Data Structures & Algorithms

Two Pointers

B.Bhuvaneswaran, AP (SG) / CSE

9791519152

bhuvaneswaran@rajalakshmi.edu.in

RAJALAKSHMI ENGINEERING COLLEGE

An AUTONOMOUS Institution

Affiliated to ANNA UNIVERSITY, Chennai

Two pointers

- Two-pointers is an extremely common technique used to solve array and string problems.
- It involves having two integer variables that both move along an iterable.
- This means we will have two integers, usually named something like i and j, or left and right which each represent an index of the array or string.

Two pointers

- Generally, the 2 Pointers approach is a good choice in those cases where:
  - The Array(s) is/are sorted
  - We are searching for a pair of numbers, or a difference etc.

Intersection of 2 Sorted Arrays

- Find the intersection of two sorted arrays or in other words, given 2 sorted arrays, find all the elements which occur in both the arrays.
- Input Format
  - The first line contains T, the number of test cases. Following T lines contain:
    - Line 1 contains N1, followed by N1 integers of the first array
    - Line 2 contains N2, followed by N2 integers of the second array
- Output Format
  - The intersection of the arrays in a single line

Sample input and output

Input:

1

3 10 17 57

6 2 7 10 15 57 246

Output:

10 57

Input:

1

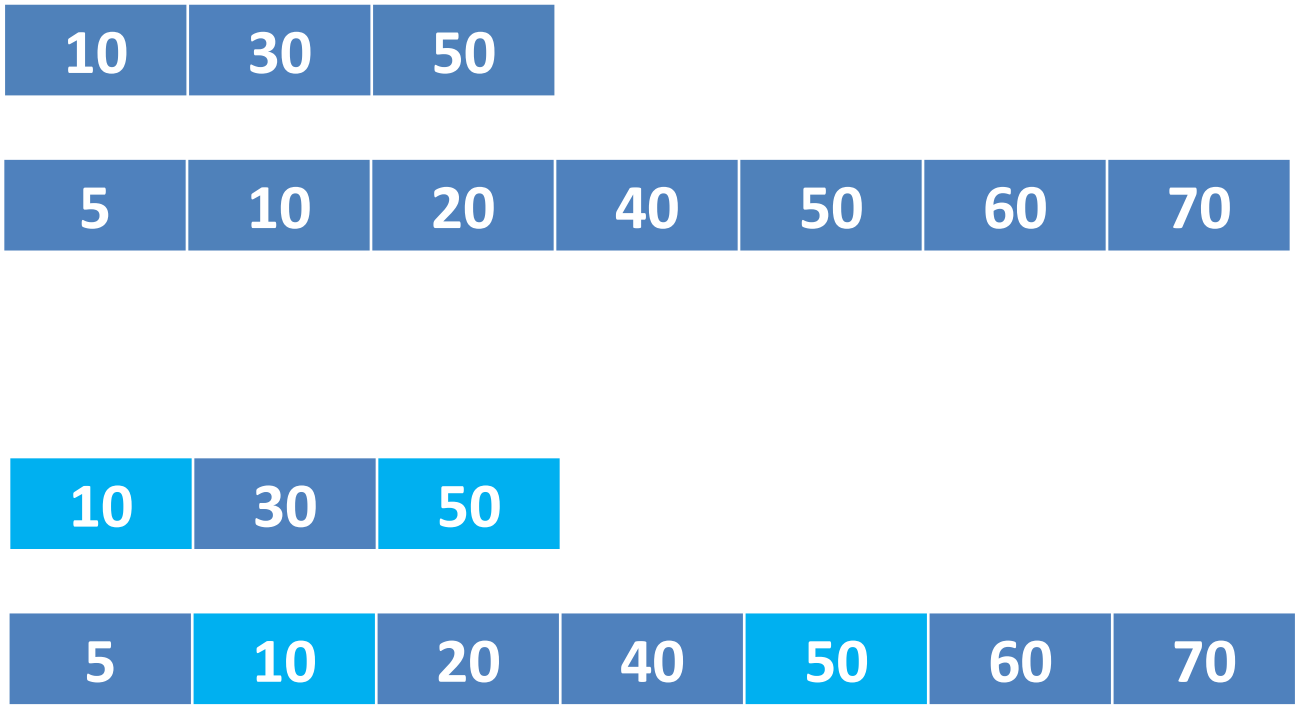
6 1 2 3 4 5 6

2 1 6

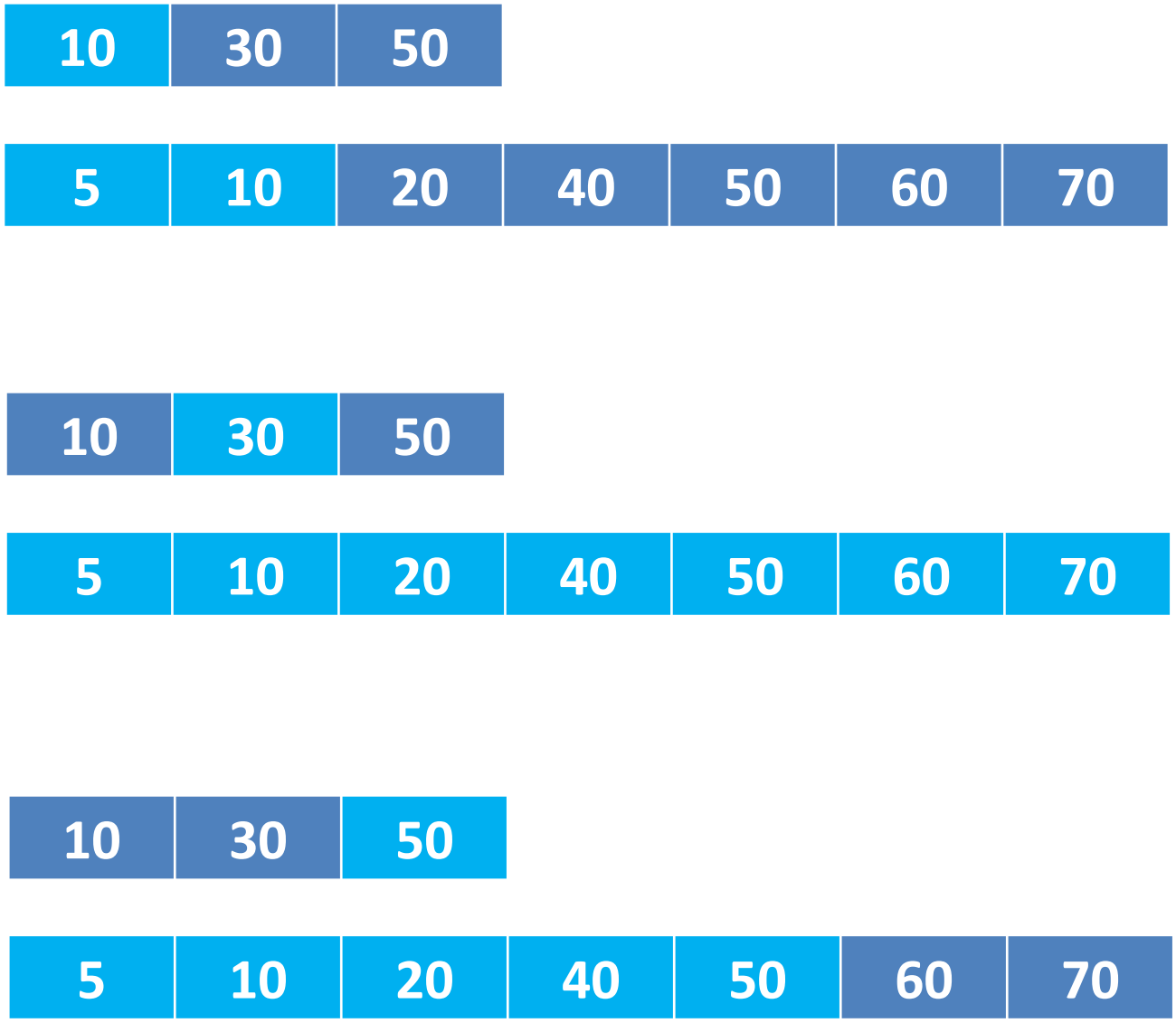
Output:

1 6

Example



Brute Force Approach



Brute Force Approach

- First Array: m
- Second Array: n
- Time Complexity: O(m * n)

