09/08/24 DAA LAB - 02

PAGE NO.:
DATE / /

Linear & Binary search.

1. ALGORITHm for Linear Search.

// This algorithm searches for target using Linear search.
// Input: 1 array containing all elements, target value
// Output: Index position if target is found, -1 if not found.

```
linear_search (arr[], target) {

    ind ← -1
    for i in range (0, arr.size()):

        if (arr[i] == target)
            ind ← i
            break
    return ind
}
```

2. ALGORITHm for Binary Search.

// This algorithm searches for target using binary search
// Input: 1 array containing all elements, target value
// output: Index position if target is found, -1, if not found.

~~binary~~

PAGE NO.:
DATE: / /

binary_search algorithm:

```
beg ← 0
end ← arr.size() - 1.
binary_search (beg, end, target)
{
    if (beg <= end)
        mid = (beg + end) / 2
        if (arr[mid] == target)
            return mid.

        else if (arr[mid] < target)
            beg ← mid + 1
        return binary_search(beg, end, target)

        else if (arr[mid] > target)
            end ← mid - 1
        return binary_search(beg, end, target)

    else
        return -1
}
```

PAGE NO.:
DATE / /

Testcases :- LINEAR SEARCH.

1. arr = [8, 2, 5, 3] target = 2

output = 2 is present at index 1 ⟹ POSITIVE

2. arr = [13, 3, 7, 9, 6] target = 6

output = 6 is found at index 4 ⟹ POSITIVE.

3. arr = [1, 7, 20, 23] target = 1

output = 1 is found at index 0 ⟹ POSITIVE.

4. arr = [8, 2, 5, 3] target = 19

output = 19 is not found in array ⟹ Negative.

5. arr = [] target = 4

output = Error: Array is empty ⟹ Negative.

PAGE NO.:
DATE / /

Testcases :- Binary search

1. arr = [2, 3, 4, 5, 6] target = 4

Output : 4 found at ~~position~~ index 2

⇒ POSITIVE

2. arr = [5, 13, 19, 27] target = 13

Output : 13 found at index 1

⇒ POSITIVE

3. arr = [4, 8, 12, 16, 20] target = 16

output : 16 found at index 3

⇒ POSITIVE

4. arr = [ ] target = 3

Output : Error : Array is empty

⇒ Negative

5. arr = [3, 6, 9, 12, 15] target = 11

output : 11 not found in array

⇒ Negative

6. arr = [6, 3, 1, 9, 13] target = 9

output : Error : Array entered is not sorted.

⇒ Negative

PAGE NO.:
DATE / /

Time Complexity → LINEAR SEARCH.

Input size is the size of the array.
Let array size = n.

∴ Input size = n

∴ Inside the loop only 1 statement is present which checks if value at current index is equal to target.

$$\therefore M(n) = \sum_{i=0}^{n-1} 1. \quad = ((n-1) - 0 + 1)$$

$$\therefore M(n) = n$$

$$T(n) \approx C_m \times M(n)$$

$$\therefore T(n) \approx C_m \times n$$

... where C_m is the time for 1 basic operation.

∴ TIME COMPLEXITY = $O(n)$

PAGE NO.:
DATE / /

Time Complexity – Binary Search

Let size of array be n.

∴ Input size = n.

We know in binary search after each iteration array size is halved.

Let time required be T(n) for array of n elements

$$\therefore T(n) = T(n/2) + Cn$$

... Cn is the constant time required for calculating and checking middle element.

1st iteration: $\therefore T(n/2) = T(n/4) + Cn$

2nd iteration: $T(n/4) = T(n/8) + Cn$

⋮

kth iteration: $T\left(\frac{n}{2^k}\right) = T\left(\frac{n}{2^{k+1}}\right) + Cn$. ∴

After k^{th} iteration, the array size becomes 1

$$\therefore \frac{n}{2^k} = 1$$

$$\therefore 2^k = n$$

$$\therefore k = \log_2 n$$

$$\therefore T(n) = \log_2 n + c$$

∴ Time Complexity = $O(\log_2 n)$

∴ Algorithm is Logarithmic.