Agenda

Two Pointer Technique

Merge function example

Two sum example

The Two-Pointer Technique

Involves the use of two or more variables (aka pointers) to keep track of the locations (index place) of elements of interest within the same list or of two different lists.

Use with: sorted lists, symmetrical lists, strings

Key Idea

•Reduce time complexity by leveraging sorted order or structured traversal.

Example 1 - Merge Function in MergeSort

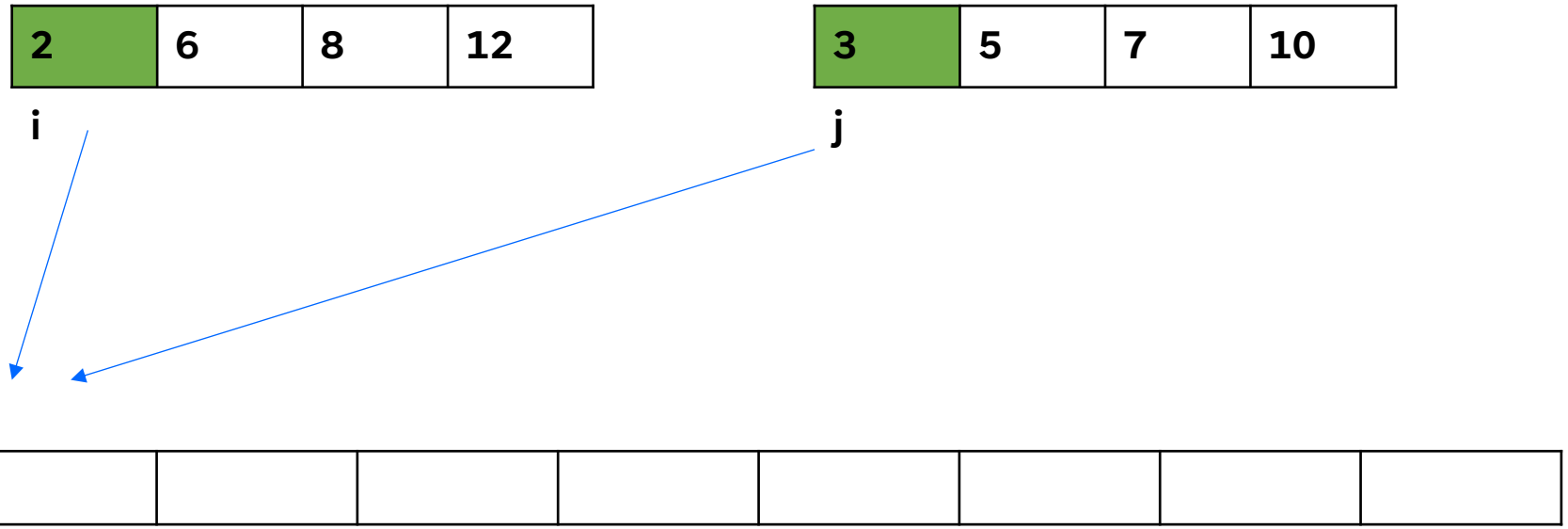
Problem

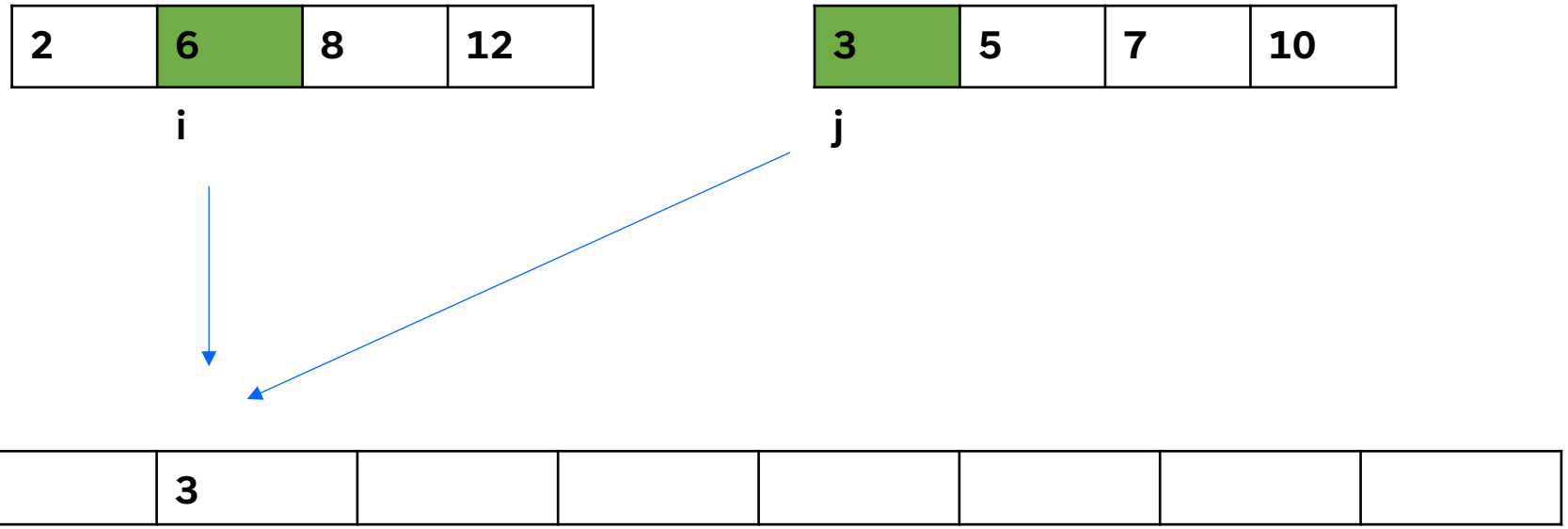
- Merge two sorted arrays into one sorted array.

