which should be much faster.

```
« → number
   « primes 2. ^ DUP SIZE 0 → squares nsquares count
      « « → prefix startpos add?
            « IF prefix number ≤ THEN
                    startpos nsquares FOR i
                        IF squares i GET number > THEN
                           nsquares 1 + 'i' STO @ exit loop
                        ELSE
                           prefix squares i GET * DUP
                           IF number > THEN
                               DROP
                           ELSE
                               DUP number SWAP / IP
                               IF add? THEN count + ELSE count SWAP - END 'count' STO
                               i 1 + add? NOT combinations EVAL
                           END
                        END
                    NEXT
                END
            »
         » → combinations
            « 1 1 1 combinations EVAL
                number count - DUP number 6 * SWAP / √
            »
      »
   »
»
```

Sudhir

EMAIL PM FIND QUOTE REPORT

19th March, 2023, 19:32 (This post was last modified: 19th March, 2023 22:33 by J-F Garnier.) **Post: #10**

J-F Garnier
Senior Member

Posts: 790
Joined: Dec 2013

RE: [VA] SRC #013 - Pi Day 2023 Special

> **Valentin Albillo Wrote:** (19th March, 2023 01:37)
>
> > **J-F Garnier Wrote:**
> > However, I'm not fully satisfied by applying a formula without understanding how and why it works.
>
> However, in this post of yours in my recent ***SRC #012e - Then and Now: Roots*** thread, you *did* use *Bornemann*'s formula
> [..]
> Same here, right ? 🙂

Understood !

> **Quote:**
> Please post your best estimates for the runtimes when using a physical **HP-71B**, for comparison purposes. Also, try and include the results/timings for ***N*** = ***10****, **25** and **33** million*, if you can.

Certainly.
Below are the timings for a HP-71B 1BBBB (636kHz), a HP-71B 2CDCC (650kHz) and Emu71/Win in Authentic Speed, respectively.
The speed of HP-71B can easily vary by 5 to 10% due to component tolerance and battery condition.
The 71B clock can be checked with the SYSTEM("CLOCK") command provided in the SYSTEMFN LEX or in my ULIB52 collection.
On the other hand, Emu71/Win in Authentic Calculator Speed mode gives reproducible timings (provided the system is not heavily loaded).

```
      N    Count  Timings (71B/1B, 71B/2C, Emu71/Auth.)
   12345   7503 26s/23s/24s
 100000  60794 78s/70s/73s
 567890 345237 194s/174s/182s
1000000 607926 260s/233s/244s
```

So your implementation seems to be about 2x faster than mine. I see some ways to gain maybe 25%, but not much more. Interesting.

More results on Emu71/Win only:

```
  N     Count      PI approx.        Timings (fast/auth.)
10E6  6079291 3.14158749068 0.8s/813s
25E6 15198180 3.14159239999 1.2s/1308s
33E6 20061593 3.14159276017 1.4s/1511s
```

J-F

EMAIL PM WWW FIND QUOTE REPORT

20th March, 2023, 03:18 **Post: #11**

2old2randr
Junior Member

Posts: 42
Joined: Jan 2018

RE: [VA] SRC #013 - Pi Day 2023 Special

I have updated the run times in my original post using the Sys RPL version of the Möbius function written by Gerald H (thanks, John Keith). This turns out to be twice as fast as the times obtained on the physical HP-71B.

Sudhir

EMAIL PM FIND QUOTE REPORT

20th March, 2023, 04:22 **Post: #12**

Valentin Albillo
Senior Member

Posts: 958
Joined: Feb 2015
Warning Level: 0%

RE: [VA] SRC #013 - Pi Day 2023 Special

.
Hi, **J-F** and **2old2randr**,

> **J-F Garnier Wrote:**
>
> > **I Wrote:**
> >
> > [...] Same here, right ? 🙂
>
> Understood !

Good.

> **J-F Garnier Wrote:**
>
> Certainly.
>
> Below are the timings for a HP-71B 1BBBB (636kHz), a HP-71B 2CDCC (650kHz) and Emu71/Win in Authentic Speed, respectively. [...] Emu71/Win in Authentic Calculator Speed mode gives reproducible timings [...] So your implementation seems to be about 2x faster than mine. I see some ways to gain maybe 25%, but not much more. Interesting.
>
> More results on Emu71/Win only:
>
> | N | Count | PI approx. | Timings (fast/auth.) |
> |---|---|---|---|
> | 10E6 | 6079291 | 3.14158749068 | 0.8s/813s |
> | 25E6 | 15198180 | 3.14159239999 | 1.2s/1308s |
> | 33E6 | 20061593 | 3.14159276017 | 1.4s/1511s |

Very good, thanks. In reciprocity, here are more of my results for you, obtained with my original *Solution 1* (my original *Solution 2* is slower):

N	Count	π approx	|Error|	*go71b @128x*	*Emu71/Win @976x*	*Physical HP-71B*
10,000,000	6,079,291	3.14158749	0.00000516	3.03"	0.40"	6' 28"
25,000,000	15,198,180	3.14159240	0.00000025	4.87"	0.64"	10' 23"
33,000,000	20,061,593	3.14159276	0.00000011	5.61"	0.74"	11' 58"
1E8	60,792,694	3.14159307	0.00000042	9.93"	1.30"	21' 11"
1E9	607,927,124	3.14159260	0.00000006	32.45"	4.26"	69' 14"

It would seem that my results for ***Emu71/Win*** @ *976x* are ~ ***2x*** faster than yours but please provide the *speed* factor *(976x in mine)* for comparison purposes. Also and for the same reason, please provide your timings for ***N*** = ***10^8*** and ***10^9***, if at all possible.

> **2old2randr Wrote:**
>
> I have updated the run times in my original post **using the Sys RPL version of the Möbius function** written by Gerald H (thanks, John Keith). This turns out to be twice as fast as the times obtained on the physical HP-71B.

Thanks but unless I'm mistaken (I don't know the first word about *RPL*, let alone *Sys RPL*) you *didn't* include the *Sys RPL* code in your edited post and I think you should, lest the posted code and timings aren't synchronized with one another.

Also, as I told **J-F** above, please provide your timings for ***N*** = ***10^8*** and ***10^9***, if at all possible.

Best regards.
V.

PM WWW FIND EDIT QUOTE REPORT

20th March, 2023, 16:43 (This post was last modified: 20th March, 2023 19:15 by J-F Garnier.) **Post: #13**

J-F Garnier
Senior Member

Posts: 790
Joined: Dec 2013

RE: [VA] SRC #013 - Pi Day 2023 Special

> **Valentin Albillo Wrote:** (20th March, 2023 04:22)
>
> In reciprocity, here are more of my results for you, obtained with my original *Solution 1* (my original *Solution 2* is slower):
>
> | N | Count | π approx | \|Error\| | *go71b @128x* | *Emu71/Win @976x* | *Physical HP-71B* |
> |---|---|---|---|---|---|---|
> | 10,000,000 | 6,079,291 | 3.14158749 | 0.00000516 | 3.03" | 0.40" | 6' 28" |
> | 25,000,000 | 15,198,180 | 3.14159240 | 0.00000025 | 4.87" | 0.64" | 10' 23" |
> | 33,000,000 | 20,061,593 | 3.14159276 | 0.00000011 | 5.61" | 0.74" | 11' 58" |
> | 1E8 | 60,792,694 | 3.14159307 | 0.00000042 | 9.93" | 1.30" | 21' 11" |
> | 1E9 | 607,927,124 | 3.14159260 | 0.00000006 | 32.45" | 4.26" | 69' 14" |
>
> It would seem that my results for ***Emu71/Win*** @ *976x* are ~ ***2x*** faster than yours but please provide the *speed* factor *(976x in mine)* for comparison purposes. Also and for the same reason, please provide your timings for ***N*** = $\mathbf{10^8}$ and $\mathbf{10^9}$, if at all possible.

Estimating the speed ratio of Emu71/Win on modern platforms is a subject by itself, but I estimate a peak ratio of ~1500x on my latest Ryzen5 laptop.

> **Quote:**
>
> please provide your timings for ***N*** = $\mathbf{10^8}$ and $\mathbf{10^9}$, if at all possible.

Execution timings on the physical HP-71B are estimated in Emu71/Win Authentic mode.

```
N          Count     PI approx.  Timings (fast, auth.)
1E8       60792694  3.14159307180 2.2s 46min
1E9      607927124  3.14159259637 7.3s 2h32min
1E10    6079270942  3.14159267337  24s N/A
1E11   60792710280  3.14159265115  81s N/A
1E12  607927102274  3.14159265250 276s N/A
```

J-F

EMAIL PM WWW FIND QUOTE REPORT

21st March, 2023, 12:45 (This post was last modified: 21st March, 2023 16:20 by J-F Garnier.) **Post: #14**

J-F Garnier
Senior Member

Posts: 790
Joined: Dec 2013

RE: [VA] SRC #013 - Pi Day 2023 Special

> **Valentin Albillo Wrote:** (20th March, 2023 04:22)
>
> It would seem that my results for ***Emu71/Win*** @ *976x* are ~ ***2x*** faster than yours...

