[Click on the logo to find **Problem Statement**]

Intuition

The code calculates the number of car fleets on a straight road, where each car has a starting position and a constant speed. The idea is to sort the cars based on their starting positions and process them in reverse order. For each car, the time it takes to reach the target is calculated, and this information is stored in a stack. If the current car takes less or equal time to reach the target than the previous car in the stack, they form a fleet, and the previous car is removed from the stack. The final result is the length of the stack, representing the total number of car fleets. This approach efficiently handles the fleet formation scenario by processing cars in reverse order and maintaining the necessary information in a stack.

Approach

1. **Pair Formation:**
    - Create a list of pairs, where each pair consists of the car's initial position and its constant speed. This pairs list is denoted as `pair`.
2. **Sort by Position:**
    - Sort the pairs based on the car's initial position in ascending order.
3. **Process in Reverse Order:**
    - Iterate through the sorted pairs in reverse order (from the car closest to the target to the farthest).
    - For each car, calculate the time it takes to reach the target using the formula `(target - position) / speed`.
    - Keep track of these times in a stack.
4. **Fleet Formation:**
    - If the current car takes less or equal time to reach the target than the previous car in the stack, they form a fleet. Remove the previous car from the stack.
5. **Count Fleets:**

- The length of the stack after processing all cars represents the total number of car fleets.

6. **Return Result:**
   - Return the count of fleets as the final result.

Complexity

- Time complexity: O(n log n)
- Space complexity: O(n)

Code

```
class Solution:
    def carFleet(self, target: int, position: List[int],
                                    speed: List[int]) -> int:
        pair = [[p,s] for p,s in zip(position, speed)]
        stack = []

        for p,s in sorted(pair)[::-1]:
            stack.append((target - p) / s)
            if len(stack) >= 2 and stack[-1] <= stack[-2]:
                stack.pop()

        return len(stack)
```

If you want to see more solutions to coding problems, you can visit: