The Collatz Conjecture. For any positive integer \( n \), let \( f(n) = \begin{cases} n/2 & \text{if } n \text{ is even} \\ 3n + 1 & \text{if } n \text{ is odd} \end{cases} \). The conjecture is that, if we start with any positive integer, and repeatedly apply \( f \), we will eventually reach 1. For example, the Collatz sequence of 5 is 5, 16, 8, 4, 2, 1, while the Collatz sequence of 9 is 9, 28, 14, 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1.

We define \( f^*(n) \) to be the number of times we need to apply \( f \) to reach 1, starting from \( n \). Thus, for example, \( f^*(5) = 5 \), and \( f^*(9) = 19 \). If we never reach 1, we define \( f^*(n) = \infty \). However, in all known cases, \( f^*(n) \) is finite.

Your project is to write a program that builds a sparse array, implemented using a search structure, which holds the values of \( f^*(n) \) for all \( n \) up to 10, and for all other numbers which appear in the Collatz sequences of those numbers. If you used an ordinary array, it would have to have size 52 for \( N = 10 \), 9232 for \( N = 100 \), and 250504 for \( N = 1000 \). What a waste of space!

The function \( f^*(n) \), also called the total stopping time of \( n \), can be defined by the following recurrence:

\[
f^*(n) = \begin{cases} 0 & \text{if } n = 1 \\ 1 + f^*(f(n)) & \text{otherwise} \end{cases}
\]

Your search structure holds ordered pairs of the \((n, f^*(n))\), where \( n \) is the key. Fetch\((n)\) is executed by searching the structure. When you find the pair \((n,V)\), you know that \( f^*(n) = V \).

Do the following steps. For the first run, \( N = 10 \).

1. Initialize your search structure as empty, and then insert the pair \((1,0)\), since \( f^*(1) = 0 \).

2. If you execute fetch\((n)\) for any \( n \), the value of \( f^*(n) \) might already have been computed, in which case the pair \((n,f^*(n))\) will already be in the search structure.

3. If \( f^*(n) \) has not been computed, you must first calculate it, then insert it. You first compute \( m = f(n) \). You then obtain the value of \( f^*(m) \) by executing fetch. You then let \( f^*(n) = 1 + f^*(m) \), then insert the pair \((n,f^*(n))\) into the search structure.

4. Of course, \( f^*(m) \) might not have already been computed, in which case you must recursively execute the steps 2. and 3. for \( m \).

5. Your output consists of the list of values of \( f^*(n) \) for \( n = 1,2,\ldots,N \). I also printed a message each time a new memo was inserted into the search structure, other than the initial memo.

6. Run the program for \( N = 10 \) and then for \( N = 100 \). Print out the number of entries in the search structure.

Refer to the Wikipedia page: [https://en.wikipedia.org/wiki/Collatz_conjecture](https://en.wikipedia.org/wiki/Collatz_conjecture) There are other pages on the internet that deal with the same problem.
I am not telling you what to use for a search structure, but I used a binary search tree. Here is my binary search tree for $N = 10$. I did not try to maintain balance, so the tree looks rather lopsided.

My output for $N = 10$ is as follows.

- $f^*(1) = 0$
- inserting $f^*(2) = 1$
- $f^*(2) = 1$
- inserting $f^*(4) = 2$
- inserting $f^*(8) = 3$
- inserting $f^*(16) = 4$
- inserting $f^*(5) = 5$
- inserting $f^*(10) = 6$
- inserting $f^*(3) = 7$
- $f^*(3) = 7$
- $f^*(4) = 2$
- $f^*(5) = 5$
- inserting $f^*(6) = 8$
- $f^*(6) = 8$
- inserting $f^*(20) = 7$
- inserting $f^*(40) = 8$

There are 22 memos  
Tree height = 9