Check if array elements are consecutive | Added Method 3

Given an unsorted array of numbers, write a function that returns true if array consists of consecutive numbers.

Examples:
a) If array is {5, 2, 3, 1, 4}, then the function should return true because the array has consecutive numbers from 1 to 5.

b) If array is {83, 78, 80, 81, 79, 82}, then the function should return true because the array has consecutive numbers from 78 to 83.

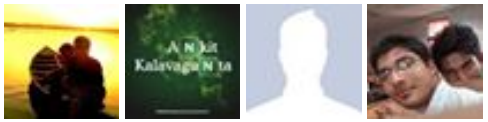
c) If the array is {34, 23, 52, 12, 3 }, then the function should return false because the elements are not consecutive.

d) If the array is {7, 6, 5, 5, 3, 4}, then the function should return false because 5 and 5 are not consecutive.

Method 1 (Use Sorting)
1) Sort all the elements.
2) Do a linear scan of the sorted array. If the difference between current element and next element is anything other than 1, then return false. If all differences are 1, then return true.

Time Complexity: O(nLogn)

Method 2 (Use visited array)
The idea is to check for following two conditions. If following two conditions are true, then return true.



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

1) *max – min + 1 = n* where max is the maximum element in array, min is minimum element in array and n is the number of elements in array.
2) All elements are distinct.

To check if all elements are distinct, we can create a visited[] array of size n. We can map the ith element of input array arr[] to visited array by using arr[i] – min as index in visited[].

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

/* Helper functions to get minimum and maximum in an array */
int getMin(int arr[], int n);
int getMax(int arr[], int n);
```

```
/* The function checks if the array elements are consecutive
  If elements are consecutive, then returns true, else returns
  false */
bool areConsecutive(int arr[], int n)
{
  if ( n <  1 )
    return false;

  /* 1) Get the minimum element in array */
  int min = getMin(arr, n);

  /* 2) Get the maximum element in array */
  int max = getMax(arr, n);

  /* 3) max - min + 1 is equal to n,  then only check all elements */
  if (max - min  + 1 == n)
  {
      /* Create a temp array to hold visited flag of all elements.
         Note that, calloc is used here so that all values are initial
         as false */
      bool *visited = (bool *) calloc (n, sizeof(bool));
      int i;
      for (i = 0; i < n; i++)
      {
         /* If we see an element again, then return false */
         if ( visited[arr[i] - min] != false )
           return false;

         /* If visited first time, then mark the element as visited */
         visited[arr[i] - min] = true;
      }
```


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

```
      /* If all elements occur once, then return true */
      return true;
  }

  return false; // if (max - min  + 1 != n)
}

/* UTILITY FUNCTIONS */
int getMin(int arr[], int n)
{
  int min = arr[0];
  for (int i = 1; i < n; i++)
   if (arr[i] < min)
     min = arr[i];
  return min;
}

int getMax(int arr[], int n)
{
  int max = arr[0];
  for (int i = 1; i < n; i++)
   if (arr[i] > max)
     max = arr[i];
  return max;
}

/* Driver program to test above functions */
int main()
{
    int arr[]= {5, 4, 2, 3, 1, 6};
    int n = sizeof(arr)/sizeof(arr[0]);
    if(areConsecutive(arr, n) == true)
        printf(" Array elements are consecutive ");
    else
        printf(" Array elements are not consecutive ");
    getchar();
    return 0;
}
```

Time Complexity: O(n)
Extra Space: O(n)

Method 3 (Mark visited array elements as negative)
This method is O(n) time complexity and O(1) extra space, but it changes the original array and it works only if all numbers are positive. We can get the original array by adding an extra step

though. It is an extension of method 2 and it has the same two steps.
1) *max – min + 1 = n* where max is the maximum element in array, min is minimum element in array and n is the number of elements in array.
2) All elements are distinct.

In this method, the implementation of step 2 differs from method 2. Instead of creating a new array, we modify the input array arr[] to keep track of visited elements. The idea is to traverse the array and for each index i (where 0 <= i < n), make arr[arr[i] - min]] as a negative value. If we see a negative value again then there is repetition.

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

/* Helper functions to get minimum and maximum in an array */
int getMin(int arr[], int n);
int getMax(int arr[], int n);
```

```
/* The function checks if the array elements are consecutive
  If elements are consecutive, then returns true, else returns
  false */
bool areConsecutive(int arr[], int n)
{

    if ( n <  1 )
        return false;

    /* 1) Get the minimum element in array */
    int min = getMin(arr, n);

    /* 2) Get the maximum element in array */
    int max = getMax(arr, n);

    /* 3) max - min + 1 is equal to n then only check all elements */
    if (max - min  + 1 == n)
    {
        int i;
        for(i = 0; i < n; i++)
        {
            int j;

            if (arr[i] < 0)
                j = -arr[i] - min;
            else
                j = arr[i] - min;
```


