Data Structures & Algorithms

22: Searching algorithms

(Linear and Binary)

Dr Ram Prasad Krishnamoorthy

Associate Professor
School of Computing and Data Science

ram.krish@saiuniversity.edu.in

SAI UNIVERSITY

Searching Problem

Searching Problem

Input: A sequence of n numbers $\langle a_1, a_2, \ldots, a_n \rangle$ stored in array $A[1:n]$ and a value x.

Output: An index i such that x equals $A[i]$ or the special value NIL if x does not appear in A.

Linear Search

Linear Search

Linear Search is also known as *Sequential Search.*

- An element in an array/list is searched sequentially one by one from the beginning to find a query element.
- Every element is considered as a potential match.
- The index of the array is located for the query element.
- If element not found, print appropriate message or NIL.

Linear Search

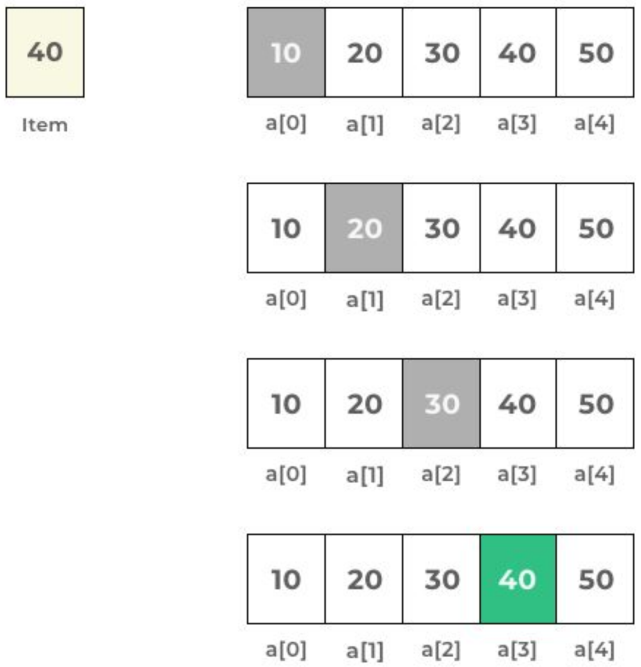
$$\text{LINEAR-SEARCH}\ (A, v)$$

$$\quad \textbf{for}\ i = 1\ \textbf{to}\ A.length$$

$$\qquad \textbf{if}\ A[i] == v$$

$$\qquad\quad \textbf{return}\ i$$

$$\quad \textbf{return}\ \text{NIL}$$



Linear Search

LINEAR-SEARCH (A, v)

for $i = 1$ **to** $A.length$

  if $A[i] == v$

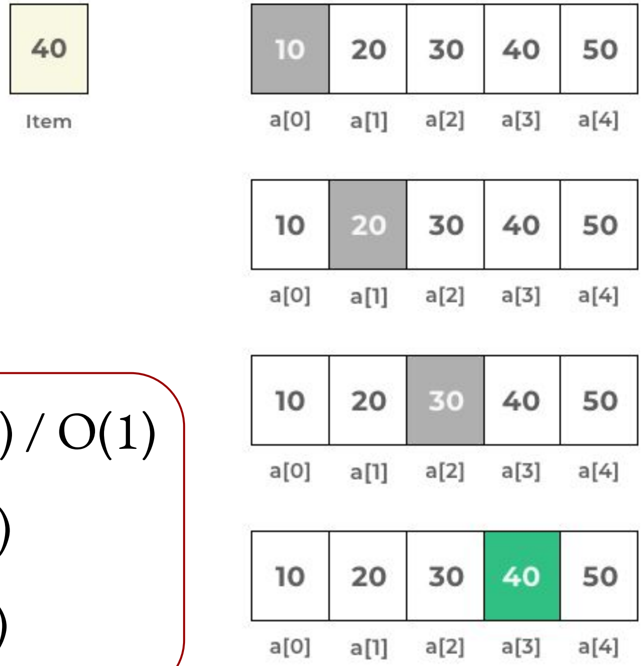
    return i

return NIL

Best case: $\Omega(1)$ / $O(1)$

Worst case: $O(n)$

Average case: $\Theta(n)$



Note: Easy to implement but not efficient for large arrays

Binary Search

Binary Search

Binary Search locates the query element in the **sorted** array.

