

Factorial Trailing Digits

Author

Novemer 26, 2025

Contents

1	Formulating Problem	2
2	Grid Construction and Introduction to Series	3
2.1	Building Solution from Column's of G	3
3	Sequence	5
3.1	Lemma L1	5
3.2	Lemma LS1; Sequence is Purely Periodic	5
3.3	Lemma LS2; Product of Periodic Sequence	6
3.3.1	Series: $(i-1)b + v \bmod b$	7
3.3.2	Series: $[(i-1)b/2 + v]/2 \bmod b$	7
3.4	Lemma LS3: Product of Series dropping 5's mod b	8
4	Helper Methods	10

1 Formulating Problem

Problem: Find the last five digits before the trailing zeroes in $N!$

Let (N_n, b_d) where $N_n := 10^n$, $b_d := 10^d$, $b_d \leq N_n$ and $n, d \in \mathbb{N}^+$ be the problem of finding the last d digits of before the trailing zeros in $N!$. Then the problem we are interested in is (N_{12}, b_5) .

Let $z(n)$ be a function that gives the number of trailing zeros in n . The $z(n!)$ is given by De Polignac's formula:

$$z(n!) = \sum_{i=1}^{\lfloor \log_5(n) \rfloor} \lfloor \frac{n}{5^i} \rfloor \quad (1)$$

Using De Polignac's formula we can remove the trailing zeros to get the digits of interest.

$$(N_n, b_d) = \frac{N!}{2^{z(N!)} 5^{z(N!)}} \mod b \quad (2)$$

2 Grid Construction and Introduction to Series

For the problem (N_n, b_d) define $\lambda := \frac{N}{b} = 10^{n-d}$. Then there is a grid G that can be constructed with dimensions is $b \times \lambda$.

$$\begin{array}{cccccccc}
 1 & 2 & \cdots & 5 & \cdots & 10 & \cdots & b \\
 b+1 & b+2 & \cdots & b+5 & \cdots & b+10 & \cdots & b+b \\
 2b+1 & 2b+2 & \cdots & 2b+5 & \cdots & 2b+10 & \cdots & 2b+b \\
 \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
 (\lambda-1)b+1 & (\lambda-1)b+2 & \cdots & (\lambda-1)b+5 & \cdots & (\lambda-1)b+10 & \cdots & (\lambda-1)b+b
 \end{array} \quad (3)$$

The product of all the elements of G is $N!$. Let c_v be the v th column of G . Then we define $c_v(i)$ as the i th element in column c_v .

$$c_v(i) = (i-1)b + v \quad i \in \{1 \cdots \lambda\} \quad (4)$$

The series c_v is an arithmetic sequence which can be seen by the following, $a_1 = v$ and $a_i = a_{i-1} + b$. Then let p_v be the product of the terms in c_v series.

$$p_v = \prod_{i=1}^{\lambda} (i-1)b + v \quad (5)$$

Then we have the series $p_1, p_2, \cdots p_b$ whose product is $N!$.

2.1 Building Solution from Column's of G

Given the series p_v from grid G built for problem (N_n, b_d) we can apply sequence lemma LS4 for $v|5$ else apply LS3. This will give the $\frac{N!}{5^{z(N!)}} \mod b$. Then to get the solution (N_n, b_d) we need to use sequence lemma LS3 to remove 2's for each 5 we removed. From LS3 we can remove λ 2's at a time. However, then we would need to multiple back the extra 2's that were taken off. This is fine and can be done mod b using lemma L1.

$$Z = \lceil \frac{z(N!)}{\lambda} \rceil \quad (6)$$

$$R = Z - z(N!) \quad (7)$$

There are $4\frac{b}{10} p_v$ that can have LS4 applied to them. The following construction hold if $Z \leq 4\frac{b}{10}$ else a construction of a similar form will need to be made which is fairly trivial with the machinery built up in the lemmas. The construction is as follows:

$$S_5 = \{5, 10, \cdots, b\} \quad (8)$$

$$S_2 \subset \{2, 4, \cdots, b\} \setminus S_5 \quad (9)$$

$$|S_2| = Z \quad (10)$$

$$S = \{1, 2, \cdots, b\} \setminus S_2 \cup S_5 \quad (11)$$

Let the $P_n(a_i) \mod b$ be the product of the first n terms of the series $a_i \mod b$ (defined by LS2). Let the $P_n^5(a_i) \mod b$ be the product of the first n terms of the series a_i dropping all 5's mod b (defined by LS3).

