Find a Fixed Point in a given array

Given an array of n distinct integers sorted in ascending order, write a function that returns a Fixed Point in the array, if there is any Fixed Point present in array, else returns -1. Fixed Point in an array is an index i such that arr[i] is equal to i. Note that integers in array can be negative.

Examples:

```
Input: arr[] = {-10, -5, 0, 3, 7}
Output: 3  // arr[3] == 3

Input: arr[] = {0, 2, 5, 8, 17}
Output: 0  // arr[0] == 0


Input: arr[] = {-10, -5, 3, 4, 7, 9}
Output: -1  // No Fixed Point
```

Asked by rajk

Method 1 (Linear Search)
Linearly search for an index i such that arr[i] == i. Return the first such index found. Thanks to pm for suggesting this solution.

```
int linearSearch(int arr[], int n)
{
    int i;
    for(i = 0; i < n; i++)
    {
        if(arr[i] == i)
```


Interview Experiences
Advanced Data Structures
Dynamic Programming
Greedy Algorithms
Backtracking
Pattern Searching
Divide & Conquer
Mathematical Algorithms
Recursion

```
            return i;
    }

    /* If no fixed point present then return -1 */
    return -1;
}

/* Driver program to check above functions */
int main()
{
    int arr[] = {-10, -1, 0, 3, 10, 11, 30, 50, 100};
    int n = sizeof(arr)/sizeof(arr[0]);
    printf("Fixed Point is %d", linearSearch(arr, n));
    getchar();
    return 0;
}
```

Time Complexity: O(n)

Method 2 (Binary Search)

First check whether middle element is Fixed Point or not. If it is, then return it; otherwise check whether index of middle element is greater than value at the index. If index is greater, then Fixed Point(s) lies on the right side of the middle point (obviously only if there is a Fixed Point). Else the Fixed Point(s) lies on left side.

```
int binarySearch(int arr[], int low, int high)
{
    if(high >= low)
    {
        int mid = (low + high)/2;  /*low + (high - low)/2;*/
        if(mid == arr[mid])
            return mid;
        if(mid > arr[mid])
            return binarySearch(arr, (mid + 1), high);
        else
            return binarySearch(arr, low, (mid -1));
    }

    /* Return -1 if there is no Fixed Point */
    return -1;
}

/* Driver program to check above functions */
int main()
```

Geometric Algorithms

Popular Posts

All permutations of a given string
Memory Layout of C Programs
Understanding “extern” keyword in C
Median of two sorted arrays
Tree traversal without recursion and without stack!
Structure Member Alignment, Padding and Data Packing
Intersection point of two Linked Lists
Lowest Common Ancestor in a BST.
Check if a binary tree is BST or not
Sorted Linked List to Balanced BST

```
{
    int arr[10] = {-10, -1, 0, 3, 10, 11, 30, 50, 100};
    int n = sizeof(arr)/sizeof(arr[0]);
    printf("Fixed Point is %d", binarySearch(arr, 0, n-1));
    getchar();
    return 0;
}
```

Algorithmic Paradigm: Divide & Conquer
Time Complexity: O(Logn)

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

Related Tpoics:

- Remove minimum elements from either side such that 2*min becomes more than max
- Divide and Conquer | Set 6 (Search in a Row-wise and Column-wise Sorted 2D Array)
- Bucket Sort
- Kth smallest element in a row-wise and column-wise sorted 2D array | Set 1
- Find the number of zeroes

- Find if there is a subarray with 0 sum
- Divide and Conquer | Set 5 (Strassen's Matrix Multiplication)
- Count all possible groups of size 2 or 3 that have sum as multiple of 3

Writing code in comment? Please use **ideone.com** and share the link here.