Since Valentin didn't post his solutions yet, here is my optimized code for speed.
It is much less readable than my first version, but still not as obscure as SysRPL :-)

```
10 ! SRC13 verC
20 INPUT N
50 T=TIME @ M=SQR(N) @ S=N-IP(N/4)-IP(N/9)
60 FOR I=4 TO M @ GOSUB 110 @ GOSUB 110 @ GOSUB 110 @ NEXT I
80 T=TIME-T @ DISP N;S;SQR(6*N/S);T
90 END
100 !
110 I=I+1 @ R=I @ C=-1 @ Q=1
120 P=PRIM(R) @ IF P=Q THEN RETURN ELSE IF P THEN C=-C @ R=R/P @ Q=P @ GOTO 120
130 IF R=Q THEN RETURN ELSE S=S+C*IP(N/(I*I)) @ RETURN
```

Execution times are now close to Valentin's results:

```
 N       Count  Timings (HP-71B)
  12345   7503 11s
 100000  60794 35s
 567890 345237 90s
1000000 607926 122s
```

More results:

```
N         Count     PI approx.  Timings (Emu71 fast, auth.)
1E7        6079291  3.14158749068 0.4s 6min59s
1E8       60792694  3.14159307180 1.3s 24min
1E9      607927124  3.14159259637 4.1s 81min
1E10   6079270942  3.14159267337  14s N/A
1E11  60792710280  3.14159265115  48s N/A
1E12 607927102274  3.14159265250 172s N/A
```

J-F

EMAIL PM WWW FIND QUOTE REPORT

21st March, 2023, 20:06 (This post was last modified: 22nd March, 2023 03:03 by Valentin Albillo.) **Post: #15**

Valentin Albillo
Senior Member

Posts: 958
Joined: Feb 2015
Warning Level: 0%

RE: [VA] SRC #013 - Pi Day 2023 Special

Hi, **all**,

About a week has elapsed since I posted this π ***Day 2023 Special*** challenge, thank you very much to those few of you who contributed your valuable solutions and/or comments, namely **2old2randr**, **J-F Garnier, vaklaff** and **John Keith**, really much appreciated.

On the one hand, it's a pity that it failed to attract the attention of other people usually interested in my productions but that's life. On the other hand, this time I got *RPL* (and even *Sys RPL !*) solutions, which were conspicuously absent from past challenges.

Now's the time for my detailed ***sleuthing*** *process* and resulting *original* ***solutions***, plus *additional* ***comments:***

My sleuthing process

First of all, if naively taking at face value the problem's statement, you'd might be tempted to just program a simple loop from ***1*** to ***N***, compute the factorization for each number and tally up the ones which have repeated factors. This is as simple as it gets but for large ***N*** (say 1 million) it's grossly *inefficient* and here we're most interested in raw *speed*.

What to do ? Search in references or the *Internet* for a better algorithm, that's what, in particular one which is better than ***O(N)***, as any such loop will take *ages* for large ***N***, even if nothing were done inside the loop. We need something which is at least ***O(√N)***, thus transforming a *1,000,000*-cycle loop into a *1,000*-cycle one.

To search for that, we notice that asking for numbers whose factorization has *no repeated prime factors* is tantamount to asking for numbers who *aren't divisible by any square* other than 1, which are usually called square-free integers, and a quick *Google* search for the term reveals many relevant links, among them one at ***Wolfram Research's MathWorld***, where we find a formula with the required order (i.e. the upper limit of the summation goes up to ***√N***,) equivalent to this one:

$$S(n) = \sum_{d=1}^{\lfloor\sqrt{n}\rfloor} \mu(d)\left\lfloor\frac{n}{d^2}\right\rfloor \quad \textit{where } \mu(d) \textit{ is the Möbius function.}$$

which also appears in text form as `a(n) = Sum_{k = 1..floor(sqrt(n))} mu(k)*floor(n/k^2)` at **OEIS A013928** Number of (positive) squarefree numbers < n, and many other sites and documents. The simple proof of this formula can be obtained using the inclusion-exclusion principle.

As for the ***Möbius function***, it's a very important *number-theoretical* function which is easily computed in various ways, like this simple **HP-71B** user-defined function **FNM** (which should be placed at the end of any program using it):

```
1  DEF FNM(N) @ IF N=1 THEN FNM=1 @ END ELSE F=1
2  D=PRIM(N) @ IF NOT D THEN FNM=(-1)^F ELSE IF MOD(N,D*D) THEN F=F+1 @ N=N/D @ GOTO 2

   >FOR N=96673 TO 96686 @ FNM(N); @ NEXT N @@      ►  1 1 0 0 0 0 0 0 1 1 1 0 -1 -1
   >S=0 @ FOR N=1 TO 1109 @ S=S+FNM(N) @ NEXT N @ S  ►  -15    { Mertens(1109) }
```

but the important thing here is that we only need ***√N*** evaluations of it instead of ***N***, which makes all the difference in the world.

My original solutions

Yes, *solutions*, plural, as I created *two* of them, optimized for different cases. My ***first solution*** is optimized for speed, regardless of memory use, and uses the ***S(n)*** formula above, but instead of computing each *individual* value for the *Möbius* function inside the loop, it does compute *all* ***√N*** *values en masse* before starting the summation loop, which can be done without factoring, multiplications or divisions by using a trivial *sieve* procedure similar to the well-known ancient Sieve of Eratosthenes algorithm used to find all prime numbers up to a limit.

The sieve procedure used here runs fast, ultimately winning over for large ***N***, and all needed values of *Möbius(n)* are left stored in an array to be later used inside the summation loop.

First solution: speed

This little *4-line*, *217-byte* **HP-71B** program implements the procedure, first sieving and storing in **INTEGER** array ***M*** all ***√N*** *Möbius* values needed, then computing the summation and outputting results and timings: *(uses **Math** and **JPC** ROMs)*

```
1  DESTROY ALL @ INPUT T @ SETTIME 0 @ N=SQR(T) @ INTEGER M(N) @ MAT M=CON @ P=2 @ WHILE P<=N
2  S=P*P @ FOR K=S TO N STEP S @ M(K)=0 @ NEXT K @ FOR K=P TO N STEP P @ M(K)=-M(K) @ NEXT K
3  P=FPRIM(P+1) @ END WHILE @ S=T @ FOR K=2 TO N @ S=S+M(K)*(T DIV (K*K)) @ NEXT K
4  DISP USING "2(3DC3DC3DC3D,2X),2(Z.8D,X),5DZ.2D";T,S,SQR(6*T/S),ABS(PI-RES),TIME
```

N	Count	π approx	|Error|	go71b	Emu71/Win	Physical
				@128x	@976x	HP-71B
12,345	7,503	3.14198205	0.00038939	0.10"	0.01"	13"
100,000	60,794	3.14155933	0.00003333	0.28"	0.04"	36"
567,890	345,237	3.14158684	0.00000582	0.69"	0.09"	1' 28"
1,000,000	607,926	3.14159550	0.00000285	0.93"	0.12"	1' 59"
10,000,000	6,079,291	3.14158749	0.00000516	3.03"	0.40"	6' 28"
25,000,000	15,198,180	3.14159240	0.00000025	4.87"	0.64"	10' 23"
33,000,000	20,061,593	3.14159276	0.00000011	5.61"	0.74"	11' 58"
1E8	60,792,694	3.14159307	0.00000042	9.93"	1.30"	21' 11"
1E9	607,927,124	3.14159260	0.00000006	32.45"	4.26"	69' 14"

The caveat is, storing all *Möbius* values in **INTEGER** array ***M*** does require large amounts of memory if ***N*** is huge, e.g.:

- For ***N*** = ***1E8***, the program uses ~ *30,055 bytes* of *RAM* (***M*** has *10,000* elements, *30,000 bytes*)
- For ***N*** = ***1E9***, the program uses ~ *94,927 bytes* of *RAM* (***M*** has *31,623* elements, *94,869 bytes*)

Larger values of ***N*** are problematic, e.g. ***N*** = ***1E10*** would require in excess of *300 Kb* of *RAM*, at the very limit of what the **HP-71B** can manage, which leads us to my ***second solution***, where I'll turn from computing all *Möbius* values *en masse* to computing them *individually*, thus avoiding the large memory requirements, albeit at the cost of speed.

