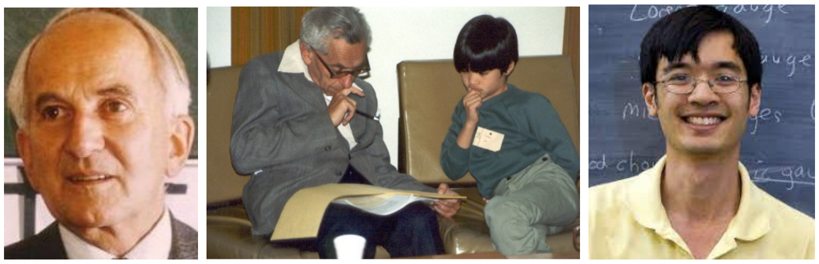
The Collatz Problem
Mathematical Programming with Python

https://people.sc.fsu.edu/∼jburkardt/classes/mpp_2023/collatz/collatz.pdf



Lothar Collatz proposed it, Paul Erdös avoided it, Terence Tao tackled it.

"Mathematics is not yet ready for such problems."
Paul Erdös, advising his student **not** to work on the Collatz conjecture.

"No problem is so intractable that something interesting cannot be said about it."
Jeffrey Lagarias.

The Collatz Conjecture

- *The Collatz mapping produces a new number from an old one;*
- *By repeatedly applying the mapping, we get a potentially infinite sequence;*
- *If a sequence includes the same number twice, it must have entered a loop;*
- *By convention, a sequence that reaches 1 is terminated (actually enters a tiny loop);*
- *The Collatz conjecture asserts that every Collatz sequence will terminate at 1;*
- *No proof has been found, but mathematicians are still attracted to the problem;*
- *Some simple variations of the Collatz mapping have very different behaviors.*

1 The Simplest "Imposible" Problem

A friend might ask for a typical mathematical problem that an "ordinary" person could understand. They may have heard of Fermat's Last Theorem. However, this simply says something is not possible, and that something is an abstract formula involving x, y, z, n that is meaningless to most people.

Some famous impossible problems like doubling the cube, the quadrature of the circle, or the trisecting of an angle, are easier to explain and visualize, but again the average listener doesn't understand the strict rules imposed on these problems which mean you can't simply trisect an angle by using a protractor to measure it, divide by 3 and get your answer.

The Collatz problem, however, involves nothing more than straightforward arithmetic, produces a surprising variety of unexpected results, has resisted mathematical analysis for decades, and yet can be understood by anybody.

Despite the claim that this is a simple problem, we will have to spend rather a lot of time going over the background. Then, given that this is an unsolved problem, we will look at mathematical and computational ideas to characterize or visualize what is going on. We will then look at turning these ideas into Python programs. At the end, you will be challenged to adapt these ideas, algorithms, and programs to analyze some variations on the original Collatz problem.

2 The Collatz transformation

In the 1930's, mathematician Lothar Collatz was studying an area of number theory, in which certain functions $f()$ operate on integers n to produce new integers $f(n)$. He realized that he could visualize this process as a kind of directed graph. The nodes of the graph would be integers, and an arrow would be drawn from each value n to its corresponding function value $f(n)$, which we might write as $n \rightarrow f(n)$. Since $f(n)$ was also a number, the graph also included the link $f(n) \rightarrow f(f(n))$ and so in. So starting at any value n, you could repeatedly apply the function $f()$ and move along the graph.

But what was the shape of this graph? Did paths go on forever? Did every node have an incoming link? Did some have several incoming links? Collatz realized that in some cases, the graph would include loops, which would imply that, for at least one starting point n, it must be the case that $f(f(...(f(n))...)) = n$. Having two ways of thinking about his problem suggested new ways to solve the problems he was interested in.

Perhaps the simplest function he looked at is now known as the Collatz function (or mapping, or transformation). We will symbolize it by $T(n)$, and define it as:

$$T(n) = \begin{cases} \frac{n}{2} & \text{if } n \text{ is even} \\ 3n+1 & \text{if } n \text{ is odd} \end{cases}$$

