Documentation

The "Majority Element II" problem involves finding all elements in a given array that appear more than ⌊n/3⌋ times, where n is the length of the array. The challenge specifies that the solution should run in linear time (O(n)) and use constant space (O(1)). This problem is a more generalized version of the classic "majority element" problem, where we had to find elements appearing more than ⌊n/2⌋ times. In this version, the array can contain up to two elements that meet the criteria, as having more than two such elements would violate the constraint of their combined frequencies exceeding the array size.

To solve this problem efficiently, the Boyer-Moore Voting Algorithm is utilized. This algorithm is particularly useful when dealing with majority voting problems due to its ability to track candidates while maintaining constant space. In this case, we keep track of two potential candidates for majority elements using two variables (candidate1 and candidate2), alongside two counters to count their occurrences. This is because, in an array of size n, there can be at most two numbers that appear more than ⌊n/3⌋ times. If we try to track more than two candidates, the total count of these elements would exceed the array's length.

The algorithm operates in two main phases. In the first phase, we iterate through the array and attempt to identify two potential majority candidates. During this process, if the current number matches one of the candidates, its count is incremented. If neither candidate matches the number and their counts are non-zero, we decrement the counts. This ensures that non-majority elements get eliminated from contention. If one of the counts reaches zero, that position is reassigned with a new candidate. At the end of this phase, we have two candidates, which may or may not actually appear more than ⌊n/3⌋ times.

In the second phase, we verify whether these candidates actually satisfy the condition of appearing more than ⌊n/3⌋ times by counting their occurrences in a second pass through the array. If any of the candidates meet this threshold, they are added to the result list. This second phase is essential because the candidates identified in the first phase are not guaranteed to meet the frequency requirement; they are simply the most likely candidates based on the initial analysis.

The Boyer-Moore Voting Algorithm offers an elegant solution to this problem. By maintaining only two candidates and two counters, it ensures that the algorithm operates in constant space, making it highly efficient. Additionally, the time complexity of O(n) is achieved because the array is processed only twice: once for candidate selection and once for candidate validation. This makes the approach optimal for scenarios where memory and performance constraints are critical.