Searching Algorithms

Content

Linear Search

Pseudocode

1. Start from the leftmost element of arr[] and one by one compare x with each element of arr[].
2. If x matches with an element, return the index.
3. If x doesn't match with any of elements, return -1.

Example

Input : arr[] = {10, 20, 80, 30, 60, 50,
                110, 100, 130, 170}
    x = 110;

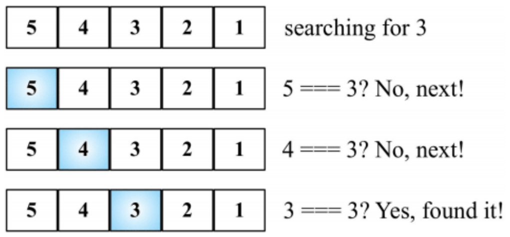
Output : 6

Element x is present at index 6

Input : arr[] = {10, 20, 80, 30, 60, 50,
110, 100, 130, 170}
x = 175;

Output : -1

Element x is not present in arr[].



Implementation -

https://github.com/lavishabhambri/Weekly-Algo-Newsletter/blob/main/Searching/Codes/LinearSearch.cpp

Time Complexity - O(n)

Linear search is rarely used practically because other search algorithms such as the binary search algorithm and hash tables allow significantly faster-searching comparison to Linear search.

Improve Linear Search Worst-Case Complexity

1. If element Found at last O(n) to O(1)
2. It is the same as the previous method because here we are performing 2 ‘if’ operations in one iteration of the loop and in the previous method we performed only 1 ‘if’ operation. This makes both the time complexities the same.

Binary Search

Problem

Given a sorted array arr[] of n elements, write a function to search a given element x in arr[].

Approach

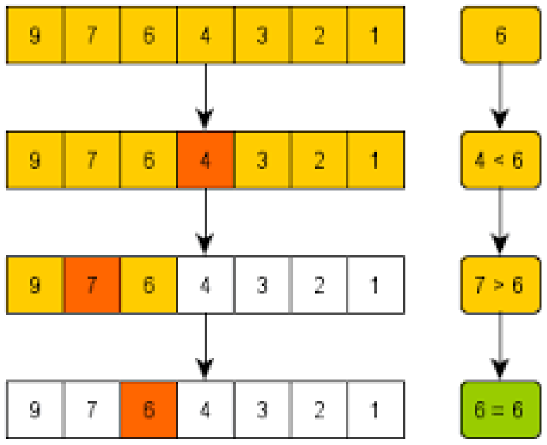
A simple approach is to do a **linear search.** The time complexity of the above algorithm is O(n). Another approach to perform the same task is using Binary Search.

Binary Search: 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.

Pseudocode

We basically ignore half of the elements just after one comparison.

1. Compare x with the middle element.
2. If x matches with the middle element, we return the mid index.
3. Else If x is greater than the mid element, then x can only lie in the right half subarray after the mid element. So we recur for the right half.
4. Else (x is smaller) recur for the left half.



Implementation

https://github.com/lavishabhambri/Weekly-Algo-Newsletter/blob/main/Searching/Codes/BinarySearch.cpp

Time Complexity

The time complexity of Binary Search can be written as

$T(n) = T(n/2) + c$

Auxiliary Space

O(1) in case of iterative implementation. In the case of recursive implementation, O(Logn) recursion calls stack space.

Algorithmic Paradigm

Decrease and Conquer