Iterated Function Project
by Samip Karki

In this project, I will explore sequences obtained by iterating the following variant of the Collatz function:

col5(n) = {3 n + 5 if n is odd
{n/2 if n is even

Preliminary Code

```
In[]:= col5[n_?OddQ] := 3 n + 5
       col5[n_?EvenQ] := n / 2

In[]:= (*Ive changed makeSeq so it stops as soon as there is a repeat in the list*)

In[]:= makeSeq[func_, start_, numIter_] := Module[{curr = start, seq = {start}},
         Do[
          (* compute the next value and append to the sequence *)
          curr = func[curr];
          AppendTo[seq, curr];
          If[Count[seq, curr] == 2, Break[]],

          {numIter} (* repeat numIter times *)
         ];

         (* return the sequence *)
         seq
        ]
```

1. What cycle (s) do you observe when in the trajectories of col5 (n) for positive integers n?

```
In[]:= Count[makeSeq[col5, 10, 100], 10]

Out[]= 2
```

```
In[∘]:= Table[
          makeSeq[col5, n, 20], {n, 1, 10}] // TableForm
Out[∘]//TableForm=
        1     8     4     2     1
        2     1     8     4     2
        3     14    7     26    13    44    22    11    38    19    62    31    98    49    152
        4     2     1     8     4
        5     20    10    5
        6     3     14    7     26    13    44    22    11    38    19    62    31    98    49
        7     26    13    44    22    11    38    19    62    31    98    49    152   76    38
        8     4     2     1     8
        9     32    16    8     4     2     1     8
        10    5     20    10
```

From initial inspection, there is a cycle 8, 4, 2, 1 and another cycle 20, 10, 5. Another cycle from 62, 31, 98, 49, 152, 76, 38, 19

```
In[∘]:=
```

I will try to get a better understanding of what kind of cycles there are:

The module below does the following:

Input a list, and then it will check each value of the list until it finds a repeated value. It will return two values, {length of the list until the repeated number AKA the stopping time, { The loop of repeating numbers}}. If there is no loop, then it will return Null.

```
In[∘]:= checkcycle[list_] := Module[{listb = {}, num, a, b},
          Do[
           num = list[[n]];
           If[MemberQ[listb, num],
            (*If True, do all this stuff*)
            AppendTo[listb, num];
            {a, b} = Flatten[Position[listb, num]];
            Return[{Length[listb],

              Table[
               listb[[n]],
               {n, a, b}]}];

            Break[],
            (*If False, do all this stuff*)
            AppendTo[listb, num]
           ]

           , {n, 1, Length[list]}]

         ]
```

```
In[]:= testlist = makeSeq[col5, 4, 5]
Out[]= {4, 2, 1, 8, 4}

In[]:= x = checkcycle[testlist]
Out[]= {5, {4, 2, 1, 8, 4}}

In[]:= checkcycle[Range[5]]
```

Looks like it is working as intended

I can use the checkcycle[ ] module to learn about what kind of cycles there are, because unlike the normal Collatz conjecture, there is more than 1 cycle.

```
In[]:=
```