33 Comments GeeksforGeeks

Sort by Newest

aspire • 10 months ago

The code fails for : {-10, -5, 3, 4, 4, 5}

```
/* Paste your code here (You may delete these lines if not writing c
```


Jagdish ↗ aspire • 9 months ago

For duplicates, solution from "Coding Interview" book.
When we can not conclude on which side fixed point on, we can searc

Compare midIndex with Midvalue,
if equals return midIndex.
else
search leftside from start to Min(MidIndex-1,midvalue)

Recent Comments

Aman Hi, Why arent we checking for conditions...
Write a C program to Delete a Tree. · 15 minutes ago

kzs please provide solution for the problem...
Backtracking | Set 2 (Rat in a Maze) · 19 minutes ago

Sanjay Agarwal bool tree::Root_to_leaf_path_given_sum(tree...
Root to leaf path sum equal to a given number · 44 minutes ago

GOPI GOPINATH @admin Highlight this sentence "We can easily...
Count trailing zeroes in factorial of a number · 45 minutes ago

newCoder3006 If the array contains negative numbers also. We...
Find subarray with given sum · 1 hour ago

newCoder3006 Code without using while loop. We can do it...
Find subarray with given sum · 1 hour ago

search rightside from max(midIndex+1,midValue

It operates almost same if all elements are distinct

```
public static int fixedPoint(int[] array, int start, int end)
                if (end < start || start < 0 || end >= array.le
                        return -1;
                }
                int midIndex = start +(end-start)/ 2;
                int midValue = array[midIndex];
                if (midValue == midIndex) {
                        return midIndex;
```

see more

• Reply • Share ›

aspire aspire • 10 months ago

I am referring to the method 2 (Binary Search).
The check (mid>arr[mid]) does not ensure that the fixed point is on the
(mid<arr[mid]) does not ensure fixed point is on the left hand side only.
I suggest doing binary search for the left and right subarrays.

int ind = binarySearch(a,0,(n/2)-1);
if(ind==-1)
ind2 = binarySearch(a,n/2,n-1);

• Reply • Share ›

skulldude aspire • 10 months ago

The question specifies that the array contains distinct integers.
Your test case does not satisfy that condition.

And, the second method fails, if there are duplicate elements.

Hope it helps

-Balasubramanian.N

abhishek08aug • a year ago

```
#include<stdio.h>

int find_fixed_point(int array[], int start, int end) {
  if(start<end) {
    int mid = (start+end)/2;
    if(array[mid]==mid){
      return mid;
    } else if(array[mid]<mid) {
      return find_fixed_point(array, mid+1, end);
    } else if(array[mid]>mid) {
      return find_fixed_point(array, start, mid-1);
    }
  } else {
    return -1;
  }
}

int main(){
  int array[]={-11, -2, 0, 2, 3, 5, 45};
  int n=sizeof(array)/sizeof(array[0]);
  printf("A fixed point in the array is %d", find_fixed_point(array, (
  return 0;
}
```

Ganesh • a year ago

You can find java code here:

[sourcecode language="JAVA"]
/**
* Given an array of n distinct integers sorted in ascending order, write a functic
* a Fixed Point in the array, if there is any Fixed Point present in array, else ret
* Fixed Point in an array is an index i such that arr[i] is equal to i.
* Note that integers in array can be negative.
* Input: arr[] = {-10, -5, 0, 3, 7} Output: 3 // arr[3] == 3
* @author GAPIITD
*
*/
public class FixedPointInGivenArray {

public static void main(String[] args) {
int arr[] = {-10, -5, 0, 3, 7};
System.out.println(FindFixedPointInGivenArray(arr));
}

see more

• Reply • Share ›

Yogesh Mani • 2 years ago

```
 #include <iostream>
#include <conio.h>
using namespace std;
int findFixedPoint(int *arr,int leftIndex,int rightIndex) {
  int mid = (leftIndex+rightIndex)/2;
  if((leftIndex == rightIndex) && (arr[mid] != mid)) {
    return -1;
  }
```

```
  if(mid == arr[mid]){
    return mid;
  }
  int fp = (mid >arr[mid])?findFixedPoint(arr,mid+1,rightIndex):
                           findFixedPoint(arr,leftIndex,mid-1);
  return fp;
}

int main(void) {
//int arr[] = {-10, -5, 0, 3, 7};
```

see more

• Reply • Share ›

Anon • 2 years ago

The binary search method will not work in case of duplicate elements in the ar
part of the array in each iteration.