Recent Comments

Aman Hi, Why arent we checking for conditions...
Write a C program to Delete a Tree. · 11 minutes ago

kzs please provide solution for the problem...
Backtracking | Set 2 (Rat in a Maze) · 15 minutes ago

Sanjay Agarwal bool tree::Root_to_leaf_path_given_sum(tree...
Root to leaf path sum equal to a given number · 40 minutes ago

GOPI GOPINATH @admin Highlight this sentence "We can easily...
Count trailing zeroes in factorial of a number · 41 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

```
            // if the value at index j is negative then
            // there is repitition
            if (arr[j] > 0)
                arr[j] = -arr[j];
            else
                return false;
        }

        /* If we do not see a negative value then all elements
           are distinct */
        return true;
    }

    return false; // if (max - min  + 1 != n)
}

/* UTILITY FUNCTIONS */
int getMin(int arr[], int n)
{
    int min = arr[0];
    for (int i = 1; i < n; i++)
        if (arr[i] < min)
            min = arr[i];
    return min;
}

int getMax(int arr[], int n)
{
    int max = arr[0];
    for (int i = 1; i < n; i++)
        if (arr[i] > max)
            max = arr[i];
    return max;
}

/* Driver program to test above functions */
int main()
{
    int arr[]= {1, 4, 5, 3, 2, 6};
    int n = sizeof(arr)/sizeof(arr[0]);
    if(areConsecutive(arr, n) == true)
        printf(" Array elements are consecutive ");
    else
        printf(" Array elements are not consecutive ");
    getchar();
    return 0;
}
```

Note that this method might not work for negative numbers. For example, it returns false for {2, 1, 0, -3, -1, -2}.

Time Complexity: O(n)
Extra Space: O(1)

Source: http://geeksforgeeks.org/forum/topic/amazon-interview-question-for-software-engineerdeveloper-fresher-9

Please write comments if you find the above codes/algorithms incorrect, or find other ways to solve the same problem.

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.

137 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

Venu Gopal • 11 days ago

@GeeksforGeeks

Can you suggest some more examples which use the logic of 2nd step of me
as negative). Also if possible give me some link which explains this approach i

• Reply • Share ›

gyane • a month ago

Algo:
1)find min in the array
2)get xor of all elements
3)get xor all numbers between min and min +size-1
4)xor the result of step 2 and 3.
5)if result of step 5 is 0 return true
else return false
time:O(n)
space:O(1)

2 • Reply • Share ›

Tarzan • 2 months ago

You can also do this in constant space by taking sum (but by array modificati

You can also do this in constant space by taking sum (but by array modificati

1.) do one traversal and find min and max values.
2.) in second iteration, subtract the min value from array elements.
3.) Add all the elements of the array in a sum variable.
4.) The sum should be equal to n(n-1)/2 where n is the number of elements.
Reason for subtracting min values is to avoid adding large numbers where it n

• Reply • Share ›

G Veera Sekhar • 2 months ago

Use an auxilary array of same size.
1. Find min(array)
2. Pick each element and place at array[i] = array[min + i]
3. Once auxillary array is ready, check size ... if not equals to n, then there is c
4. If yes, run through the auxillary, check if any array[i] is zero(default) , if yes..

