Daily Coding Problem #288

Problem

This problem was asked by Salesforce.

The number 6174 is known as Kaprekar's contant, after the mathematician who discovered an associated property: for all four-digit numbers with at least two distinct digits, repeatedly applying a simple procedure eventually results in this value. The procedure is as follows:

- For a given input x, create two new numbers that consist of the digits in x in ascending and descending order.
- Subtract the smaller number from the larger number.

For example, this algorithm terminates in three steps when starting from `1234`:

- `4321 - 1234 = 3087`
- `8730 - 0378 = 8352`
- `8532 - 2358 = 6174`

Write a function that returns how many steps this will take for a given input `N`.

Solution

Solution

To solve this imperatively, we can implement a while loop that continually runs the procedure described above until obtaining the number 6174. For each iteration of the loop we will increment a counter for the number of steps, and return this value at the end.

So that the number always remains four digits long, we can use a helper function that prepends zeros if necessary before creating the ascending and descending integers.

```
def get_digits(num):
    digits = str(num)

    if len(digits) == 4:
        return digits
    else:
        return '0' * (4 - len(digits)) + digits

def kaprekar_steps(num):
    steps = 0

    while num != 6174:
        digits = get_digits(num)
        num = int(''.join(sorted(digits, reverse=True))) - int(''.join(sorted(digits)))
        steps += 1

    return steps
```

Alternatively, we can implement a recursive solution. Here our base case will be if our input equals 6174, in which case we return the step count. Otherwise, our function will calculate

the difference between the ascending and descending permutations, and call itself with this new integer and an incremented number of steps.

```
def kaprekar_steps(num, steps=0):
    if num == 6174:
        return steps

    digits = get_digits(num)
    num = int(''.join(sorted(digits, reverse=True))) - int(''.join(sorted(digits)))

    return kaprekar_steps(num, steps + 1)
```

It can be seen experimentally that all (non-repeating) values of input between `1000` and `9999` terminate in fewer than eight steps. Since the bulk of each step consists of sorting strings of length four and performing subtraction, the time complexity is basically `O(1)`.

The first algorithm takes `O(1)` space, and the second will require space proportional to the number of steps to maintain the stack of recursion calls. However, since this number cannot exceed seven, as discussed above, this too can be considered constant.