The construction takes out λZ 2's which is R more than the number of 2's need to be taken out which is $z(N!)$. So using lemma L1 we can add back 2^R .

$$(N_n, b_d) = \prod_{\forall v_i \in S, v_j \in S_2, v_k \in S_5} [P_\lambda(c_{v_i}) \mod b] \cdot [P_\lambda(\frac{1}{2}c_{v_j}) \mod b] \cdot [P_\lambda^5(c_k) \mod b] \cdot 2^R \pmod{b} \quad (12)$$

```

1 def solve(N : int, b : int):
2     s = time()
3     assert(is_natural_number(N))
4     assert(is_natural_number(b))
5     assert(isValid_problem(N, b))
6     N, b = int(N), int(b)
7
8     z = number_of_trailing_zeros_of_factorial(N)
9     l = N // b
10    Z = math.ceil(z / l)
11    R = Z*l - z
12    assert(Z <= 4 * b/10)
13    print(f'({N}, {b}) ::<>:: lambda = {l}; z(N!) = {z}; Z = {Z} R = {R}')
14
15    product = 1
16    for v in range(1, b + 1):
17        if v % 5 == 0:
18            # pass
19            product *= product_of_sequence_drop_five(b, v, b, 1)
20        elif v % 2 == 0 and 0 < Z:
21            product *= product_of_sequence(b//2, v//2, b, 1)
22            Z -= 1
23        else:
24            product *= product_of_sequence(b, v, b, 1)
25
26        assert(not is_multiple_of_five(product))
27        product %= b
28
29    product *= mod_pow(2, R, b)
30    product %= b
31    e = time()
32    print(f'({N}, {b}) = {product} \t took {e-s} [s]')
```

3 Sequence

3.1 Lemma L1

Let $a, b \in \mathbb{N}$ then we can write a, b and ab with their 2's and 5's factored out:

$$\begin{aligned} a &= \alpha \cdot 2^{\alpha_2} \cdot 5^{\alpha_5} \\ b &= \beta \cdot 2^{\beta_2} \cdot 5^{\beta_5} \\ ab &= \lambda \cdot 2^\gamma \cdot 5^\gamma \text{ where } \gamma := \min(\alpha_2 + \beta_2, \alpha_5 + \beta_5) \end{aligned}$$

The following two relations hold for any m .

$$\alpha\beta \cdot 2^{\alpha_2+\beta_2-\gamma} \cdot 5^{\alpha_5+\beta_5-\gamma} \equiv \lambda \pmod{m} \quad (13)$$

This then allow for the following:

$$\left(\frac{a}{2^\gamma 5^\gamma} \pmod{m}\right) \left(\frac{b}{2^\gamma 5^\gamma} \pmod{m}\right) \equiv \lambda \pmod{m} \quad (14)$$

3.2 Lemma LS1; Sequence is Purely Periodic

Given a sequence $a_i = (i-1) \cdot r + v$, $a_i \pmod{m}$ where $i, r, v, m \in \mathbb{N}$ is purely periodic with period $\tau = \frac{m}{\gcd(r, m)}$. First redefine a_i as $a_0 = v$ and $a_i = a_{i-1} + r$. Then we have $a_{k+l} = a_k + lr$ from the arithmetic progression.

$$a_{k+l} \equiv a_k \pmod{m} \leftrightarrow lr \equiv 0 \pmod{m} \leftrightarrow m|lr \quad (15)$$

Let the smallest m that satisfies $m|lr$ be x . Then $m|xr$ which implies that $m = \gcd(m, xr)$. One possibility of $x = m$, is there smaller number that satisfies the condition? If xr have a common divisor $d = \gcd(x, r)$ then we get $x^*r^*d^2$ still divides m by definition. Since $m|d$ hold always we can pull d out of out $x = m$ we can reduce x to $\frac{m}{\gcd(r, m)}$ which is less than or equal to m and since there is no other larger number that we can pull out of both x^* and r^* , x is the smallest it can be then i.e. it is our period τ . [1]