The code below loops checks if there is a loop in the first 100 iterations of the col5 sequence with starting values form 1 to 100. First it prints n, then prints the output of checkcycle

```
In[]:= Do[Print[n]; Print[checkcycle[makeSeq[col5, n, 100]]]
        , {n, 1, 100}]

1
{5, {1, 8, 4, 2, 1}}
2
{5, {2, 1, 8, 4, 2}}
3
{17, {38, 19, 62, 31, 98, 49, 152, 76, 38}}
4
{5, {4, 2, 1, 8, 4}}
5
{4, {5, 20, 10, 5}}
6
{18, {38, 19, 62, 31, 98, 49, 152, 76, 38}}
7
{15, {38, 19, 62, 31, 98, 49, 152, 76, 38}}
8
{5, {8, 4, 2, 1, 8}}
9
{8, {8, 4, 2, 1, 8}}
10
{4, {10, 5, 20, 10}}
11
```

```
{10, {38, 19, 62, 31, 98, 49, 152, 76, 38}}

12

{19, {38, 19, 62, 31, 98, 49, 152, 76, 38}}

13

{13, {38, 19, 62, 31, 98, 49, 152, 76, 38}}

14

{16, {38, 19, 62, 31, 98, 49, 152, 76, 38}}

15

{9, {20, 10, 5, 20}}

16

{6, {8, 4, 2, 1, 8}}

17

{19, {38, 19, 62, 31, 98, 49, 152, 76, 38}}

18

{9, {8, 4, 2, 1, 8}}

19

{9, {19, 62, 31, 98, 49, 152, 76, 38, 19}}

20

{4, {20, 10, 5, 20}}

21

{22, {38, 19, 62, 31, 98, 49, 152, 76, 38}}

22

{11, {38, 19, 62, 31, 98, 49, 152, 76, 38}}

23

{9, {23, 74, 37, 116, 58, 29, 92, 46, 23}}

24

{20, {38, 19, 62, 31, 98, 49, 152, 76, 38}}

25

{7, {20, 10, 5, 20}}

26

{14, {38, 19, 62, 31, 98, 49, 152, 76, 38}}

27

{35, {38, 19, 62, 31, 98, 49, 152, 76, 38}}

28

{17, {38, 19, 62, 31, 98, 49, 152, 76, 38}}

29
```

```
{9, {29, 92, 46, 23, 74, 37, 116, 58, 29}}

30

{10, {20, 10, 5, 20}}

31

{9, {31, 98, 49, 152, 76, 38, 19, 62, 31}}

32

{7, {8, 4, 2, 1, 8}}

33

{17, {38, 19, 62, 31, 98, 49, 152, 76, 38}}

34

{20, {38, 19, 62, 31, 98, 49, 152, 76, 38}}

35

{18, {20, 10, 5, 20}}

36

{10, {8, 4, 2, 1, 8}}

37

{9, {37, 116, 58, 29, 92, 46, 23, 74, 37}}

38

{9, {38, 19, 62, 31, 98, 49, 152, 76, 38}}

39

{28, {38, 19, 62, 31, 98, 49, 152, 76, 38}}

40

{5, {20, 10, 5, 20}}

41

{10, {8, 4, 2, 1, 8}}

42

{23, {38, 19, 62, 31, 98, 49, 152, 76, 38}}

43

{33, {38, 19, 62, 31, 98, 49, 152, 76, 38}}

44

{12, {38, 19, 62, 31, 98, 49, 152, 76, 38}}

45

{21, {20, 10, 5, 20}}

46

{9, {46, 23, 74, 37, 116, 58, 29, 92, 46}}

47
```

```
{23, {38, 19, 62, 31, 98, 49, 152, 76, 38}}
48
{21, {38, 19, 62, 31, 98, 49, 152, 76, 38}}
49
{9, {49, 152, 76, 38, 19, 62, 31, 98, 49}}
50
{8, {20, 10, 5, 20}}
51
{16, {92, 46, 23, 74, 37, 116, 58, 29, 92}}
52
{15, {38, 19, 62, 31, 98, 49, 152, 76, 38}}
53
{13, {8, 4, 2, 1, 8}}
54
{36, {38, 19, 62, 31, 98, 49, 152, 76, 38}}
55
{16, {20, 10, 5, 20}}
56
{18, {38, 19, 62, 31, 98, 49, 152, 76, 38}}
57
{15, {38, 19, 62, 31, 98, 49, 152, 76, 38}}
58
{9, {58, 29, 92, 46, 23, 74, 37, 116, 58}}
59
{36, {38, 19, 62, 31, 98, 49, 152, 76, 38}}
60
{11, {20, 10, 5, 20}}
61
{26, {38, 19, 62, 31, 98, 49, 152, 76, 38}}
62
{9, {62, 31, 98, 49, 152, 76, 38, 19, 62}}
63
{14, {74, 37, 116, 58, 29, 92, 46, 23, 74}}
64
{8, {8, 4, 2, 1, 8}}
65
```

```
{11, {20, 10, 5, 20}}
66
{18, {38, 19, 62, 31, 98, 49, 152, 76, 38}}
67
{31, {38, 19, 62, 31, 98, 49, 152, 76, 38}}
68
{21, {38, 19, 62, 31, 98, 49, 152, 76, 38}}
69
{16, {8, 4, 2, 1, 8}}
70
{19, {20, 10, 5, 20}}
71
{34, {38, 19, 62, 31, 98, 49, 152, 76, 38}}
72
{11, {8, 4, 2, 1, 8}}
73
{21, {38, 19, 62, 31, 98, 49, 152, 76, 38}}
74
{9, {74, 37, 116, 58, 29, 92, 46, 23, 74}}
75
{19, {20, 10, 5, 20}}
76
{9, {76, 38, 19, 62, 31, 98, 49, 152, 76}}
77
{39, {38, 19, 62, 31, 98, 49, 152, 76, 38}}
78
{29, {38, 19, 62, 31, 98, 49, 152, 76, 38}}
79
{14, {92, 46, 23, 74, 37, 116, 58, 29, 92}}
80
{6, {20, 10, 5, 20}}
81
{12, {62, 31, 98, 49, 152, 76, 38, 19, 62}}
82
{11, {8, 4, 2, 1, 8}}
83
```

```
{29, {38, 19, 62, 31, 98, 49, 152, 76, 38}}

84

{24, {38, 19, 62, 31, 98, 49, 152, 76, 38}}

85

{14, {20, 10, 5, 20}}

86

{34, {38, 19, 62, 31, 98, 49, 152, 76, 38}}

87

{47, {38, 19, 62, 31, 98, 49, 152, 76, 38}}

88

{13, {38, 19, 62, 31, 98, 49, 152, 76, 38}}

89

{24, {38, 19, 62, 31, 98, 49, 152, 76, 38}}

90

{22, {20, 10, 5, 20}}

91

{34, {38, 19, 62, 31, 98, 49, 152, 76, 38}}

92

{9, {92, 46, 23, 74, 37, 116, 58, 29, 92}}

93

{37, {38, 19, 62, 31, 98, 49, 152, 76, 38}}

94

{24, {38, 19, 62, 31, 98, 49, 152, 76, 38}}

95

{22, {20, 10, 5, 20}}

96

{22, {38, 19, 62, 31, 98, 49, 152, 76, 38}}

97

{12, {74, 37, 116, 58, 29, 92, 46, 23, 74}}

98

{9, {98, 49, 152, 76, 38, 19, 62, 31, 98}}

99

{76, {38, 19, 62, 31, 98, 49, 152, 76, 38}}

100

{9, {20, 10, 5, 20}}
```

All the sequences had a loop within the first 100 iterations.

In the starting values from 1 to 100, there are the following ending cycles:

1- cycle (where 1 is the smallest number of the cycle),
19- cycle
5- cycle
23- cycle

Right now, it is not clear if these are the only cycles. It is possible as we increase the starting number, more cycles will appear.

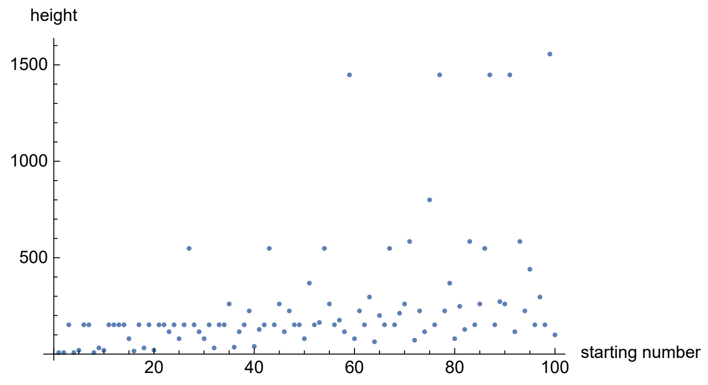
Examining Heights of the col5 seqences

```
In[ ]:= height[n_] := Max[makeSeq[col5, n, 100000]]
```

```
In[ ]:= heightslist = Table[{n, height[n]}, {n, 1, 100}];
```

```
In[ ]:= ListPlot[heightslist, AxesLabel → {"starting number", "height"}, PlotRange → All]
```

