Teemo Attacking

In LLP world, there is a hero called Teemo and his attacking can make his enemy Ashe be in poisoned condition. Now, given the Teemo's attacking **ascending** time series towards Ashe and the poisoning time duration per Teemo's attacking, you need to output the total time that Ashe is in poisoned condition.

You may assume that Teemo attacks at the very beginning of a specific time point, and makes Ashe be in poisoned condition immediately.

Example 1:

```
Input: [1,4], 2
Output: 4
Explanation: At time point 1, Teemo starts attacking Ashe and makes Ashe be poisone
d immediately.
This poisoned status will last 2 seconds until the end of time point 2.
And at time point 4, Teemo attacks Ashe again, and causes Ashe to be in poisoned st
atus for another 2 seconds.
So you finally need to output 4.
```

Example 2:

```
Input: [1,2], 2
Output: 3
Explanation: At time point 1, Teemo starts attacking Ashe and makes Ashe be poisone
d.
This poisoned status will last 2 seconds until the end of time point 2.
However, at the beginning of time point 2, Teemo attacks Ashe again who is already
in poisoned status.
Since the poisoned status won't add up together, though the second poisoning attack
 will still work at time point 2, it will stop at the end of time point 3.
So you finally need to output 3.
```

Note:

1. You may assume the length of given time series array won't exceed 10000.
2. You may assume the numbers in the Teemo's attacking time series and his poisoning time duration per attacking are non-negative integers, which won't exceed 10,000,000.

Solution 1

```
class Solution(object):
    def findPoisonedDuration(self, timeSeries, duration):
        ans = duration * len(timeSeries)
        for i in range(1,len(timeSeries)):
            ans -= max(0, duration - (timeSeries[i] - timeSeries[i-1]))
        return ans
```

written by lilixu original link here

Solution 2

For each `begin` followed by `t`
If `t` is within previous duration `[begin, begin + duration]` then increase total by `t - begin`
If `t` in out of previous duration `[begin, begin + duration]` then increase total by `duration`
In both cases update `begin` to the new begin time `t`

```
    public int findPoisonedDuration(int[] timeSeries, int duration) {
        if (timeSeries.length == 0) return 0;
        int begin = timeSeries[0], total = 0;
        for (int t : timeSeries) {
            total+= t < begin + duration ? t - begin : duration;
            begin = t;
        }
        return total + duration;
    }
```

written by yuxiangmusic original link here

Solution 3

The same idea as https://leetcode.com/problems/merge-intervals/
Algorithm:

1. Use two variable to record current start and end point.
2. If the start of new interval is greater than current end, meaning NO overlapping, we can sum the current interval length to result and then update start and end.
3. Otherwise just update the current end;

```
public class Solution {
    public int findPosisonedDuration(int[] timeSeries, int duration) {
        if (timeSeries == null || timeSeries.length == 0 || duration == 0) return
0;

        int result = 0, start = timeSeries[0], end = timeSeries[0] + duration;
        for (int i = 1; i < timeSeries.length; i++) {
            if (timeSeries[i] > end) {
                result += end - start;
                start = timeSeries[i];
            }
            end = timeSeries[i] + duration;
        }
        result += end - start;

        return result;
    }
}
```

written by shawngao original link here