Look-and-Say Sequence

Difficulty Level : Medium • Last Updated : 28 Jun, 2022

Find the n'th term in Look-and-say (Or Count and Say) Sequence. The look-and-say sequence is the sequence of the below integers:
1, 11, 21, 1211, 111221, 312211, 13112221, 1113213211, ...

How is the above sequence generated?

n'th term is generated by reading (n-1)'th term.

```
The first term is "1"

Second term is "11", generated by reading first term as "One 1"
(There is one 1 in previous term)

Third term is "21", generated by reading second term as "Two 1"

Fourth term is "1211", generated by reading third term as "One 2 O

and so on
```

How to find n'th term?

Example:

```
Input: n = 3
Output: 21

Input: n = 5
```

```
Output: 111221
```

Recommended Practice

Look and Say Pattern Try It!

The idea is simple, we generate all terms from 1 to n. First, two terms are initialized as "1" and "11", and all other terms are generated using previous terms. To generate a term using the previous term, we scan the previous term. While scanning a term, we simply keep track of the count of all consecutive characters. For a sequence of the same characters, we append the count followed by the character to generate the next term.

Below is an implementation of the above idea.

C++

```
// C++ program to find n'th term in look and say
// sequence
#include <bits/stdc++.h>
using namespace std;

// Returns n'th term in look-and-say sequence
string countnndSay(int n)
{
    // Base cases
    if (n == 1)      return "1";
    if (n == 2)      return "11";

    // Find n'th term by generating all terms from 3 to
    // n-1.  Every term is generated using previous term
    string str = "11"; // Initialize previous term
    for (int i = 3; i<=n; i++)
    {
        // In below for loop, previous character
        // is processed in current iteration. That
        // is why a dummy character is added to make
        // sure that loop runs one extra iteration.
```

```
        str += '$';
        int len = str.length();

        int cnt = 1; // Initialize count of matching chars
        string  tmp = ""; // Initialize i'th term in series

        // Process previous term to find the next term
        for (int j = 1; j < len; j++)
        {
            // If current character doesn't match
            if (str[j] != str[j-1])
            {
                // Append count of str[j-1] to temp
                tmp += cnt + '0';

                // Append str[j-1]
                tmp += str[j-1];

                // Reset count
                cnt = 1;
            }

            //  If matches, then increment count of matching
            // characters
            else cnt++;
        }

        // Update str
        str = tmp;
    }

    return str;
}

// Driver program
int main()
{
    int N = 3;
    cout << countnndSay(N) << endl;
    return 0;
}
```

Java

```java
// Java program to find n'th
// term in look and say sequence

class GFG
{

    // Returns n'th term in
    // look-and-say sequence
    static String countnndSay(int n)
    {
    // Base cases
    if (n == 1)     return "1";
    if (n == 2)     return "11";

    // Find n'th term by generating
    // all terms from 3 to n-1.
    // Every term is generated
    // using previous term

    // Initialize previous term
    String str = "11";
    for (int i = 3; i <= n; i++)
    {
        // In below for loop, previous
        // character is processed in
        // current iteration. That is
        // why a dummy character is
        // added to make sure that loop
        // runs one extra iteration.
        str += '$';
        int len = str.length();

        int cnt = 1; // Initialize count
                     // of matching chars
        String tmp = ""; // Initialize i'th
                         // term in series
        char []arr = str.toCharArray();

        // Process previous term
        // to find the next term
        for (int j = 1; j < len; j++)
        {
            // If current character
            // doesn't match
            if (arr[j] != arr[j - 1])
```

```
            {
                // Append count of
                // str[j-1] to temp
                tmp += cnt + 0;

                // Append str[j-1]
                tmp += arr[j - 1];

                // Reset count
                cnt = 1;
            }

            // If matches, then increment
            // count of matching characters
            else cnt++;
        }

        // Update str
        str = tmp;
    }

    return str;
    }

    // Driver Code
    public static void main(String[] args)
    {
        int N = 3;
        System.out.println(countnndSay(N));
    }
}

// This code is contributed
// by ChitraNayal
```

