In[155]:=
```
2 - 2
```

Out[155]=
```
0
```

Two Unsolved Recursion Problems

Nicholas Wheeler
August 31, 2020

The Lychrel Number Problem

Some of my recent work has involved aspects of the theory of orthogonal polynomials. Central to that theory are 2nd order differential equations of Sturm-Liouville form

$$L\,y(x) = \lambda\, w(x)\, y(x)$$

where

$$L \equiv \partial_x p(x)\, \partial_x + q(x)$$

I was led to fresh appreciation of the fact S-L theory owes much of its utility to the circumstance that Sturm-Liouville operators L are *palindromic*. Relatedly, if $\mathbb{A}, \mathbb{B}, \mathbb{C}$ are symmetric matrices then

$$(\mathbb{ABC})^{\mathsf{T}} = \mathbb{CBA}$$

so only palindromic products of such matrices are symmetric. Looking to this fascinating article

In[172]:=
```
Hyperlink["Palindromes", "https://en.wikipedia.org/wiki/Palindrome"]
```

Out[172]=
```
Palindromes
```

I became aware of the Lychrel Number Problem, pose by Wade Van Landingham ("Lynchrel" is a crude anagram of "Cheryl," his girlfriend at the time), which asks (of base 10 digits): Does the "reverse and add" iteration always produce palindromic number? "Lychrel numbers" are numbers that fail to do so, so the question is: Do Lychrel numbers exist?

In[173]:=
```
Hyperlink["Lychrel Number Problem",
 "https://en.wikipedia.org/wiki/Lychrel_number"]
```

Out[173]=
```
Lychrel Number Problem
```

The following command generates the sequence that grows from n:

```
NestWhileList[(# + IntegerReverse[#]) &, n, # ≠ IntegerReverse[#] &]
```

I give some examples:

In[174]:=
```
NestWhileList[(# + IntegerReverse[#]) &, 59, # ≠ IntegerReverse[#] &]
```

Out[174]=
```
{59, 154, 605, 1111}
```

That string of 4 digits resulted from 4 - 1 = 3 iterations.

In[180]:=
```
NestWhileList[(# + IntegerReverse[#]) &, 89, # ≠ IntegerReverse[#] &]
Length[%] - 1
PalindromeQ[Last[%%]]
```

Out[180]= {89, 187, 968, 1837, 9218, 17 347, 91 718, 173 437, 907 808, 1 716 517, 8 872 688, 17 735 476, 85 189 247, 159 487 405, 664 272 356, 1 317 544 822, 3 602 001 953, 7 193 004 016, 13 297 007 933, 47 267 087 164, 93 445 163 438, 176 881 317 877, 955 594 506 548, 1 801 200 002 107, 8 813 200 023 188}

Out[181]= 24

Out[182]= True

The neighboring numbers 88 and 90 are (or immediately become) palindromic, so Length[n] is quite an irregular function. The smallest potential Lychrel number is 196, which has been iterated more than 1,000,000 times without producing a palindromic number.

"Does a Lychrel number exist?" is a question with a 1-bit yes-or-no answer that has thus far yielded to neither analytical nor brute force methods,

The 3x + 1 Problem

All of which brought back to mind a Reed Physics Seminar—presented by Richard Crandall soon after he joined the Reed physics faculty—based upon his paper "On the 3x + 1 Problem" (Mathematics of Computation, **32**, 1281-1292 (1978)), a paper that has generated a considerable literature:

In[192]:=
```
Hyperlink["Crandall's 3x + 1 paper",
 "https://www.semanticscholar.org/paper/On-the-%22-3-jc-%2B-1-%22-Problem-Crandall
   /0960b14d916ac02442b13571a2a41a3ba5602306"]
```

Out[192]= Crandall's 3x + 1 paper

The paper treats aspects of another unsolved recursive problem with 1-bit yes-or-no answer:

In[189]:=
```
Hyperlink["Collatz Conjecture",
 "https://en.wikipedia.org/wiki/Collatz_conjecture"]
```

Out[189]= Collatz Conjecture

It was conjectured by Lothar Collatz (1937—remarkably, since this was before the age of computers) that for all integers n the recursive scheme

$n_{odd} \longrightarrow 3\, n_{odd} + 1$

$n_{even} \longrightarrow n_{even} / 2$

invariably produces a one, and thereafter cycles $1 \longrightarrow 4 \longrightarrow 2 \longrightarrow 1$. I use the following command

```
NestWhileList[If[OddQ[#], 3 # + 1, # / 2] &, n, #^2 ≠ # &]
```

to generate some examples:

NOTE: N - 1 iterations are required to produce an n-digit sequence, as previously remarked.

