Experiment No :- 8

Title - Write a program to implement various searching algorithms.

Objective - To implement linear and binary searching operation.

Theory -

- What is searching ?

1) Searching in data structure refers to the process of finding a desired element in set of items.
2) The desired element it is called "target"
3) Searched in any data structure like list, array, linked-list, tree or graph.
4) Element is found - search is successful.
5) Element is not found - search is unsuccessful.

- Types of searching

1) linear or sequential search
2) Binary search.

1) Linear or sequential search

1) Searching of specific element in given array is linear or sequential
2) We access each element of an array one by one sequentially & see wheather it is desired element or not.
3) It compares the target element with all the elements sequentially in an array to find if the element is present in the list or not.

Algorithm -

step 1 :- Traverse the array elements using a for loop.
step 2 :- Match/compare the key element with array element
step 3 :- If key element is found, return the index position of array element. If key element is not found, then move to the next element.
step 4 :- If there is no match or key element is not present in the given array, return -1 or null.

Time Complexity of linear search -

1) Best-case time complexity is o(1).
2) Average-case time complexity is o(n)
3) Worst-case time complexity is o(n)

steps :- 1) i=0
2) if i>n; go to step 7
3) if A[i]=x; go to step 6
4) i=i+1
5) Go to step 2
6) Print x found at location i
7) Print element not found
8) Exit.

e.g :-

```
def linearsearch(array, n, x):
    for i in range(0,n):
        if (array[i] == x):
            return i
    return -1

array = [2,4,0,1,9]
```

```
x = 1
n = len(array)
result = linear_search(array, n, x)
if (result == -1):
    print("Element not found")
else:
    print("Element found at index:", result)
```

2) Binary search

1) Binary search is a searching algorithm used in a sorted array.
2) Binary search works effeciently on sorted lists by repeatedly dividing given array.
3) Array is divided into two halves & the item is compared with the middle element of the list.

Algorithm -

step 1 :- set beg = lower_bound, end = upper_bound
step 2 :- Repeat steps 3 and 4 while beg <= end
step 3 :- set mid = (beg + end) / 2
step 4 :- if a[mid] = item
        set pos = mid
        print pos go to step 6
        else
          if a[mid] > item
          set end = mid - 1
          else
            set beg = mid + 1
step 5 :- if a[mid] != item
        print "Item is not present in the array"
step 6 :- exit.

Binary Search Complexity

1) Best Case Time Complexity is o(1)
2) Average Case Time Complexity is o(logn)
3) Worst Case Time Complexity is o(logn)

e.g ⇒ A[7] = {10,15,20,25,29,41,45} find X = 45

low = 0, high = 6, mid = low + high / 2 = 0 + 6 / 2 = 3

step	low	high	mid	A(mid)		X
1-	0	6	3	25	<	45
2	4	6	5	41	<	45
3	6	6	6	45	=	45

element found

Conclusion

I have understand various operations in linear or sequential search and Binary search.