C#

```
// C# program to find n'th
// term in look and say sequence
using System;

class GFG
{
```

```
// Returns n'th term in
// look-and-say sequence
static string countnndSay(int n)
{
// Base cases
if (n == 1)     return "1";
if (n == 2)     return "11";

// Find n'th term by generating
// all terms from 3 to n-1.
// Every term is generated using
// previous term

// Initialize previous term
string str = "11";
for (int i = 3; i <= n; i++)
{
    // In below for loop, previous
    // character is processed in
    // current iteration. That is
    // why a dummy character is
    // added to make sure that loop
    // runs one extra iteration.
    str += '$';
    int len = str.Length;

    int cnt = 1; // Initialize count of
                 // matching chars
    string tmp = ""; // Initialize i'th
                     // term in series
    char []arr = str.ToCharArray();

    // Process previous term
    // to find the next term
    for (int j = 1; j < len; j++)
    {
        // If current character
        // doesn't match
        if (arr[j] != arr[j - 1])
        {
             // Append count of
            // str[j-1] to temp
            tmp += cnt + 0;

            // Append str[j-1]
```

```
                tmp += arr[j - 1];

                // Reset count
                cnt = 1;
            }

            // If matches, then increment
            // count of matching characters
            else cnt++;
        }

        // Update str
        str = tmp;
    }

    return str;
    }

    // Driver Code
    public static void Main()
    {
        int N = 3;
        Console.Write(countnndSay(N));
    }
}

// This code is contributed
// by ChitraNayal
```

Python3

```
# Python 3 program to find
# n'th term in look and
# say sequence

# Returns n'th term in
# look-and-say sequence
def countnndSay(n):

    # Base cases
    if (n == 1):
        return "1"
    if (n == 2):
        return "11"
```

```
# Find n'th term by generating
# all terms from 3 to n-1.
# Every term is generated using
# previous term

# Initialize previous term
s = "11"
for i in range(3, n + 1):

    # In below for loop,
    # previous character is
    # processed in current
    # iteration. That is why
    # a dummy character is
    # added to make sure that
    # loop runs one extra iteration.
    s += '$'
    l = len(s)

    cnt = 1 # Initialize count
            # of matching chars
    tmp = "" # Initialize i'th
             # term in series

    # Process previous term to
    # find the next term
    for j in range(1 , l):

        # If current character
        # doesn't match
        if (s[j] != s[j - 1]):

            # Append count of
            # str[j-1] to temp
            tmp += str(cnt + 0)

            # Append str[j-1]
            tmp += s[j - 1]

            # Reset count
            cnt = 1

        # If matches, then increment
        # count of matching characters
```

```
            else:
                cnt += 1

        # Update str
        s = tmp
    return s;

# Driver Code
N = 3
print(countnndSay(N))

# This code is contributed
# by ChitraNayal
```

PHP

```
<?php
// PHP program to find
// n'th term in look
// and say sequence

// Returns n'th term in
// look-and-say sequence
function countnndSay($n)
{
    // Base cases
    if ($n == 1)
        return "1";
    if ($n == 2)
        return "11";

    // Find n'th term by generating
    // all terms from 3 to n-1.
    // Every term is generated
    // using previous term

    // Initialize previous term
    $str = "11";
    for ($i = 3;
         $i <= $n; $i++)
    {
        // In below for loop,
        // previous character is
        // processed in current
```

```
        // iteration. That is why
        // a dummy character is
        // added to make sure that
        // loop runs one extra iteration.
        $str = $str.'$';
        $len = strlen($str);

        $cnt = 1; // Initialize count of
                  // matching chars
        $tmp = ""; // Initialize i'th
                   // term in series

        // Process previous term
        // to find the next term
        for ($j = 1; $j < $len; $j++)
        {
            // If current character
            // doesn't match
            if ($str[$j] != $str[$j - 1])
            {
                // Append count of
                // str[j-1] to temp
                $tmp = $tmp.$cnt + 0;

                // Append str[j-1]
                $tmp = $tmp. $str[$j - 1];

                // Reset count
                $cnt = 1;
            }

            // If matches, then increment
            // count of matching characters
            else $cnt++;
        }

        // Update str
        $str = $tmp;
    }

    return $str;
}

// Driver Code
$N = 3;
```

```
echo countnndSay($N);
return 0;

// This code is contributed
// by ChitraNayal
?>
```