Two Pointers Approach

10	30	50
↑		

5	10	20	40	50	60	70
↑						

Two Pointers Approach

10	30	50
↑		

5	10	20	40	50	60	70
	↑					

Two Pointers Approach

10	30	50
↑		

5	10	20	40	50	60	70
	↑					

Two Pointers Approach

10	30	50
	↑	

5	10	20	40	50	60	70
		↑				

Two Pointers Approach

10	30	50
	↑	

5	10	20	40	50	60	70
			↑			

Two Pointers Approach

10	30	50
		↑

5	10	20	40	50	60	70
			↑			

Two Pointers Approach

10	30	50
		↑

5	10	20	40	50	60	70
				↑		

Two Pointers Approach

10	30	50
		↑

5	10	20	40	50	60	70
				↑		

Two Pointers Approach

- First Array: m
- Second Array: n
- Time Complexity: O(m + n)

Pseudo Code

```
i = 0 // i for first array
j = 0 // j for first array
intersectionList = []
while i < A.length - 1 && j < B.length - 1:
        if A[i] < B[j]:
                i++
        else if A[i] > B[j]:
                j++
        else if A[i] == B[j]:
                intersectionList.add(A[i])
                i++
                j++
```

Note

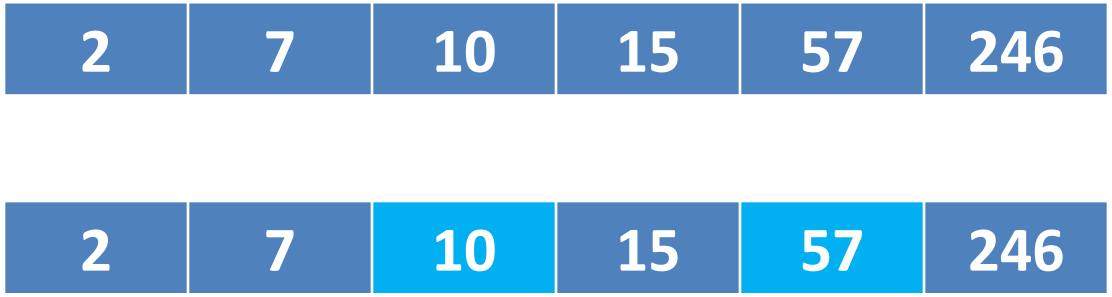
- Arrays are sorted
- Take care of edge cases

Check pair with difference k

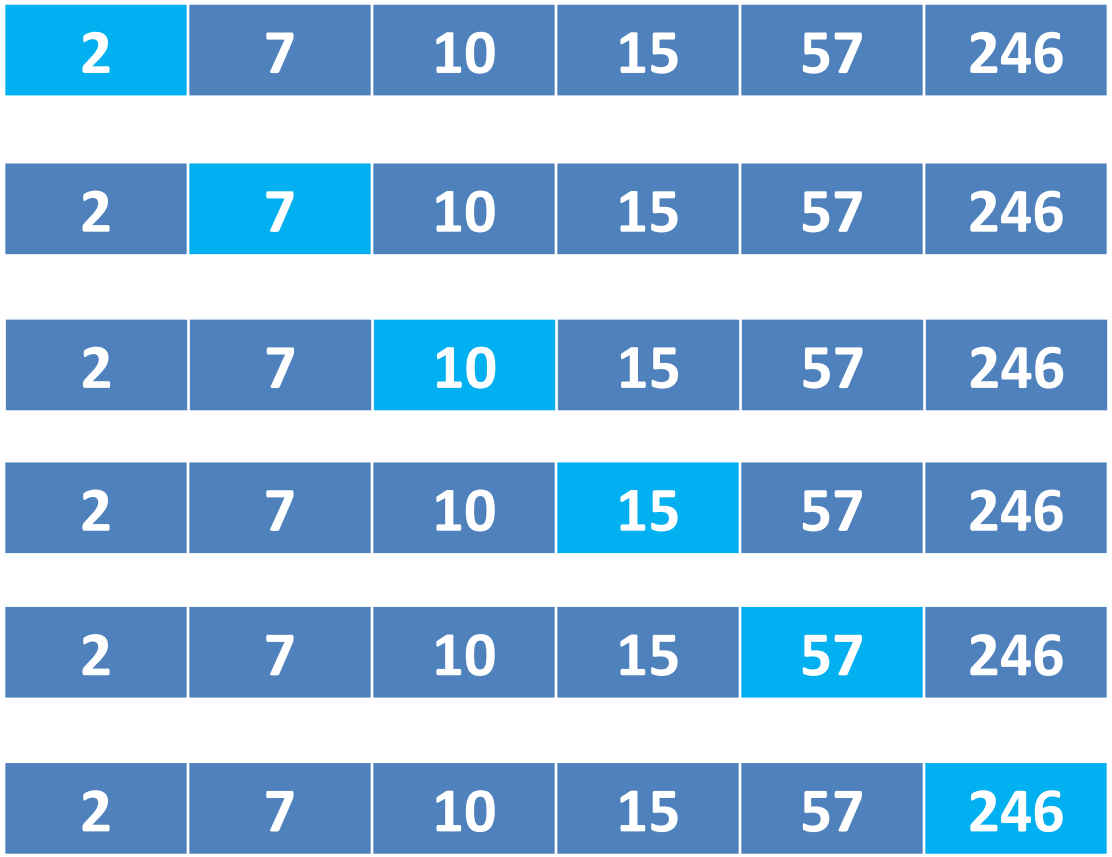
- Given an array A of sorted integers and another non negative integer k, find if there exists 2 indices i and j such that A[i] - A[j] = k, i != j.
- Input Format
  - First line is number of test cases T. Following T lines contain:
    - N, followed by N integers of the array
    - The non-negative integer k
- Output format
  - Print 1 if such a pair exists and 0 if it doesn't.