Time: O(n)
Space: O(n)

• Reply • Share ›

alien • 3 months ago

method 3 wont work if the input is as below:

-2,-1,0,1,2

• Reply • Share ›

Arunkumar Rajendran • 3 months ago

Than doing this.. we can sort the array and do the following check to find the s

```
if(arr[n-1] == arr[0]+(n-1)){
printf("True\n");
}
else
printf("false\n");
```

where n is the number of elements in the array.

• Reply • Share ›

Ankit • 4 months ago

This method uses O(n) time and O(1) space. It is based on mathematical forr
numbers, ((first_no+last_no)/2)*no_of_elements

http://ideone.com/zATln5

• Reply • Share ›

G Veera Sekhar → Ankit • 2 months ago

It might fail... check the case, where the array has elements(duplicate
consecutive elements.

• Reply • Share ›

Guest • 4 months ago

This method uses O(n) time and O(1) space. It uses mathematical formula th
((first_no+last_no)/2)*no_of_elements.

```
#include<iostream>
#include<cstdlib>
#include<climits>
using namespace std;

bool check_if_consecutive(int arr[], int len){
int minimum=INT_MAX;
int sum=0;
for(int i=0; i<len; i++){="" if(minimum="">arr[i])
minimum=arr[i];
sum+=arr[i];
}
int sum_formula=((minimum+minimum+len-1)/2)*len;
if(sum_formula==sum)
```

```
if(sum_formula == sum)
return true;
return false;
}

int main(){
int arr[]={4, 2, 3, 1, 0};
int len=sizeof(arr)/sizeof(arr[0]);
cout<<boolalpha<<check_if_consecutive(arr, len)<<endl;="" return="" exit_su
```

• Reply • Share ›

shar • 5 months ago

does the 2nd method work for the input 34 25 27 31 29 26 28 30 33 32 ?

• Reply • Share ›

Guest • 5 months ago

```
#include <iostream>
using namespace std;
int getMin(int arr[],int n)
{
int min=arr[0];
for(int i=1;i<n;i++) if(arr[i]<min)="" min="arr[i];" return="" min;="" }="" int="" get
max="arr[0];" for(int="" i="1;i&lt;n;i++)" if(arr[i]=">max)
max=arr[i];
return max;
}
bool areConsecutive(int arr[],int n)
{
int max,min,i,xori=0;
max=getMax(arr,n);
min=getMin(arr,n);
if(max-min+1==n)
{
```

```
for(i=min;i<=max;i++)
xori^=i;
for(i=0;i<n;i++) xori^="arr[i];" if(xori="=0)" return="" true;="" }="" return="" false;
arr[]="{7," 6,="" 5,="" 5,="" 3,="" 4};="" int="" n="sizeof(arr)/sizeof(arr[0]);" if(are
printf("="" array="" elements="" are="" consecutive="" ");="" else="" printf("="
consecutive="" ");="" getchar();="" return="" 0;="" }="">
```

Reply • Share ›

Nitin Khanna • 6 months ago

.

1 Reply • Share ›

Upen • 6 months ago

@admin I have a new method which will work in O(n) time and O(1) extra spa

Given Constraints are array elements should be in Consecutive manner and n
have duplicates elements :
My algorithm is :
Step 1: Find the total sum of array say Sum;
Step 2: Find the min element in array say a;
Step 3: Find the max element in array say b;
Now calculate the sum of range (a, a+1, a+2, ... , b)
Which is say __sum = (b*(b+1)/2 - a*(a+1)/2) + a;

if(__sum == Sum) return true;
else return false;

Correct me if i am wrong !!!

Reply • Share ›

Kartik → Upen • 6 months ago

This won't work. Please see the below comments.