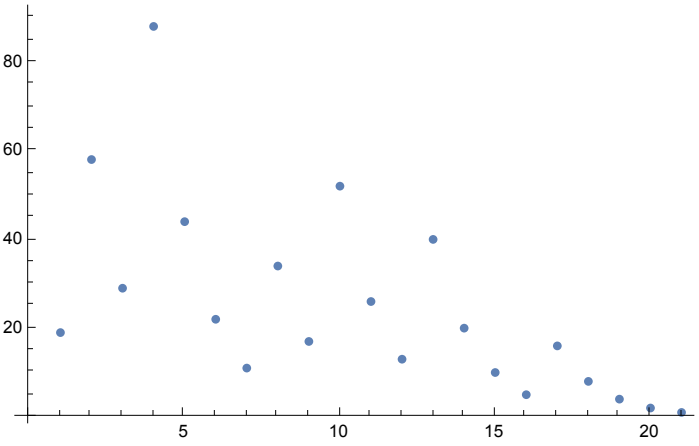
In[214]:=
```
NestWhileList[If[OddQ[#], 3 # + 1, # / 2] &, 19, #^2 ≠ # &]
Length[%] - 1
Max[%%]
ListPlot[%%%]
```

Out[214]= {19, 58, 29, 88, 44, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1}

Out[215]= 20

Out[216]= 88

Out[217]=



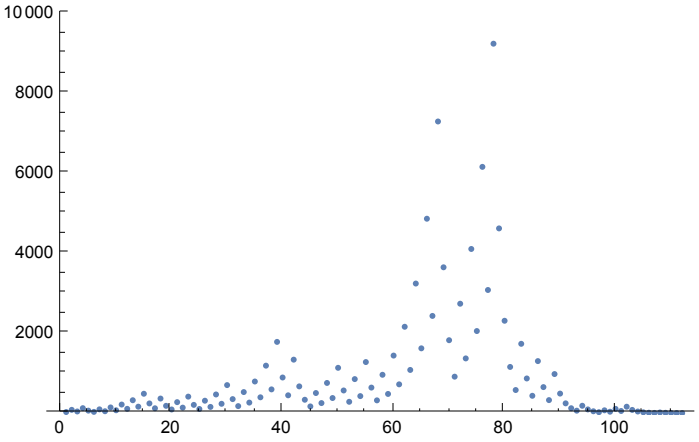
In[210]:=
```
NestWhileList[If[OddQ[#], 3 # + 1, # / 2] &, 27, #^2 ≠ # &]
Length[%] - 1
Max[%%]
ListPlot[%%%, PlotRange → 10 000]
```

Out[210]= {27, 82, 41, 124, 62, 31, 94, 47, 142, 71, 214, 107, 322, 161, 484, 242, 121, 364, 182, 91, 274, 137, 412, 206, 103, 310, 155, 466, 233, 700, 350, 175, 526, 263, 790, 395, 1186, 593, 1780, 890, 445, 1336, 668, 334, 167, 502, 251, 754, 377, 1132, 566, 283, 850, 425, 1276, 638, 319, 958, 479, 1438, 719, 2158, 1079, 3238, 1619, 4858, 2429, 7288, 3644, 1822, 911, 2734, 1367, 4102, 2051, 6154, 3077, 9232, 4616, 2308, 1154, 577, 1732, 866, 433, 1300, 650, 325, 976, 488, 244, 122, 61, 184, 92, 46, 23, 70, 35, 106, 53, 160, 80, 40, 20, 10, 5, 16, 8, 4, 2, 1}

Out[211]= 111

Out[212]= 9232

Out[213]=



The output is seen there to be highly (if irregularly) patterned.

