Binary Search

- Binary search operates by repeatedly dividing in half the portion of the list that could contain the target element and then comparing the target value with the middle element.

- The list must be sorted in ascending order for binary search to work effectively.

BinarySearch(arr, target):

```
left = 0
right = length of arr - 1
while left <= right:
    mid = (left + right) / 2
    if arr[mid] == target:
        return mid
    elif arr[mid] < target:
        left = mid + 1
    else:
        right = mid - 1
return -1  // Target not found
```

Time Complexity:

Best Case: O(1) : Target element is at the middle.

Average and Worst Case: O(log n) : Due to the halving of the search range in each step.

Space Complexity:

Binary search is an in-place algorithm, so the space complexity is O(1) (constant).

Applications:

- Binary search is widely used in searching algorithms, such as finding an item in a sorted array, binary search trees, and other data structures.

- It's a key algorithmic component in various computer science applications, including algorithms for sorting, searching, and more.