Example

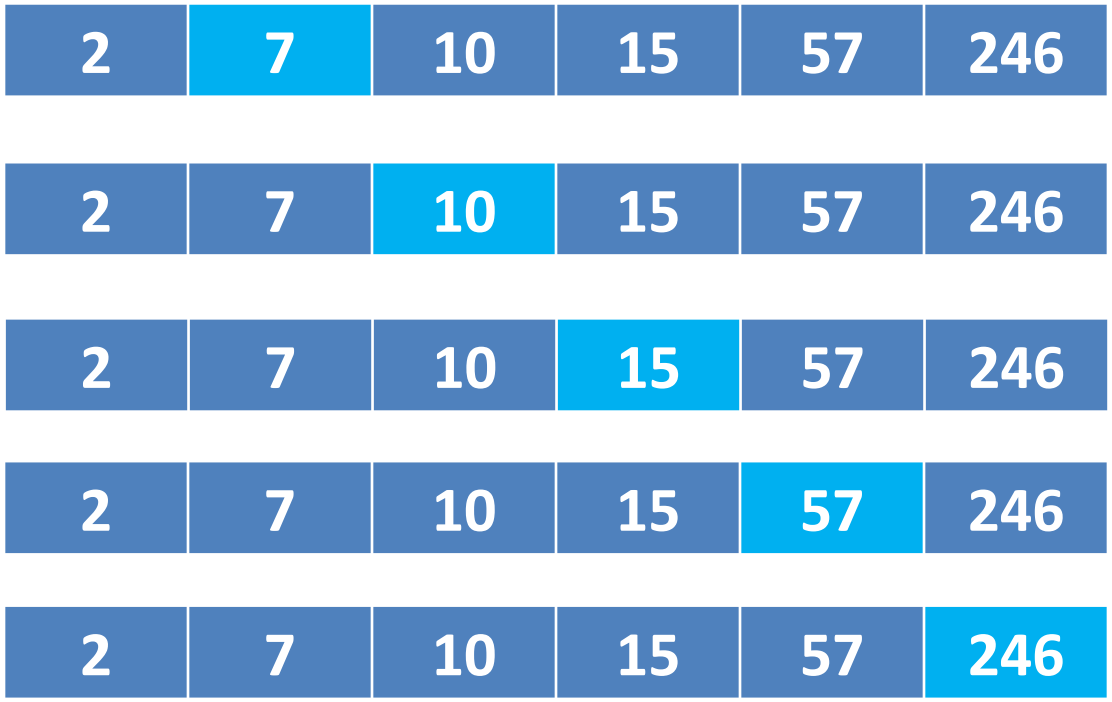
- Find a pair with difference k = 47



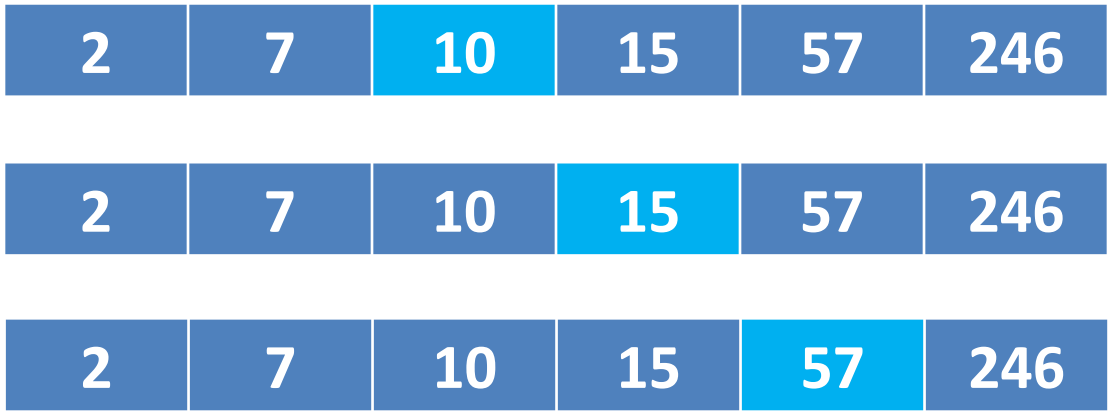
Brute Force Approach



Brute Force Approach



Brute Force Approach



Brute Force Approach

- Time Complexity: $O(n^2)$

Two Pointers Approach

2	7	10	15	57	246
↑ left	↑ right				

Rule

- Increment the right pointer to increase the difference.
- Increment the left pointer to decrease the difference.

Two Pointers Approach

2	7	10	15	57	246
↑ left	↑ right				

Two Pointers Approach

2	7	10	15	57	246
↑ left		↑ right			

Two Pointers Approach

2	7	10	15	57	246
↑ left			↑ right		

Two Pointers Approach

2	7	10	15	57	246
↑ left				↑ right	

Two Pointers Approach

2	7	10	15	57	246
	↑ left			↑ right	

Two Pointers Approach

2	7	10	15	57	246
		↑ left		↑ right	

Two Pointers Approach

- Time Complexity: O(n)

Reverse String

- Write a program that reverses a string.
- The input string is given as an array of characters s.
- You must do this by modifying the input array in-place with O(1) extra memory.

Example 1

- Input:
  - hello
- Output:
  - olleh

Example 2

- Input:
  - Hannah
- Output:
  - hannaH

Constraints

- $1 <= s.length <= 10^5$
- s[i] is a printable ASCII character.

Squares of a Sorted Array

- Given an integer array nums sorted in non-decreasing order, return an array of the squares of each number sorted in non-decreasing order.

Example 1

- Input:
  - 5
  - -4 -1 0 3 10
- Output:
  - 0 1 9 16 100

Explanation

- After squaring, the array becomes [16,1,0,9,100].
- After sorting, it becomes [0,1,9,16,100].

Example 2

- Input:
  - 5
  - -7 -3 2 3 11
- Output:
  - 4 9 9 49 121

Constraints

- $1 <= nums.length <= 10^4$
