# 38. Count and Say

Problem Statement

1. The **count-and-say** sequence is a sequence of digit strings defined by the recursive formula:

   - `countAndSay(1) = "1"`
   - `countAndSay(n)` is the run-length encoding of `countAndSay(n - 1)`.

   Run-length encoding (RLE) is a string compression method that works by replacing consecutive identical characters (repeated 2 or more times) with the concatenation of the character and the number marking the count of the characters (length of the run). For example, to compress the string `"3322251"` we replace `"33"` with `"23"`, replace `"222"` with `"32"`, replace `"5"` with `"15"` and replace `"1"` with `"11"`. Thus the compressed string becomes `"23321511"`.

   Given a positive integer `n`, return *the* `n^th` *element of the* ***count-and-say*** *sequence.*

   **Example 1:**

   > **Input:** `n = 4`
   >
   > **Output:** `"1211"`
   >
   > **Explanation:**
   >
   > ```
   > countAndSay(1) = "1"
   > countAndSay(2) = RLE of "1" = "11"
   > countAndSay(3) = RLE of "11" = "21"
   > countAndSay(4) = RLE of "21" = "1211"
   > ```

   More Intuition

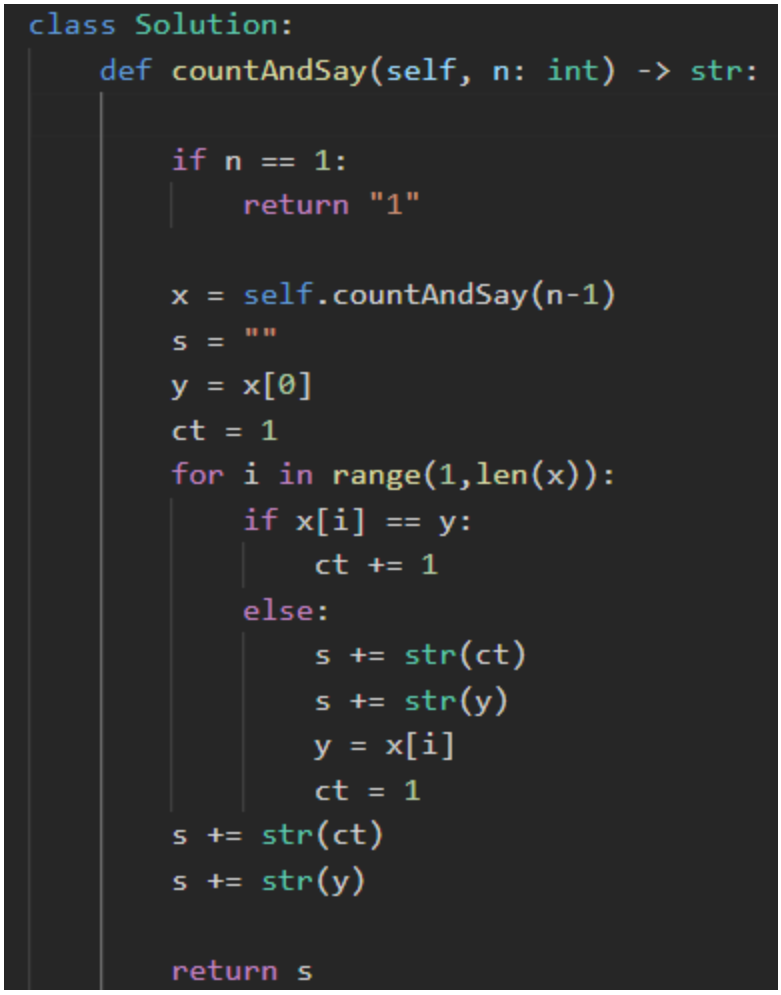
**Intuition :**

```
n = 1: return 1 is the base case
n = 2: return count of last entry i.e. 1 1
n = 3: return count of last entry i.e. two 1's so 21
n = 4: we have one 2 and one 1 so 1211
n = 5: , we have one 1 and one 2 and two 1's so -> 111221
n = 6: we have three 1's, two 2's and one 1 so -> 312211
n = 7: we have one 3, one 1, two 2's and two 1's -> 13112221
...
n = i: return counts in front of the number for entry of i-1 case
```

**The following are sequence from n=1 to n=10:**

```
1.     1
2.     11
3.     21
4.     1211
5.     111221
6.     312211
7.     13112221
8.     1113213211
9.     31131211131221
10.   13211311123113112211
```

Python code:

1. 

```python
class Solution:
    def countAndSay(self, n: int) -> str:

        if n == 1:
            return "1"

        x = self.countAndSay(n-1)
        s = ""
        y = x[0]
        ct = 1
        for i in range(1,len(x)):
            if x[i] == y:
                ct += 1
            else:
                s += str(ct)
                s += str(y)
                y = x[i]
                ct = 1
        s += str(ct)
        s += str(y)

        return s
```