• Reply • Share ›

Sreenivas Doosa Anon • 2 years ago

You are correct. But please read the problem statement. The input arra

• Reply • Share ›

Avi • 2 years ago

```
 #include<stdio.h>
#include<conio.h>
int main()
{
   int arr[] = {-10, -5, 3, 4, 7, 9};

    int len = sizeof(arr)/sizeof(int);
    static int flag;
```

```
int num = -1;
for(int i = 0; i<len; i++)
{
    if(arr[i] == i)
    {
            flag = 1;
            num = arr[i];
            break;

    }
```

see more

Reply • Share ›

PsychoCoder • 2 years ago

In the function this portion is fixed irrespective of the given data.

```
 if(mid > arr[mid])
  return binarySearch(arr, (mid + 1), high);
else
  return binarySearch(arr, low, (mid -1));
```

So, if my input is

```
int arr[10] = {-10, -1, 0, 6, 10, 11, 30, 50, 100, 9};
```

then 5(mid) < 11 (arr[mid]) so it will neglect right hand side the value '9' which

Reply • Share ›

Yogesh Batra ➦ PsychoCoder • 2 years ago

This is clearly given that sequence is in increasing order, find some otl

method.

Reply • Share ›

kartikaditya • 2 years ago

[sourcecode language="C++"]
#include <iostream>
#include <stdio.h>

using namespace std;

int getFixedPoint(int a[], int n) {
int start = 0, end = n - 1, mid = 0;
while (start <= end) {
mid = (start + end) >> 1;
if (a[mid] == mid) {
return mid;
} else if (a[mid] < mid) {
start = mid + 1;
} else {
end = mid - 1;
}
}
return -1;

see more

Reply • Share ›

vijay_kansal • 2 years ago

An optimized algorithm for the case when elements can be repeated in the arr

```
int fixed_pt(int *a,int n)
{
 int l=0,h=n-1,m,i;
```

```
while(l<=h)
{
 m=(l+h)/2;
 if(a[m]>=0)
  h=m-1;
 else
  l=m+1;
}// finding the 1st element(it will be a[l]) >=0

for(i=l;i<n;i=a[i])
{
 if(a[i]==i)
  return i;
}
return -1;
}
```

The algo will return 1st element that is a fixed pt.

 • Reply • Share ›

vijay_kansal vijay_kansal • 2 years ago

An error correction,
Replace the if condition of while loop by :

```
if(a[m]>=m)
```

 • Reply • Share ›

Nages • 2 years ago

[sourcecode language="JAVA"]
public class FixedPointInArray {

```
public static void main(String[] arg) {

int arr[] = new int[] { -10, -5, 0, 3, 7 };

System.out.println(fixedPoint(arr));

arr = new int[] { 0, 2, 5, 8, 17 };

System.out.println(fixedPoint(arr));

arr = new int[] { -10, -5, 3, 4, 7, 9 };

System.out.println(fixedPoint(arr));
}

public static int fixedPoint(int[] arr) {
int pos = -1;
```

see more

• Reply • Share ›

Karthick • 2 years ago

I think, the following code can find the first Fixed Point.
It would be good if someone can check this code for correctness.

```
[sourcecode language="C++"]
int findFirstFixedPoint(int a[],int length)
{
if(length==0){ return -1; }
int start=0,end=length-1,middle;
while(start<=end)
{
middle=(start+end)/2;
```

if(a[middle]==middle)
{
if(middle==start || a[middle-1]!=middle-1){ return middle; }
end=middle-1;
}
else if(a[middle]<middle){ start=middle+1; }
else{ end=middle-1; }
}
return -1;
}

• Reply • Share ›

Karthick ➦ Karthick • 2 years ago

Sorry for posting without the sourcecode tags

```
 int findFirstFixedPoint(int length)
{
        if(length==0){ return -1; }
        int start=0,end=length-1,middle;
        while(start<=end)
        {
                middle=(start+end)/2;
                if(a[middle]==middle)
                {
                        if(middle==start || a[middle-1]!=middle
                        end=middle-1;
                }
                else if(a[middle]<middle){ start=middle+1; }
                else{ end=middle-1; }
        }
        return -1;
}
```