• Reply • Share ›

Guest • 6 months ago

xor = (i+min(arr)) ^ arr[i] ;
if(!xor)
return true;
else
return false;

• Reply • Share ›

Tarzan → Guest • 2 months ago

This would only work if input is sorted.

• Reply • Share ›

Kuchhu • 7 months ago

Method 3 is wrong. it wont work for all inputs.

• Reply • Share ›

chriscracker • 7 months ago

Find Min , Find max , if(min+(size-1)==max) print true else print false..... t(n)=(

• Reply • Share ›

wasseypuriyan → chriscracker • 7 months ago

Oh really?
But we get incorrect answer of this sequence 1,2,2,3,4 using the meth

• Reply • Share ›

prakash • 8 months ago

one o(n) time and o(1)space solution::
influenced by xor elements to find missing number in a given range.

find min element in the array.-o(n) time

find max element in array - o(n) time
if(n!=(min-max+1)) return false;

xor_val1=min^(min+1)^.....^(max) -o(max-min) here it must be o(n)

xor_val2=arr[0]^arr[1]^....^arr[n-1]
if(!(xor_val1^xor_val2))

return true;
else
return false;

2 • Reply • Share ›

Shradha Agrawal ➦ prakash • 6 months ago

this method won't work. for eg:
arr = 5,2,2,5
min = 2 , max=5 , n = 4
so, n = (max-min+1)
xor_val1 = 0
xor_val2 = 0
but array doesn't contain contiguous numbers.

1 • Reply • Share ›

Marsha Donna • 8 months ago

```
#include<stdio.h>
#define ele 10000
void consec(int arr[],int n)
{
int min=arr[0],max=arr[0],i,counter[ele]={0};
for(i=0;i<n;i++) {="" counter[arr[i]]++;="" if(arr[i]="">max)    max=a
    if(arr[i]<min) min="arr[i];" }="" for(i="min;i&lt;max;i++)" if(cou
```

• Reply • Share ›

Guest • 8 months ago

```
#include<stdio.h>
#define ele 10000
void consec(int arr[],int n)
{
int min=arr[0],max=arr[0],i;
int counter[ele]={0};
for(i=0;i<n;i++) {="" counter[arr[i]]++;="" if(arr[i]="">max)
        max=arr[i];
    if(arr[i]<min) min="arr[i];" }="" for(i="min;i&lt;max;i++)" {="" :
```

• Reply • Share ›

Suryabhan Singh • 8 months ago

we can using hashing for O(n) space complexity
elements will be hashed from min to max hash-table

1 • Reply • Share ›

Guest • 8 months ago

```
#include<iostream>

#include<conio.h>

using namespace std;

int main()

{

    int arr[]={34,36,36,37,37 },n=5,i,max=arr[0],min=max,x_or_all,x_o
```

```
for(i=1;i<n;i++) {="" if(arr[i]="">max)

{

max=arr[i];

}

if(arr[i]<min) {="" min="arr[i];" }="" x_or_arr^=
```

• Reply • Share ›

Guest • 8 months ago

#include<iostream>

#include<conio.h>

using namespace std;

int main()

{

int arr[]={34,36,36,37,37 },n=5,i,max=arr[0],min=max,x_or_all,x_or_arr=arr[0];

for(i=1;i<n;i++) {="" if(arr[i]="">max)

{

max=arr[i];

}

if(arr[i]<min) {="" min="arr[i];" }="" x_or_arr^="arr[i];" }="" if(max-min+1!="n)" {=
}="" x_or_all="min;" for(i="min+1;i<=max;i++)" {="" x_or_all^="i;" }="" if(x_or_
}="" else="" {="" cout<<"false";="" }="" getch();="" return="" 0;="" }=">

```
}= else= {= cout<< false ,= }= getch(),= return= 0,= }= >
```

• Reply • Share ›

Guest • 8 months ago

```
#include<iostream>

#include<conio.h>

using namespace std;

int main()

{

    int arr[]={34,36,36,37,37 },n=5,i,max=arr[0],min=max,x_or_all,x_o

    for(i=1;i<n;i++) {="" if(arr[i]="">max)

                                max=arr[i];

                    if(arr[i]<min) min="arr[i];" x_or_arr^="arr[i];" 
```