Javascript

```
<script>

// Javascript program to find n'th
// term in look and say sequence

// Returns n'th term in
// look-and-say sequence
function countnndSay(n)
{

    // Base cases
    if (n == 1)
        return "1";
    if (n == 2)
        return "11";

    // Find n'th term by generating
    // all terms from 3 to n-1.
    // Every term is generated
    // using previous term

    // Initialize previous term
    let str = "11";

    for(let i = 3; i <= n; i++)
    {

        // In below for loop, previous
        // character is processed in
        // current iteration. That is
        // why a dummy character is
        // added to make sure that loop
        // runs one extra iteration.
        str += '$';
        let len = str.length;
```

```
            // Initialize count
            // of matching chars
            let cnt = 1;

            // Initialize i'th
            // term in series
            let tmp = "";
            let arr = str.split("");

            // Process previous term
            // to find the next term
            for(let j = 1; j < len; j++)
            {

                // If current character
                // doesn't match
                if (arr[j] != arr[j - 1])
                {

                    // Append count of
                    // str[j-1] to temp
                    tmp += cnt + 0;

                    // Append str[j-1]
                    tmp += arr[j - 1];

                    // Reset count
                    cnt = 1;
                }

                // If matches, then increment
                // count of matching characters
                else cnt++;
            }

            // Update str
            str = tmp;
        }
        return str;
    }

    // Driver Code
    let N = 3;
```

```
document.write(countnndSay(N));

// This code is contributed by avanitrachhadiya2155

</script>
```

Output

```
21
```

Another Approach(Using STL): There is one more idea where we can use unordered_map from c++ stl to track the count of digits. Basic idea is to use a generator function that will generate a string from the previous string. In the count and say function we will iterate over integers from 1 to n-1 and keep updating our result.

C++

```
#include <bits/stdc++.h>
using namespace std;

// generator function returns int string from prev int
// string e.g. -> it will return '1211' for '21' ( One 2's
// and One 1)
string generator(string str)
{
    string ans = "";

    unordered_map<char, int>
        tempCount; // It is used to count integer sequence

    for (int i = 0; i < str.length() + 1; i++) {
        // when current char is different from prev one we
        // clear the map and update the ans
        if (tempCount.find(str[i]) == tempCount.end()
            && i > 0) {
            auto prev = tempCount.find(str[i - 1]);
            ans += to_string(prev->second) + prev->first;
            tempCount.clear();
        }
        // when current char is same as prev one we increase
```

```
        // it's count value
        tempCount[str[i]]++;
    }

    return ans;
}

string countnndSay(int n)
{
    string res = "1"; // res variable keep tracks of string
                      // from 1 to n-1

    // For loop iterates for n-1 time and generate strings
    // in sequence "1" -> "11" -> "21" -> "1211"
    for (int i = 1; i < n; i++) {
        res = generator(res);
    }

    return res;
}

int main()
{

    int N = 3;
    cout << countnndSay(N) << endl;
    return 0;
}
```

