Find the missing number in Arithmetic Progression

Given an array that represents elements of arithmetic progression in order. One element is missing in the progression, find the missing number.

Examples:

```
Input: arr[]  = {2, 4, 8, 10, 12, 14}
Output: 6

Input: arr[]  = {1, 6, 11, 16, 21, 31};
Output: 26
```

We strongly recommend to minimize your browser and try this yourself first.

A **Simple Solution** is to linearly traverse the array and find the missing number. Time complexity of this solution is O(n).

We can solve this problem **in O(Logn) time** using Binary Search. The idea is to go to the middle element. Check if the difference between middle and next to middle is equal to diff or not, if not then the missing element lies between mid and mid+1. If the middle element is equal to n/2th term in Arithmetic Series (Let n be the number of elements in input array), then missing element lies in right half. Else element lies in left half.

Following is C implementation of above idea.

```c
// A C program to find the missing number in a given
// arithmetic progression
#include <stdio.h>
#include <limits.h>

// A binary search based recursive function that returns
// the missing element in arithmetic progression
int findMissingUtil(int arr[], int low, int high, int diff)
{
    // There must be two elements to find the missing
    if (high <= low)
        return INT_MAX;

    // Find index of middle element
    int mid = low + (high - low)/2;

    // The element just after the middle element is missing.
    // The arr[mid+1] must exist, because we return when
    // (low == high) and take floor of (high-low)/2
    if (arr[mid+1] - arr[mid] != diff)
        return (arr[mid] + diff);

    // The element just before mid is missing
    if (mid > 0 && arr[mid] - arr[mid-1] != diff)
        return (arr[mid-1] + diff);

    // If the elements till mid follow AP, then recur
    // for right half
    if (arr[mid] == arr[0] + mid*diff)
        return findMissingUtil(arr, mid+1, high, diff);

    // Else recur for left half
    return findMissingUtil(arr, low, mid-1, diff);
}

// The function uses findMissingUtil() to find the missing
// element in AP.  It assumes that there is exactly one missing
// element and may give incorrect result when there is no missing
// element or more than one missing elements.
// This function also assumes that the difference in AP is an
// integer.
int findMissing(int arr[], int n)
```

```
{
    // If exactly one element is missing, then we can find
    // difference of arithmetic progression using following
    // formula.  Example, 2, 4, 6, 10, diff = (10-2)/4 = 2.
    // The assumption in formula is that the difference is
    // an integer.
    int diff = (arr[n-1] - arr[0])/n;

    // Binary search for the missing number using above
    // calculated diff
    return findMissingUtil(arr, 0, n-1, diff);
}

/* Driver program to check above functions */
int main()
{
    int arr[] = {2, 4, 8, 10, 12, 14};
    int n = sizeof(arr)/sizeof(arr[0]);
    printf("The missing element is %d", findMissing(arr, n));
    return 0;
}
```

Output:

```
The missing element is 6
```

Exercise:
Solve the same problem for Geometrical Series. What is the time complexity of your solution? What about Fibonacci Series?

This article is contributed by **Harshit Agrawal**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Related Topics:

- Find Union and Intersection of two unsorted arrays
- Pythagorean Triplet in an array
- Maximum profit by buying and selling a share at most twice
- Design a data structure that supports insert, delete, search and getRandom in constant time
- Print missing elements that lie in range 0 – 99
- Iterative Merge Sort
- Group multiple occurrence of array elements ordered by first occurrence
- Given a sorted and rotated array, find if there is a pair with a given sum

Tags: Divide and Conquer

Tweet 8+1 3

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

41 Comments **GeeksforGeeks** 1 **Login**

Recommend 1 Share Sort by Newest

Join the discussion…

123 • a month ago

```
#include <iostream>

using namespace std;

int main()

{

int arr[] = { 1, 6, 11, 16, 21, 31 };

int n = sizeof(arr) / sizeof(arr[0]) + 1;

int sum = (double)(arr[0] + arr[n - 2]) / 2 * n;

for (int i = 0; i < n - 1; ++i)

{

sum -= arr[i];

}

cout << sum << endl;

return 0;

}
```

• Reply • Share ›

ravi • a month ago

This will not work if last element or 1st element is the one which is missing
example 2,4,6,8,9 missing is 10 but d with this method will estimate to be wrong or 1,4,6,8,10

• Reply • Share ›

Manoj Saini ravi • 17 days ago

if last or first is missing, than it is already a complete..
We can not say that element is missing.

• Reply • Share ›

anirudh • 2 months ago

what if the no. of elements given to us is odd? i dont think its working for those cases.