Second solution: memory

This even smaller *3-line*, *178-byte* **HP-71B** program uses in-lined *Möbius* function code (as oposed to an *user-defined function*) for speed, and needs only ~ *63 bytes* of *RAM* to run no matter the size of ***N***, all the way to a *trillion* (10^{12}) *- 1*, i.e. *999,999,999,999*, which is the largest integer the *HP-71B* can represent: *(uses **JPC** ROM)*

```
1 DESTROY ALL @ INPUT T @ SETTIME 0 @ U=-1 @ S=T @ FOR K=2 TO SQR(T) @ N=K @ F=1
2 D=PRIM(N) @ IF NOT D THEN S=S+U^F*IP(T/(K*K)) ELSE IF MOD(N,D*D) THEN F=F+1 @ N=N/D @ GOTO 2
3 NEXT K @ DISP USING "2(3DC3DC3DC3D,2X),2(Z.8D,X),5DZ.2D";T,S,SQR(6*T/S),ABS(PI-RES),TIME
```

N	Count	π approx	|Error|	go71b	Emu71/Win	Physical
				@128x	@976x	HP-71B
12,345	7,503	3.14198205	0.00038939	0.09"	0.01"	12"
100,000	60,794	3.14155933	0.00003333	0.26"	0.03"	33"
567,890	345,237	3.14158684	0.00000582	0.67"	0.09"	1' 26"
1,000,000	607,926	3.14159550	0.00000285	0.91"	0.12"	1' 56"
10,000,000	6,079,291	3.14158749	0.00000516	3.15"	0.41"	6' 43"
25,000,000	15,198,180	3.14159240	0.00000025	5.15"	0.68"	10' 59"
33,000,000	20,061,593	3.14159276	0.00000011	5.98"	0.78"	12' 45"
1E8	60,792,694	3.14159307	0.00000042	10.81"	1.42"	23' 4"
1E9	607,927,124	3.14159260	0.00000006	36.93"	4.84"	78' 47"

```
  1E10    6,079,270,942  3.14159267  0.00000002   2'  8"   16.82"     4h 33'
  1E11   60,792,710,280  3.14159265  0.00000000*  7' 37"   59.95"    16h 15'
1E12-1 607,927,102,274  3.14159265  0.00000000* 28'  9"    3' 41"    60h  2'
```

* the 12-digit values are *3.14159265115*, *0.00000000244* and *3.14159265250*, *0.00000000109*, respectively.

Thus, although this second solution can achieve the maximum possible integer range ***1*** .. ***10^{12} - 1*** with no memory problems and even seems to be a little faster for ***N*** up to *1 million*, this is only because the **HP-71B** hardware and *BASIC* language are optimized for fast *mathematical operations* performed entirely in *assembler* without user loops or branching, which are pretty slow and thus slow down the sieving procedure.

This is why a *matrix inversion* (**MAT INV**) or a *Fast Fourier Transform* (**FOUR**) are performed at speeds competing with much faster *CPU*s, while an empty **FOR-NEXT** construct does only *~100 loops/sec.* on a physical *HP-71B*. Even so, the second solution will get increasingly slower than the first one for large ***N***.

Additional Comments

• In the *very* distant past *(2004)*, I posted the pair of threads:

> HP Challenge VA114 - The Turtle (HP-71B) and the Hare (HP49G+)
> HP Challenge VA115 - The 71B Turtle and 49Gp Hare Harder 2nd Half

where I pitted the **HP-71B** vs. the **HP49G+**, executing various quite hard computations, and the resulting times showed that the **HP49G+** was usually about ***4.6x*** to ***6.4x*** faster than the **HP-71B**, so any timings must take this into account in order to ascertain the relative performance of the various solutions running on different hardware.

• You may be wondering why I selected ***N*** = ***33,000,000*** as a test case. It's an innocuous number in and of itself but I found it while reading this truly *excellent book eons ago:*

> ***The Mathematical Experience*** by *Phillip J. Davis* and *Reuben Hersh*
> (464 pages, *ISBN-10:* 9780395929681)

where they say:

> *"A more refined piece of natural scientific research into prime Numbers was reported in a paper by I. J. Good and R. F. Churchhouse in 1967 [...]"*

and they elaborate in *page 367*, I quote:

> *"To check their probabilistic reasoning, Good and Churchhouse did some numerical work. [...] In a separate calculation, they found that the total number of zeros of μ(n)* [VA: the **Möbius** function] *for n between 0 and **33,000,000** is 12,938,407.*
>
> *The "expected number" is **33.000.000***(1 -6/π^2). which works out to 12,938,405.6. They call this "an astonishingly close fit, better than we deserved." A nonrigorous argument has predicted a mathematical result to 8 place accuracy. In physics or chemistry. experimental agreement with theory to 8 place accuracy would be regarded as a very strong confirmation of the theory. Here, also, it is impossible to believe that such agreement is accidental. The principle by which the calculation was made must be right.* "

I was sure it took them considerable time to get the tally for *33,000,000* using some 1967-era computer and wanted to know how long it would take a 1984-era handheld **HP-71B** to get the result, which is less than *12 min.* for a physical machine and less than a *second* for a modern emulation *(Emu71/Win)*.

Both timings would be reduced at least an order of magnitude if rewritten in *assembler* and run in the same hardware (perhaps **J-F Garnier** would obligue and create a **MAT M=MOB** matrix statement which would fill up array ***M*** with the results of the sieving procedure implemented in *BASIC* in my first solution ...)

By the way, the aforementioned 1967 paper by I. J. Good and R. F. Churchhouse is:

> ***The Riemann Hypothesis and Pseudo-random Features of the Möbius Sequence***

where among other very interesting things, they say:

> *"Thus the Möbius sequence contains arbitrarily long runs of zeros, but these long runs presumably occur extremely rarely."*

I showed above such a run of zeros from ***N*** = ***96675*** to ***96680***, *six* consecutive zeros in all. It would make for an interesting challenge to locate longer runs.

• These are the *exact* counts ***S(10^N)*** for selected ***N***:

```
        N  S(10^N)
-----------------------------------------
        0  1
        1  7
        2  61
        3  608
        4  6083
        5  60794
        6  607926
        7  6079291
        8  60792694
        9  607927124
       10  6079270942
       11  60792710280
       12  607927102274

       13  6079271018294
       14  60792710185947
       16  6079271018540405
       18  607927101854022750
       24  6079271018540266628773299
       30  607927101854026628663278087296
       36  607927101854026628663276779463775476
 6/π² = 0.607927101854026628663276779258365833...
```

So you see, you only need to tally up numbers up to ***10³⁶*** with no repeated prime factors and you'll get an approximation to π correct to ***27*** *decimal digits !* 🙂

That's all for now, I hope you enjoyed it. Regrettably, due to a number of factors *(pun intended,)* I'm now taking a *leave of absence* from further challenges for a while. ***Cya***.

V.

Edit: rephrased one sentence.

PM WWW FIND EDIT ✕ QUOTE REPORT

22nd March, 2023, 11:19 **Post: #16**

EdS2
Senior Member

Posts: 517
Joined: Apr 2014

RE: [VA] SRC #013 - Pi Day 2023 Special

Nice challenge Valentin, based on an interesting finding! I've posted some reflections over here.

EMAIL PM FIND QUOTE REPORT

23rd March, 2023, 02:24 **Post: #17**

Valentin Albillo
Senior Member

Posts: 958
Joined: Feb 2015
Warning Level: 0%

RE: [VA] SRC #013 - Pi Day 2023 Special

Hi, **all**,

Casually looking at my code for ***solution 1*** I realized that I did overlook a small but obvious optimization, namely not trying to update the sum ***S*** if the *Möbius* value ***M(K)*** is zero.

The slightly modified code is now *225 bytes* instead of *217*, and runs about ***5,4%*** *faster:*

```
1  DESTROY ALL @ INPUT T @ SETTIME 0 @ N=SQR(T) @ INTEGER M(N) @ MAT M=CON @ P=2 @ WHILE P<=N
2  S=P*P @ FOR K=S TO N STEP S @ M(K)=0 @ NEXT K @ FOR K=P TO N STEP P @ M(K)=-M(K) @ NEXT K
3  P=FPRIM(P+1) @ END WHILE @ S=T @ FOR K=2 TO N @ IF M(K) THEN S=S+M(K)*(T DIV (K*K))
4  NEXT K @ DISP USING "2(3DC3DC3DC3D,2X),2(Z.8D,X),5DZ.2D";T,S,SQR(6*T/S),ABS(PI-RES),TIME
```

```
        N         Count     π approx    |Error|      go71b   Emu71/Win  Physical
                                                     @128x     @976x     HP-71B
-------------------------------------------------------------------------------
     12,345       7,503   3.14198205  0.00038939     0.09"     0.01"        12"
    100,000      60,794   3.14155933  0.00003333     0.27"     0.04"        35"
    567,890     345,237   3.14158684  0.00000582     0.65"     0.09"     1' 23"
```

```
     1,000,000     607,926  3.14159550  0.00000285    0.87"    0.11"     1' 51"

    10,000,000   6,079,291  3.14158749  0.00000516    2.86"    0.38"     6'  6"
    25,000,000  15,198,180  3.14159240  0.00000025    4.59"    0.60"     9' 48"
    33,000,000  20,061,593  3.14159276  0.00000011    5.29"    0.69"    11' 17"

           1E8  60,792,694  3.14159307  0.00000042    9.37"    1.23"    19' 59"
           1E9 607,927,124  3.14159260  0.00000006   30.70"    4.03"    65' 30"
```

V.