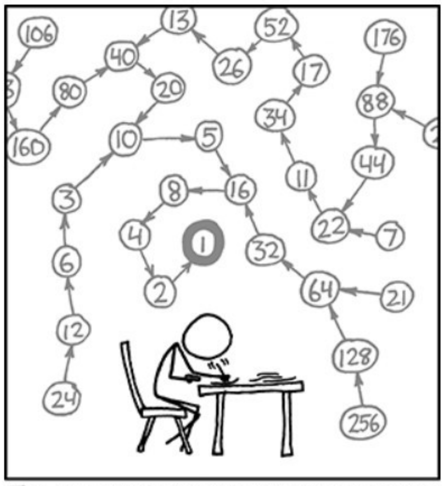
. Since the output of the Collatz transformation is another integer, we can apply the transformation again and again, producing a sequence of values. For instance, starting with $n = 7$, here are the first few terms of the corresponding sequence:

```
7 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1 4 2 1 4 2 1 ...
```

and we quickly lose interest when we see that the sequence has nothing new to tell us once the values $4, 2, 1$ show up. If you try other starting values for n, you will probably (probably?) find that after a few, several, or many steps, you reach the same three values repeating endlessly. If, instead of merely "probably happening", this is actually a mathematical fact, then it would be very nice to be able to prove it, and to understand why it happens!

Strictly speaking, the sequences we observe have entered an infinite loop. By convention, however, as soon as a Collatz sequence reaches the value 1, we will stop, and say the sequence has terminated after finitely many steps. And now we are ready for a formal statement of what seems to be happening:

Conjecture 1 *[The Collatz Conjecture]*
For any positive starting value n, the corresponding Collatz sequence $n, T(n), T(T(n)), ...$ will terminate at 1 in finitely many steps.



THE COLLATZ CONJECTURE STATES THAT IF YOU PICK A NUMBER, AND IF IT'S EVEN DIVIDE IT BY TWO AND IF IT'S ODD MULTIPLY IT BY THREE AND ADD ONE, AND YOU REPEAT THIS PROCEDURE LONG ENOUGH, EVENTUALLY YOUR FRIENDS WILL STOP CALLING TO SEE IF YOU WANT TO HANG OUT.

3 Graph theory approach

Consider the positive integers $n \in \mathbb{N}$ as nodes in a graph. For every n, draw a directed edge or "arrow" from node n to node $T(n)$. A graph with directed edges is known as a "directed graph" or *digraph*; in this case, the result is known as the Collatz digraph. The structure of the Collatz digraph may help to illustrate certain features of $T(n)$. Again, we will assume that any Collatz sequence that reaches 1 must stop there.

The number of directed edges coming into a node is its *indegree*. The number of directed edges leaving a node is its *outdegree*. Here is a fact about the indegree and outdegree of nodes of the Collatz digraph:

Theorem 1 *[Node degree of Collatz Digraph]*
Every node of the Collatz digraph:

1. *has outdegree 1, except for 1, which has outdegree 0.*
2. *has indegree 1 or 2.*

Since we decided that any Collatz sequence that reaches the value 1 must stop there, it turns out that the Collatz conjecture is equivalent to the following:

Conjecture 2 *[The Collatz Conjecture #2]*
Every node of the Collatz digraph is the beginning of a Collatz sequence that reaches the node 1.

A graph might contain a cycle, which is a path defined by a sequence of edges which returns to its starting point. For a digraph, a cycle must use directed edges in the proper order. If we didn't require Collatz sequences to stop at 1, then the Collatz digraph would include the cycle $1 \rightarrow 4 \rightarrow 2 \rightarrow 1$. We eliminated that cycle by forcing sequences to stop at 1.

But could there be other cycles in the digraph? And if so, what does that imply about the conjecture? Suppose there is another cycle somewhere, that is, a starting point $n \neq 1$, and a sequence of directed edges defined by the Collatz function, such that we have $n \rightarrow T(n) \rightarrow T(T(n)) \rightarrow ... \rightarrow n$ If such a cycle exists, then it cannot pass through the node labeled 1.

Theorem 2 *[A Cycle Disproves the Conjecture]*
If the Collatz digraph includes a cycle, then the Collatz Conjecture is false.

Is this the only way that the Collatz Conjecture could turn out to be false? Let's use our imagination, and start at some value n and follow the sequence. We don't want to reach 1 (so we can disprove the Conjecture), but we also don't want to ever repeat a value, otherwise we form a cycle, and we already worried about that. A Collatz sequence that never repeats a value and never reaches 1 must be of infinite length. That means it must eventually involve values that are arbitrarily large. We can say such a sequence "blows up".

Theorem 3 *[An Infinite Sequence Disproves the Conjecture]*
If the Collatz digraph includes an infinite (noncyclic) sequence, then the Collatz Conjecture is false.

We can turn this all around to say:

Theorem 4 *[Digraph Requirements for Collatz Conjecture]*
If there are no cycles and no infinite sequences in the Collatz digraph, then the Collatz conjecture is true.

