⚡ Binary Search Interview Quick-Check Pattern

Use this 1-minute scan to align your thinking before solving any binary search problem in interviews.

🔍 1. Is the Data Sorted or Searchable?

- Is the array **sorted** or can it be sorted?
- Is the question about **minimum or maximum** possible values?
- Does the value **increase or decrease** with position? (Monotonic)

> 💡 *Binary Search only works on a clearly ordered search space (ascending, descending, or can be converted).*

🧠 2. What Type of Binary Search Does This Problem Need?

Goal	Technique
Find exact value/index	Standard Binary Search
Find first/last occurrence	Lower/Upper Bound (variant)
Optimize smallest/largest value	Binary Search on Answer
Search in rotated array	Modified Binary Search
Search in 2D matrix	Index math → `r = mid / cols`, `c = mid % cols`
Find precision value	Float Binary Search (e.g., square root)

> 🧠 *Know what you're solving for — is it a value, a position, a boolean condition, or a range?*

🧭 3. Understand the Return Target

- Do I need to return an **index**, **boolean**, **value**, or **range**?
- Do I need to return **one result**, or **multiple** (first and last)?
- Do I need to **store a result** when a condition is satisfied?

🧪 4. Write the Correct Binary Search Logic

✅ Standard Template:

```
let left = 0, right = arr.length - 1;
while (left <= right) {
  let mid = Math.floor((left + right) / 2);
  if (arr[mid] === target) return mid;
  else if (arr[mid] < target) left = mid + 1;
  else right = mid - 1;
}
```

🔍 Ask Yourself:

- What does `mid` represent?
- Do I shrink the search from `left` or `right`?
- Do I move `mid - 1`, `mid + 1`, or allow `mid` to be reused?

🔥 5. Binary Search on Answer

👀 When to Use:

- You're **not searching a value** in an array but rather a **threshold** (e.g., min speed, max size, min time).

- You can define a function like `isFeasible(value)`.

🧩 Pattern:

```
let left = minVal, right = maxVal;
while (left <= right) {
  let mid = Math.floor((left + right) / 2);
  if (isFeasible(mid)) {
    answer = mid; // or update high = mid - 1
    right = mid - 1;
  } else {
    left = mid + 1;
  }
}
```

🪲 6. Edge Cases to Always Check

- Empty array `[]`
- Single element `[5]`
- Target at beginning `[5,7,9]` target=5
- Target at end `[5,7,9]` target=9
- Target not in array `[5,7,9]` target=6

✅ Final Mental Checklist Before You Start Coding

- Is the input **sorted** or **monotonic**?
- What exactly is the **target** — value, condition, index, or range?
- Which **binary search pattern** fits this problem?
- Am I using correct `left`, `right`, and `mid` logic?
- Have I thought through **edge cases** and **boundary values**?

🧠 MENTAL MODEL

Picture shrinking window with 3 pointers:

```
[L · · · · M · · · · R]  ← Initial state
[L · M · · R]            ← If target < mid

[· · · L M · R]          ← If target > mid
```