🏹 5 Two Pointer Patterns

💕 1. Two Sum II (Sorted Array)

🔸 **Main Problem:**
Find two numbers in a sorted array that add up to a target.
Input: `nums = [2, 7, 11, 15], target = 9` → **Output:** `[1, 2]`

🔁 **Similar Questions:**

- Find pair with given sum in sorted/rotated array
- Check if a pair exists with target sum
- Count triplets with sum less than K

<details> <summary>💻 Code</summary>

Python

```
python
Copy code
def two_sum(nums, target):
    l, r = 0, len(nums) - 1
    while l < r:
        s = nums[l] + nums[r]
        if s == target:
            return [l + 1, r + 1]
        elif s < target:
            l += 1
        else:
            r -= 1
```

C++

```
cpp
Copy code
vector<int> twoSum(vector<int>& numbers, int target) {
    int l = 0, r = numbers.size() - 1;
    while (l < r) {
        int sum = numbers[l] + numbers[r];
        if (sum == target) return {l + 1, r + 1};
        else if (sum < target) l++;
        else r--;
    }
    return {};
}
```

Java

```
java
```

```
Copy code
public int[] twoSum(int[] numbers, int target) {
    int l = 0, r = numbers.length - 1;
    while (l < r) {
        int sum = numbers[l] + numbers[r];
        if (sum == target) return new int[]{l + 1, r + 1};
        else if (sum < target) l++;
        else r--;
    }
    return new int[]{};
}
```
</details>

💧 2. Container With Most Water

🔸 **Main Problem:**
Find two lines that together with the x-axis form a container holding the most water.

Input: `[1,8,6,2,5,4,8,3,7]` → **Output:** `49`

🔁 **Similar Questions:**

- Max area under skyline
- Trapping rainwater (variation)
- Max product subarray (variation of boundary stretching)

<details> <summary>💻 Code</summary>

Python

```
python
Copy code
def max_area(height):
    l, r = 0, len(height) - 1
    res = 0
    while l < r:
        res = max(res, min(height[l], height[r]) * (r - l))
        if height[l] < height[r]:
            l += 1
        else:
            r -= 1
    return res
```

C++

```
cpp
Copy code
int maxArea(vector<int>& height) {
    int l = 0, r = height.size() - 1, res = 0;
    while (l < r) {
        res = max(res, min(height[l], height[r]) * (r - l));
        if (height[l] < height[r]) l++;
        else r--;
    }
```

```
    return res;
}
```

Java

```
java
Copy code
public int maxArea(int[] height) {
    int l = 0, r = height.length - 1, res = 0;
    while (l < r) {
        res = Math.max(res, Math.min(height[l], height[r]) * (r - l));
        if (height[l] < height[r]) l++;
        else r--;
    }
    return res;
}
```

</details>

🎯 3. 3Sum (Triplet Sum Zero)

🔸 **Main Problem:**
Find all triplets in array that sum to zero.

Input: `[-1,0,1,2,-1,-4]` → **Output:** `[[-1,-1,2],[-1,0,1]]`

🔁 **Similar Questions:**

- 4Sum
- Count triplets with target sum
- Closest 3Sum

<details> <summary>💻 Code</summary>

Python

```
python
Copy code
def three_sum(nums):
    nums.sort()
    res = []
    for i in range(len(nums)):
        if i > 0 and nums[i] == nums[i - 1]: continue
        l, r = i + 1, len(nums) - 1
        while l < r:
            total = nums[i] + nums[l] + nums[r]
            if total == 0:
                res.append([nums[i], nums[l], nums[r]])
                while l < r and nums[l] == nums[l+1]: l += 1
                while l < r and nums[r] == nums[r-1]: r -= 1
                l += 1; r -= 1
            elif total < 0:
                l += 1
            else:
                r -= 1
```

```
    return res
```

C++

```
cpp
Copy code
vector<vector<int>> threeSum(vector<int>& nums) {
    sort(nums.begin(), nums.end());
    vector<vector<int>> res;
    for (int i = 0; i < nums.size(); ++i) {
        if (i > 0 && nums[i] == nums[i - 1]) continue;
        int l = i + 1, r = nums.size() - 1;
        while (l < r) {
            int sum = nums[i] + nums[l] + nums[r];
            if (sum == 0) {
                res.push_back({nums[i], nums[l], nums[r]});
                while (l < r && nums[l] == nums[l + 1]) l++;
                while (l < r && nums[r] == nums[r - 1]) r--;
                l++; r--;
            } else if (sum < 0) l++;
            else r--;
        }
    }
    return res;
}
```

Java

```
java
Copy code
public List<List<Integer>> threeSum(int[] nums) {
    Arrays.sort(nums);
    List<List<Integer>> res = new ArrayList<>();
    for (int i = 0; i < nums.length; i++) {
        if (i > 0 && nums[i] == nums[i - 1]) continue;
        int l = i + 1, r = nums.length - 1;
        while (l < r) {
            int sum = nums[i] + nums[l] + nums[r];
            if (sum == 0) {
                res.add(Arrays.asList(nums[i], nums[l], nums[r]));
                while (l < r && nums[l] == nums[l + 1]) l++;
                while (l < r && nums[r] == nums[r - 1]) r--;
                l++; r--;
            } else if (sum < 0) l++;
            else r--;
        }
    }
    return res;
}
</details>
```

🔄 4. Reverse String / Array In-Place

🔸 **Main Problem:**
Reverse characters in-place using two pointers.

Input: `["h","e","l","l","o"]` → **Output:** `["o","l","l","e","h"]`

🔁 **Similar Questions:**

- Reverse vowels only
- Palindrome check
- Rotate array in-place

<details> <summary>💻 Code</summary>

Python

```
python
Copy code
def reverse_string(s):
    l, r = 0, len(s) - 1
    while l < r:
        s[l], s[r] = s[r], s[l]
        l += 1; r -= 1
```

C++

```
cpp
Copy code
void reverseString(vector<char>& s) {
    int l = 0, r = s.size() - 1;
    while (l < r) swap(s[l++], s[r--]);
}
```

Java

```
java
Copy code
public void reverseString(char[] s) {
    int l = 0, r = s.length - 1;
    while (l < r) {
        char temp = s[l];
        s[l++] = s[r];
        s[r--] = temp;
    }
}
```

</details>

🧊 5. Is Subsequence

🔸 **Main Problem:**
Check if `s` is a subsequence of `t`.

Input: `s = "abc", t = "ahbgdc"` → **Output:** `true`

🔁 **Similar Questions:**

- Longest Common Subsequence
- Matching string pattern
- Isomorphic string check

<details> <summary>💻 Code</summary>

Python

```
python
Copy code
def is_subsequence(s, t):
    i = j = 0
    while i < len(s) and j < len(t):
        if s[i] == t[j]:
            i += 1
        j += 1
    return i == len(s)
```

C++

```
cpp
Copy code
bool isSubsequence(string s, string t) {
    int i = 0, j = 0;
    while (i < s.size() && j < t.size()) {
        if (s[i] == t[j]) i++;
        j++;
    }
    return i == s.size();
}
```

Java

```
java
Copy code
public boolean isSubsequence(String s, String t) {
    int i = 0, j = 0;
    while (i < s.length() && j < t.length()) {
        if (s.charAt(i) == t.charAt(j)) i++;
        j++;
    }
    return i == s.length();
}
```