

3.6 Left Rotate Array by K

MEDIUM

■ What You Will Learn

- ✓ The 3-reverse trick — most elegant rotation approach
- ✓ Why $k \% n$ handles over-rotation
- ✓ Space-efficient $O(1)$ solution vs the naive $O(k)$ approach

■ Intuition First

Left rotating by K means the first K elements move to the end. There's a beautiful trick: if you reverse the first K, then reverse the remaining, then reverse the whole array — you get the rotated result. It feels like magic but it works every time.

■ The 3-Reverse Trick

- Step 1: Reverse first K elements
- Step 2: Reverse remaining n-K elements
- Step 3: Reverse the entire array
- Result: Left rotated by K positions!

■ Visual — Rotate [1,2,3,4,5] by K=2

Original — K=2 elements will move to end

1	2	3	4	5
---	---	---	---	---

Step 1: Reverse first 2 → [2,1,3,4,5]

Step 2: Reverse last 3 → [2,1,5,4,3]

Step 3: Reverse all → [3,4,5,1,2]

Left rotated by 2 ✓

3	4	5	1	2
---	---	---	---	---

■ Java Code

```
public static void leftRotateByK(int[] arr, int k) {  
    int n = arr.length;  
    k = k % n; // Handle k >= n (k=7,n=5 same as k=2)  
    if (k == 0) return;  
    reverse(arr, 0, k-1);  
    reverse(arr, k, n-1);  
    reverse(arr, 0, n-1);  
}
```

```

reverse(arr, 0, k - 1); // Step 1
reverse(arr, k, n - 1); // Step 2
reverse(arr, 0, n - 1); // Step 3
}

public static void reverse(int[] arr, int l, int r) {
    while (l < r) {
        int temp = arr[l]; arr[l] = arr[r]; arr[r] = temp;
        l++; r--;
    }
}

```

■ Complexity

Case	Time	Space	What happens?
All Cases	O(n)	O(1)	3 reversals, each O(n/2) — still O(n) total

■ Common Mistakes

- Forgetting $k = k \% n$ — rotating by n is same as no rotation
- Naive approach: copy first K to temp array, shift rest left $\rightarrow O(n)$ time but $O(k)$ space

■ Naive Approach vs 3-Reverse

- Naive: use temp array of size $k \rightarrow O(k)$ space
- 3-Reverse: no extra space $\rightarrow O(1)$ space
- Both are $O(n)$ time — but 3-reverse wins on space

■ Practice Problems

- [Med] Right rotate by K (just reverse the step order)
- [Med] Rotate matrix 90 degrees (same reversal idea)
- [Med] Check if one array is a rotation of another

■ Quick Revision

- $k = k \% n$ to normalize
- 3-Reverse: first k , last $n-k$, whole array

- $O(n)$ time, $O(1)$ space
- Works for both left and right rotation (adjust which halves to reverse first)