Out[ ]=

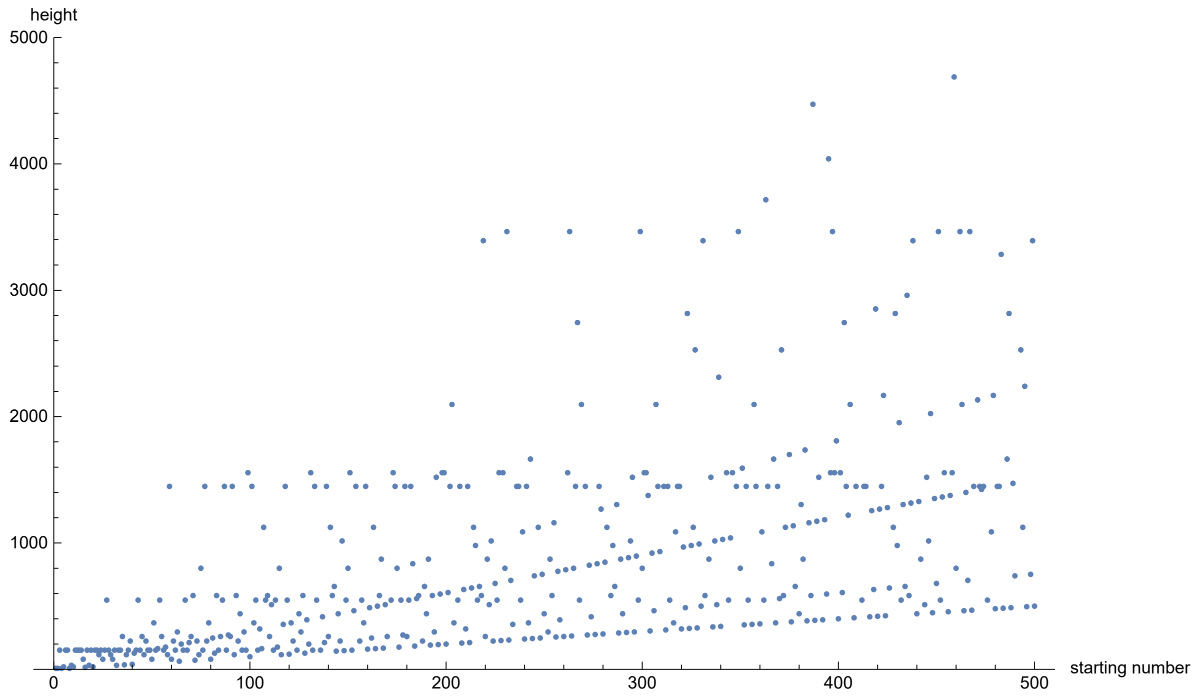


Lets look at more values, see if there is a bigger effect

```
In[ ]:= heightslist = Table[height[n], {n, 1, 500}];
```

In[∘]:=
```
ListPlot[heightslist, AxesLabel → {"starting number", "height"}, PlotRange → 5000]
```

Out[∘]=



I have limited to PlotRange to only show below height = 5000, because there are some points which go up to 50,000 which makes it hard to get the pattern out of the graph. Looking at this graph, it has some similarities to the same graph of the actual Collatz sequence. Just like the actual Collatz sequence, this one has diagonal and horizontal lines.

What are heights that make horizontal lines in the height vs starting number plot?

I can investigate this by taking the height list and tallying the values. If i get rid of the numbers with small tally counts, then the rest should be from the horizontal lines.

```
In[∘]:= tallyheights = Tally[heightslist]

Out[∘]= {{8, 4}, {152, 38}, {20, 3}, {32, 3}, {80, 6}, {16, 1}, {116, 8}, {548, 29}, {260, 12},
         {36, 1}, {224, 11}, {40, 1}, {128, 3}, {368, 9}, {164, 3}, {176, 3}, {1448, 42},
         {296, 6}, {64, 1}, {200, 3}, {212, 3}, {584, 16}, {72, 1}, {800, 9}, {248, 3},
         {272, 3}, {440, 8}, {1556, 18}, {100, 1}, {320, 3}, {1124, 10}, {512, 5}, {356, 3},
         {120, 1}, {8324, 10}, {392, 3}, {46160, 16}, {416, 3}, {656, 6}, {144, 1}, {1016, 5},
         {148, 1}, {464, 3}, {160, 1}, {488, 3}, {500, 3}, {872, 8}, {168, 1}, {10196, 10},
         {836, 3}, {184, 1}, {560, 2}, {192, 1}, {1520, 5}, {196, 1}, {596, 2}, {608, 2},
         {2096, 6}, {208, 1}, {632, 2}, {644, 2}, {980, 4}, {3392, 4}, {680, 2}, {228, 1},
         {3464, 8}, {232, 1}, {704, 2}, {1088, 4}, {240, 1}, {1664, 3}, {244, 1}, {740, 2},
         {752, 2}, {13112, 3}, {1160, 2}, {256, 1}, {776, 1}, {788, 1}, {264, 1}, {2744, 2},
         {824, 1}, {276, 1}, {1268, 2}, {280, 1}, {848, 1}, {1304, 3}, {288, 1}, {292, 1},
         {884, 1}, {896, 1}, {1376, 2}, {304, 1}, {920, 1}, {932, 1}, {312, 1}, {968, 1},
         {2816, 3}, {324, 1}, {2528, 3}, {328, 1}, {992, 1}, {336, 1}, {2312, 1}, {340, 1},
         {1028, 1}, {1040, 1}, {1592, 1}, {352, 1}, {360, 1}, {3716, 1}, {1700, 1}, {376, 1},
         {1136, 1}, {1736, 1}, {384, 1}, {4472, 1}, {388, 1}, {1172, 1}, {1184, 1}, {4040, 1},
         {1808, 1}, {400, 1}, {1220, 1}, {408, 1}, {6308, 1}, {1256, 1}, {2852, 1}, {420, 1},
         {2168, 2}, {424, 1}, {1280, 1}, {1952, 1}, {2960, 1}, {1316, 1}, {1328, 1}, {2024, 1},
         {448, 1}, {1352, 1}, {1364, 1}, {456, 1}, {4688, 1}, {1400, 1}, {468, 1}, {2132, 1},
         {1424, 1}, {480, 1}, {3284, 1}, {484, 1}, {1472, 1}, {5408, 1}, {2240, 1}, {496, 1}}
```

```
In[∘]:= tallyheights[[1, 2]]

Out[∘]= 4
```

```
In[∘]:= filteredlist = {};
        Do[If[tallyheights[[n, 2]] > 10, AppendTo[filteredlist, tallyheights[[n]]]]
          , {n, 1, Length[tallyheights]}];
         (*Only takes the tally values where the count was greater than 10*)
        filteredlist

Out[∘]= {{152, 38}, {548, 29}, {260, 12}, {224, 11}, {1448, 42}, {584, 16}, {1556, 18}, {46160, 16}}
```

Above are the most common heights of the first 500 sequences of the col5 function. Lets do this with more values

```
heights2 = Table[height[n], {n, 1, 5000}];(*5000 sequences*)
```

```
In[∘]:= tallyheights2 = Tally[heights2];
```

```
In[∘]:= filteredlist2 = {};
        Do[If[tallyheights2[[n, 2]] > 20, AppendTo[filteredlist2, tallyheights2[[n]]]]
          , {n, 1, Length[tallyheights2]}];
         (*Only takes the tally values where the count was greater than 20*)
        filteredlist2

Out[∘]= {{152, 38}, {548, 31}, {1448, 77}, {1556, 36}, {8324, 100}, {46160, 354}, {10196, 135},
         {3464, 47}, {13112, 70}, {2816, 23}, {5408, 41}, {9296, 26}, {7928, 35}, {21860, 28},
         {11168, 31}, {17648, 26}, {25748, 24}, {6416, 23}, {33632, 30}, {18944, 21}, {118088, 29}}
```

Above are the most common heights for 5000 sequences of the col5 function

What is stopping time of each cycle?