Java

```
import java.io.*;
import java.util.*;

class GFG {
  // generator function returns int string from prev int
  // string e.g. -> it will return '1211' for '21' ( One 2's
  // and One 1)
  static String generator(String str)
  {
    String ans = "";

    HashMap<Character, Integer>tempCount = new HashMap<>(); // It is usec
```

```
    for (int i = 0; i < str.length() + 1; i++) {
      // when current char is different from prev one we
      // clear the map and update the ans
      if (i == str.length()  || tempCount.containsKey(str.charAt(i)) == f
        ans += String.valueOf(tempCount.get(str.charAt(i-1))) + str.charA
        tempCount.clear();
      }
      // when current char is same as prev one we increase
      // it's count value

      if(i == str.length()){
        tempCount.put(null, 1);
      }
      else{
        if(tempCount.containsKey(str.charAt(i))){
          tempCount.put(str.charAt(i), tempCount.get(str.charAt(i))+1);
        }
        else{
          if(i != str.length())tempCount.put(str.charAt(i), 1);
        }
      }
    }
    return ans;
  }

  static String countnndSay(int n)
  {
    String res = "1"; // res variable keep tracks of string
    // from 1 to n-1

    // For loop iterates for n-1 time and generate strings
    // in sequence "1" -> "11" -> "21" -> "1211"
    for (int i = 1; i < n; i++) {
      res = generator(res);
    }

    return res;
  }

  // Driver Code
  public static void main(String args[])
  {
    int N = 3;
    System.out.println(countnndSay(N));
  }
```

```
}

// This code is contributed by shinjanpatra
```

Javascript

```
<script>
// generator function returns int string from prev int
// string e.g. -> it will return '1211' for '21' ( One 2's
// and One 1)
function generator(str)
{
    let ans = "";

    let tempCount = new Map(); // It is used to count integer sequence

    for ( i = 0; i < str.length + 1; i++)
    {

        // when current char is different from prev one we
        // clear the map and update the ans
        if (tempCount.has(str[i]) == false && i > 0) {
            let prev = tempCount.get(str[i - 1]);
            ans += prev.toString() + str[i - 1];
            tempCount.clear();
        }

        // when current char is same as prev one we increase
        // it's count value
        if(tempCount.has(str[i]) == false)
           tempCount.set(str[i],1);
        else
        tempCount.set(str[i],tempCount.get(str[i])+1);

    }

    return ans;
}

function countnndSay(n)
{
    let res = "1"; // res variable keep tracks of string
                   // from 1 to n-1
```

```
    // For loop iterates for n-1 time and generate strings
    // in sequence "1" -> "11" -> "21" -> "1211"
    for (let i = 1; i < n; i++) {
        res = generator(res);
    }

    return res;
}

// driver code
let N = 5;
document.write(countnndSay(N),"</br>");

// This code is contributed by shinjanpatra

</script>
```

Output

```
21
```

Thanks to Utkarsh and Neeraj for suggesting the above solution.

Please write comments if you find anything incorrect, or if you want to share more information about the topic discussed above

Next

Compare two Version numbers

RECOMMENDED ARTICLES

Page : 1 2 3

01 Convert an unbalanced bracket sequence to a balanced sequence
19, Aug 19

02 Find a valid parenthesis sequence of length K from a given valid parenthesis sequence
10, Sep 20

03 Juggler Sequence | Set 2 (Using Recursion)
02, Dec 21

04 Golomb sequence
09, Mar 18

05 Digit - Product - Sequence
27, Sep 17

06 Find n-th term in sequence 1, 1, 2, 1, 2, 3, 1, 2, 3, 4, ...
03, Nov 17

07 Stern-Brocot Sequence
19, Jan 18

08 Print n terms of Newman-Conway Sequence
01, Feb 18

Article Contributed By :

GeeksforGeeks

Vote for difficulty

Current difficulty : Medium

Easy | Normal | Medium | Hard | Expert

Improved By : ukasp, avanitrachhadiya2155, neerajpatil22, shinjanpatra, rkbhola5, mukulsomukesh, hardikkoriintern

Article Tags : Amazon, Facebook, pattern-printing, series, Zoho, Misc, Strings

Practice Tags : Zoho, Amazon, Facebook, Misc, Strings, pattern-printing, Misc, series

Improve Article | Report Issue

GeeksforGeeks
A-143, 9th Floor, Sovereign Corporate Tower,
Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments