

# 1 General

## 1.1 Interviewer Considerations

### Notes:

- How did the candidate **analyze** the problem?
- Did the candidate miss any special or **edge** cases?
- Did the candidate approach the problem **methodically** and logically?
- Does the candidate have a strong foundation in basic computer science **concepts**?
- Did the candidate produce **working code**? Did the candidate **test** the code?
- Is the candidate's code clean and easy to read and **maintain**?
- Can the candidate **explain** their ideas clearly?

## 1.2 Steps for Success During the Technical Interview

### Summary:

1. **Clarify the question**
  - (a) Understand what the question is asking and gather example inputs and outputs.
  - (b) Clarify constraints such as:
    - i. Can numbers be negative or repeated?
    - ii. Are values sorted or do we need to sort them?
    - iii. Can we assume input validity?
  - (c) Asking clarifying questions shows communication skills and prevents missteps.
2. **Design a solution**
  - (a) Avoid immediate coding; propose an initial approach and refine it.
  - (b) Analyze the algorithm's time and space complexity.
  - (c) Consider and address edge cases.
  - (d) Think aloud to demonstrate logical reasoning and collaboration.
  - (e) Discuss non-optimal ideas to show your thought process.
3. **Write your code**
  - (a) Structure the solution using helper functions.
  - (b) Confirm API details when uncertain.
  - (c) Use your strongest programming language and full syntax.
  - (d) Write complete, working code—not pseudocode.
4. **Test your code**
  - (a) Validate your solution with 1–2 example test cases.
  - (b) Walk through each line using inputs.
  - (c) Do not assume correctness—prove it through testing.
  - (d) Discuss any further optimizations and their trade-offs.

## 1.3 Common Mistakes to Avoid

### Warning:

1. Starting to code without clarifying the problem.
2. Failing to write or discuss sample inputs and outputs.
3. Using pseudocode instead of fully functional code.
4. Misunderstanding the problem or optimizing prematurely.

## 2 Arrays

## 3 Hashing

### 3.1 When to Use?

**Summary:**

- 

### 3.2 Hashing

**Algorithm:**

### 3.3 Common Problems

**Summary:**

Problem	Description:
---------	--------------

- |            |                                                                                                      |
|------------|------------------------------------------------------------------------------------------------------|
| 1. Two Sum | Given an array of integers, return indices of the two numbers s.t. they add up to a specific target. |
|------------|------------------------------------------------------------------------------------------------------|

- **Tricks:**
  - See if  $\text{target} - \text{nums}[i]$  is in the map.
  - If it is, return the index of the  $\text{target} - \text{nums}[i]$  (from prevMap) and  $i$ .
    - \* `prevMap[nums[i]] = i`

## 4 Two Pointers

### 4.1 When to Use?

#### Summary:

- If we need to find a pair of elements that satisfy a condition.
- If we need to find a subarray that satisfies a condition.

### 4.2 Slow and Fast Pointers

#### Algorithm:

1.

#### 4.2.1 Common Problems

#### Summary:

Problem	Description:
15. 3Sum	Given an array of integers, return all the triplets [nums[i], nums[j], nums[k]] s.t. $i \neq j$ , $i \neq k$ , and $j \neq k$ .
<ul style="list-style-type: none"> <li>• <b>Tricks:</b></li> </ul>	
125. Valid Palindrome	Given a string, determine if it is a palindrome, considering only alphanumeric characters and ignoring cases.
<ul style="list-style-type: none"> <li>• <code>s_new = ".join(char.lower() for char in s if char.isalnum())</code> to remove non-alphanumeric and lowercase.</li> <li>• Use front and back pointers. If they not equal, return False. If equal move both pointers.</li> </ul>	
167. Two Sum II - Input array is sorted	Given an array of integers that is already sorted in ascending order, find two numbers such that they add up to a target.
<ul style="list-style-type: none"> <li>• Use front and back pointers. If <math>&gt;</math> target, move back pointer left. If <math>&lt;</math> target, move front pointer right.</li> </ul>	

## 4.3 Front and Back Pointers

### Algorithm:

1. Initialize two pointers, one at the front and one at the back of the array.

### 4.3.1 Common Problems

#### Summary:

Problem	Description:
15. 3Sum	Given an array of integers, return all the triplets [nums[i], nums[j], nums[k]] s.t. $i \neq j$ , $i \neq k$ , and $j \neq k$ .
<ul style="list-style-type: none"><li>• <b>Tricks:</b></li></ul>	
125. Valid Palindrome	Given a string, determine if it is a palindrome, considering only alphanumeric characters and ignoring cases.
<ul style="list-style-type: none"><li>• <code>s_new = ".join(char.lower() for char in s if char.isalnum())</code> to remove non-alphanumeric and lowercase.</li><li>• Use front and back pointers. If they not equal, return False. If equal move both pointers.</li></ul>	
167. Two Sum II - Input array is sorted	Given an array of integers that is already sorted in ascending order, find two numbers such that they add up to a target.
<ul style="list-style-type: none"><li>• Use front and back pointers. If <math>&gt;</math> target, move back pointer left. If <math>&lt;</math> target, move front pointer right.</li></ul>	

## 5 Sliding Window

### 5.1 Fixed Sliding Window

#### Summary:

- Find a subarray/substring of a fixed size that satisfies a condition.
- Find the maximum or minimum of a subarray of a fixed size.

#### Algorithm:

```
1 initialize window_sum = 0
2 initialize max_result (or other required value)
3
4 # Set up initial window
5 for i in range(0, k):
6     window_sum += arr[i]
7
8 max_result = window_sum # Initialize result
9
10 # Slide the window
11 for i in range(k, n):
12     window_sum += arr[i] - arr[i - k] # Add new element and remove 1st element of prev window
13     max_result = max(max_result, window_sum) (or other computation)
14
15 return max_result (or other required value)
16
```

#### 5.1.1 Common Problems

#### Summary:

Problem	Description:
643. Maximum Average Subarray I	Given an integer array nums and an integer k, return the maximum average value of a subarray of length k.
<ul style="list-style-type: none"><li>• Follow template.</li></ul>	

## 5.2 Dynamic Sliding Window

### Summary:

- Find longest or shortest subarray/substring that satisfies a condition.

### Algorithm:

```
1 initialize left = 0
2 initialize window_state (sum, count, frequency map, etc.)
3 initialize min_or_max_result
4
5 for right in range(n):
6     update window_state to include arr[right] # Expand the window
7
8     while window_state violates the condition:
9         update min_or_max_result (if needed)
10        update window_state to exclude arr[left] # Shrink the window
11        move left pointer forward
12
13 return min_or_max_result
```

### 5.2.1 Common Problems

#### Summary:

Problem	Description:
3. Longest Substring Without Repeating Characters	Given a string s, find the length of the longest substring without repeating characters.
<ul style="list-style-type: none"><li>• Use a frequency map to track characters in the current window.</li><li>• If a character is repeated, move the left pointer to the right of the last occurrence.</li></ul>	

## 6 Binary Search

### 6.1 When to Use?

**Summary:**

- If array is sorted.