Day 1 - Two Pointer Pattern Notes (English Version)

Objective

Today we will learn what the Two Pointer Pattern is, when it is used, and why it is important. It is one of the most powerful patterns in DSA that helps optimize problems from O(n) to O(n).

Intuition (Simple Story)

Imagine a line a boy on one end and a girl on the other. Both move one step forward, sometimes together. When both meet, we say condition met. The same logic applies to arrays, strings, or linked lists.

Definition (Formal)

The Two Pointer Pattern is a technique where two variables or pointers traverse the same linear data structure from different positions simultaneously.

Common Data Structures:

- Array
- String
- Linked List

Characteristics

- Two variables/pointers: i and j
- Linear Data Structure: Array, String, or Linked List
- Different Start Points: Both move from different positions
- Termination Condition: Depends on the problem (e.g., i j, pointer becomes null, etc.)

Direction Types

1. Towards Each Other: e.g., Palindrome Check, Pair Sum Problem
2. Away From Each Other: e.g., Expand From Middle
3. Same Direction (Fast-Slow): e.g., Find Middle in Linked List, Detect Cycle

FastSlow Pointer Concept (Important)

- Slow pointer (i): moves one step at a time
- Fast pointer (j): moves two steps at a time

When the fast pointer reaches the end, the slow pointer is at the middle element.

Day 1 - Two Pointer Pattern Notes (English Version)

This logic is used in:

- Finding the middle of a linked list

- Detecting a loop in a linked list (Floyds Algorithm)

Why Use Two Pointer Pattern?

- Efficient: Reduces time complexity (O(n) O(n))

- Early Stopping: Stop loop when condition fails

- In-place: Requires no extra memory (Space O(1))

Mnemonic: PCS Partitioning, Comparison, Searching

Example 1 Palindrome Check (String)

Definition: A string is a palindrome if it reads the same from left to right and right to left.

Example: level Palindrome

Code:

```
-------------------------------
i = 0, j = s.length - 1
while i < j:
  if s[i] != s[j]:
    return false
  i += 1
  j -= 1
return true
-------------------------------
```

Time: O(n), Space: O(1)

Example 2 Pair Sum in Sorted Array

Problem: Find two elements in a sorted array whose sum equals a target.

Code:

```
-------------------------------
```

```
left = 0, right = n - 1
while left < right:
  if arr[left] + arr[right] == target:
    return [left, right]
  elif arr[left] + arr[right] < target:
    left += 1
  else:
    right -= 1
```

Time: O(n), Brute Force: O(n)

Example 3 Remove Duplicates from Sorted Array

Idea:

- Slow pointer stores unique values
- Fast pointer traverses the array

When a new element is found, increment slow and copy the value.

Result: In-place unique array without extra space.

Example 4 Merge Two Sorted Arrays

Concept: Merge two sorted arrays while exhausting both.

Code:

```
i = 0, j = 0, k = 0
while i < n1 and j < n2:
  if a[i] <= b[j]:
    res[k] = a[i]; i += 1
  else:
    res[k] = b[j]; j += 1
  k += 1
// append remaining elements
```

Templates Summary

Template A Single Array (Both Ends)

```
------------------------------
while left < right:
  if arr[left] == arr[right]:
    left += 1
    right -= 1
  else:
    break
------------------------------
```

Used In: Palindrome, Pair Sum

Template B Two Arrays (Exhaust Both)

```
------------------------------
while i < n1 and j < n2:
  if a[i] < b[j]:
    i++
  else:
    j++
------------------------------
```

Used In: Merge Sort, Intersection/Union Problems

Extra Tips & Notes

- Stop early when mismatch occurs.
- Avoid extra space by working in-place.
- Use fast-slow pointers in linked lists.
- If array is sorted, think Two Pointers.

When to Think Two Pointer?

Use Two Pointers when:

- Problem mentions pair, sum, compare, partition, merge, duplicate, middle, cycle.
- Array is sorted Try Two Pointer.
- Linked list has no indexing Use Fast-Slow pointer approach.

One-Shot Summary (Revision)

Technique: Two pointers traverse same data structure

Data Structures: Array, String, Linked List

Time Complexity: O(n)

Space Complexity: O(1)

Major Uses: Palindrome, Pair Sum, Duplicate Removal, Merge, Middle/Cycle Detection

Keywords: Partition, Comparison, Searching

Final Note

The Two Pointer Pattern is a powerful DSA technique that optimizes both time and space.

Practice each example and understand it with diagrams.

Keep Learning! Mastery in DSA comes from understanding patterns and the Two Pointer Pattern is the first step.