3.3 Lemma LS2; Product of Periodic Sequence

Given a periodic sequence a_i with period τ and the product of the first n terms be p_n , then following allows for the computation of higher large n .

$$\prod_{i=1}^n a_i = p_{\tau}^{\lfloor \frac{n}{\tau} \rfloor} \cdot p_{n \bmod \tau} \quad (16)$$

```
1 # series a_i = (i - 1) * r + v
2 # calculates the the product of (a_1 ... a_n) mod b
3 def product_of_sequence(r : int, v : int, b : int, n : int) -> int:
4     assert(isValid_sequence(r, v))
5     assert(is_natural_number(b))
6     assert(is_natural_number(n))
7     # period of sequence mod b
8     t = b // math.gcd(r, b)
9     pt, pr = 1, 1
10    for i in range(1, min(t, n) + 1):
11        a_i = (i - 1) * r + v
12        pt *= a_i % b
13        pt %= b
14        if i == n % t:
15            pr = pt
16    product = mod_pow(pt, n//min(n, t), b)
17    if not n < t:
18        product * pr
19    return product % b
```

3.3.1 Series: $(i-1)b + v \bmod b$

We have the sequence $c_v(i) = (i-1)10^d + v = (i-1)2^d 5^d + v$. We will use lemma L1, which requires $\gcd(c_v, 5) = 1$. The only way $\gcd(c_v, 5) > 1$ is if $\gcd(v, 5) > 1$. So for the following we will deal with cases in which $\gcd(v, 5) = 1$ that is to say v cannot be a multiple of 5. Since all our elements in our sequence not divisible by 5 lemma L1 can be used to get $\bmod b$ product. Since our sequence is arithmetic we can use sequence lemma LS1 to get our sequence $\bmod b$ has period $1 = \frac{b}{\gcd(b, 5)}$ and $c_v \bmod b = v$. Using sequence lemma LS2 since we have a periodic sequence we get the following:

$$\prod_{i=1}^n a_i \equiv (p_{\tau} \bmod b)^{\lfloor \frac{\lambda}{\tau} \rfloor} \cdot (p_{n \bmod \tau} \bmod b) \pmod{b} \quad (17)$$

$$p_v \equiv v^\lambda \pmod{b} \quad (18)$$

3.3.2 Series: $[(i-1)b/2 + v]/2 \bmod b$

For the sequence $c_v(i) = (i-1)10^d + v = (i-1)2^d 5^d + v$ we want the $\bmod b$ of the product of $\frac{1}{2}c_v(i)$ first λ terms. Since $v|2$ and $\gcd(v, 5) = 1$ we can use L1, LS1 and LS2.

$$\frac{1}{2}c_v(i) = (i-1)2^{d-1}5^d + \frac{v}{2} \quad (19)$$

Using LS1 we get the period of the new sequence as $\frac{2^d 5^d}{\gcd(2^{d-1}5^d, 2^d 5^d)} = 2$. Then using LS2 we get the following and L1.

$$p_v \equiv \left[\frac{v}{2} \cdot (2^{d-1}5^d + \frac{v}{2}) \bmod b \right]^{\lfloor \frac{\lambda}{2} \rfloor} \cdot \left(\frac{v}{2} \right)^\lambda \bmod b \pmod{b} \quad (20)$$

3.4 Lemma LS3: Product of Series dropping 5's mod b

Let the series be of the form $a_i = (i - 1) \cdot r + v$. From sequence lemma LS1 we know that a_i is purely periodic with period $\tau = \frac{b}{\gcd(r, b)}$. Find the following product:

$$\prod_{i=1}^{\lambda} \frac{a_i}{5^{p_5^i}} \pmod{b} \quad (21)$$

where p_5^i is the exponent for the factor 5 for a_i 's factorization. Let $r = r^* 5^{p_r}$ and $v = v^* 5^{p_v}$ where $\gcd(r^*, 5) = \gcd(v^*, 5) = 1$. Then the series $a_i^* = (i - 1) \cdot r^* + v^*$ product of first λ terms dropping 5's mod b is the same as our original problem.