PM WWW FIND EDIT QUOTE REPORT

23rd March, 2023, 16:03 (This post was last modified: 23rd March, 2023 16:04 by EdS2.) **Post: #18**

EdS2
Senior Member

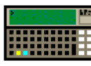
Posts: 517
Joined: Apr 2014

RE: [VA] SRC #013 - Pi Day 2023 Special

BTW, the short paper *The Riemann Hypothesis and Pseudorandom Features of the Möbius Sequence* by Good and Churchhouse can be found here. (It's an example of a large scale pure mathematics computation, in 1967, running in the background on the Atlas at Chilton. We're not told the runtime.)

EMAIL PM FIND QUOTE REPORT

24th March, 2023, 14:28 **Post: #19**



J-F Garnier
Senior Member

Posts: 790
Joined: Dec 2013

RE: [VA] SRC #013 - Pi Day 2023 Special

So this time we got two solutions from Valentin, plus as often interesting comments and references.
The first solution is nice and quite unexpected, but my preference goes to the second solution that has no limitation due to memory.

My last solution posted just a few hours before Valentin's final solutions probably didn't catch much attention.
Yet I was quite happy with the tricks I used to speed up my code by almost a factor of 2.

One trick was to skip the unnecessary evaluation of the Möbius function for all multiples of 4, i.e. one number over 4 or a 25% gain.
But this was not enough to beat Valentin's timings, and even more with his last 5% optimization of his fastest version.

So here is another attempt, pushing the optimisation further towards speed with just a slightly larger code, but no huge memory requirements.

```
5 ! SRC13 verE
10 INPUT N
15 T=TIME @ M=(SQR(N)+3) DIV 4*4 @ S=0
20 FOR I=M TO 5 STEP -1
25 I=I-1 @ R=I @ C=-1
30 P=PRIM(R) @ IF NOT P THEN S=S+C*(N DIV (I*I)) ELSE IF MOD(R,P*P) THEN C=-C @ R=R/P @ GOTO 30
35 I=I-1 @ R=I/2 @ C=1
40 P=PRIM(R) @ IF NOT P THEN S=S+C*(N DIV (I*I)) ELSE IF MOD(R,P*P) THEN C=-C @ R=R/P @ GOTO 40
45 I=I-1 @ R=I @ C=-1
50 P=PRIM(R) @ IF NOT P THEN S=S+C*(N DIV (I*I)) ELSE IF MOD(R,P*P) THEN C=-C @ R=R/P @ GOTO 50
55 NEXT I
60 S=S-(N DIV 9)-(N DIV 4)+N
65 T=TIME-T @ DISP N;S;SQR(6*N/S);T
```

Execution times are now closer ... no, *better* (slightly) than Valentin's results :-)

```
 N       Count  Timings (HP-71B)
  12345   7503 7.4s
 100000  60794 24s
 567890 345237 64s
1000000 607926 87s
```

More results on Emu71/Win, in fast and Authentic modes:

```
N          Count      PI approx.  Timings (Emu71 fast, auth.)
1E7         6079291  3.14158749068 0.3s 5min11s
1E8        60792694  3.14159307180 0.9s 18min8s
1E9       607927124  3.14159259637 3.1s 63min1s
```

```
1E10   6079270942  3.14159267337  11s N/A
1E11  60792710280  3.14159265115  39s N/A
1E12 607927102274  3.14159265250 144s N/A
```

Do we have to stop at 1E12?
No! The computing loop uses numbers from 2 to SQR(N) which is no problem.
Just the count must be carried carefully, starting from smallest terms, to the highest that can exceed 1E12.
Also some operations must be written carefully, for instance replacing IP(N/X) with N DIV X.
This explains the changes in the program above.

Of course, the final count will have only 12 significant digits, but if is correct within a few ULPs as expected, then we can get an approximation of pi with the same accuracy.

Results on Emu71/Win full speed only:

```
N          Count          PI approx.  Timings (Emu71 fast)
1E13 6.07927101830E12 3.14159265365 9min27s
1E14 6.07927101860E13 3.14159265357 39min
1E15 6.07927101854E14 3.14159265359 166min
```

J-F

EMAIL PM WWW FIND QUOTE REPORT

24th March, 2023, 14:38 **Post: #20**

EdS2
Senior Member

Posts: 517
Joined: Apr 2014

RE: [VA] SRC #013 - Pi Day 2023 Special

Bravo!

I'm thinking, on a platform which lacks the very handy PRIM predicate, one might proceed first with a sieve and then lookup into that for the PRIM calls. It will potentially take a lot of storage, and packing the bits will slow down access.

It might be that saving the primes in this way is very similar to Valentin's program which records the Möbius values, although in that case I think we still need an implementation of FPRIM.

EMAIL PM FIND QUOTE REPORT

25th March, 2023, 00:40 **Post: #21**

Valentin Albillo
Senior Member

Posts: 958
Joined: Feb 2015
Warning Level: 0%

RE: [VA] SRC #013 - Pi Day 2023 Special

Hi, **J-F Garnier** and **EdS2**,

> **J-F Garnier Wrote:**
> So this time we got **two solutions** from Valentin, plus as often **interesting comments and references**.

Thanks for your continued appreciation, **J-F**. I created first *solution 1*, as doing a sieve was the fastest way to accomplish the goal, but as *HP* didn't see fit to implement **BYTE** and/or **BOOLEAN** arrays in *71 BASIC*, like many other high-level languages do, the memory consumption of **INTEGER** arrays (3 bytes per element) required lots of *RAM* for large ***N***, and thus I created *solution 2*, which in the long run gets increasingly slower but has insignificant memory requirements.

> **J-F Garnier Wrote:**
> My last solution posted just a few hours before Valentin's final solutions probably **didn't catch much attention**. Yet I was quite happy with the tricks I used to speed up my code by almost a factor of 2.

Oh yes, it did. I noticed it at once but refrained from replying something outright as I intended to include a version of my *solution 1* able to run in puny *BASIC* dialects without using any of the *HP-71 BASIC* enhancements provided by the ***Math*** and ***JPC** ROMs*, as **EdS2** requested here for using it on *BBC BASIC* or other *"ordinary BASIC"* (his words).

> **J-F Garnier Wrote:**
> **One trick was to skip the unnecessary evaluation** of the Möbius function for all multiples of 4, i.e. one number over 4 or a 25% gain. But this was not enough to beat Valentin's timings, and even more with his last 5% optimization of his

> fastest version.

I like the way you never let go until you succeed, and your "tricks" (I'd call them "techniques") are indeed clever and I'm sure interested people will benefit from studying them.

> **J-F Garnier Wrote:**
>
> **So here is another attempt**, pushing the optimisation further towards speed with just a slightly larger code, but no huge memory requirements. [...]

Ditto. However, your use of **PRIM** means, as I've said, that in the long run that approach can't beat sieve-based methods, because finding a factor (or ensuring that there's none) for large enough numbers is quite costly.

> **J-F Garnier Wrote:**
>
> Execution times are now closer ... no, **better (slightly)** than Valentin's results :-) [...] More results on Emu71/Win, in fast and Authentic modes

Congratulations, but I think you mentioned before that your *Emu71/Win* is running at ~ *1,500x*, I quote:

*"I estimate a peak ratio of ~**1500x** on my latest Ryzen5 laptop"*

while my timings are for a ~ *976x* instance, so it would be proper to take that speed difference into account for accurate comparisons of the *Emu71/Win* timings.

> **J-F Garnier Wrote:**
>
> **Do we have to stop at 1E12? No!** [...] Of course, the final count will have only 12 significant digits, but if is correct within a few ULPs as expected, then we can get an approximation of pi with the same accuracy.

Impressive ! ... ... and that last approximation for ***N*** = ***1E15***, namely **3.14159265359**, is a sight to behold !
Congratulations, again !

> **EdS2 Wrote:**
>
> I'm thinking, on a platform which lacks the very handy **PRIM** predicate, one might proceed first with a sieve and then lookup into that for the **PRIM** calls. [...] It might be that saving the primes in this way is very similar to Valentin's program which records the Möbius values, although in that case I think we still need an implementation of **FPRIM**.

