MAA Metro NY Section
Problem of the Month

February 2025
Editor: Eric Rowland

List the positive integer palindromes in increasing order:

$$1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 22, 33, \ldots.$$

The *index* of the palindrome p is the positive integer i such that p is the ith palindrome. For example, the index of the palindrome 11 is 10. A *palindromic-index palindrome* is a palindrome whose index is also a palindrome. For example, 22 is a palindromic-index palindrome, since its index 11 is a palindrome. Moreover, 22 is the 10th palindromic-index palindrome. What is the 111th palindromic-index palindrome?

AI-generated solutions will not be accepted, but writing code is encouraged! Be sure to submit any code you wrote to solve this problem.

We feature a solution by David Feng and Indranil Biswas.

MAA Metro NY February Problem of the Month

David Feng & Indranil Biswas

February 9, 2025

1 Problem

A *palindromic number* is a number that is the same when its digits are reversed. Ex: 1, 2, 3, 11, 101, 1221.

A *palindromic-index palindrome* is a palindrome whose index is a palindrome. Ex: 1, 2, 3, 22, 131, 242.

What is the 111th palindromic-index palindrome?

1.1 Combinatorial Approach

The problem asks us to evaluate $P_{P_{111}}$, given that P is a palindrome. Hence, let's begin by solving for P_{111}.

Consider an even-length integer with $2k$ digits. The first k digits uniquely determine the last k digits due to the palindromic structure. The leading digit must be an integer between 1 and 9, while the next $k-1$ digits can be any integer from 0 to 9.

For an odd-length integer with $2k+1$ digits, the first k digits uniquely determine the last k digits, and the middle digit can be any one-digit integer.

Hence, the number of palindromes with $2k$ digits and $2k+1$ digits can be modeled by the function N, given by

$$N(2k) = 9(10)^{k-1}, \quad N(2k-1) = 9(10)^{k-1}$$

# of Digits	# of Palindromes
1	9
2	9
3	90
4	90
5	900
6	900

Table 1: Number of Digits vs. Number of Palindromes

Therefore, there are 108 palindromes with at most 3 digits, and the 108th palindrome is 999. Hence, by counting, the 111th palindrome is 1221.

$$P(108) = 999$$

$$P(109) = 1001$$

$$P(110) = 1111$$

$$P(111) = 1221$$

Then, the 111th palindromic-index palindrome is equivalent to the 1221th palindrome.

$$P_{P_{111}} = P_{1221} \tag{1}$$

From Table 1, there are 1098 palindromes with at most 5 digits, and the 1098th palindrome is 99999. Then, the 1221st palindrome is the 123rd palindrome of 6 digits.

$$1221 - 1098 = 123$$

For a 6-digit palindrome, there are $9 \times 10 \times 10 = 900$, where the first 100 palindromes are in the following form

$$1 \_\_\_\_ \_\_\_\_ \_\_\_\_ \_\_\_\_ 1$$

Then, the next 20 palindromes are in the following forms.

$$2\ 0 \_\_\_\_ \_\_\_\_ 0\ 2$$

$$2\ 1 \_\_\_\_ \_\_\_\_ 1\ 2$$

Finally, the 123rd 6th digit palindrome is 222222.

$$P_{P_{111}} = P_{1221} = 222222 \tag{2}$$

1.1.1 Python Implementation

The solution can be checked using a Python script by iterating through numbers sequentially and checking if it is palindrome.

```
def is_palindrome(num):
    return (str(num)==str(num)[::-1])

counter = 0
num = 0
palindrome = 0

while counter != 1221:
    num += 1
    if isPalindrome(num):
        counter+=1
        palindrome = num

print(palindrome)
```

The program enters a while loop that runs until the counter reaches 1221. isPalindrome(num) determines whether the integer is a palindrome by checking if a number reads the same forward and backward. If the integer is a palindrome, the counter is increased by 1. The variable palindrome is updated to store the latest palindrome found.

This script returns 222222.

1.2 Programmatic Approach

This problem can also be solved iteratively by checking each positive integer sequentially for a palindromic structure. If the integer is a palindrome, it is appended to an array and the index counter is increased by 1.

```
def is_palindrome(num):
    return (str(num)==str(num)[::-1])

def find_my_palindromic_index_palindrome(n):
    palindrome_array = []
    integer = 1
    index = 0
    palindromic_index_palindrome_index = 0

    while palindromic_index_palindrome != n:
        if is_palindrome(integer):
            palindrome_array.append(integer)
            index+=1
            if index in palindrome_array:
                palindromic_index_palindrome_index+=1
        integer+=1
    return {integer-1}


find_my_palindromic_index_palindrome(111)
```

If the index is within the palindrome array, then the index for the palindromic_index_palindrome is increased by 1. This while loop continues until the index of the palindromic_index_palindrome reaches the target nth palindromic-index palindrome, in this case, the 111th.

2 References

We thank Dr. James Sundstrom, Lecturer of Mathematics at Baruch College for providing some helpful insights on the problem.

We thank Aidan Epperly, a Ph.D. student at UC Davis for providing an alternative solution to the problem.