Now suppose the Collatz conjecture is true, so that every node n has a finite directed path to the node 1. If we ignore, for a moment, the direction of the edges of the digraph, then we can say that for any pair of nodes m and n, there is a path of edges between them. A digraph which has this property is called `weakly connected`. This allows us to rephrase the conjecture yet again:

Conjecture 3 *[The Collatz Conjecture #3]*
The Collatz digraph is weakly connected.

You can see that the existence of a cycle, or an infinite sequence, would mean that there were some nodes that could not be connected by a path. In fact, we can simply realize that node 1 could not be connected to any node in a cycle, or in an infinite sequence.

4 Trajectories

You may be used to hearing the word *trajectory* to describe the path of a bullet. Mathematically, the same word is used to describe the sequence of values taken on by an object whose motion is described by some transformation formula. Often, this is a differential equation, but for our work here, it is the Collatz tranformation $T(n)$.

Thus the trajectory of n is the sequence of values $(n, T(n), T(T(n)), ..., T^k(n), ..., 1)$ (where we are assuming that all trajectories end at 1). Equivalently, we can rewrite this trajectory as $(T^0n, T^1(n), T^2(n), ..., T^k(n), ..., T^mn)$.

5 Trajectory length

For a trajectory in the form $(T^0n, T^1(n), T^2(n), ..., T^k(n), ..., T^mn)$, where $T^m(n) = 1$, we want to report the trajectory length. While the trajectory includes $m+1$ values, it takes m steps. It is difficult to decide which number is more appropriate to focus on, but we will take the trajectory length to be the number of steps, that is, m. we naturally want to say that the trajectory involves m steps, or has length m.

For instance, starting from the initial value $n = 52$, the Collatz sequence is

```
m:  0     1     2     3     4     5     6     7     8     9   10   11
n: 52 -> 26 -> 13 -> 40 -> 20 -> 10 ->  5 -> 16 ->  8 ->  4 -> 2 -> 1
```

We can regard the trajectory length as a function `tlen(n)` of the starting point, and so we say the trajectory starting at $n = 52$ has length $tlen(52) = m = 11$.

We can imagine starting a little table to study the behavior of this newly invented function:

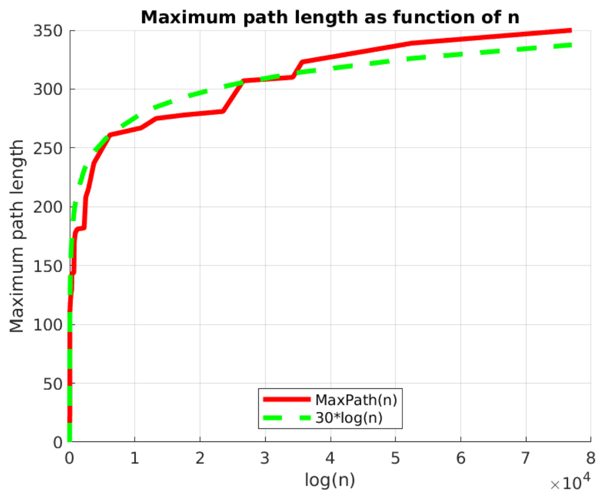
n	tlen(n)
1	0
2	1
3	7
4	2
5	5
6	8
7	16
8	3
9	19
10	6

Often, one hopes to find a simple formula for a function like `tlen(n)`. It turns out that the behavior of this function is surprisingly irregular, and no such formula has been found.

On the other hand, suppose for each n, we look at the maximum sequence length of any path with starting values between 1 and n. Then a surprising and suggestive plot emerges, if we are willing to go far in our data:

n	max(tlen(1:n)
1	0
2	1
3	7
6	8
7	16
9	19
18	20
25	23
27	111
54	112
73	115
97	118
129	121
171	124
231	127
313	130
327	143
649	144
703	170
871	178
1161	181
2223	182
2463	208
2919	216
3711	237
6171	261
10971	267
13255	275
17647	278
23529	281
26623	307
34239	310
35655	323
52527	339
77031	350

The data suggests logarithmic growth, which we can try to explore with a plot.



6 Trajectory peak

Just as we defined the length of a trajectory, we may also be interested in the largest or peak value observed. We might designate this function as `tpeak(n)`. For our Collatz sequence that starts with $n = 52$, we have $tpeak(52) = 52$, a somewhat uninteresting result. But in fact, we can see that it must always be the case that $tpeak(n) \geq n$. And if we try $n = 3$, we have

```
3 10, 5, 16 8, 4 2 1
```

so $tpeak(3) = 16$. If we add another column to our table, we once again see that, even from this small bit of evidence, there is little likelihood of a simple formula for predicting these values:

n	tlen(n)	tpeak(n)
1	1	1
2	2	2
3	8	16
4	3	4
5	6	10
6	9	16
7	17	52
8	4	16
9	20	52
10	7	16

On the other hand, we do see certain values, such as 16 and 52, showing up rather more often than random chance might suggest.