- $-10^4 <= nums[i] <= 104$
- nums is sorted in non-decreasing order.

Example 1

- Given a string s, return true if it is a palindrome, false otherwise.
- A string is a palindrome if it reads the same forward as backward. That means, after reversing it, it is still the same string. For example: "abcdcba", or "racecar".

Example 2

- Given a sorted array of unique integers and a target integer, return true if there exists a pair of numbers that sum to target, false otherwise. This problem is similar to Two Sum. (In Two Sum, the input is not sorted).
- For example, given nums = [1, 2, 4, 6, 8, 9, 14, 15] and target = 13, return true because 4 + 9 = 13.

Example 3

- Given two sorted integer arrays arr1 and arr2, return a new array that combines both of them and is also sorted.
- [1, 4, 7, 20]
- [3, 5, 6]

Is Subsequence.

- Given two strings s and t, return true if s is a subsequence of t, or false otherwise.
- A subsequence of a string is a sequence of characters that can be obtained by deleting some (or none) of the characters from the original string, while maintaining the relative order of the remaining characters. For example, "ace" is a subsequence of "abcde" while "aec" is not.

Reverse String

- Write a function that reverses a string. The input string is given as an array of characters s.
- You must do this by modifying the input array in-place with O(1) extra memory.

Examples

- Input:
  - s = ["h","e","l","l","o"]
- Output:
  - ["o","l","l","e","h“
- Input:
  - s = ["H","a","n","n","a","h"]
- Output:
  - ["h","a","n","n","a","H"]

Constraints

- $1 <= s.length <= 10^5$
- s[i] is a printable ascii character.

Queries?

Thank You…!