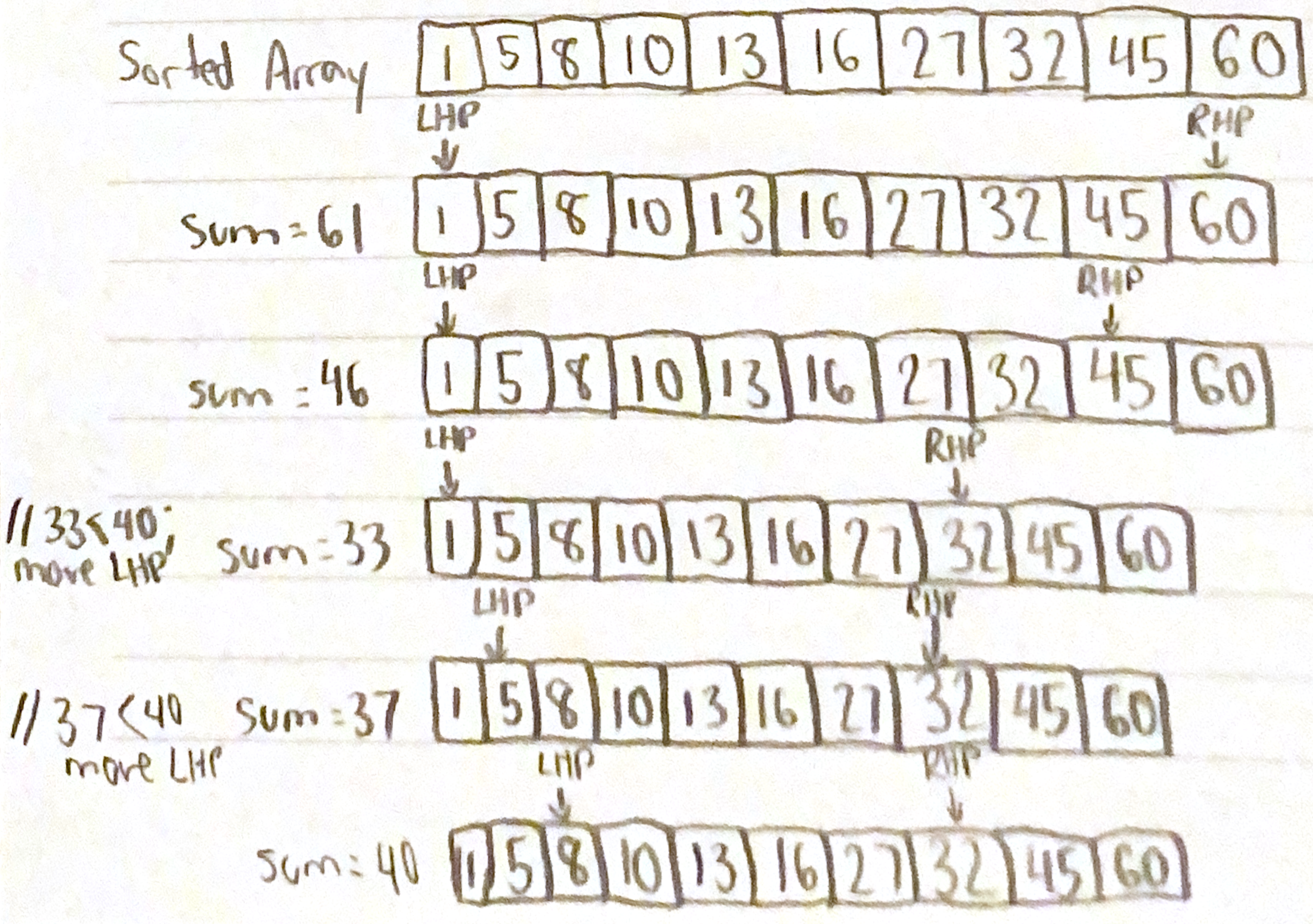
Two Pointers

Think Two Pointers when:
- the input is sorted (or could be sorted)
- you need to find a pair of elements that meet some criteria
- you need to traverse from both ends of a sequence
- you want to optimize from $O(n^2)$ to $O(n)$ by avoiding nested loops

- The general idea is to use two indices (pointers) to traverse a data structure optimally
  ↳ these pointers can move toward each other, in opposite directions, or independently

Example: Two Sum, sorted array
↳ Given a sorted (ascending order) integer array nums of n elements and a target value, find if there exists any pair of elements such that their sum is equal to target

target = 40



Two Pointers (continued)

Leetcode Problem: Two Sum II - Input Array is Sorted

$\boxed{\text{Description}}$ Given a 1-indexed array of integers numbers that are sorted in ascending order, find two numbers such that they add up to a specific target number. Let these two numbers[index1] and numbers[index2] where 1 ≤ index1 < index2 ≤ numbers.length

Return the indices of the two numbers added by 1 (remember, it's a 1-indexed array) of length 2. You are guaranteed to have only one solution, and each element can only be used once

Test Cases:

[1,3,7,12], target = 15 => return [2,4]
[1,3], target = 4 => return [1,2]
[-11, -4, 1, 3, 6], target = 2 => return [2,5]

Algorithm:

1. If the length of numbers == 2, return [1,2]
2. Set the left pointer to index 0 and the right pointer to length of numbers -1
3. Check the sum of the two elements at these indices. If it is greater than target, move the RHP to the left
4. If target is greater than the sum, move the LHP to the right
5. Add 1 to both indices and return