LAB2 DAA

Q1] Pseudo Code for Linear Search

```
void Linear_Search ( )
A[] ← input    integer k ← input
if A.size == 0
   print → Array is Empty
   return
else:
   index = -1
   for i = 0 to A.size() - 1
        if A[i] == k
          index = i
          break
   ~~return index~~
   if index == -1
      print → Element not found
   else
      print → Element found at index i
```

Test Cases with Expected Output

A1 = [10, 20, 30, 40, 50]   ~~A2 = [5, 15, 25~~

A2 = [5, 15, 25, 35, 45, 55, 65]

A3 = [ ]

1] A = A1   k = 20 → Element found at index 1

2] A = A1, k = 30 → Element found at index 2

3] A = A2, k = 55 → Element found at index 5

4] A = A1, k = 100 → Element not found

5] A = A3, k = 20 → Error!! Vector is Empty

Q2 Pseudo Code for Binary Search
→ Binary Search(arr[], left, right, k)
  if arr.size() == 0
    print → Error! vector is Empty
    return -1
  if (left > right)
    print → Element not found
    return -1
  mid = (left + right) / 2

  if arr[mid] == k
    print → Element found at index 'mid'.
    return mid
  if arr[mid] > k
    return Binary_Search(arr, left, mid-1, k)
  else
    return Binary_Search(arr, mid+1, right, k)

# Testcases
A1 = [10, 20, 30, 40, 50]
A2 = [5, 15, 25, 35, 45, 55, 65]
A3 = [ ]

1] arr = A1, left = 0, right = arr.size() - 1, k = 20
Output: Element found at index 1.
2] arr = A1; left = 0, right = arr.size() - 1; k = 30
Output: Element found at index 2
3] arr = A2, left = 0, right = arr.size() - 1, k = 55
Output → Element found at index 5
4] arr = A1; left = 0, right = arr.size() - 1, k = 100
Output → Element not found
5] arr = A3, left = 0, right = arr.size() - 1, k = 20
Output → Error! Vector is Empty

Time Complexity

1] Linear Search

$n \rightarrow$ size of array / vector

$c_i \rightarrow$ cost of i^{th} comparision

$$T(n) = \sum_{i=1}^{n} c_i$$

- In Best case, element found at first index

$$T(n) = \sum_{i=1}^{1} c_i = c_1 = \text{constant}$$

$\therefore T(n) = \cancel{O(n)}\ O(1)$

- In Worst case, element found at last index

$$T(n) = \sum_{i=1}^{n} 1 = n$$

$\therefore T(n) = O(n)$

- In Average case

$$T(n) = \frac{1}{n} \sum_{i=1}^{n} i = \frac{1}{n} \times \frac{n(n+1)}{2} = \frac{n+1}{2}$$

$\therefore T(n) = O(n)$

2] Binary Search Using Recursion

$n \rightarrow$ size of array, $c \rightarrow$ cost of each comparision

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

$T(n/2) = T(n/4) + c$

$\therefore T(n) = T(n/4) + 2c$

$\vdots$

$T(n) = \cancel{c \log_2} \; c \log_2 n + T(1)$

$\therefore T(n) = c \log_2 n + O(1)$

$\therefore T(n) = O(\log_2 n)$

- Best case $\rightarrow T(n) = O(1)$

Worst case $\rightarrow T(n) = O(\log_2 n)$

Avg case $\rightarrow T(n) = O(\log_2 n)$

Conclusion → Hence we have learned how to perform linear and Binary search on a list of number with error handling for negative test cases and time complexity for both searching algorithms. time complexity of Binary Search is less than that of linear search.