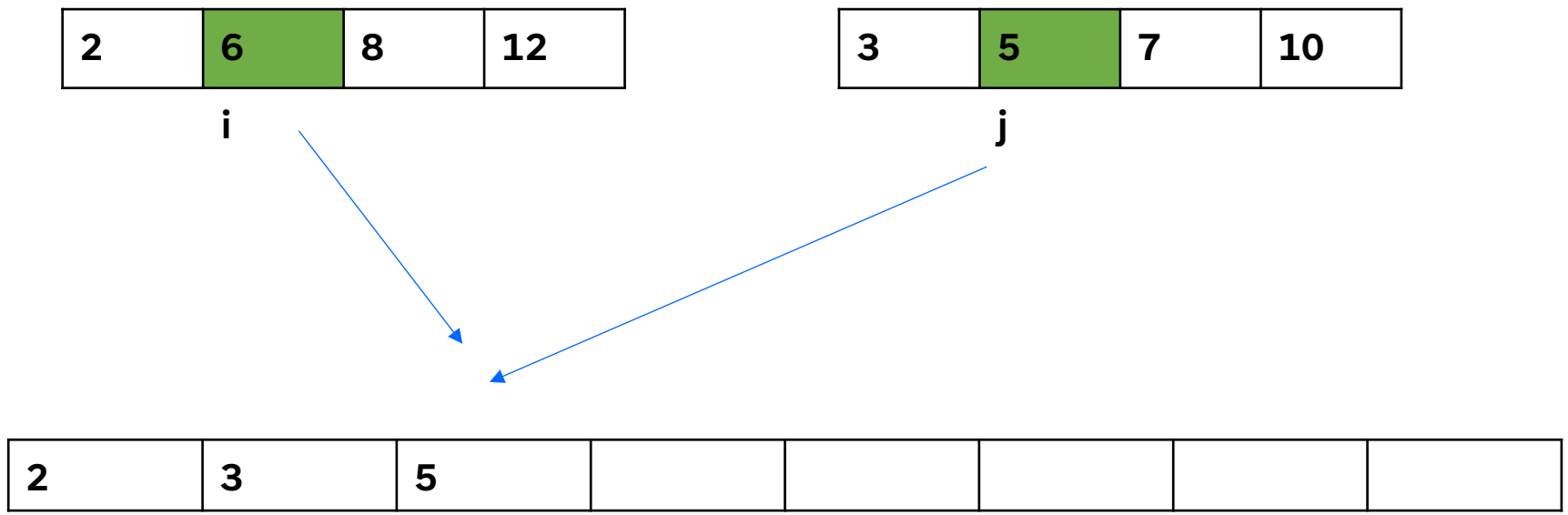
Approach

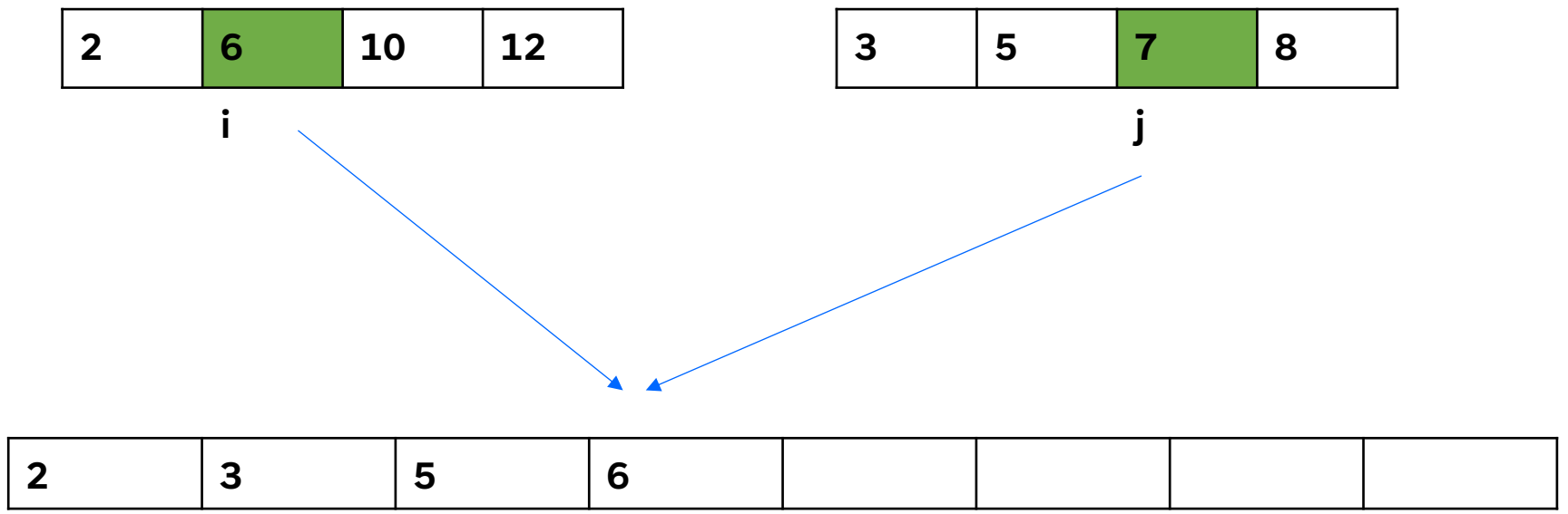
1. Use two pointers, i and j, starting at the beginning of both arrays.
2. Compare elements and pick the smaller one to add to the merged array.
3. Move the pointer of the array with the smaller element.
4. Repeat until one array is exhausted, then copy remaining elements.