Here you are, the promised code for your first *"related challenge"*, to run my ***solution 1*** for the **HP-71B** but using an *"unaugmented* [sic] *71B, or a similarly ordinary Basic"* (your words):

```
  EDS2    10 lines, 389 bytes

   10  DESTROY ALL @ INPUT T @ SETTIME 0 @ N=IP(SQR(T)) @ GOSUB 70
   20  INTEGER M(N) @ FOR K=1 TO N @ M(K)=1 @ NEXT K @ L=1 @ P=2
  *30  S=P*P @ FOR K=S TO N STEP S @ M(K)=0 @ NEXT K @ FOR K=P TO N STEP P
   40  M(K)=-M(K) @ NEXT K @ L=L+1 @ IF L<=W THEN P=F(L) @ GOTO 30
   50  S=T @ FOR K=2 TO N @ IF M(K) THEN S=S+M(K)*(T DIV (K*K))
   60  NEXT K @ R=SQR(6*T/S) @ DISP T;S;R;ABS(PI-R);TIME @ END

  *70  H=CEIL(N/2) @ INTEGER F(H) @ F(1)=1 @ FOR S=2 TO (SQR(2*H)+1)/2 @ IF F(S) THEN 90
   80  FOR K=2*S*(S-1)+1 TO H STEP 2*S-1 @ F(K)=1 @ NEXT K
  *90  NEXT S @ W=1 @ F(1)=2 @ FOR K=1 TO H @ IF NOT F(K) THEN W=W+1 @ F(W)=2*K-1
  100  NEXT K @ INTEGER F(W) @ RETURN
```

Description:

- Lines **10** to **60** are essentially the same as in my ***solution 1*** for the **HP-71B**, with changes (described below in **Notes)** to adapt it to run in plain-vanilla *BASIC* dialects. In line *10*, before beginning the sieve to compute the ***Möbius*** function values *en masse*, the subroutine beginning at line *70* is first called to obtain at once all prime numbers required

- Lines **70** to **100**, nonexistent in *solution 1*, are now necessary to implement the functionality that the **FPRIM** keyword allowed there. This subroutine (called just once, before performing the *Möbius* sieve) computes and stores in integer array **F** all prime numbers required by performing yet another sieve. Once completed, the prime numbers identified are placed consecutively in array **F**, which is then redimensioned to reclaim memory, thus leaving more available to later dimension integer array **M** for the second sieve.

- As was the case with the ***Möbius*** sieve, the new prime sieve doesn't use any factoring, multiplications or divisions, for maximum speed. As will be seen in the timings below, this adapted program takes *only **24%** longer* than the original one, which used the *assembler* keyword **FPRIM**, so in this battle between my *BASIC* code vs. *JPC Assembler* code, I declare myself the winner !

Notes:

This is based on my ***solution 1*** for the **HP-71B** but with the following changes:

- I've replaced **MAT M=CON**, **WHILE** .. **END WHILE**, **FPRIM** and **RES** by functionally equivalent code, and I've deleted **USING "***image***"**. No keywords from the ***Math*** or ***JPC*** *ROMs* remain. The loop at lines *30-40* can be reimplemented using a **WHILE** .. **WEND** construct, if available.

- I'm not sure if **DIV** *(integer division)* exists in *BBC BASIC* or other *"ordinary"* dialects. If it doesn't, **A DIV B** can be replaced by **INT(A/B)**.

- Same thing with **H=CEIL(N/2)** at line *70*. If **CEIL** isn't available, it will need to be replaced by trivial equivalent code.

- For convenience while running and timing the code, I've left in the above code the **HP-71B** *BASIC* keywords **DESTROY ALL**, **SETTIME 0** and **TIME**, which should be either deleted or replaced by equivalent code in other *BASIC* dialects.

- The integer array **F** is first dimensioned in line *70* as **INTEGER F(H)**, with a *variable* number of elements **H**. In *BASIC* dialects which do not admit dimensioning arrays with a variable number of elements given by an expression, **F** will probably need to be dimensioned to the *fixed* *maximum* number of elements to use. Same thing happens at line *20* **INTEGER M(N)**.

- Also, the same array **F** is later **re**dimensioned in line *100* as **INTEGER F(W)**, reducing its size (because **W** is always < **H**) to reclaim memory. If the plain-vanilla dialect can't redimension arrays, or if it can but it doesn't keep intact the unaffected elements (initializes or destroys the whole array's contents,) then this redimensioning will not be possible and memory won't be reclaimed, which will limit the maximum argument **N** being input to the program.

  The memory savings by redimensioning can be huge. For instance, for **N** = *100,000,000*, redimensioning **F** reduces it from *5,000* elements to just *1,229* elements.

- As it stands now, the program recomputes the array of *prime* numbers each time it's run. If there's a RAM disk available or other sufficiently fast way to store and retrieve the array, it could be computed to its maximum size and stored just once, to be retrieved in whole or in part when the program is executed afterwards. This would reduce the running time considerably, depending on the retrieval speed vs. the computation speed.

Results:

Upon running the program with the following arguments on a physical/virtual **HP-71B**, we get these results and timings:

```
    N        Count      π approx        |Error|        go71b   Emu71/Win  Physical
                                                       @128x     @976x     HP-71B
-------------------------------------------------------------------------------
   12345      7503  3.14198204634  0.00038939275    0.12"     0.02"          15"
  100000     60794  3.14155932716  0.00003332643    0.34"     0.04"          44"
  567890    345237  3.14158683822  0.00000581537    0.83"     0.11"      1' 46"
 1000000    607926  3.14159550063  0.00000284704    1.11"     0.15"      2' 22"

10000000   6079291  3.14158749068  0.00000516291    3.61"     0.47"      7' 42"
25000000  15198180  3.14159239999  0.00000025360    5.77"     0.76"     12' 19"
33000000  20061593  3.14159276017  0.00000010658    6.65"     0.87"     14' 11"

   1E8    60792694  3.14159307180  0.00000041821   11.71"     1.54"     24' 59"
   1E9   607927124  3.14159259637  0.00000005722   38.00"     4.98"     1h  21'
```

That's all. Hope it's useful, or at least enjoyable. Tell me how it goes when running it on your beloved *BBC BASIC*.

V.

25th March, 2023, 20:02 (This post was last modified: 29th March, 2023 04:17 by DavidM.) **Post: #22**

DavidM
Senior Member

Posts: 918
Joined: Dec 2013

RE: [VA] SRC #013 - Pi Day 2023 Special

Using the Möbius-based algorithm already presented by 2old2randr and Valentin above, I put together the following SysRPL implementation to see how it would perform on a real 50g. I was curious as to how a pure SysRPL implementation would compare to the hybrid version already posted by 2old2randr.

This particular implementation makes use of a customized version of the Möbius function that Gerald H created. Instead of calling the function code as a subroutine, it is simply executed inline in the loop that accumulates the final count. I also wanted the result as an approximate number instead of an exact integer in order to reduce the need for subsequent conversions. Otherwise, it's the same method as Gerald's original Möbius function.

The rest of the code is fairly standard stack-based computations. As with any SysRPL implementation, there's an argument check at the beginning. This is followed by the sum-loop, and then the final computation of SQRT(6*N/Count):

Edit: this code has been updated to correct an issue with the computation of Count (identified below in this thread)

```
!NO CODE
!RPL
::
   CK1NOLASTWD
   CK&DISPATCH1
   real ::
      %1 OVER %> caseSIZEERR                ( N must be a positive integer <= [1E12-1] )
      DUP %IP OVER %<> caseSIZEERR
      DUP % 9.99999999999E11 %> caseSIZEERR

      DUP %SQRT %IP>#                       ( maxloop: IP[SQRT[N]] )
      %1                                    ( initial i value )
      %0                                    ( initial S subtotal )
      ROT ZERO_DO                           ( for i = 0 to [maxloop-1] )
         OVERDUP %1 %= ?SKIP ::             ( skip mu[i] if i=1 )
            FPTR2 ^R>Z                      ( convert i to zint for factoring )
            FPTR2 ^MZSQFF                   ( obtain factor meta )
            #2/                             ( convert meta size to pair count )
            %1 SWAP                         ( seed mu[] result with 1 )
            ZERO_DO                         ( Gerald's mu[] method: )
               ROTDROPSWAP                  (   check each factor magnitude, )
               %1 %= ITE %CHS DROP%0_       (   alter current mu[] result for given value )
            LOOP                            ( next factor )
         ;                                  ( end of mu[] )
         2OVER                              ( SL2: N, SL1: i )
         DUP %*                             ( i^2 )
         2DUP %MOD                          ( N MOD i^2 )
         ROTSWAP %-                         ( Q = N - [N MOD i^2] )
         SWAP %/                            ( Q = Q / i^2 )
         %*                                 ( Q * mu[i] )
         %+SWAP                             ( increment/decrement S, swap with i )
         %1+ SWAP                           ( increment i, swap with S )
      LOOP                                  ( next i )
      SWAPDROPDUP                           ( drop i, dup S )
      %6 4ROLL %*                           ( 6*N )
      SWAP %/                               ( 6*N/S )
      %SQRT                                 ( SQRT[6*N/S] )
   ;
;
@
```

Size: 182.5 bytes
CRC: #79C1h

Here's the results of similar inputs to those posted by others. Except where noted, the runtimes listed are averages of 3 runs:

				Physical 50g
N	**Count**	**Approximation**	**|error|**	**Runtime**
10	7	2.92770021885	0.21389243474	00:00:00.05
12,345	7,503	3.14198204634	0.00038939275	00:00:02.36
100,000	60,794	3.14155932716	0.00003332643	00:00:06.65
567,890	345,237	3.14158683822	0.00000581537	00:00:15.95
1,000,000	607,926	3.14159550063	0.00000284704	00:00:21.22
10,000,000	6,079,291	3.14158749068	0.00000516291	00:01:09.68
25,000,000	15,198,180	3.14159239999	0.00000025360	00:01:52.44
33,000,000	20,061,593	3.14159276017	0.00000010658	00:02:09.21
1E8	60,792,694	3.14159307180	0.00000041821	00:03:51.44
1E9	607,927,124	3.14159259637	0.00000005722	00:12:55.88

```
        1E10     6,079,270,942   3.14159267337   0.00000001978    00:44:36.81
        1E11    60,792,710,280   3.14159265115   0.00000000244    02:39:38.69 *
      1E12-1   607,927,102,274   3.14159265250   0.00000000109    10:03:57.12 *
```

* Time shown is for a single run

PM FIND QUOTE REPORT

26th March, 2023, 03:31 (This post was last modified: 26th March, 2023 03:32 by 2old2randr.) **Post: #23**

2old2randr
Junior Member

Posts: 42
Joined: Jan 2018

RE: [VA] SRC #013 - Pi Day 2023 Special

Thank you, David for that very nicely annotated (and indented) code. I am learning Sys RPL and example programs are very helpful.

