

# Searching – Binary Search

## 1. Introduction

**Binary Search** is an efficient searching algorithm used to find an element in a **sorted array**.

Instead of checking elements one by one, binary search **divides the search space into halves**, significantly reducing the number of comparisons.

Because of its efficiency, binary search is widely used in **large datasets and real-world systems**.

---

## 2. What is Binary Search?

Binary search works by:

- Comparing the target element with the **middle element** of the array
- Eliminating half of the array based on the comparison
- Repeating the process on the remaining half

This continues until:

- The element is found, or
  - The search space becomes empty
- 

## 3. Prerequisite for Binary Search

### Important Requirement

The array **must be sorted** (either ascending or descending).

Binary search **does not work on unsorted arrays**.

---

## 4. Why Do We Use Binary Search?

Binary search is used because:

- It is much faster than linear search

- It reduces time complexity drastically
  - Efficient for large datasets
  - Used in databases, libraries, and system-level applications
- 

## 5. Basic Idea Behind Binary Search

The idea is based on the **divide and conquer** technique:

1. Find the middle element
  2. Compare it with the target value
  3. If equal → element found
  4. If target is smaller → search left half
  5. If target is larger → search right half
  6. Repeat until found or range becomes invalid
- 

## 6. Logic for Binary Search (Plain English)

1. Start with the first and last index of the array
  2. Find the middle index
  3. Compare the middle element with the target
  4. If they match, stop the search
  5. If the target is smaller, move to the left half
  6. If the target is larger, move to the right half
  7. Repeat until the element is found or search range ends
- 

## 7. Visualization of Binary Search

Given sorted array:

```
[10, 20, 30, 40, 50, 60]
```

Target:

40

Search steps:

Middle → 30 (target > 30)

Right half → [40, 50, 60]

Middle → 50 (target < 50)

Left half → [40]

Found

## 8. Types of Binary Search

### 1 Iterative Binary Search

- Uses a loop
- More space-efficient
- Preferred in practice

### 2 Recursive Binary Search

- Uses function calls
- Easier to understand conceptually
- Uses extra stack space

## 9. Time and Space Complexity

Case	Time Complexity
Best Case	$O(1)$
Average Case	$O(\log n)$
Worst Case	$O(\log n)$

- **Space Complexity:**

- Iterative:  $O(1)$
  - Recursive:  $O(\log n)$
- 

## 10. Advantages of Binary Search

- Very fast searching
  - Efficient for large datasets
  - Fewer comparisons
  - Widely used in real applications
- 

## 11. Limitations of Binary Search

- Works only on sorted data
  - Sorting may add extra cost
  - Not suitable for frequently changing datasets
  - Implementation is more complex than linear search
- 

## 12. Real-World Applications

- Searching records in databases
  - Dictionary word lookup
  - Searching in phone directories
  - System-level file searching
  - Competitive programming problems
- 

## 13. Comparison: Linear Search vs Binary Search

Feature	Linear Search	Binary Search
Data Requirement	Unsorted allowed	Must be sorted
Time Complexity	$O(n)$	$O(\log n)$

Feature	Linear Search	Binary Search
Speed	Slow	Fast
Implementation	Simple	Moderate

---

## 14. Summary

- Binary search is a fast searching algorithm
  - Uses divide and conquer strategy
  - Requires sorted data
  - Time complexity is  $O(\log n)$
  - Highly efficient for large arrays
-