Preimage Size of Factorial Zeroes Function

Let `f(x)` be the number of zeroes at the end of `x!`. (Recall that `x! = 1 * 2 * 3 * ... * x`, and by convention, `0! = 1`.)

For example, `f(3) = 0` because 3! = 6 has no zeroes at the end, while `f(11) = 2` because 11! = 39916800 has 2 zeroes at the end. Given `K`, find how many non-negative integers `x` have the property that `f(x) = K`.

```
Example 1:
Input: K = 0
Output: 5
Explanation: 0!, 1!, 2!, 3!, and 4! end with K = 0 zeroes.

Example 2:
Input: K = 5
Output: 0
Explanation: There is no x such that x! ends in K = 5 zeroes.
```

Note:

- `K` will be an integer in the range `[0, 10^9]`.

Solution 1

```
class Solution {
public:
    int preimageSizeFZF(int K) {
        int sum[13]={1};
        for (int i=1; i<13; i++) sum[i]=sum[i-1]*5+1;
        for (int i=12; i>=0; i--) {
            if (K/sum[i]==5) return 0;
            K=K%sum[i];
        }
        return 5;
    }
};
```

Let g(x)=f(x)-f(x-1), so f(x)=sum{g(x)}.
We can find that
g(x)=0 if x%5!=0,
g(x)>=1 if x%5==0,
g(x)>=2 if x%25==0,
...
List g(x) as follows:
x: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 ...
g(x): 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 ...
x: 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100 105 110 115 120 125 ...
g(x): 1 1 1 1 2 1 1 1 1 2 1 1 1 1 2 1 1 1 1 2 1 1 1 1 3 ...
so g(x) will always repeat a sequnce for 5 times and +1 at the last number of the sequence.
Some K is/are skipped if g(x)>1.
For example, when x=25, g(x)=2, K=5 is skipped(there is no x that f(x)=5), and when x=125, g(x)=3, K=29, 30 are skipped.
We can find that 5(=1*5), 11(=6*1+5), 17(=6*2+5), 23(=6*3+5), 29(=6*4+5), 30(=6*5), 36(=31+5), 42(=31+6+5), 48(=31+6*2+5), ... are skipped.
Let me know if there are ambiguities.

written by JohnsonTau original link here

Solution 2

1. when `n>=5`, the number of zeros in `n!` is the count of factors of 5. This function `nzero` has a complexity of `O(log(n))`.
2. therefore, `(5K)!` should have more than `K` zeros, if `K >= 1`.
3. use a binary search in `[1, 5K]` to find the smallest number `n`, so that `nzero(n) >= K`. This has a complexity of `O(log(5K))`.
4. if 'nzero(n) == K', the numbers in range `[n, n + 5 - n % 5)` are all valid candidates.
5. otherwise, there is no valid numbers.
6. the total complexity is `O(log(5K) * log(5K))`, which is very small.

```
class Solution:
    def preimageSizeFZF(self, K):
        def nzero(n):
            f = 5
            cnt = 0
            while f <= n:
                cnt += n // f
                f *= 5
            return cnt

        if K == 0:
            return 5

        min = 1
        max = K * 5
        while min < max:
            mid = (min + max) // 2
            if nzero(mid) < K:
                min = mid + 1
            else:
                max = mid

        if nzero(min) != K:
            return 0
        next = (min // 5 + 1) * 5
        return next - min
```

written by ambling original link here

Solution 3

```
Given N – No. of zeroes in N! can be found by N/5, N/25, N/125 so on
So Here use Binary search, from 0 to 10^9 – take mid element and check for No. of Z
eroes

Find lower range and Higher Range : in between that is the numbers

Lower Range : Strictly <K zeroes –
Higher Range : Strictly >=K zeroes


No. of zeroes Range once it starts – it will never decrease
 Example
 0! – 4! – 0 zeroes
5! – 9! – 1 zero
10! onwards 2 zeroes, so on
```

```java
class Solution {


    public int preimageSizeFZF(int K) {

        /*
        findRange(K)- All elements factorial <= Kzeroes

        findRange(K-1) -All elements factorial <= K-1 zeroes

        */
        return findRange(K)-findRange(K-1);



    }

    // Using Binary Search
    int findRange(int k){
        long low = 0, high = Long.MAX_VALUE;

        while(low<=high){
            long mid = low + (high-low)/2;
            if(getZeroes(mid)>k) high = mid-1;
            else
                low = mid+1;
        }

        return (int)low-1;
    }

  // get zeroes in N!
    long getZeroes(long N){
        long count = 0;
        for(long i=5;N/i>=1;i=i*5){
            count+=N/i;
        }

        return count;
    }

}
```

written by ag1119 original link here