Note, stopping time is calculated differently for this Collatz function because not all the sequences end in the same cycle. I am calculating the stopping time as the number of iterations it to get a repeating number, which indicates the start of a loop.

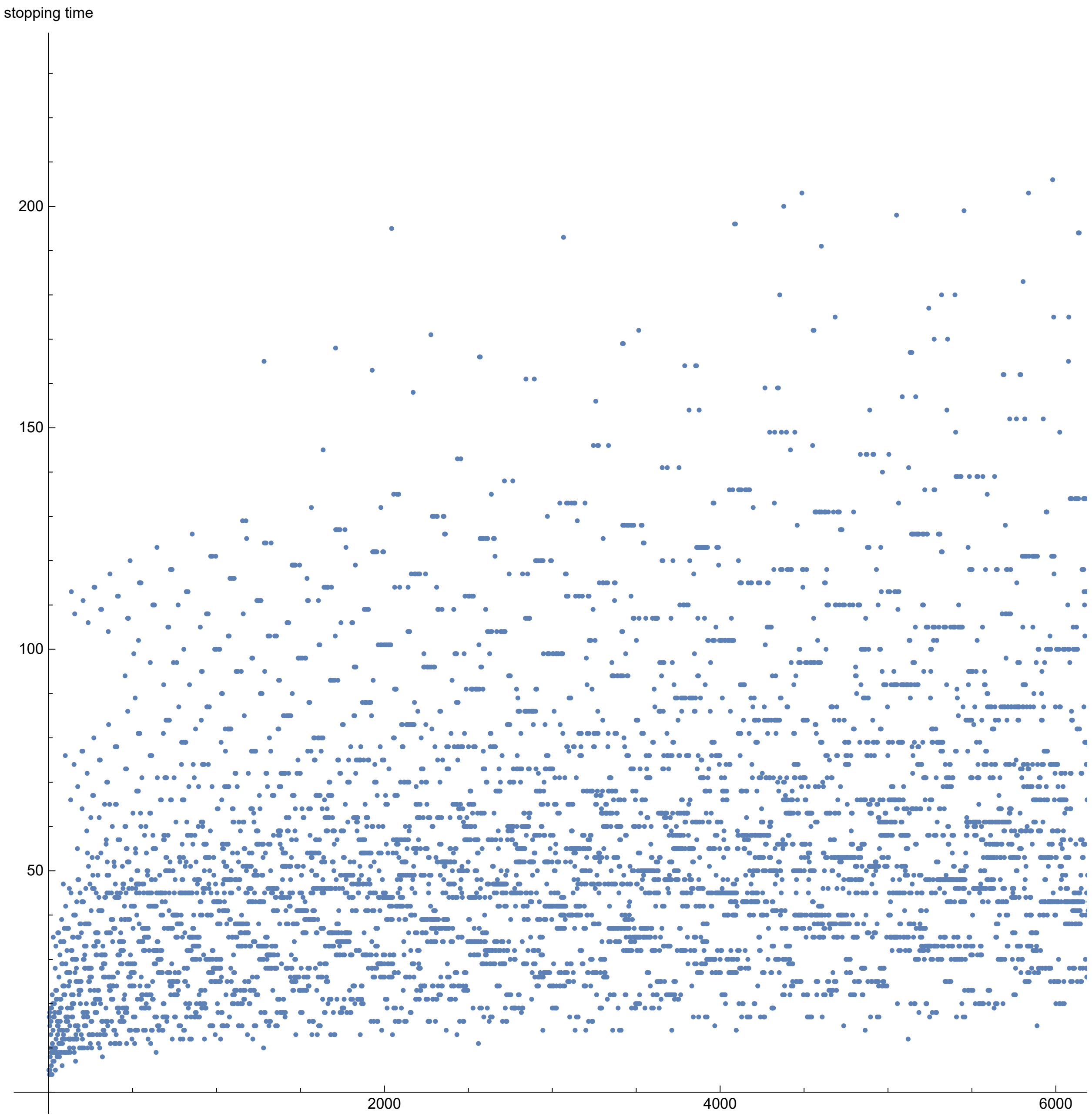
```
checkcycle[makeSeq[col5, 10, 1000]][[1]](*This code will give me the stopping time*)
```

```
Out[•]= 4
```

```
In[•]:= stoppingtimelist = Table[{n, checkcycle[makeSeq[col5, n, 100 000]][[1]]}, {n, 1, 10 000}];
```

```
In[•]:= ListPlot[stoppingtimelist, AxesLabel → {"starting value", "stopping time"}]
```

Out[•]=



There are similarities and differences between this plot and the same one that was made for the actual Collatz Sequences. Just like before, there are streaks of numbers where the stopping distance is the same for some consecutive numbers. There are also seems to be a similar curving pattern where it looks like dots on the plot are outlining shapes that look like leaves. Something that is different in this one than the one we saw in class is that the individual dots and steaks are denser in this one, which makes the overall curving pattern harder to see.

Investigating the Streaks: What Are the most common streaks?