A couple of questions regarding your program - what are the two lines (!NO CODE and !RPL) at the beginning of the program for? Secondly, what is the meaning of the underscore in DROP%0_?

Thanks
Sudhir

EMAIL PM FIND QUOTE REPORT

26th March, 2023, 05:37 **Post: #24**

DavidM
Senior Member

Posts: 918
Joined: Dec 2013

RE: [VA] SRC #013 - Pi Day 2023 Special

> **2old2randr Wrote:** (26th March, 2023 03:31)
>
> ...what are the two lines (!NO CODE and !RPL) at the beginning of the program for? Secondly, what is the meaning of the underscore in DROP%0_?

"!NO CODE" and "!RPL" are directives for the built-in MASD compiler that designate the appropriate modes for translation and output. They are specific to that particular compiler, and may not be needed if you set system flag -92. Other compilers (such as the HP Developer Tools) may not recognize them at all.

The underscore suffix is a convention used by the HP designers to indicate that the opcode is for an entry point that is not official, but is in a safe area and can be used in programs as needed. The underscore is actually part of the opcode's name, and doesn't have any special meaning for the compiler. You may also see certain SysRPL opcodes written within parentheses, which is another way to designate the same thing. You just need to know what your particular compiler is expecting.

PM FIND QUOTE REPORT

27th March, 2023, 16:02 **Post: #25**

EdS2
Senior Member

Posts: 517
Joined: Apr 2014

RE: [VA] SRC #013 - Pi Day 2023 Special

Thanks Valentin, David, J-F. I'm hoping to be able to apply some brain power to this and will post if I get something working. If I don't post, it's because I haven't marshalled the effort.

EMAIL PM FIND QUOTE REPORT

27th March, 2023, 16:28 **Post: #26**

DavidM
Senior Member

Posts: 918
Joined: Dec 2013

RE: [VA] SRC #013 - Pi Day 2023 Special

I hadn't noticed that the final Count value I listed in the table in post #22 was a bit off until Valentin sent me a note about it. We both assumed a typo, but it turns out not to be. The program I posted definitely produces the Count value specified for an input of 999,999,999,999.

While it certainly feels like some sort of rounding discrepancy, I haven't yet found the spot where things break down. I'm still working on it as time permits, but I wanted to let everybody know that it appears there's still another "puzzle in the puzzle". At least in the version I posted. 🙂

PM FIND QUOTE REPORT

27th March, 2023, 17:48 **Post: #27**

J-F Garnier
Senior Member

Posts: 790
Joined: Dec 2013

RE: [VA] SRC #013 - Pi Day 2023 Special

> **DavidM Wrote:** (27th March, 2023 16:28)
>
> I hadn't noticed that the final Count value I listed in the table in post #22 was a bit off until Valentin sent me a note about it. We both assumed a typo, but it turns out not to be. The program I posted definitely produces the Count value specified for an input of 999,999,999,999.
>
> While it certainly feels like some sort of rounding discrepancy, I haven't yet found the spot where things break down. I'm still working on it as time permits, but I wanted to let everybody know that it appears there's still another "puzzle in the puzzle". At least in the version I posted.

I didn't notice this discrepancy !
However ... running Valentin's 2nd solution also returns the same wrong value:

```
1 DESTROY ALL @ INPUT T @ SETTIME 0 @ U=-1 @ S=T @ FOR K=2 TO SQR(T) @ N=K @ F=1
2 D=PRIM(N) @ IF NOT D THEN S=S+U^F*IP(T/(K*K)) ELSE IF MOD(N,D*D) THEN F=F+1 @ N=N/D @ GOTO 2
3 NEXT K @ DISP USING "2(3DC3DC3DC3D,2X),2(Z.8D,X),5DZ.2D";T,S,SQR(6*T/S),ABS(PI-RES),TIME

>RUN
?1E12-1
999,999,999,999 607,927,102,272 3.14159265 0.00000000 178.90
```

J-F

EMAIL PM WWW FIND QUOTE REPORT

27th March, 2023, 21:46 **Post: #28**

Valentin Albillo
Senior Member

Posts: 958
Joined: Feb 2015
Warning Level: 0%

RE: [VA] SRC #013 - Pi Day 2023 Special

.
Hi, **J-F** and **DavidM**,

> **J-F Garnier Wrote:** (27th March, 2023 17:48)
>
> > **DavidM Wrote:** (27th March, 2023 16:28)
> >
> > I hadn't noticed that the final Count value I listed in the table in post #22 was a bit off until Valentin sent me a note about it. We both assumed a typo, but it turns out not to be. The program I posted definitely produces the Count value specified for an input of 999,999,999,999 [...]
>
> I didn't notice this discrepancy !
> However ... running Valentin's 2nd solution also returns **the same wrong value**:
>
> ```
> 1 DESTROY ALL @ INPUT T @ SETTIME 0 @ U=-1 @ S=T @ FOR K=2 TO SQR(T) @ N=K @ F=1
> 2 D=PRIM(N) @ IF NOT D THEN S=S+U^F*IP(T/(K*K)) ELSE IF MOD(N,D*D) THEN F=F+1 @ N=N/D @ GOTO 2
> 3 NEXT K @ DISP USING "2(3DC3DC3DC3D,2X),2(Z.8D,X),5DZ.2D";T,S,SQR(6*T/S),ABS(PI-RES),TIME
>
> >RUN
> ?1E12-1
> 999,999,999,999 607,927,102,272 3.14159265 0.00000000 178.90
> ```

Shocking ! I didn't notice at the time I posted my two original solutions because *I never saw the wrong count*, just the correct one. A little trip down the memory lane reveals what happened and why I didn't saw it:

When I ran my *solution 2* for various ***N***, the last one I entered at the input prompt was **1E12**, *exactly*, without the " **-1**" used to make it the largest 12-digit integer **999,999,999,999**. The program merrily run and output this:

```
      >RUN
              ?1E12
                      WRN L3:IMAGE Ovfl
                      ***,***,***,*** 607,927,102,274 [...]
```

and I thought, *"Gosh, the image 3DC3DC3DC3D caters only for* ***12*** *digits and* ***1E12*** *=* ***1,000,000,000,000*** *which has* ***13****, thus the image overflow. But the output count,* ***607,927,102,274****, is fully correct as per tables, so no problem."*

Thus, instead of changing the already sizeable image, and as ***Moebius(1E12)*** = ***0*** (because **1E12** is divisible my many squares *4*, for instance,) it's adding nothing to the tally, so the count for **1E12-1** must be the same value, **607,927,102,274**, and thus I simply posted the input as **1E12-1** to avoid the ungainly *IMAGE* overflow, fully expecting that the program would produce the exact same value as the one computed for **1E12** proper so I didn't bother to check it out by actually running it again.

Alas, as **J-F** discovered, it does *not*, possibly because of some rounding error, but the solution is quite simple because the only operations which can result in such errors are the square root (which is innocent) and divisions. A little examination reveals that the culprit is **IP(T/(K*K))** at *line 2*, and changing it to **(T DIV (K*K))**, like this:

```
  2 D=PRIM(N) @ IF NOT D THEN S=S+U^F*(T DIV (K*K)) ELSE IF MOD(N,D*D) THEN F=F+1 @ N=N/D @ GOTO 2
```

completely solves the problem *(177 bytes)*:

```
     1 DESTROY ALL @ INPUT T @ SETTIME 0 @ U=-1 @ S=T @ FOR K=2 TO SQR(T) @ N=K @ F=1
     2 D=PRIM(N) @ IF NOT D THEN S=S+U^F*(T DIV (K*K)) ELSE IF MOD(N,D*D) THEN F=F+1 @ N=N/D @ GOTO 2
     3 NEXT K @ DISP USING "2(3DC3DC3DC3D,2X),2(Z.8D,X),5DZ.2D";T,S,SQR(6*T/S),ABS(PI-RES),TIME

     >RUN
           ?1E12-1
                999,999,999,999  607,927,102,274 [...]
```

Probably something like this may be affecting **DavidM**'s program and the solution might possibly be along the same lines.