7 Average length and peak

We have seen that the length and peak for a Collatz sequence varies wildly as we increase n; however, we were able to get a more regular behavior by concentrating on the maximum length, and maximum peak, for all sequences with a starting value n between 1 and some value n_{max}

But if you make a plot showing every sequence length, or every sequence peak, over an interval, you will see that most of the values attained are much smaller than the maximums. This suggests that it might be appropriate to repeat the investigations made above, but this time to make plots of the *average* value of the length and peak over all sequences with starting value n in the interval $[1, n\_max]$.

To be clear, here is how the first 10 rows of such a table would look:

nmax	ave(tlen)	ave(tpeak)
1	1.00	1.00
2	1.50	1.55
3	3.66	6.33
4	3.50	5.75
5	4.00	6.60
6	4.83	8.16
7	6.57	14.42
8	6.25	14.62
9	7.77	18.77
10	7.77	18.50

This short table suggests that the sequence length function is smoother and slower rising than the sequence peak function. However, it would be worth while to extend the table out to $n\_max = 100$ or $n\_max = 1000$ to see if this initial pattern persists, and to see whether, for the average values, a logarithmic behavior is observed, as we saw for the maximum length and peak values.

8 Inverse Trajectory

If we think about the Collatz digraph, then at a node n, the Collatz function tells us where to move next, in order to approach the terminal node 1. This is a straightforward calculation. But suppose, instead, given a value n, we ask "how did we get here?", or better yet, "how *might* we get here?" Instead of computing $T(n)$, we are trying to compute something like $T^{-1}n$, the inverse function. Mathematically, this is not a well-defined function. If that bothers you, let's consider the essentially equivalent idea of the *preimage* of n, that is, the set of values m such that $T(m) = n$.

To make this clear, let's consider $n = 10$. With a little thought, or a glance at a picture of the Collatz digraph, we can see that $T^{-1}(10) = \{20, 3\}$. In fact, for any n, one element of the preimage is always $2n$, using the "divide an even value by 2" rule. In what cases is there a second item in the preimage? If there is a second item, with value m, then $n = 3 * m + 1$. In other words, it must be the case that $mod(n, 3) = 1$. Luckily, we see that $10 = 3 * 3 + 1$, so in fact, there is a second item in the preimage of 10, and it's 3.

What's the preimage of 3? Obvious 6 is in there, by doubling. But is there an odd number so that $3 = 3 * m + 1$? No, obviously not, because the right hand side cannot be divisible by 3. What about 6? Again, 12 is in the preimage, and because 6 is divisible by 3, it can't have an odd number if the preimage. In fact, the entire inverse trajectory of 3 consists of a single sequence of even values extending to infinity, of the form $3 * 2^k$.

The other "ancestor" of 10 was 20. Since 20=3*6+2, it can't have an odd ancestor, only 40. But 40=3*13+1, so 40 has ancestors 80 and 13. Both 80 and 13 only have one ancestor, but if we proceed further along either subbranch, we pick up more cases where there are two ancestors.

Theorem 5 *[Inverse trajectories are infinitely long]*
Every value n defines a set of inverse trajectories. Every such inverse trajectory extends infinitely far backwards.

But wait a minute, didn't we say earlier that if the Collatz conjecture is true, there can't be any infinite trajectories? And now you say the inverse trajectories are infinite. So...?

Let's suppose that the Collatz conjecture is true. Now pick any node n in the Collatz digraph. It must be the case that the trajectory length from n to 1 is finite. On the other hand, the inverse trajectories from n going backwards must all be infinite. So, perhaps to be more clear, if the Collatz conjecture is true, there can't be any infinite `forward` trajectories. It can take a while for you mind to get used to the idea that every node n sits on a trajectory that is finite (forwards to 1) and infinite (going backwards).

9 Climbing Backwards up the Collatz Digraph

Let's go back and think about the Collatz digraph. From every node n except 1, there is a single arrow pointing to the "next" node, whose value can be computed as $T(n)$. On the other hand, into every node n there is always an arrow from the even node $m_e = 2n$, and there is possibly a second arrow coming from an odd node m_o with a special property, namely, that $3 * m_o + 1 = n$. But for this second "parent" node to exist, it has to be the case that $m_o = \frac{n-1}{3}$ is an odd integer. This is not a rare occurrence; examples include $4 = 3*1+1, 10 = 3*3+1, 16 = 3*5+1, 22 = 3*7+1, ...$ So given a value n, we can expect an odd ancestor if `mod ( n, 6 ) = 4`.

