3.7 BINARY SEARCH IMPLEMENTATION WITH COMPARISON COUNT

Question:

Implement the Binary Search algorithm in a programming language of your choice and test it on the array 5,10,15,20,25,30,35,40,45 to find the position of the element 20. Execute your code and provide the index of element 20. Modify your implementation to count the number of comparisons made during the search process. Print this count along with the result.

AIM

To implement Binary Search in Python and count the number of comparisons made while locating a target element.

ALGORITHM

1. Initialize low = 0 and high = len(arr) - 1.
2. While low <= high:
    - Increment comparison counter.
    - Compute mid = (low + high) // 2.
    - If arr[mid] == target, return index.
    - If arr[mid] < target, search right half.
    - Else, search left half.
3. If not found, return -1.

PROGRAM

```
def binary_search(arr, key):
    left, right = 0, len(arr) - 1
    comparisons = 0
    while left <= right:
        mid = (left + right) // 2
        comparisons += 1
        if arr[mid] == key:
            return mid, comparisons
        elif arr[mid] < key:
            left = mid + 1
        else:
            right = mid - 1
    return -1, comparisons

def run_binary_search():
    N = int(input("Enter number of elements: "))
    arr = list(map(int, input("Enter sorted array elements: ").split()))
    key = int(input("Enter search key: "))
    index, count = binary_search(arr, key)
    print("Index:", index if index != -1 else "Not found")
    print("Comparisons:", count)

run_binary_search()
```

Input:

12, 54, 69, 73, 92

key=2,54

Output:

```
Enter number of elements: 5
Enter sorted array elements: 12 54 69 73 92
Enter search key: 2
Index: Not found
Comparisons: 2
>>>
================================================
Enter number of elements: 5
Enter sorted array elements: 12 54 69 92 97
Enter search key: 54
Index: 1
Comparisons: 3
>>>
```

RESULT:

Thus the program is successfully executed and the output is verified.

PERFORMANCE ANALYSIS:

- Time Complexity:
    - O(log n).
- Space Complexity:
    - O(1).