1491. Average Salary Excluding the Minimum and Maximum Salary

Easy Array Sorting Leetcode Link

Problem Description

The given LeetCode problem presents you with an array named `salary` which contains integers that represent the salaries of employees. Each integer in the array is unique and corresponds to the salary of an individual employee. Your task is to calculate the average salary after excluding the maximum and minimum salary from the calculation. The result should be close to the actual average, within a tolerance of `10^-5`. Basically, you're filtering out the outliers (the highest and the lowest salaries) to find the average salary that represents the majority of employees more accurately.

Intuition

To solve this problem, the intuitive approach is to first eliminate the outliers since we want an average that doesn't include the extremes (minimum and maximum values). We do this by:

1. Calculating the sum of all elements in the `salary` array, which would initially include the minimum and maximum salaries.
2. Then, subtract the minimum salary and the maximum salary from this sum to exclude their influence on the overall average.
3. Finally, we divide the adjusted sum by the total count of salaries excluding the two removed salaries (the minimum and maximum ones).

The length of the adjusted salary array is `len(salary) - 2` because we have removed two elements (the smallest and the largest).

By performing this simple arithmetic operation, we reach the efficient solution to calculate the required average. The provided code neatly encapsulates this approach and directly uses Python's built-in functions `sum()`, `min()`, and `max()` to compute it. The results are then returned after performing the division, giving us the average salary excluding the minimum and maximum salaries.

Solution Approach

The Reference Solution Approach leverages built-in functions in Python without the need for any additional algorithms, data structures, or patterns. The approach is very straight-forward and efficient, following a simple set of operations:

1. **Calculate Total Salary**: This is done by using Python's `sum()` function that adds up all elements in the `salary` array.

   ```
   total_salary = sum(salary)
   ```

2. **Find Minimum and Maximum Salary**: Python provides `min()` and `max()` functions that are used to find the smallest and largest values in the `salary` array, respectively.

   ```
   minimum_salary = min(salary)
   maximum_salary = max(salary)
   ```

3. **Exclude Minimum and Maximum Salaries**: To exclude these values from our calculations, we subtract them from the `total_salary`.

   ```
   adjusted_salary = total_salary - minimum_salary - maximum_salary
   ```

4. **Calculate Average Salary**: To find the average, we divide the `adjusted_salary` by the number of remaining salaries. Since we excluded the minimum and maximum salaries, we have `len(salary) - 2` employees remaining.

   ```
   average_salary = adjusted_salary / (len(salary) - 2)
   ```

5. **Return the Result**: The final step in our solution is to return the calculated average salary.

   ```
   return average_salary
   ```

The provided code executes these steps sequentially in a single method, compacting the operations into a concise one-liner. This straightforward implementation minimizes complexity and makes the code easy to understand and maintain. It's a typical pattern for solving problems involving arrays and basic statistics in programming, utilizing only the fundamental built-in functions and operations of the Python language.

Example Walkthrough

Let's walk through a small example to illustrate the solution approach. Suppose we have an array `salary` that represents the salaries of employees as follows:

```
salary = [4000, 3000, 5000, 2000, 6000]
```

Step 1: Calculate Total Salary

Using the `sum()` function, we sum up all salaries:

```
total_salary = sum(salary) # total_salary = 20000
```

Step 2: Find Minimum and Maximum Salary

We find the minimum and maximum salaries using `min()` and `max()` functions:

```
minimum_salary = min(salary) # minimum_salary = 2000
maximum_salary = max(salary) # maximum_salary = 6000
```

Step 3: Exclude Minimum and Maximum Salaries

Next, we subtract both these values from the total salary to exclude their effect:

```
adjusted_salary = total_salary - minimum_salary - maximum_salary # adjusted_salary = 12000
```

Step 4: Calculate Average Salary

To calculate the average, we now divide by the number of remaining salaries (which is `len(salary) - 2` because we're not counting the minimum and maximum):

```
count = len(salary) - 2 # count = 3
average_salary = adjusted_salary / count # average_salary = 4000
```

Step 5: Return the Result

Finally, we return the average salary, excluding the highest and lowest salaries:

```
return average_salary # returns 4000
```

This average salary (`4000`) is the output, and in this example, it is equal to one of the employee's salaries, which is more representative of the majority when the outliers are removed.

