

plus_one

May 21, 2025

1 Plus One

1.1 Problem Definition

You are given a large integer represented as an integer array `digits`, where each `digits[i]` is the *i*th digit of the integer. The digits are ordered from most significant to least significant in left-to-right order. The large integer does not contain any leading 0's.

Increment the large integer by one and return the resulting array of digits.

1.1.1 Example Cases

Example 1:

Input: `digits = [1,2,3]`

Output: `[1,2,4]`

Explanation: The array represents the integer 123.

Incrementing by one gives $123 + 1 = 124$.

Thus, the result should be `[1,2,4]`.

Example 2:

Input: `digits = [4,3,2,1]`

Output: `[4,3,2,2]`

Explanation: The array represents the integer 4321.

Incrementing by one gives $4321 + 1 = 4322$.

Thus, the result should be `[4,3,2,2]`.

Example 3:

Input: `digits = [9]`

Output: `[1,0]`

Explanation: The array represents the integer 9.

Incrementing by one gives $9 + 1 = 10$.

Thus, the result should be `[1,0]`.

1.1.2 Constraints

Constraints:

- $1 \leq \text{digits.length} \leq 100$
- $0 \leq \text{digits}[i] \leq 9$
- `digits` does not contain any leading 0's.

1.2 Example test cases

```
[3]: def test_cases():
    assert plusOne([1,2,3]) == [1,2,4]
    assert plusOne([4,3,2,1]) == [4,3,2,2]
    assert plusOne([9]) == [1,0]
    print("All test cases passed!")
```

1.3 Solutions

$O(n)$

The problem here is when there are 9's at the end. This can be solved by changing 9s with 0s and adding 1 to the next element. So, a good idea is to iterate from the back. All 9s should be replaced with 0s and 1 should be transferred to the next digit.

```
[4]: def plusOne(l):
    if l[-1] != 9:
        l[-1] += 1
        return l
    else:
        if len(l) == 1:
            return [1, 0]
        else:
            pointer = -1
            while pointer != -len(l):
                if l[pointer] >= 9:
                    l[pointer] = 0
                    l[pointer-1] += 1
                    pointer -= 1
                else:
                    break
            if l[0] == 10:
                l[0] = 0
                return [1] + l
            return l
```

```
[5]: test_cases()
```

All test cases passed!

For this solution, I was trying to replicate what I wrote above. Below, I just cleaned it.

```
[15]: def plusOne(l):
    pointer = -1
    while pointer >= -len(l):
```

```
    if l[pointer] + 1 != 10:  
        l[pointer] += 1  
        return l  
  
    l[pointer] = 0  
    pointer -= 1  
  
    return [1] + l
```

```
[16]: test_cases()
```

All test cases passed!

```
[18]: plusOne([9])
```

```
[18]: [1, 0]
```