Best regards.
V.

PM WWW FIND EDIT QUOTE REPORT

28th March, 2023, 17:10 (This post was last modified: 29th March, 2023 04:22 by DavidM.) **Post: #29**

DavidM
Senior Member

Posts: 918
Joined: Dec 2013

RE: [VA] SRC #013 - Pi Day 2023 Special

> **Valentin Albillo Wrote:** (27th March, 2023 21:46)
>
> Probably something like this may be affecting **DavidM**'s program and the solution might possibly be along the same lines.

That was indeed the problem. I've updated the SysRPL code in post #22 to reflect a fix for the issue.

Unfortunately there is no built-in SysRPL operator for integer division with real/approximate arguments, so I couldn't use the same approach that Valentin was able to use.

I could have opted to refactor the code using ZINTs (exact integers) so as to use the supported integer division operator for those, but that would reduce the performance gains that I was seeking. A simple test showed about a 10% runtime hit going that route.

While there's no supported integer division operator that I could tap into, there *is* a **%MOD** operator that allowed me to "back into" the integer quotient by subtracting the remainder from N prior to dividing by i^2. Although somewhat convoluted, it works in this situation and only cost about a 2-3% penalty in runtime.

I'm still working on documenting the new runtimes, and will update post #22 when I've got the corrected data.

Edit: Runtimes have now been updated as well as the program code.

PM FIND QUOTE REPORT

29th March, 2023, 03:16 **Post: #30**

Valentin Albillo
Senior Member

Posts: 958
Joined: Feb 2015
Warning Level: 0%

RE: [VA] SRC #013 - Pi Day 2023 Special

.
Hi, all,

Adding to this:

> **Valentin Albillo Wrote:**
>
> Alas, as **J-F** discovered, it does *not*, possibly because of some rounding error, but the solution is quite simple because the only operations which can result in such errors are the square root (which is innocent) and divisions. A little examination reveals that the culprit is **IP(T/(K*K))** at *line 2*, and changing it to **(T DIV (K*K))**, like this: [...]

Indeed **IP(T/(K*K))** and **T DIV (K*K)**, which would appear at first sight to be equivalent, do really differ at times (though *very* rarely and for large values of **T**, it seems,) when the former's *rounding* does not match the latter's *truncation*.

A trivial program I wrote (relatively) quickly finds all mismatches for various very large integer **T** and for **K** from *2* to *IP(√T)* (i.e. ~ *one million* possible cases for the first eight values of **T** listed):

```
          T                # Mismatches            K
     -----------------------------------------------------------
     999,999,999,999      31 instances      2, 5, 8, 16, 20, ...
     999,999,999,998      19 instances      2, 3, 8, 20, 25, ...
     999,999,999,997      12 instances      3, 8, 25, 80, ...
     999,999,999,996       3 instances      3, 3125, 31250, ...
     999,999,999,995       3 instances      2, 3, 254
     999,999,999,994       1 instance       254
     999,999,999,993       1 instance       254
     999,999,999,992       0 instances       -
      99,999,999,999       0 instances       -
```

As you can see, for ***T*** = ***999,999,999,999*** there are **31 different instances** (in about a million) where `IP(T/(K*K))` differs from `T DIV (K*K)`, for ***K*** ranging from *1* to *IP(√T)*. The instances begin at ***K*** = **2** (*249,999,999,999* vs. *250,000,000,000*, respectively) and end at ***K*** = **500,000** (*3* vs. *4*, respectively).

Doing the same with ***T*** = ***999,999,999,998***, there's just ***19*** instances reported instead of *31*, and with ***T*** = ***999,999,999,997*** just ***12***. By the time ***T*** equals *999,999,999,995*, a mere ***3*** faulty instances remain (namely for *K* = *2*, *3* and *254*), then *999,999,999,994* and *999,999,999,993* have just the *one* mismatch *(in a million !)* and for *999,999,999,992* and below there seems to be *none*.

Also, as expected, running this small program for input values with less than 12 digits, say ***T*** = **99,999,999,999** instead, i.e. **1E11 - 1**, no instances of mismatches appear at all, and probably the same happens for all smaller ***T***.

V.

29th March, 2023, 17:42 (This post was last modified: 29th March, 2023 18:20 by EdS2.) **Post: #31**

EdS2
Senior Member
Posts: 517
Joined: Apr 2014

RE: [VA] SRC #013 - Pi Day 2023 Special

Right then, after a false start yesterday afternoon, resulting in a need for aspirin and a nap, I've managed to do a bit of coding. I chose J-F's simplest (first) submission as a template, and wanted to make use of sieve code from a previous discussion on Stardot. After a great deal of uglification and debugging, and using some more of J-F's ideas, I have the following (can run it here):

```
100 REM based on SRC13 by J-F Garnier (first attempt)
110 DATA 1000,12345,1E5,1E6
120 REPEAT:READ N%
130 T%=TIME
140 L%=SQR(N%)
150 REM start with a sieve of smallest prime factors
160 Z%=SQR(L%)
170 DIM V% L%+5
180 FORI%=1TOL%+1STEP4:V%!I%=0:NEXT
190 P%=2:REPEATFORI%=P%*P%TOL%STEPP%:IFV%?I%ELSEV%?I%=P%
200 NEXT:REPEATP%=P%+1:UNTILV%?P%=0
210 UNTILP%>Z%:IF P%>255 STOP
220 REM now the counting of the squarefree
230 S%=N%:FOR I%=2 TO L%
240 R%=I%:C%=-1:Q%=1
250 P%=V%?R%:IFP%ELSEP%=R%
260 IFP%=Q%M%=0ELSEIFP%=R%M%=C%ELSEC%=-C%:R%=R%DIVP%:Q%=P%:GOTO250
270 S%=S%+M%*(N%DIV(I%*I%)):I%=I%+1
```

```
280 R%=I%:C%=-1:Q%=1
290 P%=V%?R%:IFP%ELSEP%=R%
300 IFP%=Q%M%=0ELSEIFP%=R%M%=C%ELSEC%=-C%:R%=R%DIVP%:Q%=P%:GOTO290
310 S%=S%+M%*(N%DIV(I%*I%)):I%=I%+2
320 R%=I%:C%=-1:Q%=1
330 P%=V%?R%:IFP%ELSEP%=R%
340 IFP%=Q%M%=0ELSEIFP%=R%M%=C%ELSEC%=-C%:R%=R%DIVP%:Q%=P%:GOTO330
350 S%=S%+M%*(N%DIV(I%*I%))
360 NEXT
370 T%=TIME-T%
380 PRINT N%" "S%" "T%/100"s"
390 U.FA.:END
```

The timings and counts are as follows:

```
       1000        608       0.33s
      12345       7503       1.20s
     100000      60794       3.59s
     567890     345237       8.98s
    1000000     607926      12.07s
```

I was able to push this as far as 7.8E8 with result 474183174, which I can't readily check, using a faster unrealistic emulation mode. (Can't get Wolfram to give me large results - any tips?)

Other than that, all the above I ran in accurate emulation of a BBC Micro, a 2MHz 6502 machine with 32k RAM and a rather good 16k Basic. There are models with more available RAM and faster CPUs and I should be able to get to 1E9 one way or another, and probably higher.

The only peculiarity of BBC Basic which you might need to understand is the use of DIM, ? and ! to allocate and directly access RAM. Fairly obviously, the use of the % sign is to denote integer variables.

Thanks to everyone for the materials and discussion! And to Valentin for the challenge. I've posted over on Stardot too, of course.

I note that some vintage HP desktops offer a Basic, it might be interesting to see what they can do.

EMAIL PM FIND QUOTE REPORT

2nd April, 2023, 01:39 **Post: #32**

Valentin Albillo
Senior Member

Posts: 958
Joined: Feb 2015
Warning Level: 0%

RE: [VA] SRC #013 - Pi Day 2023 Special

Hi, all,

To end this thread, there's the subject of finding values of ***N*** which result in an approximation to π much closer than what would be statistically expected. Of course, the larger ***N***, the better the appoximation on the long run, but perhaps there are values of ***N*** which defy the odds and result in unexpectedly close approximations.