Python Solution

```python
from typing import List  # Import List from typing to use for type hinting

class Solution:
    def average(self, salaries: List[int]) -> float:
        """
        Calculate the average salary after excluding the minimum and maximum salary.

        :param salaries: List of salaries from which to calculate the average.
        :return: The average salary as a float.
        """
        # Calculate the sum of all salaries and subtract the minimum and maximum salary
        total_salary_except_min_max = sum(salaries) - min(salaries) - max(salaries)

        # Calculate the average salary by dividing the total_salary_except_min_max by
        # the number of salaries minus 2 (since we excluded the min and max salaries)
        average_salary = total_salary_except_min_max / (len(salaries) - 2)

        return average_salary

```

Java Solution

```java
class Solution {
    public double average(int[] salary) {
        // Initialize sum of salaries to 0
        int sum = 0;
        // Initialize minimum salary to the largest possible integer value
        int minSalary = Integer.MAX_VALUE;
        // Initialize maximum salary to the smallest possible integer value
        int maxSalary = Integer.MIN_VALUE;

        // Iterate over all the salaries
        for (int value : salary) {
            // Update minimum salary if a smaller salary is found
            minSalary = Math.min(minSalary, value);
            // Update maximum salary if a larger salary is found
            maxSalary = Math.max(maxSalary, value);
            // Accumulate sum of all salaries
            sum += value;
        }

        // Subtract the extreme values (min and max salary) from the total sum
        sum -= (minSalary + maxSalary);

        // Calculate average: divide the modified sum by the number of elements excluding the two extremes
        return sum * 1.0 / (salary.length - 2);
    }
}

```

C++ Solution

```cpp
#include <vector>
#include <algorithm>

class Solution {
public:
    // Function to calculate the average salary excluding the minimum and maximum salary.
    double average(std::vector<int>& salary) {
        int sum = 0; // Initialize sum of all salaries to 0
        int minSalary = INT_MAX; // Initialize minimum salary to the highest possible integer
        int maxSalary = INT_MIN; // Initialize maximum salary to the lowest possible integer

        // Loop to calculate the total sum and find min and max salary
        for (int currentValue : salary) {
            sum += currentValue; // Add the current value to the sum
            minSalary = std::min(minSalary, currentValue); // Update minSalary if the current value is smaller
            maxSalary = std::max(maxSalary, currentValue); // Update maxSalary if the current value is larger
        }

        // Remove the min and max salary from the sum
        sum -= (minSalary + maxSalary);

        // Calculate the average by dividing the sum by the size of the array minus 2 (since we excluded 2 salaries)
        return static_cast<double>(sum) / (salary.size() - 2);
    }
};

```

Typescript Solution

```typescript
// Define a function to calculate the average salary excluding the minimum and maximum salary values.
function average(salary: number[]): number {
    // Initialize maximum salary seen so far to the lowest possible number.
    let maxSalary = Number.MIN_VALUE;
    // Initialize minimum salary seen so far to the highest possible number.
    let minSalary = Number.MAX_VALUE;
    // Initialize sum of all salaries to zero.
    let totalSalary = 0;

    // Iterate through each salary value in the array.
    for (const value of salary) {
        // Add current salary to the total sum.
        totalSalary += value;
        // Update maximum salary if the current value is greater.
        maxSalary = Math.max(maxSalary, value);
        // Update minimum salary if the current value is lower.
        minSalary = Math.min(minSalary, value);
    }

    // Subtract the maximum and minimum salary from the total sum
    // and divide by the length of the array minus 2 (excuding min and max salaries)
    return (totalSalary - maxSalary - minSalary) / (salary.length - 2);
}

```

Time and Space Complexity

Time Complexity

The given Python code consists of finding the minimum and maximum values, summing the elements of the salary list, subtracting the minimum and maximum values from the sum, and finally computing the average by dividing by the number of elements minus 2. Here's how the time complexity breaks down:

- `min(salary)` and `max(salary)` each run in `O(n)` time where `n` is the number of elements in the salary list, as the function needs to check each element to determine the minimum or maximum value.
- `sum(salary)` also runs in `O(n)` time, as it adds each element in the list once.
- Subtracting the minimum and maximum from the sum and dividing by `n - 2` are constant time operations, `O(1)`.

All of these are sequential operations. Therefore, the overall time complexity is `O(n)` since we sum the individual complexities: `O(n) + O(n) + O(n) + O(1)` which simplifies to `O(n)`.

Space Complexity

The space complexity of the code is the amount of memory used above and beyond the input itself. Since the code only uses additional space for a few variables (`s` which holds the sum after subtracting the minimum and maximum salaries), the space complexity is `O(1)`, or constant space, because the space used does not increase with the size of the input list.