A small change of n can produce a radically changed result.

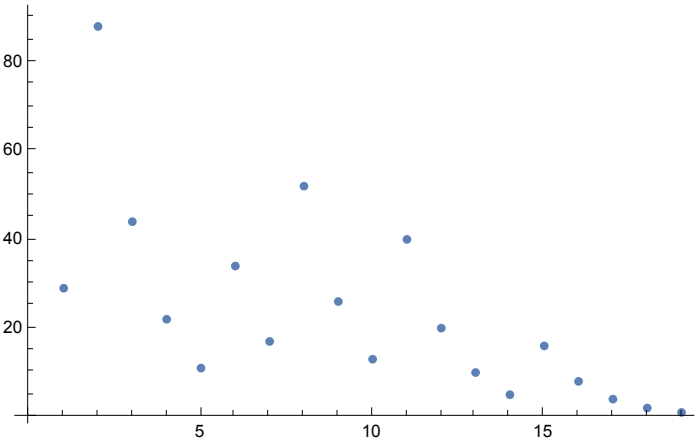
In[202]:=
```
NestWhileList[If[OddQ[#], 3 # + 1, # / 2] &, 29, #^2 ≠ # &]
Length[%] - 1
Max[%%]
ListPlot[%%%]
```

Out[202]= {29, 88, 44, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1}

Out[203]= 18

Out[204]= 88

Out[205]=



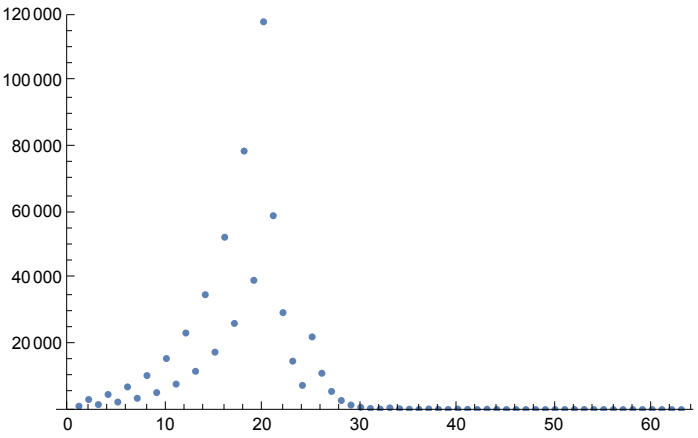
In[247]:=
```
NestWhileList[If[OddQ[#], 3 # + 1, # / 2] &, 2^10 - 1, #^2 ≠ # &]
Length[%] - 1
Max[%%]
ListPlot[%%%, PlotRange → 120 000]
ListPlot[%%%%, PlotRange → 500]
```

Out[247]= {1023, 3070, 1535, 4606, 2303, 6910, 3455, 10 366, 5183, 15 550, 7775, 23 326, 11 663, 34 990, 17 495, 52 486, 26 243, 78 730, 39 365, 118 096, 59 048, 29 524, 14 762, 7381, 22 144, 11 072, 5536, 2768, 1384, 692, 346, 173, 520, 260, 130, 65, 196, 98, 49, 148, 74, 37, 112, 56, 28, 14, 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1}

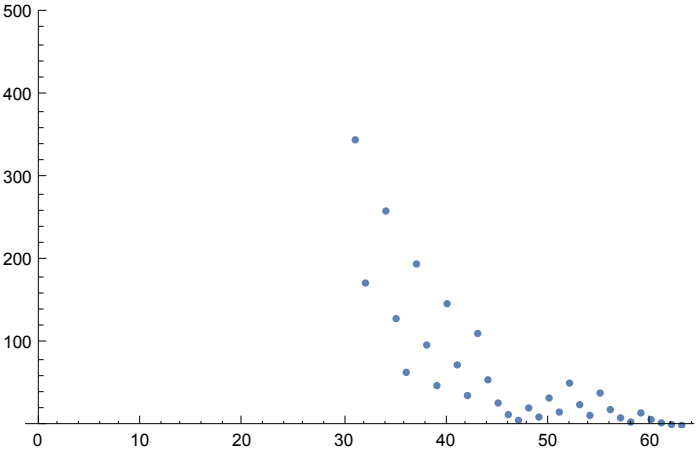
Out[248]= 62

Out[249]= 118 096

Out[250]=



Out[251]=



The following example shows one characteristic pattern to arise from erasing powers of 2:

