GeeksQuiz

Computer science mock tests for geeks

Binary Search

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.

```
#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 present
        // 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.

```
#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 right 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 $\Theta(Logn)$.

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

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.

Category: Searching and Sorting

PG • 9 months ago

You guys are doing great work. Hats off to you GeeksForGeeks team.

8 • Reply • Share ›

Legolas • a year ago

Auxiliary space for recursive binary search O(log n) is correct but for practical case since this is tail recursion these space is optimized we dont need a different stack for a recursive call

7 • Reply • Share ›

reenu Legolas • a month ago

how the auxillary space for recursive binary search is o(logn)...m not getting.....if supoose element is not der...then m getting more than o(logn)..plz clarify

• Reply • Share ›

DS+Algo=Placement • 9 months ago

mid can easily be calculated by (l+r)/2, why to make it complicated by l+(r-l)/2 #Cormen-Instruction-Manual

2 • Reply • Share ›

DS+Algo=Placement DS+Algo=Placement • 9 months ago

I think the advantage of l+(r-l)/2 is that it avoids the overflow for large l

2 • Reply • Share ›

GeeksforGeeks Mod DS+Algo=Placement • 7 months ago

Please see http://www.geeksforgeeks.org/p... for details

2 • Reply • Share ›

Aditya Goel DS+Algo=Placement • 7 months ago

right, but even if l is small but r is large overflow can happen. One should keep in mind that overflow happens if (l+r) exceeds integer boundary

• Reply • Share ›

RK ➚ Aditya Goel • 3 months ago

Overflow will never happen for l+(r-l)/2, if the actual values of l and r are within the bounds of their data type.

^ | v • Reply • Share ›

rtshah ➚ DS+Algo=Placement • 2 months ago

That's for negative numbers . there is a rounding error in that case . So we use the following formula. I read that in the topcoder tutorial

1 ^ | v • Reply • Share ›

Hengameh • 16 days ago

How we check array is empty or not?
If both are "l" and "r" are 0, it means array has one element.
But what will happen when array is empty?

1 ^ | v • Reply • Share ›

RICKY KUMAR • 3 months ago

for int ary[]={1,2,4,6,8,10,12}; and x=6 its does not gives correct result

1 ^ | v • Reply • Share ›

Rohith Yeravothula • 7 months ago

what if there are multiple entries and we need all those indices ??

1 ^ | v • Reply • Share ›

GeeksforGeeks Mod ➚ Rohith Yeravothula • 7 months ago

We can find indexes of first and last occurrence in O(Logn) time. See http://www.geeksforgeeks.org/c... for details.

1 ^ | v • Reply • Share ›

ramboww • 9 months ago

can anbody tell me, if l+(r-l)/2 equal to (l+r)/2 ?

1 ^ | v • Reply • Share ›

DS+Algo=Placement ➚ ramboww • 9 months ago

Both are same but the advantage of l+(r-l)/2 is that it avoids the overflow for large l

1 ^ | v • Reply • Share ›

ramboww ➚ DS+Algo=Placement • 9 months ago

ok,i get it.3Q

^ | v • Reply • Share ›

12345@yopmail.com ➚ DS+Algo=Placement • 8 months ago

can u give an example wen it happens i mean the overflow

• Reply • Share ›

TenPowZero ramboww • 9 months ago

l+(r-l)/2 = l+ r/2 -l/2 = l-l/2+r/2 = l/2 + r/2 = (l+r)/2

1 • Reply • Share ›

ramboww TenPowZero • 9 months ago

but x/3*3 != x . eg: 7/3*3=6 != 7.so i'm not sure.

1 • Reply • Share ›

Hengameh • 8 days ago

for "lo=0" & "r=9", mid will be "4.5"; it will convert to "4'? since mid is an int?

• Reply • Share ›

Hengameh • 8 days ago

what is the point of last line in "Driver program"? " return 0; "
When it will be executed?

• Reply • Share ›

MingWei Jie • 2 months ago

I doubt the iterative version has a bug. Consider the following case, on a 32-bit machine, max signed-int may is 2147483647, the array has 2147483648 elements, l=0, r=2147483647, x is bigger than all elements. in while loop, l will get closer to r, when l is equal to r(2147483647), m==l, arr[m]<x, then="" l="m+1," overflow,="" l="">-2147483648, then the loop is endless. I think it should add check: when l is growing equal to r, if arr[m]!=x, then break the loop.

• Reply • Share ›

Guest • 2 months ago

as in the code:
l=0,r=4, then mid=0+(4-1)/2=1, but it should give mid=2.
So, plz explain the expression of getting mid index

• Reply • Share ›

guest Guest • 15 days ago

mid = L+(R-L)/2 NOT L+(R-1)/2

• Reply • Share ›

Giire • 3 months ago

Your code i.e. recursive one, is causing stack over flow after it is executed. Could u fix that problem please.

• Reply • Share ›

Ankit Bansal • 6 months ago

```
public void binary(int arg[],int start,int last,int searchingVariable) {

int middle = (start+last)/2;

if(arg[middle]==searchingVariable){

System.out.println("index ="+ middle);

}

else if (arg[middle]>=searchingVariable){

last=middle-1;

binary(arg, start, last, searchingVariable);

}

else {

start=middle+1;

binary(arg, start, last, searchingVariable);

}

}
```

• Reply • Share ›

paradise • 6 months ago

What is auxiliary space?

• Reply • Share ›

Guest • 7 months ago

what if there are multiple entries ??

• Reply • Share ›

atul • 8 months ago

while (l <= r)

condition will create array out of bound error.
as it will try to access the data which is at arr[l+1] location
it may cause crash....
why cant while (l < r) ?????????

• Reply • Share ›

beingawesomesince1991 atul • 7 months ago

try l<r and="" search="" for="" the="" element="" at="" index="" 0,="" you="" will="" get="" the="" answer="">

• Reply • Share ›

tihcar • 8 months ago

In iterative implementation , small correction in the comment
instead of
else r = m - 1; // If x is smaller, ignore left half

Should be

else r = m - 1; // If x is smaller, ignore right half

• Reply • Share ›

GeeksforGeeks Mod ➚ tihcar • 8 months ago

Thanks for pointing this out. We have updated the post.

1 • Reply • Share ›

shubhpy • 8 months ago

Which search method will be less time taking for an unsorted array?
I think it should be Linear.

• Reply • Share ›

deepak ➚ shubhpy • 5 months ago

If you have to search only few element, use linear search to search the element in O(n).

But you want to search many elements in same array, binary search will take less time as sorting is done only once in O(nlogn) and searching of each element in O(logn) compared to O(n) for each element in linear search.

• Reply • Share ›