□ Teemo Attacking Documentation

1. Problem Statement

You are given:

- A non-decreasing list of time series of integers, where each element represents the time Teemo attacks Ashe.
- An integer duration is the number of seconds each poison attack lasts.

Each attack causes Ashe to be poisoned from time t to t + duration - 1. If a new attack happens before the current poison effect ends, the timer resets, extending the poisoned time instead of stacking.

Goal: Return the total number of seconds Ashe is poisoned.

2. Intuition

Each attack poisons Ashe for seconds. However, if two attacks are close enough to overlap, we must not double-count the overlapping seconds.

So we:

- Traverse each pair of consecutive attacks.
- Add either:
    - The full duration if no overlap.
    - Or the gap between attacks (which is the non-overlapping part) if an overlap exists.
- Don't forget to add the full duration for the last attack.

3. Key Observations

- Poison does not stack; it extends the poisoned time.
- If next attack time < current poison end, an overlap happens.
- If next attack time ≥ current poison end, no overlap.
- We must compare the gap between attacks and duration.

4. **Approach**

- Initialize total_poisoned = 0.
- Iterate through timeSeries from i = 1 to end:
  - Let gap = timeSeries[i] - timeSeries[i - 1].
  - Add min(gap, duration) to total_poisoned.
- After the loop, add duration for the last attack.
- Return total_poisoned.

5. **Edge Cases**

- timeSeries is empty → Return 0.
- Only one attack → Return duration.
- duration = 0 → Always return 0 regardless of timeSeries.

6. **Complexity Analysis**

Time Complexity

- O(n) where n = len(timeSeries) we loop through the list once.

Space Complexity

- O(1) only a few variables used, no extra data structures.

7. **Alternative Approaches**

- Brute-force (Inefficient): Use a set to track all poisoned seconds and return its size.
  - Time: O(n·duration)
  - Space: O(n·duration)
  - Not optimal for large inputs.
- Sliding Window / Interval Merging: Not necessary since each interval is exactly duration long and non-decreasing.

8. Test Cases

timeSeries	duration	Expected Output	Explanation
[1, 4]	2	4	Poisoned: [1,2] and [4,5] = 4 seconds
[1, 2]	2	3	Poisoned: [1,2,3] (overlap at 2)
[1]	5	5	Only one attack
[]	10	0	No attack
[1, 100]	1	2	No overlap

9. Final Thoughts

This is a great example of:

- Handling intervals with overlapping logic.
- Efficient linear solutions by smart arithmetic.

It also reinforces that not every problem needs a complicated structure like interval merging or segment trees simple math and clean iteration are enough.