Modifying the code from Class

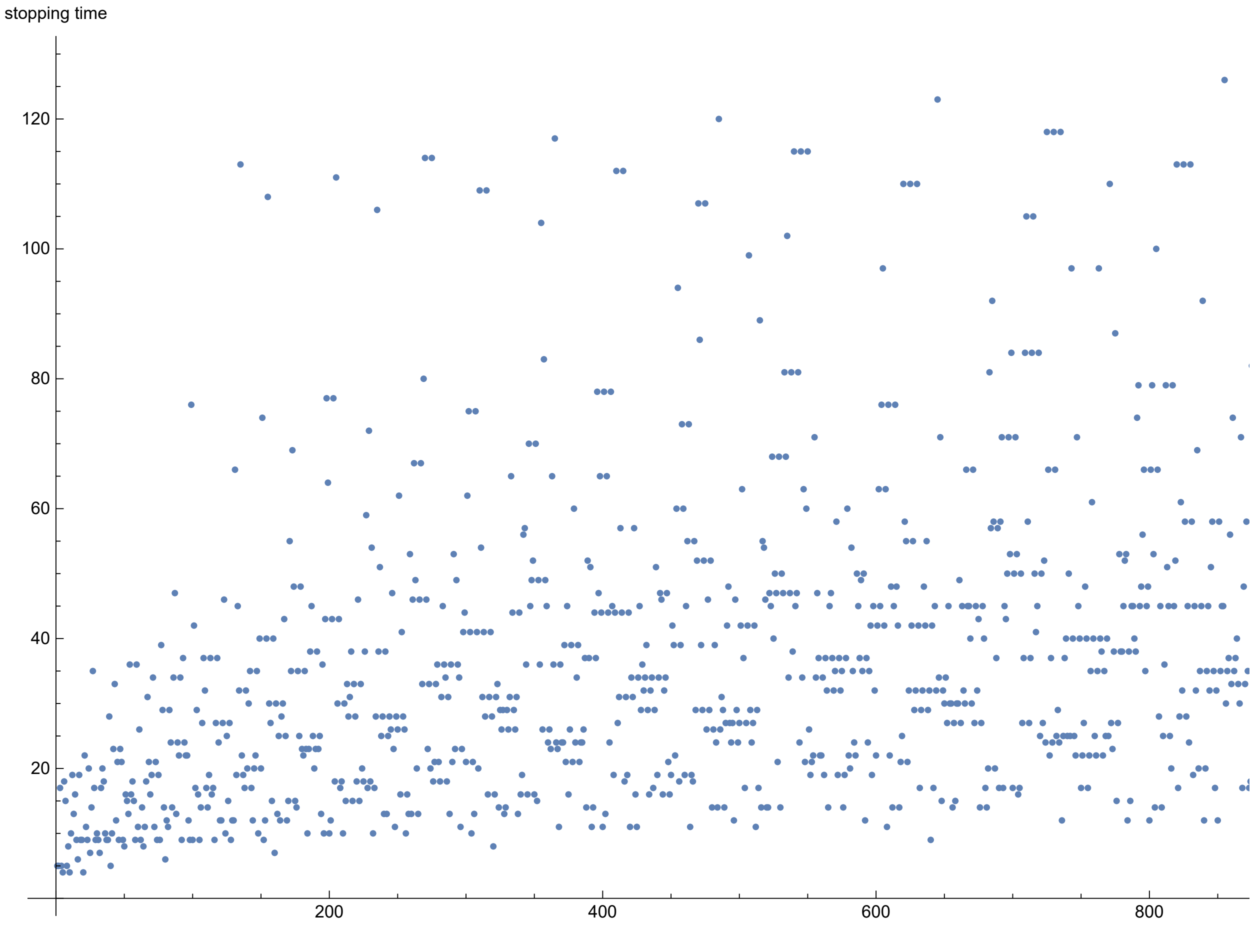
This detectStreak module takes in a list, goes through the list element by element, and whenever the value changes, it will tell you the streak of that value.

```
In[•]:= detectStreak[list_] := Module[{pos = 2, currStreak = 1, newlist = {}},

          (* loop over list entries until we find a streak *)
          While[ pos <= Length[list],
           If[list[[pos]] == list[[pos - 1]],
            (*true*)currStreak += 1,

            (*else*)AppendTo[newlist, currStreak]; currStreak = 1];
           pos += 1
          ];

          (* return value *)
          newlist
         ]

In[•]:= testlist = {1, 1, 2, 3, 4, 5, 5, 5, 6};

In[•]:= detectStreak[testlist]

Out[•]= {2, 1, 1, 1, 3}
```

The code above shows how the module is supposed to work. There is a streak of 2 for the two ones, a streak of 1 for the two three and four, and a streak of 3 for the fives. It does not give the streak for the last value, but that is not that big of a deal, because our list we want to use will have thousands of points. Missing one value is not a big deal. Note, that if there is a streak of 10, it only reports the value of 10, and not the values 10 through 1.

Using this code, I will take the stoppingtime list and find how many streaks of each number there are.

```
In[•]:= stoppingtimelist[[ ;; 1000]];

        stoppingtimelist[[All, 2]]; (*This is all the stopping values*)

In[•]:= Tally[detectStreak[stoppingtimelist[[All, 2]]]]

Out[•]= {{2, 59}, {1, 9881}}

In[•]:= ListPlot[stoppingtimelist[[ ;; 1000]], AxesLabel → {"starting value", "stopping time"}]

Out[•]=
```



Interestingly, there is no streak greater than 2 in the col5 sequence for starting values from 0 to 10,000. There was 59 streaks of 20 in the first 10,000 I zoomed into the first 1,000 points, and it is clearer to see. Some of the lines of points that look like consecutive streaks actually have gaps in them.

I will also compare this with the actual collatz sequences below.

```
col[n_?OddQ] := 3 n + 1
col[n_?EvenQ] := n / 2
computeTrajectory[start_, maxIter_ : 100000] := Module[{traj = {start}, n = start},
  (* compute the trajectory until we find 1 *)
  While[n != 1 && Length[traj] < maxIter,
   n = col[n];
   AppendTo[traj, n];
  ];
  (* return the trajectory *)
  traj
 ]
```

```
In[]:= stop[n_] := Length[computeTrajectory[n]] - 1

In[]:= colstoplist = stop /@ Range[1000];

In[]:= Tally[detectStreak[colstoplist]]

Out[]= {{1, 394}, {2, 150}, {3, 72}, {5, 10}, {4, 5}, {6, 2}, {7, 1}}
```

The first 1,000 sequences for that actual collatz function have streaks up to 7. Interestingly, the col5 seems to be more dense of points, but actually non of the streaks have totally consecutive points.

How does the proportion of each cycle change as the starting number increase?

```
Min[checkcycle[makeSeq[col5, 10, 1000]][[2]]]
(*This gives me the smallest number in the repeating cycle,
this will be how I will figure out if a certain starting value ends in a cycle*)

Out[]= 5

cycleslist = Table[
   Min[
    checkcycle[makeSeq[col5, n, 1000]][[2]]
   ], {n, 1, 1000}];(*The numbers in this list
 represent the cycle of the nth sequence with starting value n*)

In[]:= Tally[cycleslist]

Out[]= {{1, 142}, {19, 494}, {5, 200}, {23, 96}, {187, 36}, {347, 32}}
```

In the first 1,000 sequences, along with the 1-cycle, 19-cycle, 5-cycle, and 23 cycle, there are two more that show up, the 187-cycle, and the 347-cycle.

How does the proportion of 1 cycle change with time.

