Binary Search

January 28, 2014

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

A simple approach is to do **linear search**, i.e., start from the leftmost element of arr[] and one by one compare x with each element of arr[], if x matches with an element, return the index. If x doesn't match with any of elements, return -1.

```
// Linearly search x in arr[].  If x is present then return its
// location,  otherwise return -1
int search(int arr[], int n, int x)
{
    int i;
    for (i=0; i<n; i++)
        if (arr[i] == x)
         return i;
    return -1;
}
```

The time complexity of above algorithm is O(n).

The idea of binary search is to use the information that the array is sorted and reduce the time complexity to O(Logn). We basically ignore half of the elements just after one comparison.

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

Following is **Recursive** C implementation of Binary Search.

```c
#include <stdio.h>

// A recursive binary search function. It returns location of x in
// given array arr[l..r] is present, otherwise -1
int binarySearch(int arr[], int l, int r, int x)
{
   if (r >= l)
   {
        int mid = l + (r - l)/2;

        // If the element is present at the middle itself
        if (arr[mid] == x)  return mid;

        // If element is smaller than mid, then it can only be prese
        // in left subarray
        if (arr[mid] > x) return binarySearch(arr, l, mid-1, x);

        // Else the element can only be present in right subarray
        return binarySearch(arr, mid+1, r, x);
   }

   // We reach here when element is not present in array
   return -1;
}

int main(void)
{
   int arr[] = {2, 3, 4, 10, 40};
   int n = sizeof(arr)/ sizeof(arr[0]);
   int x = 10;
   int result = binarySearch(arr, 0, n-1, x);
   (result == -1)? printf("Element is not present in array")
                 : printf("Element is present at index %d", result);
   return 0;
}
```

Output:

```
Element is present at index 3
```

Following is **Iterative** C implementation of Binary Search.

```c
#include <stdio.h>
```

```
// A iterative binary search function. It returns location of x in
// given array arr[l..r] if present, otherwise -1
int binarySearch(int arr[], int l, int r, int x)
{
  while (l <= r)
  {
    int m = l + (r-l)/2;

    if (arr[m] == x) return m;  // Check if x is present at mid

    if (arr[m] < x) l = m + 1; // If x greater, ignore left half

    else r = m - 1; // If x is smaller, ignore left half
  }
  return -1; // if we reach here, then element was not present
}

int main(void)
{
   int arr[] = {2, 3, 4, 10, 40};
   int n = sizeof(arr)/ sizeof(arr[0]);
   int x = 10;
   int result = binarySearch(arr, 0, n-1, x);
   (result == -1)? printf("Element is not present in array")
                 : printf("Element is present at index %d", result);
   return 0;
}
```

Output:

```
Element is present at index 3
```

Time Complexity:
The time complexity of Binary Search can be written as

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

The above recurrence can be solved either using Recurrence T ree method or Master method. It falls in case II of Master Method and solution of the recurrence is .

Auxiliary Space: O(1) in case of iterative implementation. In case of recursive implementation, O(Logn) recursion call stack space.

Algorithmic Paradigm: Divide and Conquer

Recent Discussions

Sumit Khatri this is the sorting technique which can work...

Insertion Sort · 7 hours ago

Sumit Khatri no, quick sort requires more swaps than...

Selection Sort · 7 hours ago

Sumit Khatri yes, it is the only sorting technique which...

Selection Sort · 7 hours ago

Sudhakar Mishra I think it should be 2n + 1

Following are some interesting articles based on Binary Search.
The Ubiquitous Binary Search
Interpolation search vs Binary search
Find the minimum element in a sorted and rotated array
Find a peak element
Find a Fixed Point in a given array
Count the number of occurrences in a sorted array
Median of two sorted arrays
Floor and Ceiling in a sorted array
Find the maximum element in an array which is first increasing and then decreasing

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Related Questions:

- Bubble Sort
- Selection Sort
- QuickSort
- Heap Sort
- Merge Sort
- Insertion Sort

Data Structures | Binary Trees | Question 12 · 8 hours ago

Sudhakar Mishra (2n)!/((n+1)!*n!)

Data Structures | Binary Trees | Question 6 · 1 day ago

Sudhakar Mishra Always Y will be more than one because after...

Data Structures | Stack | Question 7 · 1 day ago

5 Tweet 0 1

0 Comments GeeksQuiz

Sort by Best

Start the discussion…

Be the first to comment.

Subscribe Add Disqus to your site