Max Consecutive Ones

Given a binary array `nums`, return the maximum number of consecutive 1's in the array.

Examples

Example 1:

```
Input: nums = [1,1,0,1,1,1]
    Output: 3
    Explanation: The first two digits or the last three digits are consecutive 1s.
The maximum number of consecutive 1s is 3.
```

Example 2:

```
Input: nums = [1,0,1,1,0,1]
    Output: 2
```

Constraints:

$1 \leq \text{nums.length} \leq 10^5$
nums[i] is either 0 or 1.

Optimal Approach – Single Pass

Initialize two variables:

`currentCount` → to count current streak of 1s
`maxCount` → to keep track of the maximum streak seen so far

Traverse the array:

If `nums[i] == 1`, increment `currentCount`
If `nums[i] == 0`, compare `currentCount` with `maxCount`, update `maxCount`, then reset `currentCount` to 0

After the loop, return the maximum of `maxCount` and `currentCount` (to handle case where array ends in 1s)

Dry Run

```
Input: nums = [1, 1, 0, 1, 1, 1]

    i = 0 → nums[i] = 1 → currentCount = 1
    i = 1 → nums[i] = 1 → currentCount = 2
    i = 2 → nums[i] = 0 → maxCount = 2, currentCount = 0
    i = 3 → nums[i] = 1 → currentCount = 1
    i = 4 → nums[i] = 1 → currentCount = 2
    i = 5 → nums[i] = 1 → currentCount = 3

    Final return: max(2, 3) = 3
```

Time and Space Complexity

Time Complexity: O(n) → One pass through the array of n elements
Space Complexity: O(1) → No extra space used beyond a few variables

JavaScript | C++ | C | Java | Python

```javascript
var findMaxConsecutiveOnes = function(nums) {
  let currentCount = 0;
  let maxCount = 0;
  for (let i = 0; i < nums.length; i++) {
    if (nums[i] == 1) {
      currentCount++;
    } else {
      maxCount = Math.max(currentCount, maxCount);
      currentCount = 0;
    }
  }
  return Math.max(maxCount, currentCount);
};
```