• Reply • Share ›

Siya anirudh • a month ago
I am not able to see problem with odd number of elements. If (arr[n-1] - arr[0])/n; is an integer than there will be no problem otherwise this algo will fail.
• Reply • Share ›

rtp • 2 months ago
From this condition:

if (mid > 0 && arr[mid] - arr[mid-1] != diff)

we can eliminate the check for mid >0. Am i right? I couldn't come up with a test case for which mid >0 check is required.
• Reply • Share ›

Guest rtp • 2 months ago
Consider the AP {2,4,6,8,10}..........then you will keep recursive on the left subarray......and at the final left sub array call where you have only one element (So, mid = 0). There, you will try to do A[mid-1] which will give you an Array exception(Out of bound).

Please correct if I am wrong
• Reply • Share ›

rtp • 2 months ago
Why is the midpoint of array found like this

int mid = low + (high - low)/2;

instead of this:

int mid = (low + high) /2;
• Reply • Share ›

cfh rtp • 2 months ago
(low+high) can overflow but (high-low) can not.
1 • Reply • Share ›

Nayanava De • 2 months ago
Hi my friends. I feel that this solution won't work for cases when the AP is actually decreasing say 14,12,10,8,4,2. That case has to be handled using the condition :-
if(diff > 0){
if(array[mid] == array[0] + mid*diff)
return findMissingUtil(array,low, mid-1, diff);

return findMissingUtil(array,mid+1, high, diff);
}
else{
if(array[mid] == array[0] + mid*diff)
return findMissingUtil(array,mid+1, high, diff);
return findMissingUtil(array,low, mid-1, diff);
}

• Reply • Share ›

Subhadeep Karan Nayanava De • 2 months ago

It certainly works fine.

• Reply • Share ›

Harsh • 3 months ago

If I am not wrong we can solve it in O(n) .
Sum of AP if number is not missing =(n+1)/2*(a[0]+a[n-1]), as current a[n-1] will be the last number of AP.

Missing number = Above Sum - sum of array ??

O(n) is for sum of array.

• Reply • Share ›

Harsh Harsh • 2 months ago

http://ideone.com/DzWR6S

• Reply • Share ›

Subhadeep Karan Harsh • 2 months ago

why you would you want to solve in O(n)..when it can be solved in O(log n). As, O(log n) is much better than O(n)

• Reply • Share ›

prateek479 • 3 months ago

Hi @Harshit Agrawal.,
We have a o(n) solution available for this

we know sum of ap is n/2(first-term + last-term)

we also know that one term is missing in the given ap

what we will do is calculate the sum of array at time of insertion of array element let it be S.

now we store temp_sum as :

temp_sum = (arraySize+1)(a[0] + a[last]) // +1 coz one term is missing .

now
if(temp_sum < sum)
// definitely last term is missing coz first term can not be missing anyways.
missing-term = a[last] + (a[last] = a[last-1])

else
missing-term = sum - temp_sum;

• Reply • Share ›

Rajat__Gupta • 3 months ago

hey this one is a simple code.
plz do tell if it fails for any test condition.
http://ideone.com/NuUEmO

• Reply • Share ›

frank • 3 months ago

Hi, have you consider such test cases as that the missing number appear at the first position or the end. if so, the formular diff = (arr[n-1] - arr[0])/n would be wrong.

2 • Reply • Share ›

John frank • 3 months ago

If either the first or last number was missing then the array would show up as not missing any number in progression. Only one number can be missing, that's why.

• Reply • Share ›

harendra • 4 months ago

this is very simple way for solving this problem..

```
#include<stdio.h>

int main()

{

int arr[10],n,i,total=0,dif;

scanf("%d",&n);

for(i=0;i<n;i++) scanf("%d",&arr[i]);="" dif="(arr[n-1]-arr[0])/n;" total+="arr[0];" for(i="1;i&lt;n;i++)"
{="" if((total+dif)="=arr[i])" {="" total="total+dif;" }="" else="" {="" printf("%d",total+dif);=""
break;}="" }="" }="">
```

1 • Reply • Share ›

-{Smack}- • 4 months ago

If the input is {4, 6, 8, 10, 12, 14}

output is 9?

1 ^ | v • Reply • Share ›

Debanjan Chanda • 4 months ago

Any particular reason for which we are using recursion here?

Same approach (iterative way): http://ideone.com/2yoX7N with O(1) space.

^ | v • Reply • Share ›

Kenneth • 4 months ago

Here is my iterative solution:

http://ideone.com/fUooKc

^ | v • Reply • Share ›

Puneet Bhadauriya • 4 months ago

If we take a case like a[0] and a[n-1] element never be missing and logically it must not be then
step-1 Get sum of given array SumA=a[0]+....+a[n-1]
step-2 Get sum of Arithmetic series SumB=((lenght of array+1)(a[0]+a[n-1]))/2

Step-3 Missing term = (SumB-SumA)

2 ^ | v • Reply • Share ›

Mysterious Mind ➦ Puneet Bhadauriya • 4 months ago

In your case, the complexity will be o(n).

