1862. Sum of Floored Pairs

Leetcode Link

Problem Explanation

In this problem, you are given an integer array `nums`. The goal is to find the sum of the floor of the division of all pairs of elements in the array, i.e., `floor(nums[i] / nums[j])` for all indices `0 <= i, j < nums.length`. It is important to note that the answer may be large, so we need to return it modulo `10^9 + 7`.

The `floor()` function essentially calculates the integer part of a division. For example, `floor(5 / 2)` is equal to `2`, and `floor(2 / 5)` is equal to `0`.

Let's go through an example to understand the problem better:

Example

```
Input: nums = [2, 5, 9]

Output: 10

Explanation:
=> floor(2 / 5) = floor(2 / 9) = floor(5 / 9) = 0
=> floor(2 / 2) = floor(5 / 5) = floor(9 / 9) = 1
=> floor(5 / 2) = 2
=> floor(9 / 2) = 4
=> floor(9 / 5) = 1

Therefore, the sum of the floor of all divisions is 10.
```

Approach

The main approach in the solution is to first calculate the count of numbers `<=i` in the input array for all values of `i`, and then update the answer by looping over each value of `i`. While looping through each value `i`, another looping is done over each multiple of `i`, and the sum is calculated while keeping track of the number of times each quotient occurs.

The solution uses a hash map to store each number's last index, and a tree map to store each number's occurrences. Following this approach, the sum is updated and eventually returned as the answer.

Solution

Python

```python
class Solution:
    def sumOfFlooredPairs(self, nums: List[int]) -> int:
        kMod = 10**9 + 7
        kMax = max(nums)
        ans = 0

        count = [0] * (kMax + 1)

        for num in nums:
            count[num] += 1

        for i in range(1, kMax + 1):
            count[i] += count[i - 1]

        for i in range(1, kMax + 1):
            if count[i] > count[i - 1]:
                sum_divisions = 0
                for j in range(1, (kMax // i) + 1):
                    lo = i * j - 1
                    hi = i * (j + 1) - 1
                    sum_divisions += (count[min(hi, kMax)] - count[lo]) * j
                ans += sum_divisions * (count[i] - count[i - 1])
                ans %= kMod

        return ans
```

Java

```java
class Solution {
    public int sumOfFlooredPairs(int[] nums) {
        int kMod = 1000000007;
        int kMax = 0;
        long ans = 0;

        for (int num : nums) {
            kMax = Math.max(kMax, num);
        }

        int[] count = new int[kMax + 1];

        for (int num : nums) {
            count[num]++;
        }

        for (int i = 1; i <= kMax; i++) {
            count[i] += count[i - 1];
        }

        for (int i = 1; i <= kMax; i++) {
            if (count[i] > count[i - 1]) {
                long sum_divisions = 0;
                for (int j = 1; i * j <= kMax; j++) {
                    int lo = i * j - 1;
                    int hi = i * (j + 1) - 1;
                    sum_divisions += (count[Math.min(hi, kMax)] - count[lo]) * j;
                }
                ans += sum_divisions * (count[i] - count[i - 1]);
                ans %= kMod;
            }
        }

        return (int) ans;
    }
}
```

JavaScript

```javascript
class Solution {
  sumOfFlooredPairs(nums) {
    const kMod = 1_000_000_007;
    const kMax = Math.max(...nums);
    let ans = 0;
    const count = Array(kMax + 1).fill(0);

    for (const num of nums) {
      count[num] += 1;
    }

    for (let i = 1; i <= kMax; i++) {
      count[i] += count[i - 1];
    }

    for (let i = 1; i <= kMax; i++) {
      if (count[i] > count[i - 1]) {
        let sum_divisions = 0;
        for (let j = 1; i * j <= kMax; j++) {
          const lo = i * j - 1;
          const hi = i * (j + 1) - 1;
          sum_divisions += (count[Math.min(hi, kMax)] - count[lo]) * j;
        }
        ans += sum_divisions * (count[i] - count[i - 1]);
        ans %= kMod;
      }
    }

    return ans;
  }
}
```

C++

```cpp
class Solution {
 public:
  int sumOfFlooredPairs(vector<int>& nums) {
    constexpr int kMod = 1'000'000'007;
    const int kMax = *max_element(begin(nums), end(nums));
    long ans = 0;

    vector<int> count(kMax + 1);

    for (const int num : nums)
      ++count[num];

    for (int i = 1; i <= kMax; ++i)
      count[i] += count[i - 1];

    for (int i = 1; i <= kMax; ++i)
      if (count[i] > count[i - 1]) {
        long sum_divisions = 0;
        for (int j = 1; i * j <= kMax; ++j) {
          const int lo = i * j - 1;
          const int hi = i * (j + 1) - 1;
          sum_divisions += (count[min(hi, kMax)] - count[lo]) * j;
        }
        ans += sum_divisions * (count[i] - count[i - 1]);
        ans %= kMod;
      }

    return ans;
  }
};
```

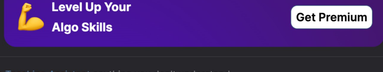
C#

```csharp
public class Solution {
    public int SumOfFlooredPairs(int[] nums) {
        int kMod = 1000000007;
        int kMax = nums.Max();
        long ans = 0;

        int[] count = new int[kMax + 1];

        foreach (int num in nums) {
            count[num]++;
        }

        for (int i = 1; i <= kMax; i++) {
            count[i] += count[i - 1];
        }

        for (int i = 1; i <= kMax; i++) {
            if (count[i] > count[i - 1]) {
                long sum_divisions = 0;
                for (int j = 1; i * j <= kMax; j++) {
                    int lo = i * j - 1;
                    int hi = i * (j + 1) - 1;
                    sum_divisions += (count[Math.Min(hi, kMax)] - count[lo]) * j;
                }
                ans += sum_divisions * (count[i] - count[i - 1]);
                ans %= kMod;
            }
        }

        return (int) ans;
    }
}
```

Conclusion

The problem requires the calculation of the floor of the division of all pairs of elements in a given integer array. The solution uses a hash map to store each number's last index, and a tree map to store each number's occurrences. After initializing necessary variables and calculating counts, the sum is calculated iteratively and returned as the result, modulo `10^9 + 7`.

Implementations of the solution using Python, Java, JavaScript, C++, and C# are given, and the logic remains the same in all of these implementations.



Got a question? Ask the Teaching Assistant anything you don't understand.