Python3 Fibonacci-ish four lines (beating 99.84%)

ye15 ★ 324 November 7, 2019 8:26 AM 98 VIEWS

4

The key to solve this problem is to derive the below recursive formula
`f(n) = (f(n-1) + f(n-2))*(k-1)`,
where `f(n)` is the number of ways to paint `n` poles following given rules, and initial condition
`f(1) = k` and `f(2) = k**2`.
Assume when painting the `n` th pole you already know `f(n-1)`, `f(n-2)`, etc. To get `f(n)`, there are two means:

1. add `n` th pole whose color is different from that of `(n-1)` th;
2. add `n` th pole whose color is the same as that of `(n-1)` th.

The contribution from 1) is `f(n-1)*(k-1)` as there are already `f(n-1)` ways to paint the first `n-1` poles and the new pole's color ought to be different from the `(n-1)` th (i.e. `k-1` choices).
The contribution from 2) is `f(n-2)*(k-1)` as this is effectively to add two poles of the same colors to first `n-2` poles. The two new poles need to have color different from that of `(n-2)` th pole which again has `k-1` choices.
`f(1) = k` is obvious since one could use any of `k` colors on one pole and `f(2) = k**2` is also obvious since it is allowed for two consecutive poles to have the same color.

The above argument is where below implemented is based upon.

```
class Solution:
    def numWays(self, n: int, k: int) -> int:
        if n == 0: return 0
        a, b = k, k*k
        for i in range(n-1): a, b = b, (a+b)*(k-1)
        return a
```

The below implementation is slightly more efficient (beating 100%).

```
class Solution:
    def numWays(self, n: int, k: int) -> int:
        if n == 0: return 0
        elif n == 1: return k
        a, b = k, k*k
        for i in range(n-2): a, b = b, (a+b)*(k-1)
        return b
```

python 3