www.logofoundation.org

Easy as 1 1 2 2 3

by

Michael Tempel

On the front page of the science section of the New York Times, there was an article headlined "Intellectual Duel: Brash Challenge, Swift Response." John Conway, a mathematician best known for inventing the Game of Life (an early version of the Phantom Fish Tank), made a presentation to a symposium at the Bell Labs in New Jersey. He presented a number series that he found interesting. It starts like this:

[1 1 2 2 3 4 4 4 5 6 7 7]

Can you guess Conway's rule?

Here's a hint. It's similar to the Fibonacci series in that the next number in the series is calculated by adding together two other numbers in the series. The Fibonacci series looks like this:

[1 1 2 3 5 8 13 21 34 55 89] etc.

By adding together the last two numbers in the Fibonacci series, we get the next number. The series starts as

[1 1]

1 + 1 = 2 so we have

[1 1 2]

1 + 2 = 3 so it becomes

[1 1 2 3]

The next addition, 2 + 3 produces the series

[1 1 2 3 5]

and so it grows.

In Conway's series the selection of the two numbers which are added together is not so simple. Do you give up?

Here's how it works.

Take the last number in the series and use it as a counter. Count backwards that many places and see what number you find. If the series is up to

[1 1 2 2 3 4 4 4]

then the counting number is 4. In the fourth position from the end, we find a 3. Remember that. Now use the same counting number to count forward through the list. The number in the fourth position from the front of the list is 2. Now add 3 and 2 to get the next number. The series becomes

[1 1 2 2 3 4 4 4 5]

Now the counter is 5. In the fifth position from the end is 3. In the fifth position from the beginning is 3. (They happen to be the same number.) The next number is 6, so the series becomes

[1 1 2 2 3 4 4 4 5 6]

So what? Why does this rate front page treatment in the New York Times? Well, Conway noticed something interesting about the series. The last number in the series, divided by the length of the series, is always a number close to one half. When the series is

[1 1 2 2 3 4 4 4]

the last number is 4. The length of the series is 8. The ratio is exactly .5. Here are the next four steps of the series followed by the ratios of the last number to the length of the series:

[1 1 2 2 3 4 4 4 5] 0.5556

[1 1 2 2 3 4 4 4 5 6] 0.6

[1 1 2 2 3 4 4 4 5 6 7] 0.6364

[1 1 2 2 3 4 4 4 5 6 7 7] 0.5833

Conway and his wife proved that, as the series gets longer, this ratio converges on .5. Is this worthy of a New York Times headline? Maybe not, but Conway offered $1,000 to anyone in the audience who could find the point in the series beyond which the ratio never deviated from .5 by more than 10%. Collin Mallows took up the challenge. After a few days of "...messing around on the backs of envelopes," he pulled out his Cray, determined that it was the 3,173,375,556th position in the series and collected his prize. That's news. Actually, due to a slip of the tongue, Conway offered $10,000, not $1,000. He claimed otherwise, but the people at the Bell Labs had the whole show on video tape. Mallows agreed that Conway probably meant $1,000 and accepted the lesser amount.

Well, I don't have a Cray, but an hour after reading the article, I found myself confined to an airplane seat on the way to Montréal, with a Toshiba 1000 running LogoWriter. I started to play.

You may have already noticed that I've been representing Conway's series as a bracketed list. The Times used the text book convention

1,1,2,2,3,4,4,4,5,6,...

Since I know Logo, representing the series as a list seemed natural. Also, it put it in a form that may be manipulated by Logo procedures.

To start with, I figured I'd write a Logo program to generate Conway's series. My approach was to write a procedure that would take any instance of the series as input and report the series with the next number stuck on the end.

```
conway [1 1]
```

should report [1 1 2]

```
conway [1 1 2]
```

should report [1 1 2 2]

...and so on.

The new number is obtained by adding two numbers, so my Logo procedure will need to use +. The last number in the series is used as a counter. I can use the Logo primitive **last** to report that number. **Item** can be used to report the number found at a particular position in the list.

Lput could append the new number to the end of the current series. Here's what I came up with:

```
to conway :series
output lput
(item last :series :series) +
(item last :series reverse :series)
:series
end
```

But wait. **Reverse** isn't a primitive. No problem:

```
to reverse :list
if (count :list) = 1 [op :list]
op lput
first :list
reverse bf :list
end
```

I tried my procedure.

```
show conway [1]

[1 2]

show conway [1 2]

[1 2 3]

show conway [1 2 3]

[1 2 3 4]
```

Something was wrong. These were just the counting numbers. The procedure looked right. After puzzling over this for a while, I realized that the procedure was fine. I had started with the wrong initial series. The minimum series, as with Fibonacci, is [1 1].

```
show conway [1 1]

[1 1 2]

show conway [1 1 2]

[1 1 2 2]
```

```
show conway [1 1 2 2]

[1 1 2 2 3]
```

It worked!

