Lower Bound and Upper Bound

Binary Search Based Interview Concepts

Placement Preparation Notes

Contents

Chapter 1

Introduction and Motivation

Binary Search is commonly misunderstood as an algorithm only used to *find elements*. In reality, most interview problems use Binary Search to **find positions and boundaries**.

Interview Reality: You are rarely asked to just "find X". You are asked:

- Where does X start?
- Where does X end?
- How many X exist?
- Where should X be inserted?

This chapter introduces **Lower Bound** and **Upper Bound** — two essential binary search patterns.

Chapter 2

Layman Analogy for Intuition

Analogy: Standing in a Queue by Height
Students are standing in increasing order of height.
Lower Bound: Where is the first student whose height is *at least* 160 cm?
Upper Bound: Where is the first student who is *strictly taller* than 160 cm?

This analogy works for all boundary-based binary search problems.

Chapter 3

Formal Definitions

3.1 Lower Bound

Lower Bound is the **smallest index** i such that:

$$arr[i] \geq target$$

Meaning:

- First element not smaller than target
- First valid insertion position

3.2 Upper Bound

Upper Bound is the **smallest index** i such that:

$$arr[i] > target$$

Meaning:

- First element strictly greater than target
- Points just after the last occurrence

Chapter 4

Worked Example

Given the sorted array:

$$arr = [1, 2, 4, 4, 4, 6, 8]$$

Target = 4

Concept	**Index**	**Value**
Lower Bound	2	4
Upper Bound	5	6

$$\textbf{Frequency of 4} = UB - LB = 5 - 2 = 3$$

Chapter 5

Binary Search Logic Building

5.1 Core Insight

Binary Search does NOT search for elements. It searches for the point where a condition changes from FALSE to TRUE.

5.2 Lower Bound Condition

$$arr[mid] \geq target$$

Decision Rule:

- Condition TRUE $\rightarrow$ move left
- Condition FALSE $\rightarrow$ move right

5.3 Lower Bound Algorithm

$$\begin{aligned}
low &= 0 \\
high &= n \\
\text{while } low < high &: \\
mid &= \lfloor \frac{low + high}{2} \rfloor \\
\text{if } arr[mid] < target &: \\
low &= mid + 1 \\
\text{else:} & \\
high &= mid \\
\text{return } low &
\end{aligned}$$

5.4 Upper Bound Difference

Only one line changes:

$$\text{if } arr[mid] \leq target \Rightarrow low = mid + 1$$

Chapter 6

Complexity Analysis

Metric	**Value**
Time Complexity	$O(\log n)$
Space Complexity	$O(1)$

Chapter 7

Common Mistakes and Tricky Parts

- Using binary search on unsorted arrays
- Confusing $\geq$ with $>$
- Using $high = n - 1$ instead of n
- Infinite loop due to incorrect mid update

Golden Rule:
Lower Bound uses ¡ Upper Bound uses $\leq$

Chapter 8

Most Asked Interview Questions

Q1. How do you count occurrences of an element?

$$\text{Count} = UpperBound - LowerBound$$

Q2. What if the element is not present?

Lower Bound returns the correct insertion position.

Q3. Can binary search work on an unsorted array?

No. Sorting is mandatory.

Q4. Why does Upper Bound skip all equal elements?

Because it uses the condition $>$ instead of $\geq$.

Chapter 9

Practice Questions with Answers

Q1

Array: $[2, 4, 6, 8]$, Target = 5
Lower Bound = 2

Q2

Array: $[1, 1, 1, 1]$, Target = 1
LB = 0, UB = 4, Frequency = 4

Q3

Array: $[3, 5, 7]$, Target = 10
LB = 3 (insert at end)

Chapter 10

Daily Memory Anchors

- Binary search finds boundaries, not values
- Lower Bound is first $\geq$ target
- Upper Bound is first $>$ target
- Frequency equals UB minus LB
- Conditions decide direction

Chapter 11

Final Summary

Lower Bound and Upper Bound are advanced binary search patterns used to locate positions, ranges, and counts.
They are critical for:

- Range queries
- Frequency counting
- Insert position problems
- Competitive programming
- Technical interviews