Linear Search Algorithm -

// Input: An array of size n, size n, a integer to be searched key.

// output: Index of occurence of ~~code~~ key in array.

```
for i to n
  if arr[i] == key    // if value of key is found
    return i          // return index of key
  else return -1      // if
```

- If the value is not found return -1 which will print ~~index~~ key not found.

LINEAR SEARCH

M T W T F S S
Page No.
Date
YOUVA

Test cases for Linear Search

positive test cases

Testcase1: Input: arr = { 4, 3, 7, 2, 5 } key = 2
Expected output : 3
as the key is found at index 3.

Testcase2: Input : arr = { 12, 7, 14, 5, 8 } key = 7
Expected output : 1
as the key is found at index 1

Testcase3: Input : arr = { 4, 7, 3, 5, 18 } key = 12
Expected output : key not found
as no index value matches the key value.

Negative test cases

Testcase4: Input : arr = { 4, 7, 3, 7, 12 } key = 7
Expected output : 0
as the first index of occurence will be returned

Testcase5: Input : arr = { } key = 8
Expected output : array is empty
as the array has no elements.

Testcase6: Input : arr = { 4, 8, 5, 15, 25, 30 } key = 15
Expected output : 3
as key is at index 3 of array.

Time complexity →
Worst case - O(n)
Best case - O(1)

Binary Search Algorithm -

// Input : An array of size n, size n, an integer to be searched key
// Output : Index of occurence of key in array

Set low to 0 and high to len(arr) - 1
while low ≤ high
- Compute mid as (low + high)/2
- If arr[mid] equals key return mid
- If arr[mid] < key, set low to mid + 1
- Else set high to mid - 1

If loop ends and key is not found
return -1 ; // Return not found.

BINARY SEARCH

Page No.
Date:
YOUVA

Test Cases for Binary Search

positive Test Cases →

Case 1: Input: arr = {1, 3, 7, 9, 14} key = 7
Expected output = 2
as the key is at middle position 2

Case 2: Input: arr = {3, 4, 5, 7, 9, 11, 15} key = 11
Expected output = 5
as the key is at index 5.

Case 3: Input: arr = {10, 20, 30, 40, 50} key = 40
Expected output = 3
as the key is at index 3.

Negative Test Cases →

case 4: Input: arr = {5, 7, 12, 13, 15, 18} key = 11
Expected output = -1
as the key is not present

case 5: Input: arr = { } key = 5
Expected output = -1
as the key is not present in empty array.

case 6: Input: arr = {10, 20, 30, 40, 50} key = 7
Expected output = -1
as the key is smaller than smallest value.

Time complexity:
Best case - O(1)
Worst case - O(log n)