If we think about this, we should realize that we can determine, for any node n, whether it has one "parent" or two, and we can compute the values of those parents.

n	mod(n,6)	m_e	m_o
1	1	2	-
2	2	4	-
3	3	6	-
4	4	8	1* (disallowed)
5	5	10	-
6	0	12	-
7	1	14	-
8	2	16	-
9	3	18	-
10	4	20	3
11	5	22	-
12	0	24	-
13	1	26	-
14	2	28	-
15	3	30	-
16	4	32	5
17	5	34	-
18	0	36	-
19	1	38	-
20	2	40	-
21	3	42	-
22	4	44	7

The table suggests that, in the column m_o, we will see each of the odd numbers appear, with a spacing of 6, to generate the second parent.

The "parent" function is actually a sort of $T^{-1}()$ operator. However, because of the occasional existence of two parents, we have to regard $T^{-1}()$ as a "pre-image" operator that returns a set of values; in our case we will be getting either one or two values returned.

Now recall the story of Jack and the Beanstalk, and suppose you are Jack, currently at node n, and the giant is chasing you from below, climbing up one node at a time. Where do you go? Using $T^{-1}(n)$, you can quickly decide whether you have one or two options, and then make a move to m_e or, if it exists, m_o. Obviously, you can repeat this operation for ever, or at least until the giant gets tired!

It is an interesting programming task to write a code that accepts a node n, and returns the value of m_e, and of m_o if it exists.

10 Level Sets of the Collatz Digraph

Assuming the Collatz conjecture is true, then every positive integer n corresponds to a node on the Collatz digraph whose distance from the node labeled 1 is $tlen(n)$. This means the Collatz digraph can be organized into levels. Level 0 is the node 1. Levels 1, 2, 3 and 4 contain respectively the nodes labeled 2, 4, 8, and 16. After this skimpy start, the levels start to fill out. Level 5 contains the nodes 5, and 32. Level 6 also contains 2 nodes, but after that, the levels gradually start to increase in size. (Each successive level must be at least as numerous as the previous one!)

To compute the size of the successive levels, we can work from the approach suggested above about climbing up the Collatz digraph. We begin by setting $n = 1$ and $level = 0$, and then use our $T^{-1}()$ preimage function, initially on the level set $\{1\}$, to move up the digraph level by level.

n	level size(n)	members
0	1	1
1	1	2
2	1	4
3	1	8
4	1	16
5	2	5, 32
6	2	10, 64
7	4	3, 20, 21, 128
8	4	6, 40, 42, 256
9	6	12, 13, 80, 84, 85, 512
10	6	24, 26, 160, 168, 170, 1024
11	8	48, 52, 55, 320, 336, 340, 341, 2048
12	10	17, 96, 104, 106, 113, 640, 672, 680, 682, 4096

It would be interesting to understand how the size of the levels grows. For this, we would need a lot more data than we have in this short table!

11 Python's dictionary variable

Along with lists, Python includes another data type, called a `dict`, for "dictionary". A `dict` can be thought of as a set of values that can be retrieved by keys, instead of an index. Suppose, for instance, that we wanted to store a list of the atomic weights of the elements. Since the elements are numbered, a natural approach would be to create a list. Restricting ourselves to the first 10 elements for this example, we write:

```
atw_list = [ 1.008, 4.0026, 6.94, 9.0122, 10.81, 12.011, 14.007, 15.999, 18.998, 20.180 ]
```

If the user wants the atomic weight of Carbon, they have to remember that this is element 6, and hence stored in `atw_list[5]`. If we are interacting with users, it might be more convenient to be able to retrieve the value of the atomic weight given the key of the element name

A suitable Python `dict` can be initialized by the command

```
atw_dict = dict ( [ ("Hydrogen":1.008), ("Helium":4.0026), ("Lithium":6.94), \
("Beryllium":9.0122), ("Boron":10.81), ("Carbon":12.011), ("Nitrogen":14.007), \
  ("Oxygen":15.999), ("Fluorine":18.998), ("Neon":20.180) ] )
```

Now, instead of requesting `atw_list[5]`, the user can ask for `atw_dict['Carbon']`.