I wrote a procedure to automatically generate ever longer Conway series:

```
to many.conways :series
print :series
many.conways conway :series
end
```

I started generating successive instances of the series along with the ratio of the last number to the length of the list. In Logo this ratio is

```
(last :series) / (count :series)
```

I graphed the changes in this ratio as Mallows did with his Cray, only I used the turtle. Here are the procedures I used:

```
to setup
rg
pu
setpos list
minus 159
50
pd
end

to graph :series
if xcor > 155 [stop]
setpos list
xcor + .5
100 * (last :series) / (count :series)
graph conway :series
end
```

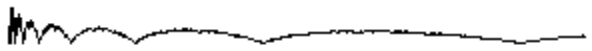
Setup puts the turtle near the left edge of the screen and at a **ycor** of 50. The first line of **graph** checks to see if the turtle is getting close to the right edge of the screen and stops the procedure if it is. Then the turtle's **xcor** is moved half a turtle step to the right and the **ycor** is set to be 100 times the ratio of the last number in the Conway series to the length of the series. Multiplying the ratio by 100 expands the fluctuations so they are visible.

Then **graph** is called again with an input that is the next instance of the series. Start

the graph with

```
graph [1 1]
```

Here's what it looks like after the series reaches a length of about 300 numbers.



The low points on the curve are where the ratio of **last :series** to **count :series** is .5. As the series grows, two things happen. The interval between the points, where the ratio exactly equals .5, stretches out. Second, the high points become lower. From this Logo graph, it certainly does look like the ratio will converge on .5. The graph is noticeably flattening even before it reaches a length of 300.

I added a line to see exactly how these low points were spaced.

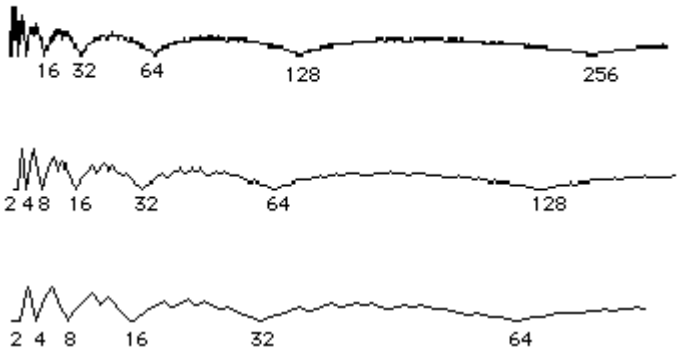
```
to graph :series
if xcor > 155 [stop]
if ((last :series) / (count :series)) = .5
[print count :series]
setpos list
xcor + .5
100 * (last :series) / (count :series)
graph conway :series
end
```

This prints the length of the series whenever the ratio of the last number to the length of the series is exactly .5. Here's what I got:

```
2
4
8
16
32
64
128
256
```

Wow! That looks familiar.

The left side of my graph is unclear, kind of squashed together. I rewrote the graph procedure to take larger steps in the x direction, 2 instead of .5. This spreads out the graph but it doesn't get as far into the series. I then increased the x to step to 4 to spread things out even more. Here are all three graphs:



The numbers show the length of the series at each low point; that is, when the crucial ratio is .5.

Now what about the high points? As you can see from the graphs, the rise and fall between each pair of low points are not smooth. However, there is a single high point in each interval. Here are the positions where those high points occur:

Position in the series	**Number in that position**	**Ratio**
3	2	.6667
6	4	.6667
11	7	.6364
23	14	.6087
44	26	.5909
92	53	.5761
178	101	.5674

Is there any pattern here? Except for the first two ratios, which are equal, each one is smaller than the one before. Also, the difference between one ratio and the next gets smaller as we move along. According to Mallows, by the time the series reaches the 3,173,375,556th place that ratio will be below .55 and remain that way as the series continues to grow.

Well, we're nowhere near that point and Logo is running out of space. I suppose we can get back to this once Logo is running on a Cray.

Now let's get back to my initial error of using [1] as an input to **conway** instead of [1 1]. A closer look shows why this generates the counting numbers. Counting backwards and forwards one place in the list [1] gives two ones to add to get the next number. Then with the series at [1 2], counting to the second place backwards gives 1. Counting to the second place forwards goes to the end of the list and results in 2. Adding them gives 3 and the new list [1 2 3]. Since the last number is the counter,

we will always count forward to the end of the list and backwards to the beginning of the list where we find the number 1. Thus we always add 1 to the last number in the list to get the next number.

Conway said, "I used to invent series every night in hopes of finding something interesting. This sequence was the only one I found that had interesting qualities." And Mallows said that "the series has beautiful symmetries and repeating properties" but "no practical use whatever."

By making the mistake of using the wrong starting point for the series, the same algorithm generated an uninteresting series with many practical uses.

Well, I think that's enough for now. I intend to keep playing with these ideas. Maybe you will, too. Let me know if you come up with something interesting.