Reply • Share ›

kaka09 • 2 years ago

```
#include<stdio.h>
int main
{
        int n,i,j,flag=-1,pos;
        scanf("%d",&n);()
        int a[n];
        for(i=0;i<n;i++)
                                        scanf("%d",&a[i]);
    i=0;
    while(a[i++]<0)
    for(j=i;j<n;j++)
    {
                                        if(j==a[j])
                                        {
                                                        flag=0;
                                                        pos=j;
                                                        break;
                            }
```

see more

Reply • Share ›

kaka09 • 2 years ago

```
#include<stdio.h>
int main()
{
        int n,i,j,flag=-1,pos;
        scanf("%d",&n);
```

```
        int a[n];
        for(i=0;i<n;i++)
                                        scanf("%d",&a[i]);
    i=0;
    while(a[i++]<0)
    for(j=i;j<n;j++)
    {
                                        if(j==a[j])
                                        {
                                                        flag=0;
                                                        pos=j;
                                                        break;
                            }
```

see more

• Reply • Share ›

punit • 2 years ago

for handling equal values like [-1, 0, 4, 4, 4, 4, 4] just tweak binary search as fo

```
int bs(vector v, int l, int r)
{
int m = (l+r)/2;
if(l > r)
return -1;
if(v[m] == m)
return m;
if((v[m] > m) && (v[m] != v.at(m+1)))
{
return bs(v, l, m-1);
}
else
return bs(v, m+1, r);
```

}

Reply • Share ›

AG ➚ punit • 2 years ago

I think it fails for this test case:-

[0, 2, 5, 5, 5, 6, 7]

```
/* Paste your code here (You may delete these lines if not wri
```

Reply • Share ›

Yogesh Batra ➚ AG • 2 years ago

It's working, I don't think there is any problem with the above co

Reply • Share ›

yogendra ➚ Yogesh Batra • 2 years ago

@Yogesh
I checked the above code is failing in case of [0, 2, 5, 5,
in first ltr
l=0,r=5 => m=2;
so a[2]>2 and here a[2]==a[3]==5
so l=3;
in second ltr l=3,r=5 =>m=4;
s0 a[4]>6 hence it will return -1; int next iter.

Reply • Share ›

punit • 2 years ago

[sourcecode language="C++"]
/* Paste your code here (You may delete these lines if not writing code) */
[#include
using namespace std;

```
int bs(vector v, int l, int r)
{
int m = (l+r)/2;
if(l > r)
return -1;
if(v[m] == m)
return m;
if(v[m] > m)
{
return bs(v, l, m-1);
}
else
return bs(v, m+1, r);
```

see more

• Reply • Share ›

Daniel • 2 years ago

The non-distinct case would be interesting.

• Reply • Share ›

kartik ➔ Daniel • 2 years ago

I don't think binary search can be applied in case of non-distict case. F
to middle index 3 and see value is greater then index. We can't have a
array could be {-1, 0, 2, 4, 6, 8, 10} or {-1, 4, 4, 4, 5, 5, 5}. Let me know

• Reply • Share ›

krishna • 2 years ago

The problem would be more interesting if we ask for the first Fixed Point instea

• Reply • Share ›

@Jing, thanks for pointing the error. Given below is updated one.

```
#include <stdio.h>

int binarySearchLeft(int A[], int l, int r)
{
    int mid;
    int traced = -1;

    while( (r - l) > 1 )
    {
        mid = l + (r - l)/2;

        if( A[mid] == mid ) traced = mid;

        (mid <= A[mid] ? r : l) = mid;
    }
```

see more

Reply • Share ›

Venki krishna • 2 years ago

Driver code.

```
int main()
{
    int arr[] = {-10, 1, 2, 3, 10, 11, 30, 50, 100};
    int n = sizeof(arr)/sizeof(arr[0]);
    printf("Fixed Point is %d", binarySearch(arr, 0, n-1));

    return 0;
```

```
}
```


Venki krishna • 2 years ago

@krishna, Good question. Binary search is such a powerful technique, power. Solution to your question is given below. Let me know if there a

```
 int binarySearch(int A[], int l, int r)
{
    int mid;
    int traced = -1;

    while( (r - l) > 1 )
    {
        mid = l + (r - l)/2;

        if( A[mid] == mid )
            traced = mid;

        // C/C++ smart enough to confuse
        (mid <= A[mid] ? r : l) = mid;
    }

    if( A[l] == l )// Corner case : )
        traced = l;

    return traced;
}
```

Jing ➦ Venki • 2 years ago

For A=[-1, 0, 2], it seems to return -1 instead of 2, doesn't it?

```
/* Paste your code here (You may delete these lines if
```

  • Reply • Share ›

kartik ➦ krishna • 2 years ago

@Krishna: Thanks for suggesting a more advanced version of the prob
first FP. The method 2 can also be tweaked to find the first FP.

  • Reply • Share ›