Ahaan Desai
231070016
Design and Analysis of Algorithms Lab

Laboratory 2 – Implementation of Linear and Binary Search

Aim: To implement Linear and Binary Searching algorithms.

Algorithm:

Linear Search Algorithm:

```
linear_search(array, element)
//Input: array -> Array of Integers, element -> Integer
//Output: Index of first occurrence of element in array (if found) or -1.
n = length(array)
for i = 0 to n-1:
        if array[i] == element:
                return i
return -1
```

Binary Search Algorithm:

```
binary_search(array, element, start, end)
//Input: array -> Array of Integers, element -> Integer, start -> Integer (default = 0),
end -> Integer (default = length(array) – 1)
//Output: Index of element in array (if found) or -1.
if start > end return -1
mid = (start + end) / 2
if array[mid] == element
   return mid
else if array[mid] > element
   return binary_search(array, element, start, mid - 1)
else
   return binary_search(array, element, mid + 1, end)
```

Testcases:

1. Linear Search Testcases

Array	Element	Expected Output
[1, 5, 4, 2, 3]	4	2
[10, 7, 15, 203, 51]	16	-1
[20, 25, 31, 400, 65]	25	1
[1, 9, -1, -2, -100, -61]	-2	3
[-105, 66, 111, 215, 330]	-60	-1

2. Binary Search Testcases

Array	Element	Expected Output
[6, 7, 8, 9, 10]	9	3
[100, 102, 104, 110, 115]	115	4
[21, 23, 24, 25, 28]	23	1
[52, 56, 57, 58, 60, 62]	55	-1
[-4, -3, -2, -1, 0]	-5	-1

Program:

```
def linear_search(arr, element):
    """Performs a linear search for the element in the array."""
    for i in range(len(arr)):
        if arr[i] == element:
            return i
    return -1

def binary_search(arr, element, start, end):
    """Performs a binary search for the element in the array, using recursion."""
    if start > end:
        return -1

    mid = (start + end) // 2

    if arr[mid] == element:
        return mid
    elif arr[mid] > element:
        return binary_search(arr, element, start, mid - 1)
    else:
        return binary_search(arr, element, mid + 1, end)

def linear_search_test():
    """Run testcases for the linear search function"""
    testcases = [
        # (array, element, expected_output)
        ([1, 5, 4, 2, 3], 4, 2),
        ([10, 7, 15, 203, 51], 16, -1),
        ([20, 25, 31, 400, 65], 25, 1),
        ([1, 9, -1, -2, -100, -61], -2, 3),
        ([-105, 66, 111, 215, 330], -60, -1),
    ]

    print("Linear Search tests:")
    for i, (array, element, expected_output) in enumerate(testcases):
        print(f"Test {i+1} - ", end="")
        output = linear_search(array, element)

        if output == expected_output == -1:
            print("Element not found")
        elif output == expected_output:
            print(f"Element found at index {expected_output}")
        else:
            print("Test failed!")

def binary_search_test():
    """Run testcases for the binary search function"""
```

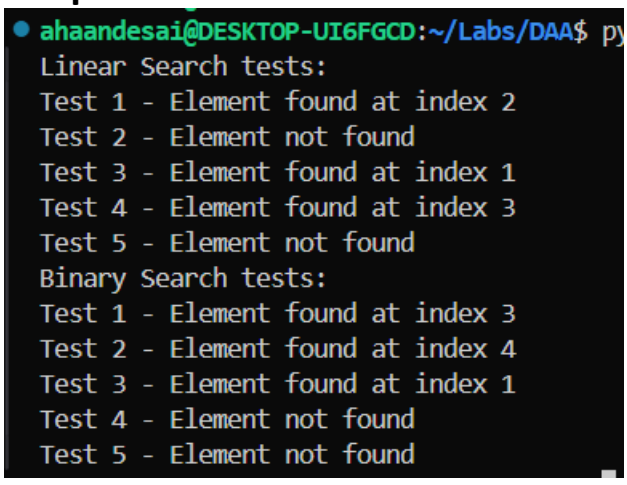
```
    testcases = [
        # (array, element, expected_output)
        ([6, 7, 8, 9, 10], 9, 3),
        ([100, 102, 104, 110, 115], 115, 4),
        ([21, 23, 24, 25, 28], 23, 1),
        ([52, 56, 57, 58, 60, 62], 55, -1),
        ([-4, -3, -2, -1, 0], -5, -1)
    ]

    print("Binary Search tests:")
    for idx, (array, element, expected_output) in enumerate(testcases):
        print(f"Test {idx+1} - ", end="")
        output = binary_search(array, element, 0, len(array) - 1)

        if output == expected_output == -1:
            print("Element not found")
        elif output == expected_output:
            print(f"Element found at index {expected_output}")
        else:
            print("Test failed!")

# Call tests
linear_search_test()
binary_search_test()
```

Output:



Conclusion:

We have studied the Linear and Binary Searching Algorithms

- In Linear Search, the array is traversed in a sequential manner, from index = 0 to index = length(array) – 1. If the element is found, its index is returned.
- In Binary Search, the element is compared to the median/middle value of the array. If they are equal, the index of the middle value is returned. Else, the function is called recursively by changing the range of the array considered, depending on whether the median value is larger or smaller than the element to be found.