Date : 16th - 09 - 2020
Morning Session : 9am – 11.00 PM
By ~ Rohan Kumar

Topics: Binary Search

Search algorithms, or solutions to search problems, are an important part of many computer programs. Many programming applications depend on them. A general search problem will contain the following:

- Input: A list or an array (*A*) and the key element (*k*)
- Output: One of two return messages:
    - 1. k is found - if *k* exists in *A*
    - 2. k is not found - if *k* does not exist in *A*.

```
arr = [2,3,9,0,1,4,6]
x = 5
x in arr
```

```
False
```

Linear Search :

Using this search technique, we can compare each array element with the key element for which we're looking. It's a very simple algorithm, but we may need to check every element of the array. If the key element is found in the array, then the search is successful. If not, then the search is not successful.

```
arr = [2,3,9,0,1,4,6]
x = 5
def search(arr,x):
    for i in range(len(arr)):
        if arr[i] == x:
            return i
    return -1
search(arr,x)
```

```
-1
```

In a linear search, the time complexity is $O(n)$, where n is the length of the array A.

This complexity arises because, in the worst case:

- The key element may be in the last position of the array (n steps).
- The key element does not exist in the array, so we'll have to run through all of the array elements (n steps).

Binary Search

Binary Search a sorted array by repeatedly dividing the search interval in half. Begin with an interval covering the whole array. If the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half. Otherwise narrow it to the upper half. Repeatedly check until the value is found or the interval is empty.

Let's assume that array A is sorted. We can take advantage of this fact to improve the search time $O(n)$ by using the binary search algorithm.

The main idea behind the binary search algorithm is to divide the given sorted array A into two subarrays at each step and look for the key element k in one of the two subarrays.

First, we'll select the middle element of the array (p). Here, p is located at the index $(l + (h - 1)) / 2$ of the array, where l is the first index of the array and h is the last one. If p is equal to k, then a found message is returned. If p is greater than k, then k will be searched in the sub-array to the left of the middle item.

Otherwise, k is searched in the sub-array to the right of the middle item. This process will continue on the sub-array as well. In each step, we'll update the h and l values to match the current subarray. If l becomes greater than h (k does not exist in A), then we'll terminate the search and a message "k is not found" message will be returned.

The idea of binary search is to use the information that the array is sorted and reduce the time complexity to O(Log n).

binarysearch.py > lastOccurence

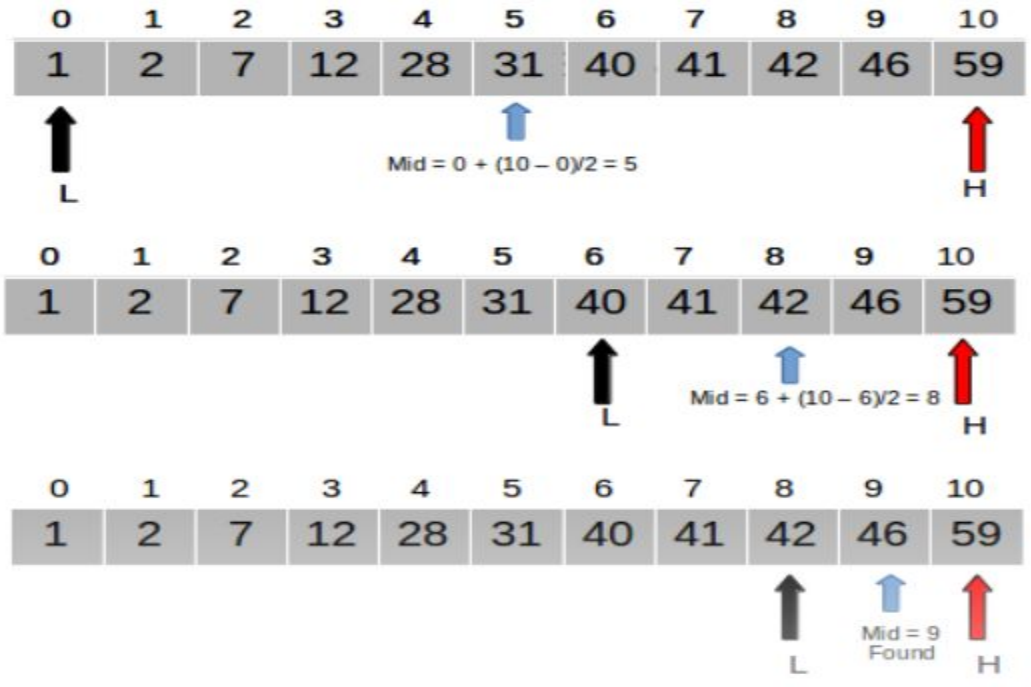
```python
def binarySearch(arr,x):
    start = 0
    end = len(arr) - 1
    while start <= end:
        mid = (start + end)//2
        if arr[mid] == x:
            return mid
        elif arr[mid] < x:
            start = mid + 1
        else:
            end = mid - 1
    return -1
def firstOccurence(arr,x):
    start = 0
    end = len(arr) - 1
    first = -1
    while(start <= end):
        mid = (start + end)//2
        if arr[mid] == x:
            first = mid
            end = mid - 1
        elif arr[mid] < x:
            start = mid + 1
        else:
            end = mid - 1
    return first
```

```python
def lastOccurence(arr,x):
    start = 0
    end = len(arr) - 1
    last = -1
    while(start <= end):
        mid = (start + end)//2
        if arr[mid] == x:
            last = mid
            start = mid + 1
        elif arr[mid] < x:
            start = mid + 1
        else:
            end = mid - 1
    return last

arr = [2,5,7,9,11,11,11,13,15]
x = 11
print(lastOccurence(arr,x))
```

Output: 6

EXample2:



MCQ 1:

1. Where is linear searching used?
a) When the list has only a few elements
b) When performing a single search in an unordered list
c) Used all the time
d) When the list has only a few elements and When performing a single search in an unordered list

Answer : D

MCQ 2:

6. What is the best case and worst case complexity of ordered linear search?
a) O(nlogn), O(logn)
b) O(logn), O(nlogn)
c) O(n), O(1)
d) O(1), O(n)

Answer : D

MCQ 3:

10. Which of the following is a disadvantage of linear search?
a) Requires more space
b) Greater time complexities compared to other searching algorithms
c) Not easy to understand
d) Not easy to implement

Answer : B

MCQ 4:

What is the worst case for binary search?

a) O(nlogn)

b) O(logn)

c) O(n)

d) O(1)

Answer : B