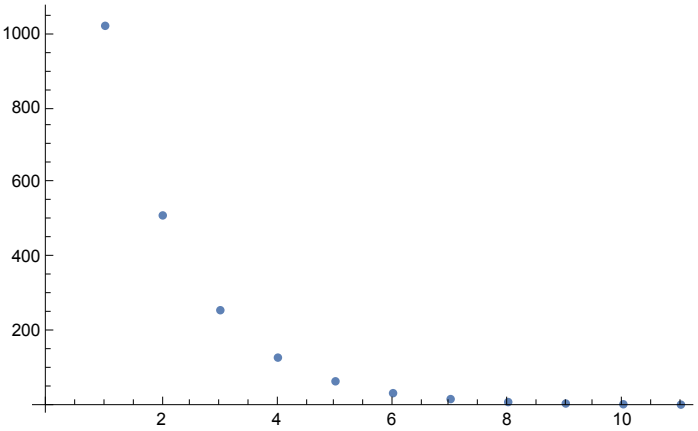
In[226]:=
```
NestWhileList[If[OddQ[#], 3 # + 1, # / 2] &, 2^10, #^2 ≠ # &]
Length[%] - 1
Max[%%]
ListPlot[%%%]
```

Out[226]= {1024, 512, 256, 128, 64, 32, 16, 8, 4, 2, 1}

Out[227]= 10

Out[228]= 1024

Out[229]=



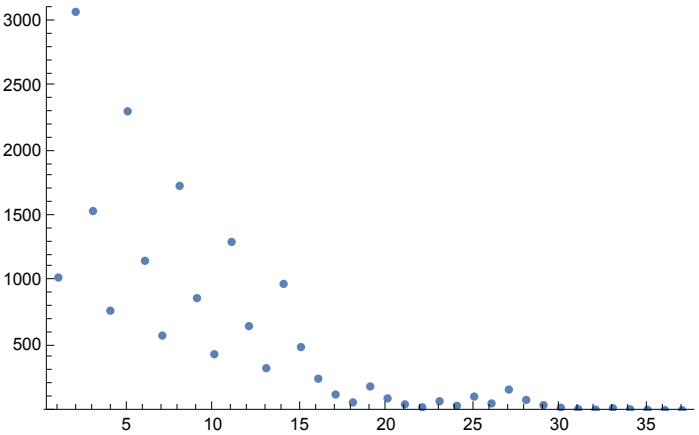
In[234]:=
```
NestWhileList[If[OddQ[#], 3 # + 1, # / 2] &, 2^10 + 1, #^2 ≠ # &]
Length[%] - 1
Max[%%]
ListPlot[%%%, PlotRange → 3100]
```

Out[234]= {1025, 3076, 1538, 769, 2308, 1154, 577, 1732, 866, 433, 1300, 650, 325, 976, 488, 244, 122, 61, 184, 92, 46, 23, 70, 35, 106, 53, 160, 80, 40, 20, 10, 5, 16, 8, 4, 2, 1}

Out[235]= 36

Out[236]= 3076

Out[237]=



The slight adjustment $n = 2^{10} \longrightarrow n = 2^{10} + 1$ has produced a major change in the iterative sequence. The effect of the adjustment $n = 2^{10} + 1 \longrightarrow n = 2^{10} - 1$ is even more dramatic.