To settle the matter, this *179-byte **4**-liner* finds them up to a given maximum ***N*** very, very quickly, listing ***all*** successive ***record-breakers***:

```
1  DESTROY ALL @ INPUT N @ SETTIME 0 @ INTEGER M(N) @ MAT M=CON @ P=2 @ WHILE P<=N @ S=P*P
2  FOR K=S TO N STEP S @ M(K)=0 @ NEXT K @ P=FPRIM(P+1) @ END WHILE @ V=10 @ S=7
3  FOR T=11 TO N @ S=S+(M(T)#0) @ X=SQR(6*T/S) @ Y=ABS(PI-X) @ IF Y<V THEN V=Y @ DISP T;X;V
4  NEXT T @ DISP "OK";TIME
```

Let's find all record-breakers for ***N*** up to ***30,000*** *(requires ~ 90 Kb of RAM):*

```
>RUN ->  ? 30000

     N        Pi Approx.        |Error|
  -------- ----------------------------
 (smallest values, which result in irrelevant approximations, not listed)
      28    3.14362099197    0.00202833838
     153    3.14180962853    0.00021697494
     426    3.14145282771    0.00013982588
```

```
      862   3.14169206124   0.00009940765
      931   3.14153751379   0.00005513980
      936   3.14164722334   0.00005456975
      982   3.14155164428   0.00004100931
     1033   3.14156437967   0.00002827392
     1061   3.14161860223   0.00002594864
     1135   3.14158641730   0.00000623629
     1186   3.14159601478   0.00000336119
     2094   3.14159185312   0.00000080047
     5147   3.14159305181   0.00000039822
     5374   3.14159277156   0.00000011797
     7241   3.14159270516   0.00000005157
    14709   3.14159260812   0.00000004547
    25684   3.14159262180   0.00000003179
```

OK *timing (**go71b**: 23.37", **Emu71/Win**: 3.06", physical **HP-71B**: 49' 51")*

Adding up more *RAM* to a virtual/physical **HP-71B**, we can go further, say up to ***N*** = ***100,000*** using *~300 Kb*. We obtain the following additional record-breakers:

```
    65623   3.14159266166   0.00000000807
    67490   3.14159265758   0.00000000399
    89440   3.14159265330   0.00000000029
```

The above list of record-breakers would seem to end the discussion, but there's something which doesn't look good, the fact that there are several series of them which feature very close ***N*** and *errors*, such as these:

```
931 3.14153751379 0.00005513980
936 3.14164722334 0.00005456975
982 3.14155164428 0.00004100931
```

looking almost redundant, as they're not that different.

It would seem preferable to find and list only those record-breakers which are a significant improvement over their predecessors, where *"significant improvement"* obeys some suitable criterium, and this ***6**-liner* implements just that:

```
1  DESTROY ALL @ INPUT N @ SETTIME 0 @ INTEGER M(N) @ MAT M=CON @ P=2 @ WHILE P<=N @ S=P*P
2  FOR K=S TO N STEP S @ M(K)=0 @ NEXT K @ P=FPRIM(P+1) @ END WHILE @ U=1 @ V=10 @ Q=1 @ S=7
3  FOR T=11 TO N @ S=S+(M(T)#0) @ X=SQR(6*T/S) @ Y=ABS(PI-X) @ IF Y>=V THEN 6
4  W=T/Q @ Z=V/Y @ H=(Z-1)/(W-1) @ IF H<3 THEN 6
5  Q=T @ V=Y @ DISP T;X;V;FNR(Z,3);FNR(W,3);FNR(H,3)
6  NEXT T @ DISP "OK";TIME @ DEF FNR(N,D)=IROUND(10^D*N)/10^D

>RUN ->  ? 30000

       N        Pi Approx.        |Error|       Epre/E  N/Npre    Merit
    -------- ---------------------------------------------------------
   (smallest values, which result in irrelevant approximations, not listed)
     1135   3.14158641730   0.00000623629  325.248  40.536    8.201
     1186   3.14159601478   0.00000336119    1.855   1.045   19.036
     2094   3.14159185312   0.00000080047    4.199   1.766    4.178
     5374   3.14159277156   0.00000011797    6.785   2.566    3.693
     7241   3.14159270516   0.00000005157    2.288   1.347    3.706
```

OK *timing (**go71b**: 25.4", **Emu71/Win**: 3.33", physical **HP-71B**: 54' 11")*

Again, using more *RAM* we can search up to, say, ***N*** = ***100,000*** and we get two more:

```
    67490   3.14159265758   0.00000000399    2.023   1.028   35.942
    89440   3.14159265330   0.00000000029   13.759   1.325   39.229
```

Why do we exclude, e.g. **5,147** from the list ? Because **5,147** 's error is ***2.010x*** *smaller* than its predecessor's, **2,094**, but at the cost of ***2.458x*** *larger* ***N***, so it's no real improvement over **2,094** and doesn't really pay off having to waste so much more time for such a meager improvement. Thus, its *merit factor* is too low, doesn't meet the suitability criterium and is excluded from the list; these are the relevant numbers:

```
2094   3.14159185312   0.00000080047   4.199   1.766   4.178
5147   3.14159305181   0.00000039822   2.010   2.458   0.693
```

and the same goes for all other omitted values. Only the ones which result in a marked improvement as per the criterium are deemed worthy to be listed.

Finally, we can do the search for ***N*** much greater than *100,000*, up to ***1,500,000*** and beyond by using *boolean* operators and variables and I've written such a version of this program implementing them, but that's left to the reader as an exercise ... 😀

Thanks to all of you who participated, much appreciated and I hope you enjoyed it all.

V.

PM WWW FIND EDIT QUOTE REPORT

9th April, 2023, 00:41 **Post: #33**

Valentin Albillo
Senior Member

Posts: 958
Joined: Feb 2015
Warning Level: 0%

RE: [VA] SRC #013 - Pi Day 2023 Special

Hi again !

I thought I was done with this thread for good but while converting it to a *PDF* document for uploading it to my *HP* site's *HP Calculator Challenges* section, I realized there was a bit of *unfinished business* which had eluded me so far.

Namely, I said in Post #15 the following, I quote:

> "As for the ***Möbius function*** *{μ(N) henceforth}*, it's a very important *number-theoretical* function which is easily computed in various ways, like this simple **HP-71B** user-defined function **FNM** (which should be placed at the end of any program using it): *{requires the **JPC ROM**}*
>
> ```
> 1  DEF FNM(N) @ IF N=1 THEN FNM=1 @ END ELSE F=1
> 2  D=PRIM(N) @ IF NOT D THEN FNM=(-1)^F ELSE IF MOD(N,D*D) THEN F=F+1 @ N=N/D @ GOTO 2
>
>    >FOR N=96673 TO 96686 @ FNM(N); @ NEXT N @@  ►  1 1 0 0 0 0 0 0 1 1 1 0 -1 -1 [...]
> ```
>
> *[... Good and Churchhouse said ...] 'Thus the Möbius sequence contains **arbitrarily long runs of zeros**, but these long runs presumably occur extremely rarely.'*
>
> I showed above such a run of zeros from *N = 96675* to *96680*, *six* consecutive zeros in all. ***It would make for an interesting challenge to locate longer runs***."

So, here I address this remaining sub-challenge by creating an *HP-71B* program to find the first values of ***N*** which are the start of a run of *1, 2, 3, ...,* consecutive zeros of *μ(N)*. Specifically, this *169-byte **4**-liner* finds them up to a given maximum ***N*** very quickly: *{requires the **JPC ROM**}*

```
1  DESTROY ALL @ INPUT N @ SETTIME 0 @ INTEGER M(N) @ P=2 @ WHILE P<=N @ S=P*P
2  FOR K=S TO N STEP S @ M(K)=1 @ NEXT K @ P=FPRIM(P+1) @ END WHILE @ Z=0 @ U=0
3  FOR K=1 TO N @ IF M(K) THEN Z=Z+1 ELSE IF Z>U THEN U=Z @ DISP U;K-Z @ Z=0 ELSE Z=0
4  NEXT K @ DISP "OK";TIME
```

- Lines *1-2* perform the initialization and identify the zeros of ***μ(N)***, while line *3* contains all the logic to locate the start of the first run of *1, 2, 3, ...,* consecutive zeros.

Let's find the first initial ***N*** for ***L*** = *1, 2, 3, ...,* consecutive zeros for ***N*** up to ***25,000*** *(requires ~ 75 Kb of RAM):*

```
>RUN ->  ? 25000

L    Initial N
----------------
1           4
2           8
3          48
4         242
5         844
6       22020
```

OK *timing (**go71b**: 10.60", **Emu71/Win**: 1.39", physical **HP-71B**: 22' 37")*

This means that e.g. for the case ***L*** = ***5*** we have that *μ(844), μ(845), μ(846), μ(847)* and *μ(848)* are all *zero*, thus the first run of *5 consecutive zeros of μ(N)* starts at ***N*** = *844*. This also means that all five numbers in the run are divisible by some *square*, namely:

844 = 2^2 *x 211;* ***845*** = *5 x* 13^2*;* ***846*** = *2 x* 3^2 *x 47;* ***847*** = *7 x* 11^2*;* ***848*** = 2^4 *x 53*

Adding up more *RAM* and/or using boolean arrays and operations we'd be able to extend the search up to ***N*** = ***1,200,000***, say, which would allow us to find two additional runs of lengths ***7*** and **8**, respectively:

```
     7      217070
     8     1092747
```

Longer runs would require much more complex programming and longer execution times.

V.