Editorial: Digital Divisibility Challenge

December 29, 2025

Core Observation

The problem requires counting substrings in a digit string that are divisible by 2^L, where L is the substring length. A critical insight is that for $L > 60$, 2^L exceeds the maximum value of any L-digit number (which is $10^L - 1$). Consequently, no L-digit number can be divisible by 2^L when $L > 60$. For $L \leq 60$, we efficiently compute substring values modulo 2^L using a sliding window with modular arithmetic, leveraging the property that $10^L \equiv 0 \pmod{2^L}$ (since $10^L = (2 \cdot 5)^L = 2^L \cdot 5^L$).

Solution

We iterate over all valid substring lengths L from 1 to $\min(n, 60)$, where n is the input string length. For each L:

1. **Initialization**:
   - If $L > n$, skip this L.
   - Set $M = 2^L$ and $\text{mask} = M - 1$ (enabling fast modulo via bitwise AND).
   - Initialize $\text{current} = 0$ to store the numeric value of the current window modulo M.
2. **First window computation**: Process the initial window $s[0..L-1]$:

   $$\text{for } j = 0 \text{ to } L - 1: \quad \text{current} \leftarrow (\text{current} \times 10 + (s[j] - \text{'0'}))\ \&\ \text{mask}$$

   If $\text{current} = 0$, the substring is divisible by 2^L; increment the count.
3. **Sliding window updates**: For each subsequent window starting at index i $(0 \leq i < n - L)$:
   - The window shifts from $s[i..i + L - 1]$ to $s[i + 1..i + L]$.
   - Using $10^L \equiv 0 \pmod{M}$, the update simplifies to:

     $$\text{current} \leftarrow (\text{current} \times 10 + (s[i + L] - \text{'0'}))\ \&\ \text{mask}$$

   - If $\text{current} = 0$, increment the count.

The total count is the sum of valid substrings across all $L \leq 60$. Substrings with $L > 60$ are skipped per the core observation.

Complexity Analysis

- **Time Complexity**: The algorithm processes L from 1 to 60 (constant). For each L, it scans the string once in $O(n)$ time (each window update is $O(1)$ due to bitwise operations). Total time is $O(60 \cdot n) = O(n)$.
- **Space Complexity**: Only $O(1)$ auxiliary space is used per L (storing M, mask, current, and loop variables). With L bounded by 60, total space remains $O(1)$.

This approach efficiently handles the problem by eliminating $L > 60$ cases and using modular arithmetic for linear-time processing of small L.