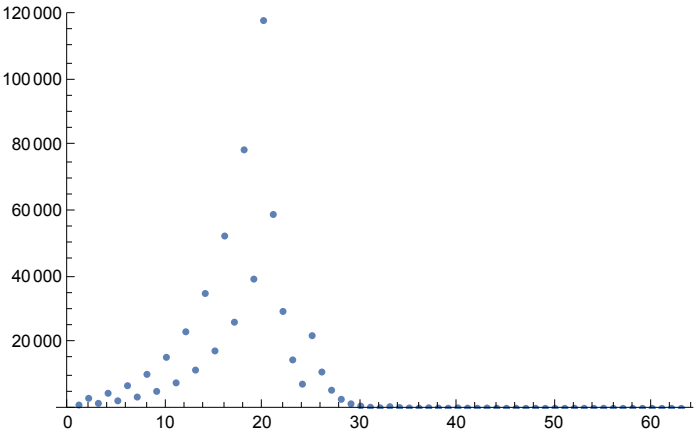
In[256]:=
```
NestWhileList[If[OddQ[#], 3 # + 1, # / 2] &, 2^10 - 1, #^2 ≠ # &]
Length[%] - 1
Max[%%]
ListPlot[%%%, PlotRange → 120 000]
```

Out[256]= {1023, 3070, 1535, 4606, 2303, 6910, 3455, 10 366, 5183, 15 550, 7775, 23 326, 11 663, 34 990, 17 495, 52 486, 26 243, 78 730, 39 365, 118 096, 59 048, 29 524, 14 762, 7381, 22 144, 11 072, 5536, 2768, 1384, 692, 346, 173, 520, 260, 130, 65, 196, 98, 49, 148, 74, 37, 112, 56, 28, 14, 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1}

Out[257]= 62

Out[258]= 118 096

Out[259]=



The large number n = 10^6 terminates after only 152 iterations

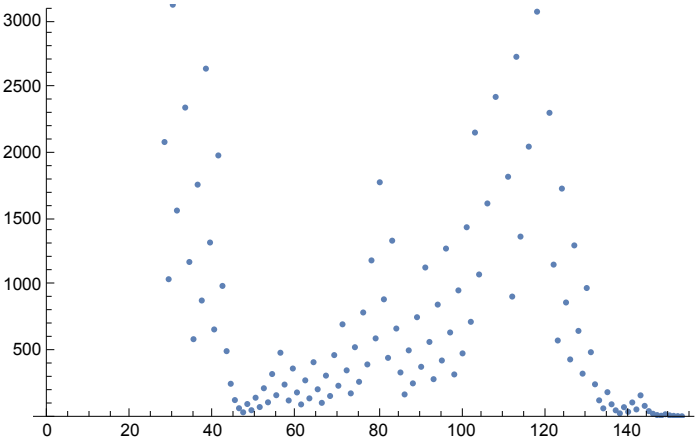
In[238]:=
```
NestWhileList[If[OddQ[#], 3 # + 1, # / 2] &, 10^6, #^2 ≠ # &]
Length[%] - 1
Max[%%]
ListPlot[%%%, PlotRange → 3100]
```

Out[238]= {1 000 000, 500 000, 250 000, 125 000, 62 500, 31 250, 15 625, 46 876, 23 438, 11 719, 35 158, 17 579, 52 738, 26 369, 79 108, 39 554, 19 777, 59 332, 29 666, 14 833, 44 500, 22 250, 11 125, 33 376, 16 688, 8344, 4172, 2086, 1043, 3130, 1565, 4696, 2348, 1174, 587, 1762, 881, 2644, 1322, 661, 1984, 992, 496, 248, 124, 62, 31, 94, 47, 142, 71, 214, 107, 322, 161, 484, 242, 121, 364, 182, 91, 274, 137, 412, 206, 103, 310, 155, 466, 233, 700, 350, 175, 526, 263, 790, 395, 1186, 593, 1780, 890, 445, 1336, 668, 334, 167, 502, 251, 754, 377, 1132, 566, 283, 850, 425, 1276, 638, 319, 958, 479, 1438, 719, 2158, 1079, 3238, 1619, 4858, 2429, 7288, 3644, 1822, 911, 2734, 1367, 4102, 2051, 6154, 3077, 9232, 4616, 2308, 1154, 577, 1732, 866, 433, 1300, 650, 325, 976, 488, 244, 122, 61, 184, 92, 46, 23, 70, 35, 106, 53, 160, 80, 40, 20, 10, 5, 16, 8, 4, 2, 1}

Out[239]= 152

Out[240]= 1 000 000

Out[241]=



Here the first 26 data points are too large to appear on the display.