• Reply • Share ›

Guest • 8 months ago

O(n) and O(1) time and space respectively using xor

```
#include<iostream>

#include<conio.h>

using namespace std;
```

```
using namespace std;

int main()

{

    int arr[]={34,36,36,37,37 },n=5,i,max=arr[0],min=max,x_or_all,x_o

    for(i=1;i<n;i++) {="" if(arr[i]="">max)

                                max=arr[i];

                    if(arr[i]<min) min="arr[i];" x_or_arr^="arr[i];" 
```

• Reply • Share ›

Guest • 8 months ago

Another O(n) solution with O(1) space complexity and without changing the ar

```
#include<iostream>

#include<conio.h>

using namespace std;

int main()

{

int arr[]={34,36,36,37,37 },n=5,i,max=arr[0],min=max,x_or_all,x_or_arr=arr[0];

for(i=1;i<n;i++) {="" if(arr[i]="">max)

max=arr[i];

if(arr[i]<min) min="arr[i];" x_or_arr^="arr[i];" }="" if(max-min+1!="n)" {="" cout<<
```

x_or_all="min;" for(i="min+1;i<=max;i++)" {="" x_or_all^="i;" }="" if(x_or_all^x
cout<<"false";="" getch();="" return="" 0;="" }="">

Reply • Share ›

asunel • 8 months ago

Code : Time complexity:O(n) Space complexity:O(1)

```
#include<stdio.h>

int areConsecutive(int arr[], int n)
{
    int i, min=arr[0], max=arr[0], xor=arr[0];
    for(i=1; i<n; i++)="" {="" xor^="arr[i];" min="arr[i]&lt;min?arr[
    }

    for(i=min; i<=max; i++)
    {
        xor^=i;
    }
    return (xor==0 && n==max-min+1)?1:0;
}
```

see more

Reply • Share ›

asunel • 8 months ago

Another solution in O(n) time complexity :
a) xor1 = xor of all elements of the given array
b) min = minimum element of the array
max = maximum element of the array

count = max-min+1
c) xor2 = xor of xor1 and all the integers in the range [min, max]
d) if (xor2==0 && count==size of the array) return true
else return false

• Reply • Share ›

Guest • 8 months ago

Another solution for the above problem in O(n) time complexity.

Main Concept :
For a given set of consecutive numbers, the sum of all elements will always b
n*(n+1)/2. Here n is the number of elements in the array.

In above equation, the second part is the sum of first n natural numbers.
The first part is added if the series starts from some number other than 1.

Hence, traverse the array once, sum all the elements on the go and get the mi
the sum to be equal to value from above equation.

Cons of above solution :
- Special behavior needed if 0 present in the array.
- If the consecutive series starts from a high number for ex 1235132. Integer o
multiplication. In that case, we can change the value of the elements by subtra
elements and verify the sum to be equal to n*(n+1)/2.

• Reply • Share ›

Kushagra Kumar • 10 months ago

In Step 2 of method 3, how about this modification to make it work for negative
entire array. If the array elements are consecutive then what we have is numb
size N. Simply, do this , arr[arr[i]%N] += N;.
Run a loop once again, ar[i]=ar[i]/N;.
If all elements == 1, return true; else false.

• Reply • Share ›

Srinivasa Rao Nalluri • 10 months ago

just kidding..... but to be specific you should say index of max and min.

• Reply • Share ›

Sarath Chandra Prasad • 10 months ago

Srinivasa Rao Nalluri ,
max-min=length of array-1, I think it&#039s fine according to my logic...can u p
where this logic will fail..

• Reply • Share ›

Srinivasa Rao Nalluri • 10 months ago

are you sure about your step3?