If we want to start a dictionary from scratch, we create an empty `dict` using a pair of curly brackets (not square brackets!):

```
capital_dict = {}
```

We can add pairs of keys and values

```
capital_dict["Pennsylvania"] = "Harrisburg"
capital_dict["Virginia"] = "Richmond"
atw_dict["Sodium"] = 22.990
```

and so on. To see the contents of a `dict`, we can, as usual, rely on the `print()` command:

```
print ( capital_dict )
```

12 A computational example

For a given starting value n, we can code a function that returns $T(n)$, ignoring for a moment the question of what to do if n is 1:

```
  if ( ( n % 2 ) == 0 ):
    Tn = n // 2
  else:
    Tn = 3 * n + 1
```

A Collatz sequence starting from n is the sequence of values $n, T(n), T(T(n)), ...$ By convention, the sequence is terminated if it reaches the value 1. So far, every Collatz sequence examined has terminated in this way, but there is no proof that this must be so.

As an example, the Collatz sequence starting at 6 is:

```
6 3 10 5 16 8 4 2 1
```

and the Collatz sequence starting at 19 is

```
19 58 29 88 44 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
```

Notice that the first sequence jumps from 3 to 10, the second from 20 to 10, and after that the sequence are (and must be) identical.

If we are going to compute many Collatz sequences, we can take advantage of this merging process. When we start computing a new sequence, we can check to see if we encounter a value already computed by another sequence. Since that sequence presumaby terminated at 1, so does this one, and we can stop.

To make this happen efficiently, we can use a Python `dict`. It starts out empty. Given any n, we check to see if it is already in the dict. If so, we stop. Otherwise, we compute T(n), add (n,T(n)) to the dict, and then let T(n) become our next value of n.

```
  if ( not collatz_dictionary ):
    collatz_dictionary = {}

  while ( not ( n in collatz_dictionary ) ):
    Tn = collatz ( n )
    collatz_dictionary[n] = Tn
    n = Tn
```

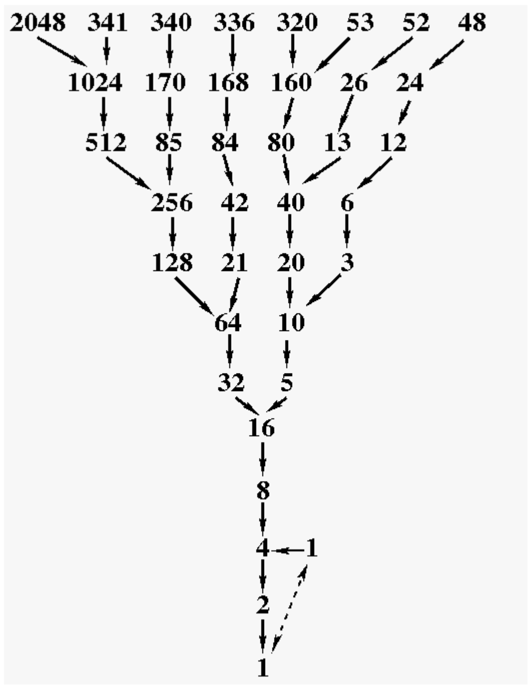
To see the efficiency of this approach, here is what the `dict` looks like after computing Collatz sequences starting at 1, 5 and 15:

```
[(1, 2), (2, 1), (4, 2), (5, 16), (8, 4), (10, 5), (15, 46), (16, 8), (20, 10), (23, 70),
(35, 106), (40, 20), (46, 23), (53, 160), (70, 35), (80, 40), (106, 53), (160, 80)]
```

and here is how the `dict` is updated, after we compute the additional sequences starting at 2, 3, 4, 6, 7, 8, 9 and 10:

```
[(1, 2), (2, 1), (3, 10), (4, 2), (5, 16), (6, 3), (7, 22), (8, 4), (9, 28), (10, 5),
(11, 34), (13, 40), (14, 7), (15, 46), (16, 8), (17, 52), (20, 10), (22, 11), (23, 70),
(26, 13), (28, 14), (34, 17), (35, 106), (40, 20), (46, 23), (52, 26), (53, 160), (70, 35),
(80, 40), (106, 53), (160, 80)]
```

If you continue this experiment by computing the sequence that starts with $n = 27$, you will see that the `dict` is able to compactly to incorporate new data for a long sequence whose values range into the thousands. If we had, instead, tried to use an array in some way, or a list, we would have wasted a lot of unused storage, and had to deal with slower access to a much larger data structure.



By the way, if you imagine the integers as nodes of a directed graph, then the Collatz transformation is telling you how to draw directed edges from n to $T(n)$, in such a way that we end up with a directed tree diagram rooted at 1, (assuming that every sequence does actually reach 1!)

