📓 Contiguous Array – Documentation

1. Problem Statement

Given a binary array nums, return the maximum length of a contiguous subarray with an equal number of 0 and 1.

Example:

Input: nums = [0,1,0]
Output: 2
Explanation: [0,1] or [1,0] are valid subarrays with equal numbers of 0s and 1s.

2. Intuition

The core idea is to **transform the problem** into a **prefix sum problem**:

- Treat each 0 as -1.
- A subarray with a total sum of 0 means it has equal numbers of 1s and 0s.
- Track prefix sums and use a hash map to store the first index where each sum appears.

3. Key Observations

- A balanced subarray means the sum (after conversion) is zero.
- If the same cumulative sum appears more than once, the segment between them must sum to 0.
- We only need to remember the **first** time each sum appears to maximize the subarray length.

4. Approach

- Replace 0 with -1 during computation.
- Traverse the array while maintaining a **running sum**.
- Use a dictionary to map each running sum to its **first occurrence index**.

- If the same sum appears again at index i, calculate subarray length: i - first_index.
- Keep updating the **maximum length** found.

5. Edge Cases

- Input array of length 1 → return 0 (no possible pair).
- All elements are the same (all 0s or all 1s) → return 0.
- Entire array is balanced → return len(nums).

6. Complexity Analysis

Time Complexity

- **O(n)**: One pass through the array.

Space Complexity

- **O(n)**: In the worst case, we store all prefix sums in a hash map.

7. Alternative Approaches

a. Brute Force:

- Check all subarrays and count 0s and 1s.
- **Time:** $O(n^2)$, **Space:** $O(1)$
- Not feasible for $n = 10^5$.

b. Stack-based or Sliding Window:

- Doesn't directly apply due to the non-monotonic behavior of binary subarrays.

8. Test Cases

```
# Test Case 1
assert Solution().findMaxLength([0,1]) == 2

# Test Case 2
assert Solution().findMaxLength([0,1,0]) == 2

# Test Case 3
assert Solution().findMaxLength([1,1,1,0,0,0]) == 6

# Test Case 4
assert Solution().findMaxLength([1,1,1,1]) == 0

# Test Case 5
assert Solution().findMaxLength([0]) == 0
```

9. Final Thoughts

- This problem demonstrates the power of transforming inputs (0 to -1) to fit well-known patterns like prefix sums.
- The efficient use of hash maps allows solving it in linear time.
- Useful in problems involving balance or equality in counts within subarrays.