Problem 3627: Maximum Median Sum of Subsequences of Size 3

Problem Information

Difficulty: Medium
Acceptance Rate: 63.83%
Paid Only: No
Tags: Array, Math, Greedy, Sorting, Game Theory

Problem Description

You are given an integer array `nums` with a length divisible by 3.

You want to make the array empty in steps. In each step, you can select any three elements from the array, compute their **median** , and remove the selected elements from the array.

The **median** of an odd-length sequence is defined as the middle element of the sequence when it is sorted in non-decreasing order.

Return the **maximum** possible sum of the medians computed from the selected elements.

Example 1:

Input: nums = [2,1,3,2,1,3]

Output: 5

Explanation:

* In the first step, select elements at indices 2, 4, and 5, which have a median 3. After removing these elements, `nums` becomes `[2, 1, 2]`. * In the second step, select elements at indices 0, 1, and 2, which have a median 2. After removing these elements, `nums` becomes empty.

Hence, the sum of the medians is `3 + 2 = 5`.

Example 2:

Input: nums = [1,1,10,10,10,10]

Output: 20

Explanation:

* In the first step, select elements at indices 0, 2, and 3, which have a median 10. After removing these elements, `nums` becomes `[1, 10, 10]`. * In the second step, select elements at indices 0, 1, and 2, which have a median 10. After removing these elements, `nums` becomes empty.

Hence, the sum of the medians is `10 + 10 = 20`.

Constraints:

* `1 <= nums.length <= 5 * 105` * `nums.length % 3 == 0` * `1 <= nums[i] <= 109`

Code Snippets

C++:

```
class Solution {
public:
long long maximumMedianSum(vector<int>& nums) {

}
};
```

Java:

```
class Solution {
public long maximumMedianSum(int[] nums) {

}
}
```

Python3:

```
class Solution:
def maximumMedianSum(self, nums: List[int]) -> int:
```