13 Graphical Analysis

For any starting value n, we could try to visualize the Collatz sequence by plotting an (x, y) line graph, that is $(n, T(n))$. It's possible to make a nice plot for the simple sequence that starts at $n = 19$. But when we try $n = 27$, the existence of some extreme values makes the plot difficult to view. A more digestible plot would handle the data as a true graph, that is, each value of n that we work with becomes a node, and there is an arrow or link or directed line from n to $T(n)$. To make such a plot, however, you would need to download and import a package like `graphviz`. There is information in the Python lesson about advanced plotting that will show you more about how to do this.

To see the behavior of the sequence length function $tlen(n)$, a bar graph might be more useful than a line plot, since most of the values are small, with a few "explosions". Looking at the interval $1 \leq n \leq 100$ should give a reasonable idea of the variation in sequence length.

A similar bar plot can be made for the sequence peak function $tpeak(n)$.

Because of the smoothing properties of averaging, you can make an (x, y) line plot of the average sequence length, or the average sequence peak, over each interval $1 \leq n \leq n_max$, as you vary n_max from 1 to 100.

The $level\_size(k)$ function, which counts the number of integers at a distance k from the node 1, can be displayed with an (x, y) plot, since it is a smooth and slowly increasing function.

14 The Nollatz sequence

Here is a simple variation on the Collatz transformation, playfully termed the *Nollatz transformation*:

$$N(n) = \begin{cases} \frac{n}{2} & \text{if } n \text{ is even} \\ n+1 & \text{if } n \text{ is odd} \end{cases}$$

This alteration in the treatment of odd values has a very interesting effect on the behavior of Nollatz sequences. See the exercises for a challenge.

15 The Mollatz sequence

The definition of the Mollatz transformation replaces **+1** by **-1** when handling an odd input value n.

$$M(n) = \begin{cases} \frac{n}{2} & \text{if } n \text{ is even} \\ 3n-1 & \text{if } n \text{ is odd} \end{cases}$$

The exercises ask you to explore this sequence, and then prove a statement.

16 The Pollatz sequence

The Pollatz transformation replaces **3n+1** by **5n-1** when handling an odd input value n:

$$P(n) = \begin{cases} \frac{n}{2} & \text{if } n \text{ is even} \\ 5n-1 & \text{if } n \text{ is odd} \end{cases}$$

The exercises ask you to explore this sequence, and conjecture whether it can fall into a cycle.

17 The Lollatz permutation and sequence

What we know as the "Collatz sequence" is not the first such transformation that Lothar Collatz considered. His notebook for 1 July 1932 records his first investigation based on the following function:

$$L(n) = \begin{cases} \frac{2n}{3} & \text{if mod(n,3) = 0} \\ \frac{4n-1}{3} & \text{if mod(n,3) = 1} \\ \frac{4n+1}{3} & \text{if mod(n,3) = 2} \end{cases}$$

If we examine the effect of $L()$ on the first 10 integers, we see:

n:	1	2	3	4	5	6	7	8	9	10
L(n):	1	3	2	5	7	4	9	11	6	13

Unlike the Collatz transformation, this "Lollatz transformation" is a permutation. In other words, $L()$ simply rearranges the natural numbers. That means, in particular, that $L()$ is defined for all positive integers, and for every positive integer n, there is always a positive integer m such that $L(m) = n$, and this integer m is unique.

If we form the Lollatz digraph, then all the nodes have indegree and outdegree 1; there is no instance in which two nodes flow into one, or one flows out into two. Does this imply that the Lollatz digraph is a single infinite line? That can't be! We already see that $2 \rightarrow 3 \rightarrow 2$, a cycle involving just two values. From the table above, you should be able to spot another cycle, connecting five values. So the Lollatz digraph includes at least a few, possibly infinitely many, cycles or loops. Is it also possible that it contains one or more infinitely long sequences, with no beginning or end? At first, this might seem to be impossible, because the permutation property suggests that every value n has a preimage $L^{-1}(n)$ and a postimage $L(n)$. But it turns out that this fact doesn't rule out an infinite sequence.

In fact, Collatz himself was particularly interested in the sequence generated by starting with $n = 8$ and repeatedly applying the Lollatz permutation. Recall that the Lollatz permutation simply produces a new value from an old one. A Lollatz sequence applies this permutation repeatedly, just like the Collatz case. Therefore, if we follow the value 8 forward, we get, initially:

```
8, 11, 15, 10, 13, 17, 23, 31, 41, 55, 73, 97, 129, 86, 115, 153, 102, 68, 91, 121 ...
```

and we have to worry that this might be a very long sequence...or even infinite. This is another question that Collatz left unanswered.