^ | v • Reply • Share ›

++C ➦ Mysterious Mind • 4 months ago

Not exactly , sum needs to be only computed once which in real life scenario can be done parallely as inputs are coming in. It's not a traversal per se and can be claimed to be constant.

1 ^ | v • Reply • Share ›

Abhishek • 4 months ago

```
#include <iostream>

using namespace std;

int main()

{

int arr[] = {1, 6, 11, 16, 21, 31};

int n;
```

int sn;

int m = 0;

n = sizeof(arr)/sizeof(arr[0]);

sn = ((n+1)*(arr[0]+arr[n-1]))/2;

for (int i = 0; i < n; i++)

m+=arr[i];

cout << sn-m << endl;

return 0;

}

^ | v • Reply • Share ›

Dev ➦ Abhishek • 4 months ago

Input = {1, 2, 3, 4,5, 6};
output = 3

Which is wrong,

^ | v • Reply • Share ›

Abhishek ➦ Dev • 4 months ago

what should be the output of your input.?

^ | v • Reply • Share ›

Dev ➦ Abhishek • 4 months ago

It should be either 0 or 7, isn't it ?
Its the case where first or last term is missing.

^ | v • Reply • Share ›

dev • 4 months ago

Hey ! why my comment has been deleted ?
Input = {1, 2, 3, 4,5, 6};
output = 3

Which is wrong,

^ | v • Reply • Share ›

Dev • 4 months ago

----- Wrong output -----
input = {1, 2, 3, 4, 5, 6}

output = 3

What if the first or last element of AP is missing.

Reply • Share ›

Abhishek → Dev • 4 months ago

That is not possible in this case.
Or if you consider this situation, then this problem is always present in every case. Let say for your case {1,2,3,4,5,6} the possible outputs are 0 and 7.

Reply • Share ›

Aman Sadhwani • 4 months ago

using Python

```
def enter(l):
diff=l[1]-l[0]
try:
for i in range(len(l)):
if l[i+1] - l[i] == diff:
pass
else:
missing=l[i] + diff
print missing
except IndexError:
pass

enter([1, 6, 11, 16, 21, 31])
```

Reply • Share ›

Dev → Aman Sadhwani • 4 months ago

Wrong,
what if second element is missing ? the concept of diff will go wrong

Reply • Share ›

Mandeep Rawat • 4 months ago

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace fibonacci
```

{

class missingTerm

{

public static void Main()

see more

Reply • Share ›

maddy • 4 months ago

can't we do it as checking:
if(low == high)
return array[low]-diff;

mid = low+(high-low)/2;
n = mid - low; //number of elements in between
if(array[mid] != low + diff*n) //first half
findMissingUtil(array,low,mid,diff)

else
findMissingUtil(array,mid+1,high,diff)

Reply • Share ›

Rainer Hoffmann • 4 months ago

Hello,

called with

arr[] = {2, 4, 8, 10, 12, 14}

returns -1

Could you please check?

Thanks

Reply • Share ›

GeeksforGeeks Mod Rainer Hoffmann • 4 months ago

Thanks for pointing this out. We have added one more if condition to check this type of cases.

Reply • Share ›

Mohit Jain • 4 months ago

Hey output of

Input: arr[] = {1, 6, 11, 16, 21, 31};
should be 26 not 27..

• Reply • Share ›

GeeksforGeeks Mod → Mohit Jain • 4 months ago

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

• Reply • Share ›

Amit Shakya → GeeksforGeeks • 3 months ago

We need to handle when no number is missing also.
Please check here: http://ideone.com/GyKy46
if(diff == arr[0])
{

}
else
{
//nothing is missing.
}

• Reply • Share ›

Subscribe Add Disqus to your site Privacy DISQUS

Google™ Custom Search

•
•
•

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

Follow @GeeksforGeeks

Recent Comments

- Nikhil kumar

  public class...

  Print missing elements that lie in range 0 – 99 · 5 minutes ago

- Ashish Aggarwal

  Try Data Structures and Algorithms Made Easy -...

  Algorithms · 27 minutes ago

- Vlad

  Thanks. Very interesting lectures.

  Expected Number of Trials until Success · 1 hour ago

- cfh

  My implementation which prints the index of the...

Longest Even Length Substring such that Sum of First and Second Half is same · 1 hour ago

- Gaurav pruthi

  forgot to see that part ;)

  Bloomberg Interview | Set 1 (Phone Interview) · 2 hours ago

- saeid aslami

  thanks

  Greedy Algorithms | Set 7 (Dijkstra's shortest path algorithm) · 2 hours ago

•