• Reply • Share ›

Sarath Chandra Prasad • 10 months ago

1. Sort the given array of elements.
2. check if duplicates exists.

```
//pseudo code
for(int i=0;i<n-1;i++) {.
if(array[i]==array[i+1]) {
duplicate found; flag=false;break.
}
}
```

If no duplicates exists then do following steps.

1. Find the min element.
2. Find the max element.
3. if max-min= length of array then the elements are consecutive else not.
4. Time complexity O(nlogn), space complexity O(1).

• Reply • Share ›

Siva Bhaskar • 10 months ago

3,5,5,5...try this

• Reply • Share ›

??????? ?????? • 10 months ago

1. sort the array.
2. sum=sum of the all elements of array.
3. min sum=sum of first n natural numbers where n=min(nothing but first elem
4.max sum=sum of first n natural numbers where n=max(last element of sort
5. if(sum== max sum-min sum).

return 1 (means consecutive).
else.

return 0 (not consecutive).

• Reply • Share ›

Prity Bhudolia • 10 months ago

@Amit jain got it. :)

• Reply • Share ›

Amit Jain • 10 months ago

xor won&#039t work for a[]={0,2,4,6,2,4.1}

• Reply • Share ›

Shekhar Singh • 10 months ago

If the array elements are large, then addition can overflow..
see the method given by shek8034
Taking XOR is the best method, it will never overflow.

• Reply • Share ›

Prity Bhudolia • 10 months ago

can be done this way.
1) Get the minimum element in array
2) Get the maximum element in array
3)Find xor of elements from min to max and store as result
4)xor the elements of array with result
5)if result is 0, return true as xor of same elements is 0

```
int areConsecutive(int arr[], int n)
{

if ( n < 1 )

return 0;

/* 1) Get the minimum element in array */

int min = getMin(arr, n);

/* 2) Get the maximum element in array */

int max = getMax(arr, n);
```

see more

• Reply • Share ›

Prity Bhudolia • 10 months ago

Thanks Vishal Singh my bad. Got the point :D.

• Reply • Share ›

Vishal Singh • 10 months ago

suppose you have array elements as (1,2,4,4,4,6)
then your getMin=1 and getMax=6 and sumofarrayelements=21
now sumrequired=21 as well
your function will return true which is wrong in this case.

• Reply • Share ›

pritybhudolia • 10 months ago

I think this is very simple approach and works for all cases.
1)get the min number
2)get maximum number
3)calculate the sum from min to max.
4)find sum of arrray elements
5)If both sum are equal , return true else false.

```
 int areConsecutive(int arr[], int n)
{
  if ( n <  1 )
    return 0;
  int i,sumrequired=0,sumofarrayelements=0;

  /* 1) Get the minimum element in array */
  int min = getMin(arr, n);

  /* 2) Get the maximum element in array */
  int max = getMax(arr, n);
```

see more

• Reply • Share ›

pritybhudolia ➦ pritybhudolia • 10 months ago

sorry ignore it as it wont work for many cases.

• Reply • Share ›

Asap ➦ pritybhudolia • 10 months ago

For your xor solution:
What about input {1,2,2,5,5}?

• Reply • Share ›

Prity Bhudolia • 10 months ago

GeeksforGeeks why cant we solve it this way. I think this is simple approach a
1)get the min number.
2)get maximum number.
3)calculate the sum from min to max.
4)find sum of arrray elements.
5)If both sum are equal , return true else false.
int areConsecutive(int arr[], int n).
{
if (n < 1).

return 0;.
int i, sumrequired=0, sumofarrayelements=0;.

/* 1) Get the minimum element in array */.
int min = getMin(arr, n);.

/* 2) Get the maximum element in array */.
int max = getMax(arr, n);.
/* 3)calculate the sum of all elements in array */.

see more

• Reply • Share ›

bpsingh • 10 months ago

O(n) time, O(1) space solution without changing original array
1. Get min element(say min)
2. Get max element(say max)
3. let x = (min*(min-1))/2;
4. let y = (max*(max+1))/2;
5. Add all elements(say total)
6. If y = x + total, return true else return false;