…while $10^6 + 1$ terminates after only 113 iterations:

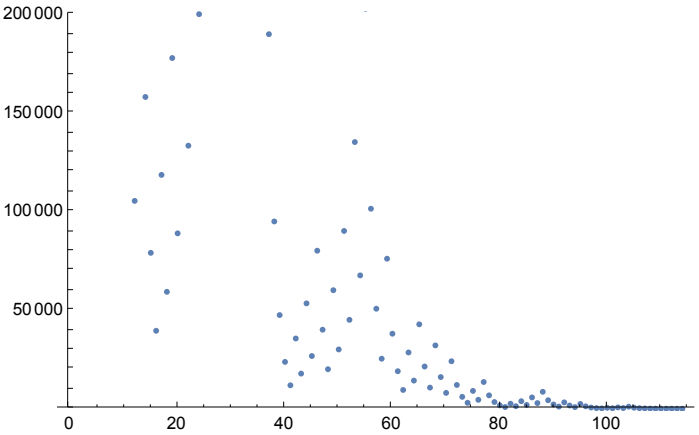
In[264]:=
```
NestWhileList[If[OddQ[#], 3 # + 1, # / 2] &, 10^6 + 1, #^2 ≠ # &]
Length[%] - 1
Max[%%]
ListPlot[%%%, PlotRange → 200 000]
```

Out[264]= {1 000 001, 3 000 004, 1 500 002, 750 001, 2 250 004, 1 125 002, 562 501, 1 687 504, 843 752, 421 876, 210 938, 105 469, 316 408, 158 204, 79 102, 39 551, 118 654, 59 327, 177 982, 88 991, 266 974, 133 487, 400 462, 200 231, 600 694, 300 347, 901 042, 450 521, 1 351 564, 675 782, 337 891, 1 013 674, 506 837, 1 520 512, 760 256, 380 128, 190 064, 95 032, 47 516, 23 758, 11 879, 35 638, 17 819, 53 458, 26 729, 80 188, 40 094, 20 047, 60 142, 30 071, 90 214, 45 107, 135 322, 67 661, 202 984, 101 492, 50 746, 25 373, 76 120, 38 060, 19 030, 9515, 28 546, 14 273, 42 820, 21 410, 10 705, 32 116, 16 058, 8029, 24 088, 12 044, 6022, 3011, 9034, 4517, 13 552, 6776, 3388, 1694, 847, 2542, 1271, 3814, 1907, 5722, 2861, 8584, 4292, 2146, 1073, 3220, 1610, 805, 2416, 1208, 604, 302, 151, 454, 227, 682, 341, 1024, 512, 256, 128, 64, 32, 16, 8, 4, 2, 1}

Out[265]= 113

Out[266]= 3 000 004

Out[267]=



NOTE that if a Collatz sequence contained a repeated digit N > 1 it would cycle, and then assuredly never produce a terminal 1. So a weak form of the Collatz conjecture is that such sequences contain no repeated digits.

Mathematicians have discovered a large collection of initial n-values that exhaust the capacity of computers without terminating, but that does not *prove* that they *never* terminate. Paul Erdös—who was of the opinion that "Mathematics may not be ready for such problems"—offered a $500 prize for a proof of the Collatz conjecture. John Conway proved (1972) that a natural generalization of the Collatz problem can be cast as a halting problem, from which it follows that the problem is algorithmically undecidable.

ADDENDUM: Peter Renz (3 September) has brought to my attention the fact that Jeffrey Lagarias has published a collection of papers relating to the Collatz problem: *The Ultimate Challenge: The 3x + 1*

Problem (AMS, 2010), which provides an annotated bibliography (1963-1999). Günther J. Wirsching published *The Dynamical System Generated by the 3x + 1 Function* (Springer, 1998).