

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

```
1
2
3 Input: nums = [2, 5, 9]
4
5 Output: 10
6
7 Explanation:
8 => floor(2 / 5) = floor(2 / 9) = floor(5 / 9) = 0
9 => floor(2 / 2) = floor(5 / 5) = floor(9 / 9) = 1
10 => floor(5 / 2) = 2
11 => floor(9 / 2) = 4
12 => floor(9 / 5) = 1
13
14 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

```
1
2 python
3 class Solution:
4     def sumOfFlooredPairs(self, nums: List[int]) -> int:
5         kMod = 10**9 + 7
6         kMax = max(nums)
7         ans = 0
8
9         count = [0] * (kMax + 1)
10
11         for num in nums:
12             count[num] += 1
13
14         for i in range(1, kMax + 1):
15             count[i] += count[i - 1]
16
17         for i in range(1, kMax + 1):
18             if count[i] > count[i - 1]:
19                 sum_divisions = 0
20                 for j in range(1, (kMax // i) + 1):
21                     lo = i * j - 1
22                     hi = i * (j + 1) - 1
23                     sum_divisions += (count[min(hi, kMax)] - count[lo]) * j
24                 ans += sum_divisions * (count[i] - count[i - 1])
25                 ans %= kMod
26
27         return ans
```

Java

```
1
2 java
3 class Solution {
4     public int sumOfFlooredPairs(int[] nums) {
5         int kMod = 1000000007;
6         int kMax = 0;
7         long ans = 0;
8
9         for (int num : nums) {
10             kMax = Math.max(kMax, num);
11         }
12
13         int[] count = new int[kMax + 1];
14
15         for (int num : nums) {
16             count[num]++;
17         }
18
19         for (int i = 1; i <= kMax; i++) {
20             count[i] += count[i - 1];
21         }
22
23         for (int i = 1; i <= kMax; i++) {
24             if (count[i] > count[i - 1]) {
25                 long sum_divisions = 0;
26                 for (int j = 1; i * j <= kMax; j++) {
27                     int lo = i * j - 1;
28                     int hi = i * (j + 1) - 1;
29                     sum_divisions += (count[Math.min(hi, kMax)] - count[lo]) * j;
30                 }
31                 ans += sum_divisions * (count[i] - count[i - 1]);
32                 ans %= kMod;
33             }
34         }
35
36         return (int) ans;
37     }
38 }
```

JavaScript

```
1
2 javascript
3 class Solution {
4     sumOfFlooredPairs(nums) {
5         const kMod = 1_000_000_007;
6         const kMax = Math.max(...nums);
7         let ans = 0;
8         const count = Array(kMax + 1).fill(0);
9
10         for (const num of nums) {
11             count[num] += 1;
12         }
13
14         for (let i = 1; i <= kMax; i++) {
15             count[i] += count[i - 1];
16         }
17
18         for (let i = 1; i <= kMax; i++) {
19             if (count[i] > count[i - 1]) {
20                 let sum_divisions = 0;
21                 for (let j = 1; i * j <= kMax; j++) {
22                     const lo = i * j - 1;
23                     const hi = i * (j + 1) - 1;
24                     sum_divisions += (count[Math.min(hi, kMax)] - count[lo]) * j;
25                 }
26                 ans += sum_divisions * (count[i] - count[i - 1]);
27                 ans %= kMod;
28             }
29         }
30
31         return ans;
32     }
33 }
```

C++

```
1
2 cpp
3 class Solution {
4 public:
5     int sumOfFlooredPairs(vector<int>& nums) {
6         constexpr int kMod = 1'000'000'007;
7         const int kMax = *max_element(begin(nums), end(nums));
8         long ans = 0;
9
10         vector<int> count(kMax + 1);
11
12         for (const int num : nums)
13             ++count[num];
14
15         for (int i = 1; i <= kMax; ++i)
16             count[i] += count[i - 1];
17
18         for (int i = 1; i <= kMax; ++i)
19             if (count[i] > count[i - 1]) {
20                 long sum_divisions = 0;
21                 for (int j = 1; i * j <= kMax; ++j) {
22                     const int lo = i * j - 1;
23                     const int hi = i * (j + 1) - 1;
24                     sum_divisions += (count[min(hi, kMax)] - count[lo]) * j;
25                 }
26                 ans += sum_divisions * (count[i] - count[i - 1]);
27                 ans %= kMod;
28             }
29
30         return ans;
31     }
32 };
```

C#

```
1
2 csharp
3 public class Solution {
4     public int SumOfFlooredPairs(int[] nums) {
5         int kMod = 1000000007;
6         int kMax = nums.Max();
7         long ans = 0;
8
9         int[] count = new int[kMax + 1];
10
11         foreach (int num in nums) {
12             count[num]++;
13         }
14
15         for (int i = 1; i <= kMax; i++) {
16             count[i] += count[i - 1];
17         }
18
19         for (int i = 1; i <= kMax; i++) {
20             if (count[i] > count[i - 1]) {
21                 long sum_divisions = 0;
22                 for (int j = 1; i * j <= kMax; j++) {
23                     int lo = i * j - 1;
24                     int hi = i * (j + 1) - 1;
25                     sum_divisions += (count[Math.Min(hi, kMax)] - count[lo]) * j;
26                 }
27                 ans += sum_divisions * (count[i] - count[i - 1]);
28                 ans %= kMod;
29             }
30         }
31
32         return (int) ans;
33     }
34 }
```

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.



Level Up Your
Algo Skills

Get Premium

Got a question? [Ask the Teaching Assistant](#) anything you don't understand.