Search Algorithms

ddas.tech/search-algorithms/

October 2, 2022

Created By : Debasis Das (Oct 2022)

Table of Contents

Linear Search Algorithm

- Search Algorithm is one of the simplest algorithm where we Start from the leftmost element of arr[] and one by one compare x with each element of arr[]
- If x matches with an element, we return the index.
- If x doesn't match with any of elements, we return -1.
- Time Complexity of the linear search algorithm is – O(n), The best case is O(1) if the first element in the array matches the search element and worst case is O(n)

```
arr = [1,2,3,6,76,21,32]

if 2 in arr:
    print("2 found at index = ",arr.index(2))

if 29 in arr:
    print("29 found at index = ",arr.index(2))
else:
    print("29 not found in the list")

2 found at index =  1
29 not found in the list

def search (arr, search_val):
    for i in range(len(arr)):
        if arr[i] == search_val:
            return i

    return -1  # -1 corresponds to not found

arr = [1,2,3,6,76,21,32]
print(search(arr,32))
print(search(arr,654))
```

```
6
-1
```

Binary Search Algorithm

Binary Search algorithm is a very efficient algorithm for collections which are already sorted.

- **Search** a collection of elements that is **already sorted** by ignoring half of the elements after just one comparison.
- Compare x (Search element) with the middle element.
- If x matches with the middle element, we return the mid index.
- Else if x is greater than the mid element, then x can only lie in the right (greater) half subarray after the mid element. Then we apply the algorithm again for the right half.
- Else if x is smaller, the target x must lie in the left (lower) half. So we apply the algorithm for the left half.

Iterative Binary Search

```
# Iterative Binary Search
def binarySearch(data,search_value):
    low = 0
    mid = 0
    high = len(data) - 1

    while low <= high:
        mid = (low + high) // 2
        if data[mid] < search_value:
            low = mid + 1
        elif data[mid] > search_value:
            high = mid - 1
        else:
            return mid
    return -1

array = [1,2,3,4,5,6,7,8,9,10]
print(binarySearch(array,11)) # -1
print(binarySearch(array,5)) # 4
```

Recursive Binary Search

```
# Recursive Binary Search
def recursiveBinarySearch(data,low,high,x):
    if high >= low:
        mid = low + (high - low) // 2
        if data[mid] == x:
            return mid

        elif data[mid] > x:
            # Object falls in the left side of the array
            return recursiveBinarySearch(data,low,mid-1,x)
        else:
            #Object falls in the right side of the array
            return recursiveBinarySearch(data,mid+1,high,x)

    else:
        return -1


array = [1,2,3,4,5,6,7,8,9,10]
searchElement1 = 4
print(recursiveBinarySearch(array,0,len(array)-1,searchElement1))

searchElement2 = 100
print(recursiveBinarySearch(array,0,len(array)-1,searchElement2))
```

This post will be continuously updated with an intention to demonstrate all search Algorithms