case: $p_v < p_r$

The series will not have any multiple of 5's i.e. $\gcd(a_i^*, 5) = 1 \forall i \in \mathbb{N}$. Therefore sequence lemma LS2 can be used to find the product mod b.

case: $p_r \leq p_v$

The series will of the following form $a_i^* = (i - 1) \cdot r^* + v^* 5^{p_v - p_r}$. When does $a_i^* | 5$? The second term will have end in $\{5\}$ when $p_r < p_v$ and will be $\{1, 2, 3, 4, 6, 7, 8, 9\}$ when $p_r = p_v$, it could end in 0 iff $v = 0$. Without specifics analysis to find to smallest i for $a_i^* | 5$ must be done manually. To do this take the array $[0, \dots, 9](r^* \bmod 10) + (v^* \bmod 10)$ and find the first instance of a multiple of 5, the index (1-indexed) of this position be O_5 . Then $a_{O_5}^* | 5$ is the first element in the series that is divisible by 5. The sequence $a_{O_5}^*, a_{O_5+5}^*, \dots$ are all the terms that are multiple of 5.

Therefore, we can use sequence lemma LS2 for the terms in a_i^* where $i \notin \{O_5, O_5 + 5, \dots\}$ iff $O_5 \leq 5 \wedge \tau | 5$. The condition on the period of the sequence is from the fact that each period of the sequence needs to take out each instance of $a_i^* | 5$ and need to line up with with in the period; this is true if the period of $a_i^* | 5$ is a factor of τ and $i \neq O_5 + 5k \forall k \in \mathbb{Z}$.

Therefore here does not seem to be a pattern for the terms $a_{O_5+5k}^*$ this can be seen by $a_{O_5+5k}^* | 5^2$ is periodic but it will have its own O_5^2 that needs to be calculated. There will be at most $n_O = \lfloor \log_5 \lambda \rfloor$ offsets that will needed to be calculated. And each offset $O_{5^{p_O}}$ the following must be true:

$$\tau | 5^{p_O} \quad (22)$$

$$O_{5^{p_O}} \leq \tau \quad (23)$$

$$O_{5^{p_O}} \leq 5^{p_O} \quad (24)$$

$$5^{p_O} \leq \tau \quad (25)$$

$$(26)$$

These condition are not generally going to be hold for $1 < p_O$, so the series $a_{O_5+5k}^*$ will need to be brute forced.


```

1 # series a_i = (i - 1) * r + v
2 # calculates the the product of (a_1 ... a_n) / 5^p mod b
3 # where p is the number of factors in the product
4 def product_of_sequence_drop_five(r : int, v : int, b : int, n : int) -> int:
5     assert(isValid_sequence(r, v))
6     assert(is_natural_number(b))
7     assert(is_natural_number(n))
8     assert(is_multiple_of_five(b))
9     # remove trivial factors of 5 in the series
10    pr = factors_of_five(r)
11    pv = factors_of_five(v)
12    min_factors_of_five = min(pr, pv)
13    reduce_sequence_by = 5**min_factors_of_five
14    r = r // reduce_sequence_by
15    v = v // reduce_sequence_by
16    assert(not is_multiple_of_five(math.gcd(r, v)))
17
18    if pv < pr:
19        return product_of_sequence(r, v, b, n)
20
21    # period of the sequence
22    t = b // math.gcd(r, b)
23
24    # check if series has a any more multiple of 5's
25    offset = first_multiple_of_five(r, v)
26    assert(offset <= 5)
27    # ignore the a_i|5
28    pt, pr = 1, 1
29    for i in range(1, min(t, n) + 1):
30        if offset is not None and is_multiple_of_five(i - offset):
31            continue
32        a_i = (i - 1) * r + v
33        assert(not is_multiple_of_five(a_i))
34        pt *= a_i % b
35        pt %= b
36        if i == n % t:
37            pr = pt
38    ignore_ever_fifth = mod_pow(pt, n//min(n, t), b)
39    if not n < t:
40        ignore_ever_fifth = (ignore_ever_fifth * pr) % b
41
42    # get product of a_i that where skipped
43    p5s = 1
44    if offset is not None:
45        for i in range(offset, n + 1, 5):
46            a_i = drop_fives((i - 1) * r + v)
47            p5s *= a_i % b
48            p5s %= b
49    return (ignore_ever_fifth * p5s) % b

