**1.9** Checks if a given number x exists in a sorted array arr using **binary search**. Analyze its time complexity using Big-O notation.

**AIM**:

To design an algorithm that checks if a given number x exists in a sorted array using binary search and analyze its performance.

**ALGORITHM:**

1. Start with two pointers:

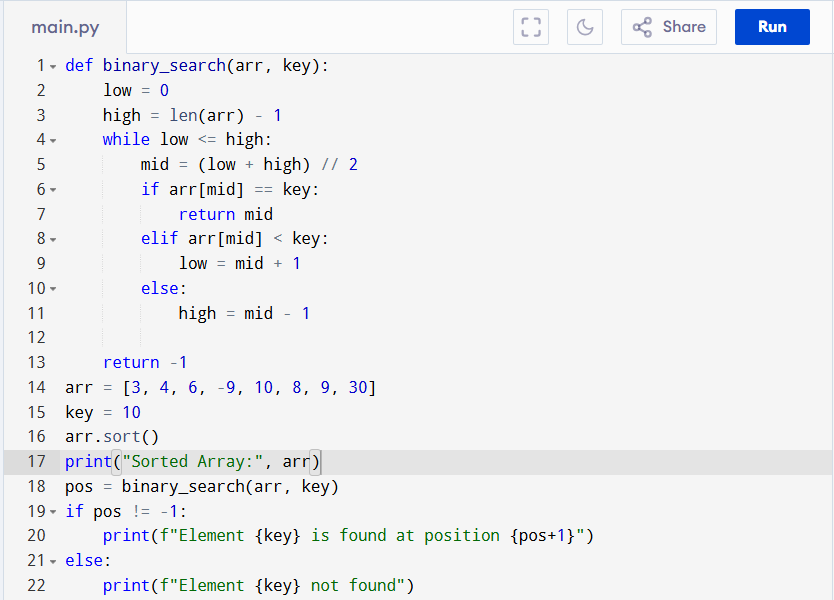
low = 0, high = len(arr)-1

1. Find the middle index:

mid = (low + high) // 2

1. If arr[mid] == key, element is found → return index.
2. If arr[mid] < key, search in the right half (low = mid+1).
3. If arr[mid] > key, search in the left half (high = mid-1).
4. Repeat until low > high.
5. If not found, return -1.

**PROGRAM:**



Input:

Array: [3, 4, 6, -9, 10, 8, 9, 30]

Key: 10

Output:

A screenshot of a computer program

AI-generated content may be incorrect.

**RESULT:**

Thus the binary search program is successfully executed, and the output is verified.

**PERFORMANCE ANALYSIS:**

Time Complexity

1. Best Case: Key found at the middle on first try → O(1)

2. Worst Case: Array repeatedly divided in half until one element left →

Number of comparisons = log₂(n) → O(log n)

3. Average Case:

Same as worst case since search space halves each step → O(log n)

Space Complexity:

* Iterative version uses O(1) extra space.
* Recursive version would use O(log n) stack space.