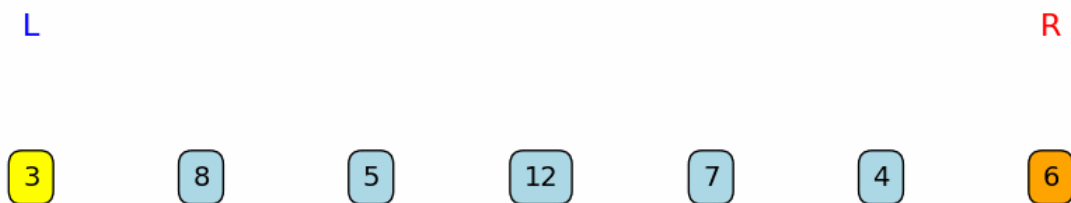


# Two Pointers: Stage Traversal



Start partitioning by parity: even numbers (L) and odd numbers (R)

## Problem: Partition Array by Parity

### Description:

Given an integer array `nums`, rearrange the array **in-place** such that all even numbers appear before all odd numbers. The order of the elements within the even or odd group does not matter.

Return the array after rearrangement.

You must solve the problem without using extra space (i.e. in  $O(1)$  extra space) and in one pass if possible.

### Example 1:

Input: `nums = [3, 8, 5, 12, 7, 4, 6]` Output: `[8, 12, 4, 6, 3, 5, 7]` Explanation: The even numbers [8, 12, 4, 6] appear before the odd numbers [3, 5, 7]. Note that the order within each group does not matter.

### Example 2:

Input: `nums = [1, 3, 5, 7]` Output: `[1, 3, 5, 7]` Explanation: Since there are no even numbers, the array remains unchanged.

### Constraints:

- $1 \leq \text{nums.length} \leq 10^5$

- $0 \leq \text{nums}[i] \leq 10^5$

### 1. Initialization:

The script starts with two pointers:

- `left` at the beginning (index 0)
- `right` at the end (last index)

### 2. Stage Traversal:

In each loop iteration:

- It prints the current array and the positions (and values) of the two pointers.
- If the number at the left pointer is even, that element is already in the correct half, so the left pointer is moved right.
- If the number at the right pointer is odd, that element is in the correct half, so the right pointer is moved left.
- If neither of those conditions holds (meaning the left number is odd and the right number is even), the two values are swapped. This moves the even number toward the left and the odd number toward the right.

### 3. Termination:

The loop ends when the left pointer is no longer less than the right pointer. At that point, the array is partitioned with evens on the left and odds on the right.

```
def partition_by_parity(nums):
    left = 0                # Initialize the left pointer at the beginning.
    right = len(nums) - 1   # Initialize the right pointer at the end.
    print("Initial array:", nums)

    while left < right:
        print("\nCurrent array:", nums)
        print(f"Left pointer at index {left} with value {nums[left]}")
        print(f"Right pointer at index {right} with value {nums[right]}")

        # If the left element is even, it's already in the correct partition.
        if nums[left] % 2 == 0:
            print(f"{nums[left]} is even, so move the left pointer right.")
            left += 1

        # If the right element is odd, it's already in the correct partition.
        elif nums[right] % 2 == 1:
            print(f"{nums[right]} is odd, so move the right pointer left.")
            right -= 1

        # Otherwise, the left element is odd and the right element is even.
        # In that case, swap them.
        else:
```

```
print(f"Swapping {nums[left]} (odd) and {nums[right]} (even).")
nums[left], nums[right] = nums[right], nums[left]
left += 1
right -= 1
```

```
print("\nFinal partitioned array:", nums)
```

```
# Example usage:
```

```
nums = [3, 8, 5, 12, 7, 4, 6]
```

```
partition_by_parity(nums)
```

---

Revision #2

Created 28 December 2024 22:13:03 by victor

Updated 10 February 2025 00:22:34 by victor