<https://leetcode.com/problems/count-and-say/description/>

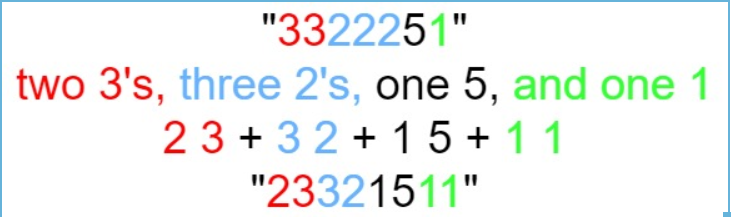
The **count-and-say** sequence is a sequence of digit strings defined by the recursive formula:

countAndSay(1) = "1"

countAndSay(n) is the way you would "say" the digit string from countAndSay(n-1), which is then converted into a different digit string.

To determine how you "say" a digit string, split it into the **minimal** number of substrings such that each substring contains exactly **one** unique digit. Then for each substring, say the number of digits, then say the digit. Finally, concatenate every said digit.

For example, the saying and conversion for digit string "3322251":



Given a positive integer n, return *the* nth *term of the* ***count-and-say*** *sequence*.

**Example 1:**

Input: n = 1

Output: "1"

Explanation: This is the base case.

**Example 2:**

Input: n = 4

Output: "1211"

Explanation:

countAndSay(1) = "1"

countAndSay(2) = say "1" = one 1 = "11"

countAndSay(3) = say "11" = two 1's = "21"

countAndSay(4) = say "21" = one 2 + one 1 = "12" + "11" = "1211"

**Constraints:**

1 <= n <= 30

**Attempt 1: 2023-10-22**

**Solution 1: DFS (10 min)**

**Style 1: Recursive + Iterative (getCur() method is iterative) way**

class Solution {

public String countAndSay(int n) {

return helper(n);

}

private String helper(int n) {

if(n == 1) {

return "1";

}

String last = helper(n - 1);

return getCur(last);

}

private String getCur(String s) {

String cur = "";

int i = 0;

for(int j = 1; j <= s.length(); j++) {

String tmp = s.charAt(i) + "";

while(j < s.length() && s.charAt(i) == s.charAt(j)) {

j++;

}

tmp = (j - i) + tmp;

i = j;

cur += tmp;

}

return cur;

}

}

**Style 2: Pure Recursive way**

class Solution {

public String countAndSay(int n) {

return helper(n);

}

private String helper(int n) {

if(n == 1) {

return "1";

}

String last = helper(n - 1);

return getCur(last);

}

private String getCur(String s) {

if(s.length() == 0) {

return "";

}

int num = getRepeatNum(s);

return num + "" + s.charAt(0) + getCur(s.substring(num));

}

private int getRepeatNum(String s) {

int count = 1;

char c = s.charAt(0);

for(int i = 1; i < s.length(); i++) {

if(c == s.charAt(i)) {

count++;

} else {

break;

}

}

return count;

}

}

**Style 3: Pure Iterative way**

class Solution {

public String countAndSay(int n) {

String res = "1";

// 从第一行开始，一行一行产生

while(n > 1) {

String temp = "";

for(int i = 0; i < res.length(); i++) {

int num = getRepeatNum(res.substring(i));

temp = temp + num + "" + res.charAt(i);

// 跳过重复的字符

i = i + num - 1;

}

n--;

// 更新

res = temp;

}

return res;

}

private int getRepeatNum(String s) {

int count = 1;

char c = s.charAt(0);

for(int i = 1; i < s.length(); i++) {

if(c == s.charAt(i)) {

count++;

} else {

break;

}

}

return count;

}

}

**Refer to**

<https://leetcode.wang/leetCode-38-Count-and-Say.html>

难在了题目是什么意思呢？

初始值第一行是 1。

第二行读第一行，1 个 1，去掉个字，所以第二行就是 11。

第三行读第二行，2 个 1，去掉个字，所以第三行就是 21。

第四行读第三行，1 个 2，1 个 1，去掉所有个字，所以第四行就是 1211。

第五行读第四行，1 个 1，1 个 2，2 个 1，去掉所有个字，所以第五航就是 111221。

第六行读第五行，3 个 1，2 个 2，1 个 1，去掉所以个字，所以第六行就是 312211。

然后题目要求输入 1 - 30 的任意行数，输出该行是啥。

# **解法一 递归**

可以看出来，我们只要知道了 n - 1 行，就可以写出第 n 行了，首先想到的就是递归。

第五行是 111221，求第六行的话，我们只需要知道每个字符重复的次数加上当前字符就行啦。

1 重复 3 次，就是 31，2 重复 2 次就是 22，1 重复 1 次 就是 11，所以最终结果就是 312211。

public String countAndSay(int n) {

//第一行就直接输出

if (n == 1) {

return "1";

}

//得到上一行的字符串

String last = countAndSay(n - 1);

//输出当前行的字符串

return getNextString(last);

}

private String getNextString(String last) {

//长度为 0 就返回空字符串

if (last.length() == 0) {

return "";

}

//得到第 1 个字符重复的次数

int num = getRepeatNum(last);

// 次数 + 当前字符 + 其余的字符串的情况

return num + "" + last.charAt(0) + getNextString(last.substring(num));

}

//得到字符 string[0] 的重复个数，例如 "111221" 返回 3

private int getRepeatNum(String string) {

int count = 1;

char same = string.charAt(0);

for (int i = 1; i < string.length(); i++) {

if (same == string.charAt(i)) {

count++;

} else {

break;

}

}

return count;

}

时间复杂度：

空间复杂度：O（1）。

# **解法二 迭代**

既然有递归，那就一定可以写出它的迭代形式。

public String countAndSay(int n) {

String res = "1";

//从第一行开始，一行一行产生

while (n > 1) {

String temp = "";

for (int i = 0; i < res.length(); i++) {

int num = getRepeatNum(res.substring(i));

temp = temp + num + "" + res.charAt(i);

//跳过重复的字符

i = i + num - 1;

}

n--;

//更新

res = temp;

}

return res;

}

//得到字符 string[0] 的重复个数，例如 "111221" 返回 3

private int getRepeatNum(String string) {

int count = 1;

char same = string.charAt(0);

for (int i = 1; i < string.length(); i++) {

if (same == string.charAt(i)) {

count++;

} else {

break;

}

}

return count;

}

时间复杂度：

空间复杂度：O（1）。