**1641. Count Sorted Vowel Strings**

Given an integer n, return *the number of strings of length*n*that consist only of vowels (*a*,*e*,*i*,*o*,*u*) and are****lexicographically sorted****.*

A string s is **lexicographically sorted** if for all valid i, s[i] is the same as or comes before s[i+1] in the alphabet.

**Example 1:**

**Input:** n = 1

**Output:** 5

**Explanation:** The 5 sorted strings that consist of vowels only are ["a","e","i","o","u"].

**Example 2:**

**Input:** n = 2

**Output:** 15

**Explanation:** The 15 sorted strings that consist of vowels only are

["aa","ae","ai","ao","au","ee","ei","eo","eu","ii","io","iu","oo","ou","uu"].

Note that "ea" is not a valid string since 'e' comes after 'a' in the alphabet.

**Example 3:**

**Input:** n = 33

**Output:** 66045

Notice that the 15 strings in the second example can be partitioned into two groups: Those which start with “a”, and those which do not. The ones which start with “a” ("aa","ae","ai","ao","au") are just the strings from example 1 with an “a” prepended, so it makes sense that there are of them. The ones which do not start with an “a” are all of the two-letter strings made from “e”, “i”, “o”, and “u”. This suggests a recursion of the form , or more generally,

where denotes the number of vowels available, and the length of the string. The boundary conditions are (with only one vowel, we can make only string consisting of that vowel repeated times) and (each vowel appears once in its own string). Given this recursion, the solution to the problem is . Coding this with top-down memorization, we find

class Solution:

    def countVowelStrings(self, n: int) -> int:

        @cache

        def v(k,n):

            if n == 1:

                return k

            if k == 1:

                return 1

            return v(k,n-1) + v(k-1,n)

        return v(5,n)