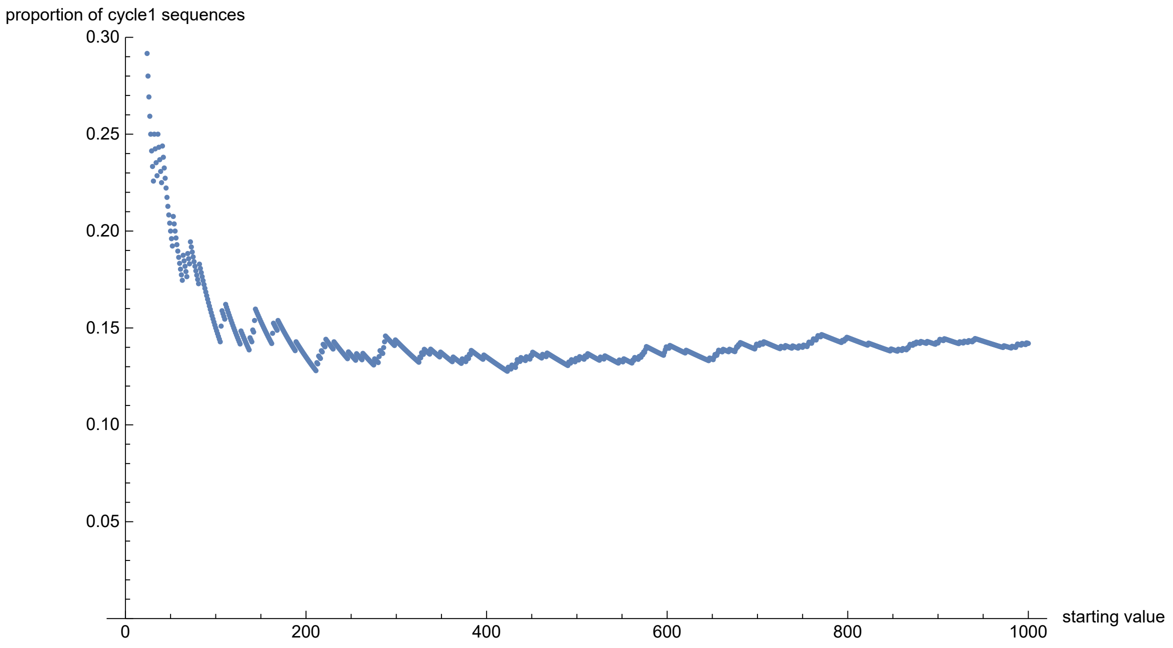
In[∘]:=

```
count1 = 0;
cycle1prop = ConstantArray[0, Length[cycleslist]];
Do[If[cycleslist[[n]] == 1, count1 = count1 + 1];
 cycle1prop[[n]] = count1 / n;
 , {n, 1, Length[cycleslist]}]
```

In[∘]:=

```
ListPlot[cycle1prop,
 AxesLabel → {"starting value", "proportion of cycle1 sequences"}, PlotRange → 0.3]
```

Out[∘]=

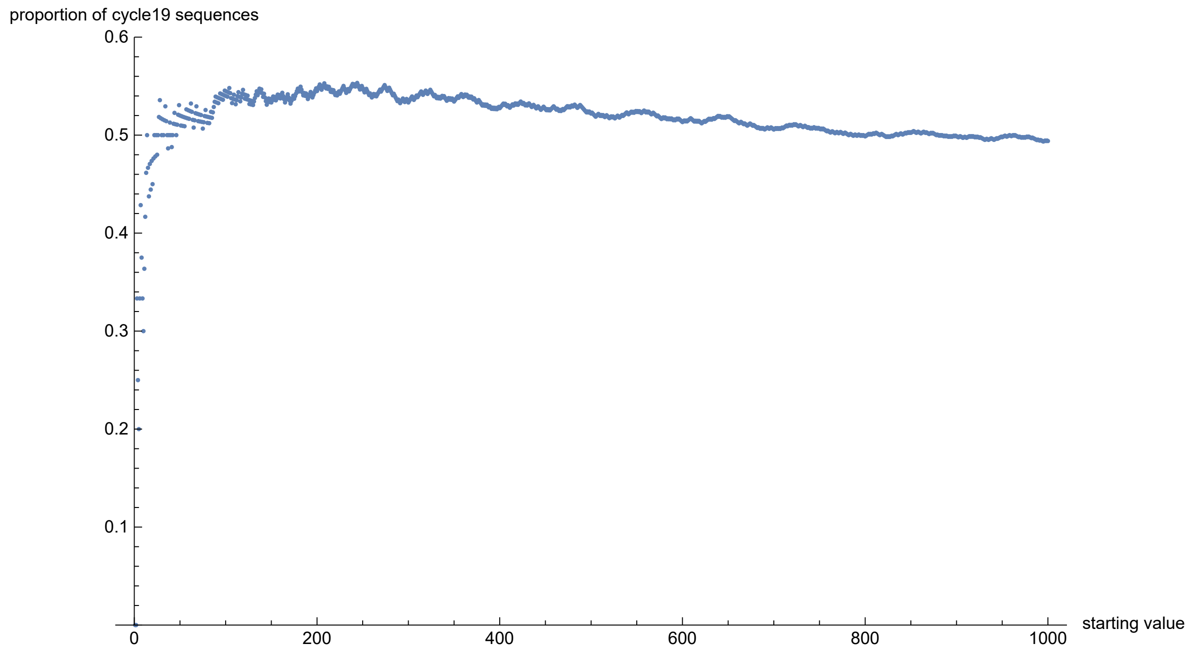


How does the proportion of 19- cycle change with time?