18 Exercises

1. Create a Python function `tn=collatz_next(n)` which is given an integer `n` and returns `tn`, the next item in the Collatz sequence. For now, if `n` is 1, return the value 4; that is, don't worry about terminating the sequence at 1.
2. Modify your Python function `tn=collatz_next(n)` from the previous exercise so that it somehow returns a special output value or signal if the input value `n` is 1. Demonstrate that your function works by starting at `n=8`, and using a loop that executes the statement `n=collatz_next(n)` and prints each output value of `n`, until the sequence should be stopped, and the loop broken, after reaching 1.
3. Create a Python function `tlen=trajectory_length(n)` which reports the length of the Collatz trajectory starting at `n`. Use your function to compute trajectory lengths for $0 \leq n \leq 30$.
4. Copy the maximum trajectory length data from the table. Create a Python function `trajectory_length_plot()` which reproduces the plot in the text comparing the trajectory length against the function $y = 30 \log(n)$. Your function will need the command `import matplotlib.pyplot as plt`.
5. Create a Python function `tmax=trajectory_max(n)` which reports the maximum value encountered in a Collatz trajectory that begins at n and ends at 1. Test it by trying to reproduce the table of data given in the text.
6. Create a Python function `[]=trajectory_inverse(n)` which returns m_e and, if it exists, m_o, the even and odd numbers that are transformed to n by the Collatz transformation. Use your function to generate a "backward trajectory" that starts at 10, and goes backwards 20 steps, always choosing the odd value m_o if it exists.
7. Create a Python program that uses the `dict` data structure to remember Collatz information. Following the example in the text, compute the Collatz sequences for 1 through 10. Print out the information

stored in the `dict`. If you are brave, then also compute the Collatz sequence for 27 and see how the `dict` has changed.

8. If you are adventurous, try to download the package `graphviz` and define the nodes and edges of your graph using a program like this:

```
from graphviz import Digraph
dot = Digraph ( comment = 'The Collatz graph', format = 'png' )
dot.node ( '1', '1' )  # dot.node ( name, label )
dot.node ( '2', '2' )
dot.node ( '3', '3' )
...
dot.edge ( '2', '1' )  # dot.edge ( node, target node )
dot.edge ( '3', '10' )
dot.edge ( '4', '2' )
dot.edge ( '5', '16' )
...
dot.render ( 'collatz.dot', view = True )
```

9. For the "Nollatz" sequence, prove that:
   (a) For any positive n, the corresponding Nollatz sequence must converge to 1. This is the *Nollatz Theorem*...it's **not** a conjecture, after you prove it!
   (b) The peak value of a Nollatz sequence starting at n is $n + 1$ or less.
   (c) The number of steps required for a Nollatz sequence starting at n to reach 1 is no more than n.
10. For the "Mollatz" sequence, compute a few test sequences, look at the behavior of sequence length and maximum value. Decide whether it is possible for a Mollatz sequence to enter a cycle. Make a convincing argument one way or the other.
11. For the "Pollatz" sequence, compute a few test sequences. Look at the sequence that starts with $n = 7$. Look at the behavior of sequence length and maximum value. Conjecture whether it is possible for a Pollatz sequence to enter a cycle. Give a convincing argument to back up your conjecture.
12. For the "Lollatz" sequence, create a Python function that implements the Lollatz permutation. Create a second Python function that produces a Lollatz sequence by repeatedly applying the permutation. Since many starting values may quickly cycle, and others might (?) continue forever, you might try to terminate the sequence if it repeats a previously seen value, or if it exceeds some fixed maximum number of steps that you impose.
13. Because the Lollatz sequence is a permutation, every positive integer n has a unique preimage m such that $L(m) = n$. Based on the formula for $L(n)$, finish the following definition for the inverse function $L^{-1}(n) = m$:

$$L^{-1}(n) = \begin{cases} \frac{3n}{2} & \text{if mod(n,2)} = 0 \\ ? & \text{if mod(n,?)} = ? \\ ? & \text{if mod(n,?)} = ? \end{cases}$$

   Check that your formula for $L^{-1}(n)$ returns $3 \to 2, 4 \to 6, 5 \to 4, 6 \to 9, 7 \to 5, 11 \to 8, 13 \to 10$

   Create a Python function `lollatz_inverse(n)` that implements $L^{-1}()$ and use it to compute the 20 immediate ancestors of the value $n = 8$.