Discrete Structures: CMPSC 102

Oliver BONHAM-CARTER

Spring 2024
Week 07

Let's Discuss

Key Questions

How do I place an equation and a system of logic into Python code?

Learning Objectives

To **remember** and **understand** some the concepts involved with placing mathematical logic into code.

An Easy, *Hard* Problem

A problem for which Mathematics is not ready. Paul Erdös

Rules: the **Collatz** or **Hailstone** Problem

$$f(x) = \begin{cases} \frac{n}{2} & \text{if } n \equiv 0 \ (mod\ 2) \\ 3n+1 & \text{if } n \equiv 1 \ (mod\ 2) \end{cases}$$

- The 3x+1 problem concerns an iterated function
- The question is to determine whether the function always reaches a value of 1 when starting from any positive integer.

Trajectory

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

- Start $= 4$ (is even)
  - $f(4) = \frac{4}{2} = 2$ (is even)
  - $f(2) = \frac{2}{2} = 1$ (stop here)
- Last three elements of sequence: $\{4, 2, 1\}$

- Start $= 5$ (is odd)
  - $f(5) = 3(5) + 1 = 16$ (is even)
  - $f(16) = \frac{16}{2} = 8$ (is even)
  - $f(8) = \frac{8}{2} = 4$ (is even)
  - $f(4) = \frac{4}{2} = 2$ (is even)
  - $f(2) = \frac{2}{2} = 1$ (stop here)
- Last three elements of sequence: $\{4, 2, 1\}$

Trajectory

```
[+] Seed Number: 7
  1.   7        odd
  2.   22       even
  3.   11       odd
  4.   34       even
  5.   17       odd
  6.   52       even
  7.   26       even
  8.   13       odd
  9.   40       even
  10.  20       even
  11.  10       even
  12.  5        odd
  13.  16       even
  14.  8        even
  15.  4        even
  16.  2        even
  17.  1        odd
```

```
[+] Seed Number: 15
  1.   15       odd
  2.   46       even
  3.   23       odd
  4.   70       even
  5.   35       odd
  6.   106      even
  7.   53       odd
  8.   160      even
  9.   80       even
  10.  40       even
  11.  20       even
  12.  10       even
  13.  5        odd
  14.  16       even
  15.  8        even
  16.  4        even
  17.  2        even
  18.  1        odd
```

```
[+] Seed Number: 70
  1.   70       even
  2.   35       odd
  3.   106      even
  4.   53       odd
  5.   160      even
  6.   80       even
  7.   40       even
  8.   20       even
  9.   10       even
  10.  5        odd
  11.  16       even
  12.  8        even
  13.  4        even
  14.  2        even
  15.  1        odd
```

- All sequences end with $\{8, 4, 2, 1\}$

Trajectory

```
[+] Seed Number: 106
  1.   106       even
  2.   53        odd
  3.   160       even
  4.   80        even
  5.   40        even
  6.   20        even
  7.   10        even
  8.   5         odd
  9.   16        even
  10.  8         even
  11.  4         even
  12.  2         even
  13.  1         odd
```

```
[+] Seed Number: 600
  1.   600       even
  2.   300       even
  3.   150       even
  4.   75        odd
  5.   226       even
  6.   113       odd
  7.   340       even
  8.   170       even
  9.   85        odd
  10.  256       even
  11.  128       even
  12.  64        even
  13.  32        even
  14.  16        even
  15.  8         even
  16.  4         even
  17.  2         even
  18.  1         odd
```

```
  39.  29        odd
  40.  88        even
  41.  44        even
  42.  22        even
  43.  11        odd
  44.  34        even
  45.  17        odd
  46.  52        even
  47.  26        even
  48.  13        odd
  49.  40        even
  50.  20        even
  51.  10        even
  52.  5         odd
  53.  16        even
  54.  8         even
  55.  4         even
  56.  2         even
  57.  1         odd
```

- All sequences end with $\{8, 4, 2, 1\}$

Further Learning

- Wikipedia: Collatz Conjecture
  - `https://en.wikipedia.org/wiki/Collatz_conjecture`
- The Simplest Math Problem No One Can Solve - Collatz Conjecture
  - `https://www.youtube.com/watch?v=094y1Z2wpJg`
- UNCRACKABLE? The Collatz Conjecture - Numberphile
  - `https://www.youtube.com/watch?v=5mFpVDpKX70`

Collatz Conjecture in Sets

File: `collatz_sets.py`

```
def collatz_conjecture(n):
    sequence = set()

    while n != 1:
        if n in sequence:
            # Detected a cycle, exit to avoid infinite loop
            break
        sequence.add(n)
        if n % 2 == 0:
            n = n // 2
        else:
            n = 3 * n + 1
    sequence.add(n)
    return sequence

def main():
    try:
        starting_number = int(input("Enter a positive integer: "))
        if starting_number <= 0:
            raise ValueError("Please enter a positive integer.")
        collatz_sequence = collatz_conjecture(starting_number)
        print(f"Collatz sequence starting from {starting_number}:")
        print(collatz_sequence)
        print(f"Length of the sequence: {len(collatz_sequence)}")

    except ValueError as ve:
        print(f"Error: {ve}")

if __name__ == "__main__":
    main()
```

Collatz Conjecture in Lists

File: `collatz_lists.py`

```
def collatz_conjecture(n):
    sequence = [n]

    while n != 1:
        if n % 2 == 0:
            n = n // 2
        else:
            n = 3 * n + 1
        sequence.append(n)
    return sequence

def main():
    # Get user input for the starting number
    starting_number = int(input("Enter a positive integer: "))
    if starting_number <= 0:
        print("Please enter a positive integer.")
        return

    # Generate and print the Collatz sequence
    result_sequence = collatz_conjecture(starting_number)
    print(f"The Collatz sequence starting with {starting_number}")
    print(f"{result_sequence}")
    print(f"Length of the sequence: {len(result_sequence)}")

if __name__ == "__main__":
    main()
```

Creating Solutions

Go check out the fun code about sets in the `sandbox/`!