In[∘]:=

```
count19 = 0;
cycle19prop = ConstantArray[0, Length[cycleslist]];
Do[If[cycleslist[[n]] == 19, count19 = count19 + 1];
 cycle19prop[[n]] = count19 / n;
 , {n, 1, Length[cycleslist]}]
```

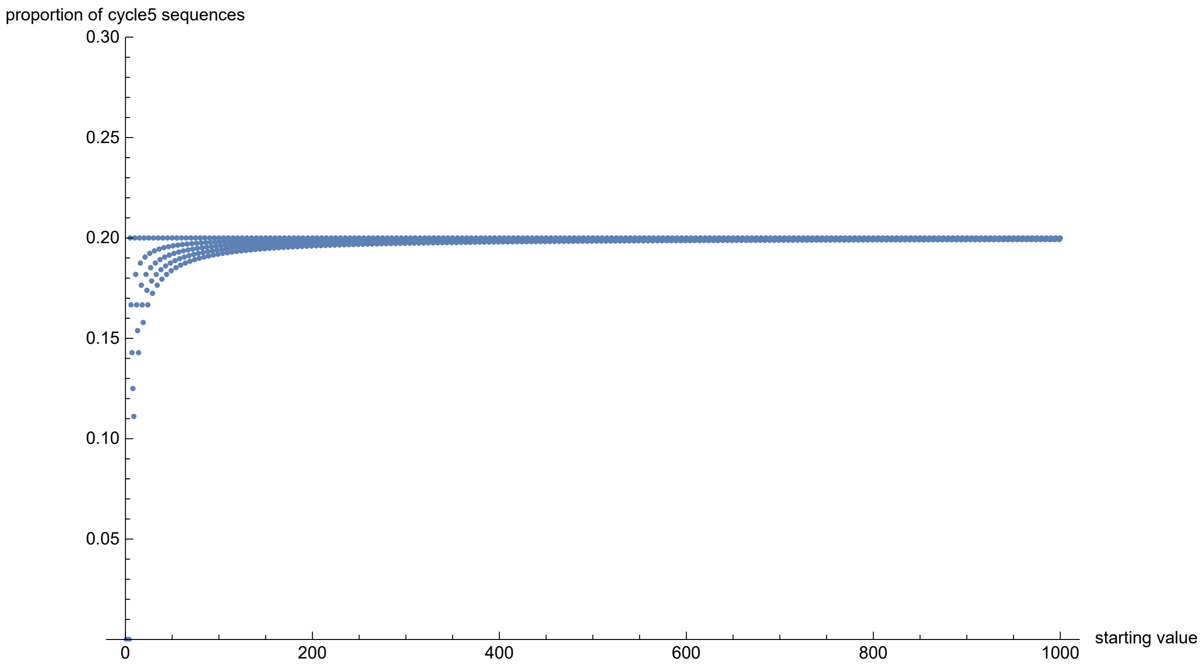
In[ ]:=
```
ListPlot[cycle19prop,
 AxesLabel → {"starting value", "proportion of cycle19 sequences"}, PlotRange → 0.6]
```

Out[ ]=


How does the proportion of 5 - cycle change with time?

In[ ]:=
```
count5 = 0;
cycle5prop = ConstantArray[0, Length[cycleslist]];
Do[If[cycleslist[[n]] == 5, count5 = count5 + 1];
 cycle5prop[[n]] = count5 / n;
 , {n, 1, Length[cycleslist]}]
```

In[]:=
```
ListPlot[cycle5prop,
 AxesLabel → {"starting value", "proportion of cycle5 sequences"}, PlotRange → 0.3]
```

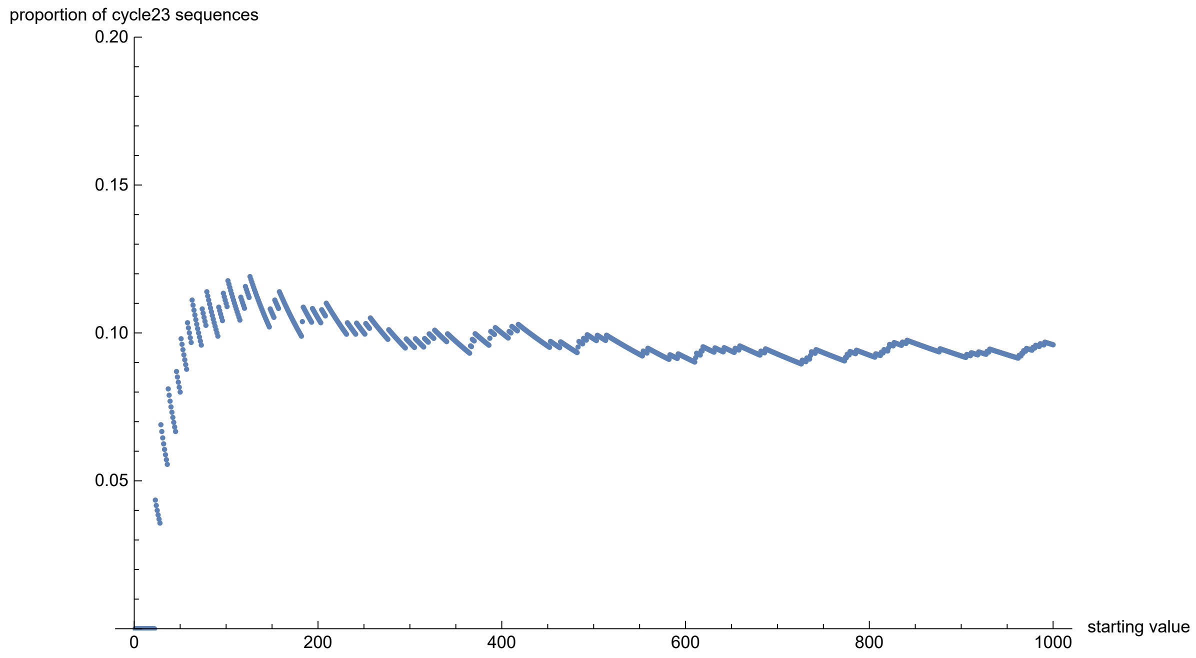
Out[]=


Proportion of 23-cycle:

In[]:=
```
count23 = 0;
cycle23prop = ConstantArray[0, Length[cycleslist]];
Do[If[cycleslist[[n]] == 23, count23 = count23 + 1];
 cycle23prop[[n]] = count23 / n;
 , {n, 1, Length[cycleslist]}]
```

In[ ]:=
```
ListPlot[cycle23prop,
 AxesLabel → {"starting value", "proportion of cycle23 sequences"}, PlotRange → 0.20]
```