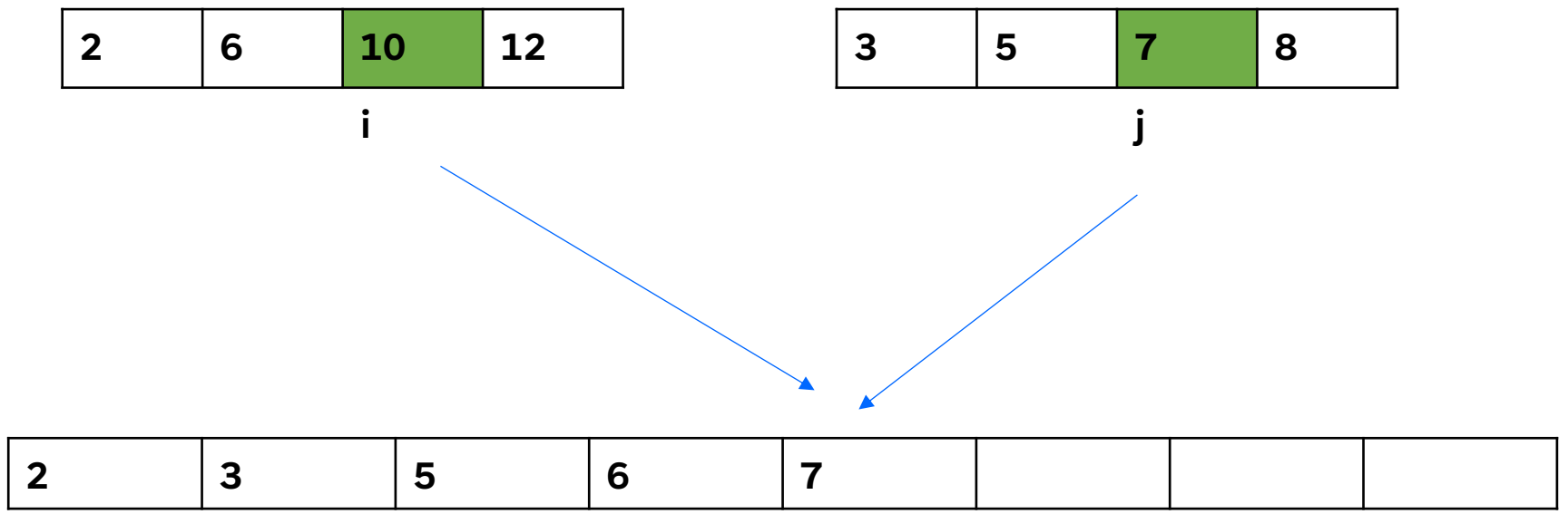
5. O(n) runtime

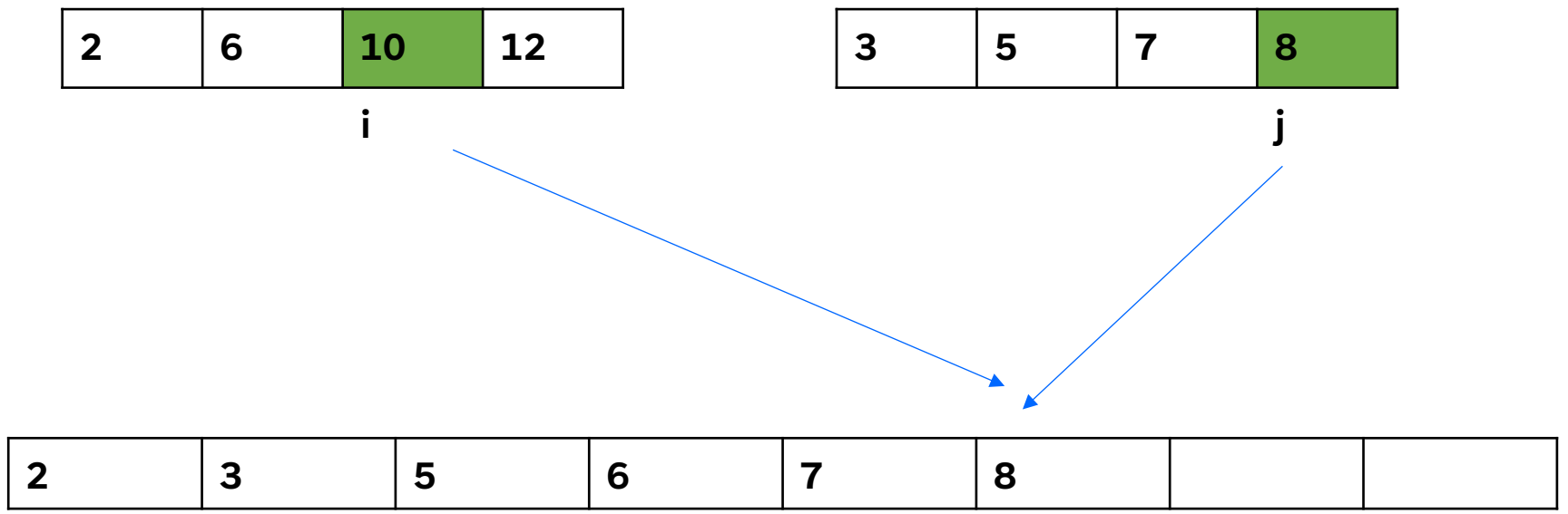

2
6
8
12
3
5
7
10
i
j
2

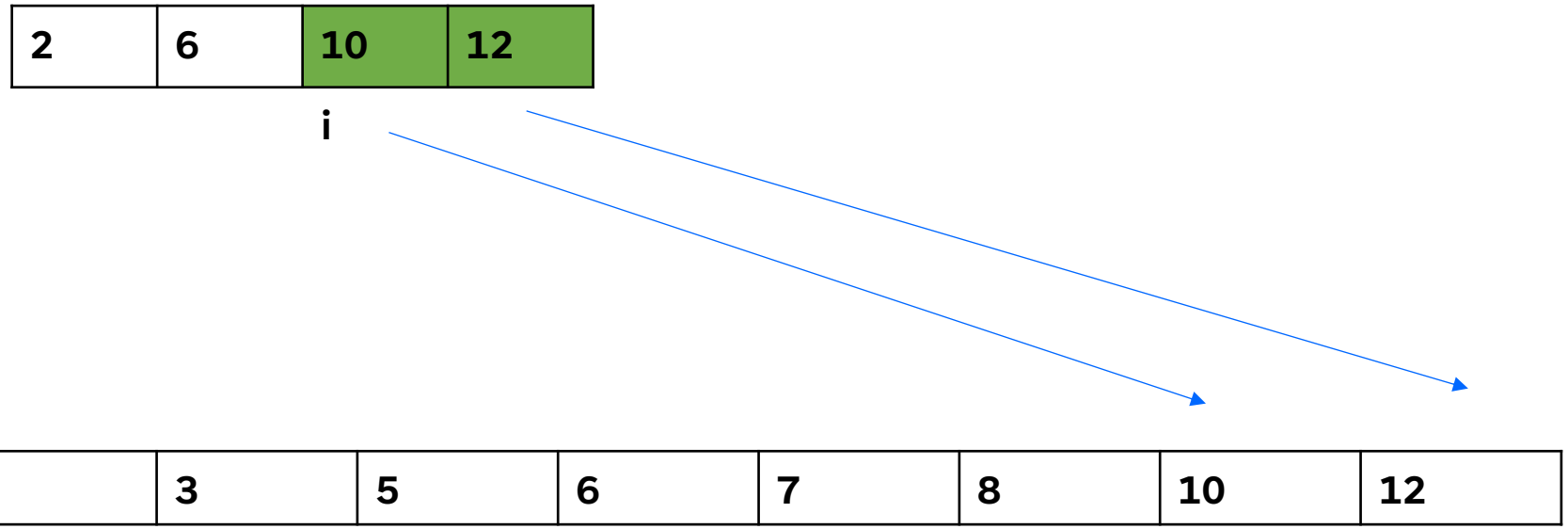

2
6
8
12
i
3
5
7
10
j
2
3


2
6
8
12
i
3
5
7
10
j
2
3
5


2
6
10
12
i
3
5
7
8
j
2
3
5
6


2
6
10
12
i
3
5
7
8
j
2
3
5
6
7


2
6
10
12
i
3
5
7
8
j
2
3
5
6
7
8


2
6
10
12
i
2
3
5
6
7
8
10
12

Example 2 - Two Sum in a Sorted Array

Find two numbers in a sorted array that sum to a target value.

Approach

1. Use two pointers:
    - left starts at the beginning of the array.
    - right starts at the end of the array.
2. Calculate the sum of arr[left] + arr[right].
3. If the sum equals the target, return the pair.
4. If the sum is too small, move left forward.
5. If the sum is too large, move right backward.

Problem: Find two numbers in the list that sum to a target

Target = 10

Sum = 12
12 > target
Decrement right pointer

Problem: Find two numbers in the list that sum to a target

Target = 10

Sum = 9
9 < target
Increment left pointer

Problem: Find two numbers in the list that sum to a target

Target = 10

Sum = 11
11 > target
Decrement right pointer

Problem: Find two numbers in the list that sum to a target

Target = 10

Sum = 9
9 < target
Increment left pointer

Problem: Find two numbers in the list that sum to a target

Target = 10

Sum = 10

Found!

Why Use Two-Pointer?

Benefits

- **Reduced Complexity**: Often brings down time complexity from $O(n^2)$ to $O(n)$.
- **Low Memory Overhead**: No need for additional data structures in most cases.

Limitations

- Often requires sorted data (or pre-processing to sort).
- May not work if the problem involves unsorted arrays or unstructured traversal.