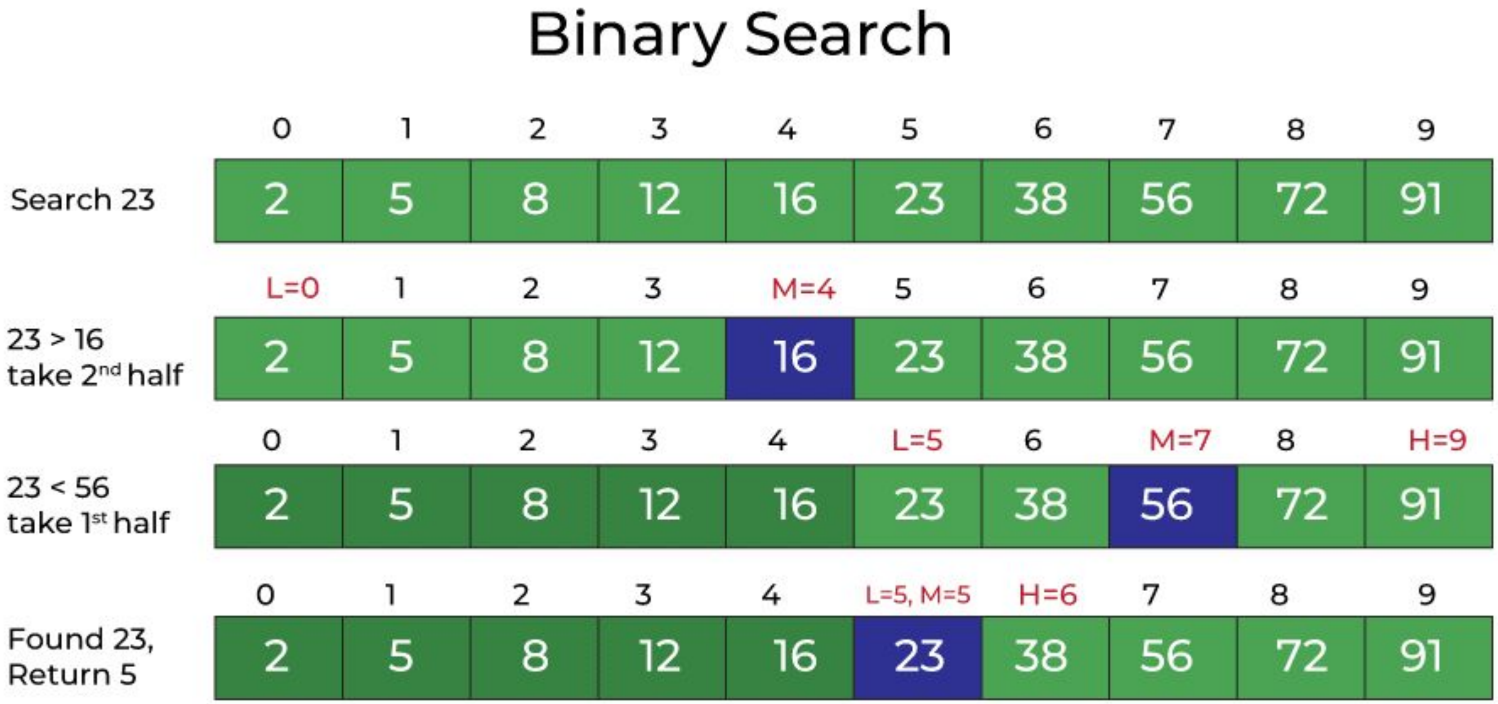
- The sorted array is divided into half and the query is compared with **middle** value.

- Based on the comparison, we decide whether to check the left-half or right-half of the array.

- If query is greater than mid-element, search in right-half other in the left-half.

- Keep repeating this process until element is found or no more division is possible.

Binary Search



Binary Search

$\textsc{Iterative-Binary-Search}\,(A, v)$

```
low = 1
high = A.length
while low ≤ high
    mid = ⌊(low + high)/2⌋
    if v == A[mid]
        return mid
    elseif v > A[mid]
        low = mid + 1
    else
        high = mid − 1
return NIL
```

Binary Search

ITERATIVE-BINARY-SEARCH (A, v)

```
low = 1
high = A.length
while low ≤ high
    mid = ⌊(low + high)/2⌋
    if v == A[mid]
        return mid
    elseif v > A[mid]
        low = mid + 1
    else
        high = mid − 1
return NIL
```

Best case: $O(1)$

Worst case: $O(\mathit{log}\ n)$

Average case: $\Theta(\mathit{log}\ n)$

Binary Search

$\text{Recursive-Binary-Search}\ (A, v, low, high)$

```
if low > high
    return NIL
mid = ⌊(low + high)/2⌋
if v == A[mid]
    return mid
elseif v > A[mid]
    RECURSIVE-BINARY-SEARCH(A, v, mid + 1, high)
else
    RECURSIVE-BINARY-SEARCH(A, v, low, mid − 1)
```