Out[ ]=

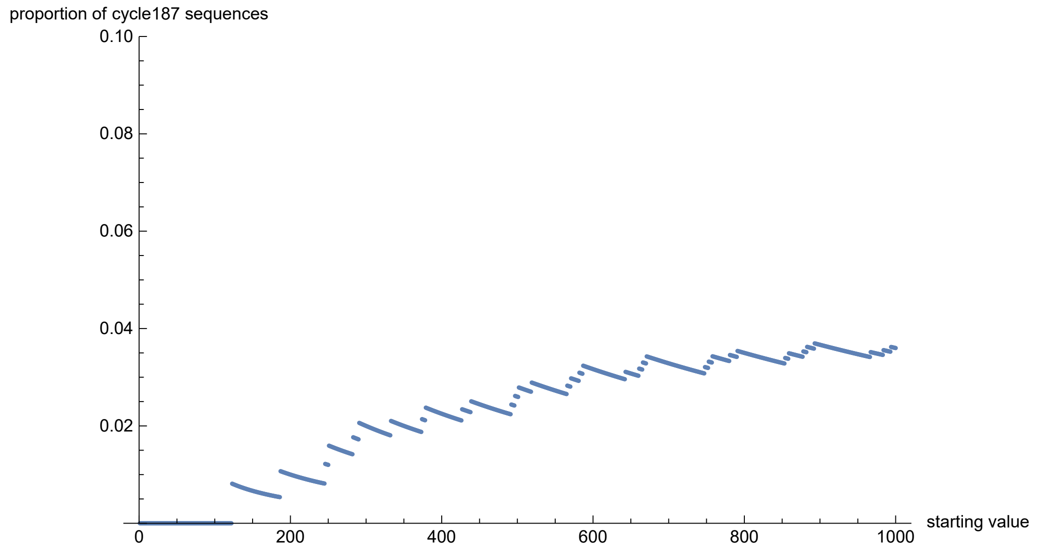


Proportion of 187 cycles:

In[ ]:=
```
count187 = 0;
cycle187prop = ConstantArray[0, Length[cycleslist]];
Do[If[cycleslist[[n]] == 187, count187 = count187 + 1];
 cycle187prop[[n]] = count187 / n;
 , {n, 1, Length[cycleslist]}]
```

In[ ]:=
```
ListPlot[cycle187prop,
 AxesLabel → {"starting value", "proportion of cycle187 sequences"}, PlotRange → 0.1]
```

Out[ ]=

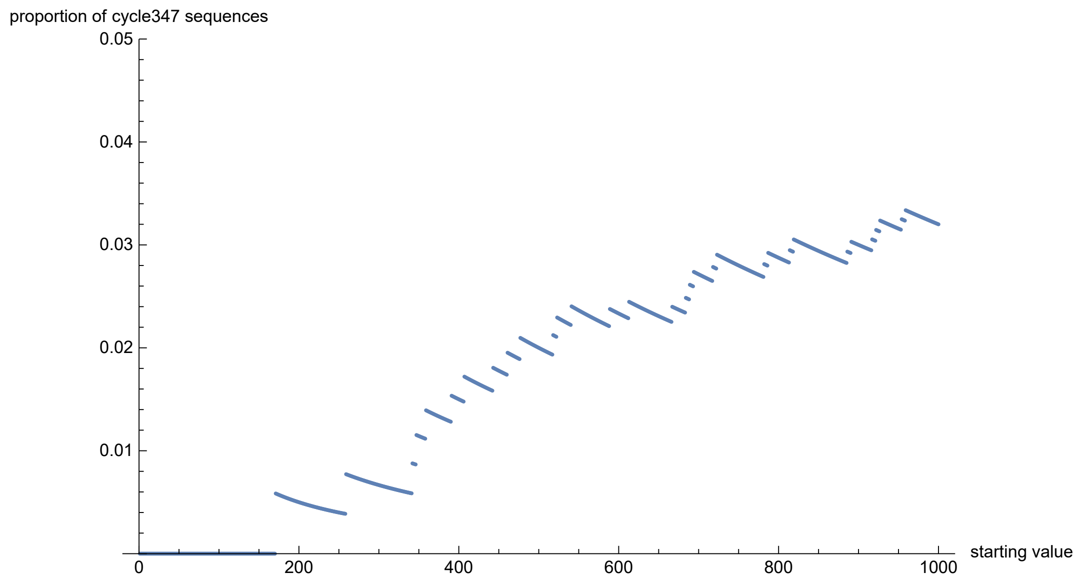


Proportion of 347 cycles:

In[ ]:=
```
count347 = 0;
cycle347prop = ConstantArray[0, Length[cycleslist]];
Do[If[cycleslist[[n]] == 347, count347 = count347 + 1];
 cycle347prop[[n]] = count347 / n;
 , {n, 1, Length[cycleslist]}]
```

In[ ]:=
```
ListPlot[cycle347prop,
 AxesLabel → {"starting value", "proportion of cycle347 sequences"}, PlotRange → 0.05]
```

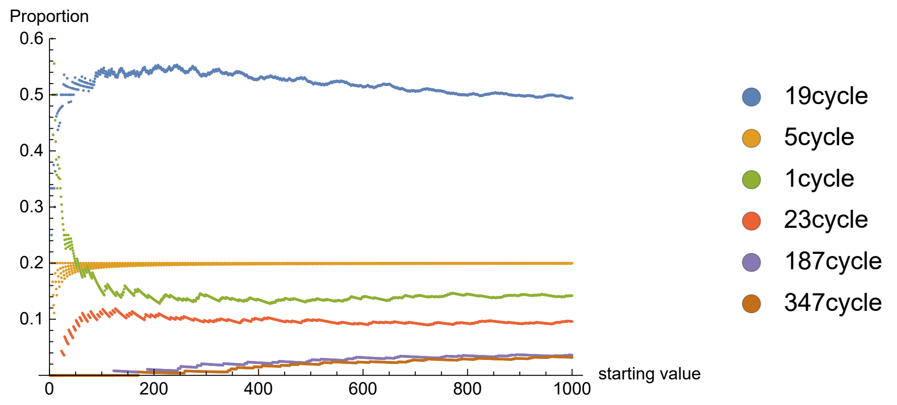
Out[ ]=



Let’s summarize our findings by putting all these on the same plot

In[ ]:=
```
ListPlot[{cycle19prop, cycle5prop, cycle1prop, cycle23prop, cycle187prop, cycle347prop},
 AxesLabel → {"starting value", "Proportion"}, PlotLegends →
  PointLegend[{"19cycle", "5cycle", "1cycle", "23cycle", "187cycle", "347cycle"},
   LegendMarkers → Graphics[Disk[]], LegendMarkerSize → Medium], PlotRange → 0.6]
```

Out[ ]=



Looking at this plot, we can see some interesting things. First, the proportions for large n are very stable. We can see that the proportion of each cycle oscillates a little bit about some value but they never deviate too far. The 19 cycle is by far the most common, with about 50% of the sequences ending in this cycle. However, as n increases, you can see that the proportion of 19 cycles slowly decreases, especially as the 187 cycle and 347 cycle starting rising up. The 1 cycle stays around 15% and the 23 cycle stays around 10% for most starting values. Interestingly, the 5 cycle stays almost completely constant exactly at 20%. This is likely because every 5th starting value (any number divisible by 5) will end up in a 5 cycle. Right now, there is not enough data to say whether or not the 187 or 347 cycles will stabilize.