```

4 Helper Methods

```
1 from functools import lru_cache
2 import math
3 from time import time
4
5 @lru_cache
6 def is_whole_number(n : int) -> bool:
7     if isinstance(n, int):
8         return True
9     elif isinstance(n, float):
10        return n.is_integer()
11    else:
12        return False
13
14 @lru_cache
15 def is_natural_number(n) -> bool:
16     return not n == 0 and is_whole_number(n)
17
18 @lru_cache
19 def is_multiple_of_five(n : int) -> bool:
20     return is_whole_number(n) and (n % 10 in {0, 5})
21
22 # number of trailing zeros for n!
23 def number_of_trailing_zeros_of_factorial(n : int) -> int:
24     assert(is_natural_number(n))
25     n = int(n)
26     R = range(1, int(math.log(n) / math.log(5)) + 1)
27     z = lambda i : n // 5**i
28     return sum(map(z, R))
29 assert(number_of_trailing_zeros_of_factorial(9) == 1)
30 assert(number_of_trailing_zeros_of_factorial(10) == 2)
31 assert(number_of_trailing_zeros_of_factorial(20) == 4)
32
33 PRIMES = [2]
34
35 @lru_cache
36 def prime_factors(n : int) -> dict:
37     assert(is_natural_number(n))
38     global PRIMES
39     n = int(n)
40     f : dict[int][int] = {}
41
42     last_prime = None
43     for p in PRIMES:
44         while n % p == 0:
45             n //= p
46             f[p] = f.get(p, 0) + 1
47         last_prime = p
48         if n < p*p:
49             break
50     assert(last_prime is not None)
51     q = last_prime
52     while q*q <= n:
53         if n % q == 0:
54             PRIMES.append(q)
55         while n % q == 0:
56             n //= q
```

```

57         f[q] = f.get(q, 0) + 1
58         q += 1
59     if n > 1:
60         PRIMES.append(n)
61         f[n] = f.get(n, 0) + 1
62     return f
63 assert(prime_factors(2**2 * 5**33 * 7) == {2:2, 5:33, 7:1})
64
65 @lru_cache
66 def factors_of_five(n):
67     if n == 0:
68         return 0
69     p = 0
70     while n % 5 == 0:
71         n //= 5
72         p += 1
73     return p
74
75 # computes a^n mod b
76 # implements Right-to-left binary method
77 # [https://en.wikipedia.org/wiki/Modular_exponentiation#Right-to-
78     left_binary_method]
79 @lru_cache
80 def mod_pow(a : int, n : int, b : int) -> int:
81     assert(is_natural_number(a))
82     assert(is_natural_number(b))
83     assert(is_whole_number(n) and 0 <= n)
84     if b == 1:
85         return 0
86     # (b-1) * (b-1) should not overflow
87     product = 1
88     a = a % b
89     while 0 < n:
90         if n % 2 == 1:
91             product = (product * a) % b
92         n = n >> 1
93         a = (a * a) % b
94     return product
95
96 def isValid_b(b : int) -> bool:
97     return math.log10(b) % 1 == 0 and b > 1
98
99 def isValid_N(N : int) -> bool:
100     return math.log10(N) % 1 == 0 and N > 1
101
102 def isValid_problem(N : int, b: int) -> bool:
103     return isValid_N(N) and isValid_b(b) and b <= N
104
105 def isValid_sequence(r : int, v : int) -> bool:
106     return r % 1 == 0 and is_natural_number(v) and 0 < v
107
108 def drop_fives(n : int) -> int:
109     if n == 0:
110         return n
111     while n % 5 == 0:
112         n //= 5
113     return n
114 assert(drop_fives(5**23 * 7) == 7)

```

```
115 def first_multiple_of_five(r : int, v : int) -> int | None:
116     assert(is_natural_number(r))
117     assert(is_natural_number(v))
118     for d in range(10):
119         if is_multiple_of_five((d*r + v) % 10):
120             return d + 1
121     return None
```