Find the element that appears once

Given an array where every element occurs three times, except one element which occurs only once. Find the element that occurs once. Expected time complexity is O(n) and O(1) extra space.
Examples:

```
Input: arr[] = {12, 1, 12, 3, 12, 1, 1, 2, 3, 3}
Output: 2
```

We can use sorting to do it in O(nLogn) time. We can also use hashing, but the worst case time complexity of hashing may be more than O(n) and hashing requires extra space.

The idea is to use bitwise operators for a solution that is O(n) time and uses O(1) extra space. The solution is not easy like other XOR based solutions, because all elements appear odd number of times here. The idea is taken from here.

Run a loop for all elements in array. At the end of every iteration, maintain following two values.

ones: The bits that have appeared 1st time or 4th time or 7th time .. etc.

twos: The bits that have appeared 2nd time or 5th time or 8th time .. etc.

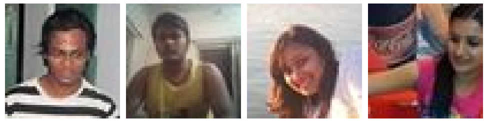
Finally, we return the value of 'ones'

How to maintain the values of 'ones' and 'twos'?
'ones' and 'twos' are initialized as 0. For every new element in array, find out the common set bits in the new element and previous value of 'ones'. These common set bits are actually the bits that should be added to 'twos'. So do bitwise OR of the common set bits with 'twos'. 'twos' also gets some extra bits that appear third time. These extra bits are removed later.

Google™ Custom Search

GeeksforGeeks

53,526 people like GeeksforGeeks.



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

Update ‘ones’ by doing XOR of new element with previous value of ‘ones’. There may be some bits which appear 3rd time. These extra bits are also removed later.

Both ‘ones’ and ‘twos’ contain those extra bits which appear 3rd time. Remove these extra bits by finding out common set bits in ‘ones’ and ‘twos’.

```
#include <stdio.h>

int getSingle(int arr[], int n)
{
    int ones = 0, twos = 0 ;

    int common_bit_mask;

    // Let us take the example of {3, 3, 2, 3} to understand this
    for( int i=0; i< n; i++ )
    {
        /* The expression "one & arr[i]" gives the bits that are
           there in both 'ones' and new element from arr[].  We
           add these bits to 'twos' using bitwise OR

           Value of 'twos' will be set as 0, 3, 3 and 1 after 1st,
           2nd, 3rd and 4th iterations respectively */
        twos  = twos | (ones & arr[i]);


        /* XOR the new bits with previous 'ones' to get all bits
           appearing odd number of times

           Value of 'ones' will be set as 3, 0, 2 and 3 after 1st,
           2nd, 3rd and 4th iterations respectively */
        ones  = ones ^ arr[i];


        /* The common bits are those bits which appear third time
           So these bits should not be there in both 'ones' and 'twos'
           common_bit_mask contains all these bits as 0, so that the b
           be removed from 'ones' and 'twos'

           Value of 'common_bit_mask' will be set as 00, 00, 01 and 10
           after 1st, 2nd, 3rd and 4th iterations respectively */
        common_bit_mask = ~(ones & twos);


        /* Remove common bits (the bits that appear third time) from '
```

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

```
           Value of 'ones' will be set as 3, 0, 0 and 2 after 1st,
           2nd, 3rd and 4th iterations respectively */
        ones &= common_bit_mask;


        /* Remove common bits (the bits that appear third time) from '

           Value of 'twos' will be set as 0, 3, 1 and 0 after 1st,
           2nd, 3rd and 4th itearations respectively */
        twos &= common_bit_mask;

        // uncomment this code to see intermediate values
        //printf (" %d %d \n", ones, twos);
    }

    return ones;
}
```

```
int main()
{
    int arr[] = {3, 3, 2, 3};
    int n = sizeof(arr) / sizeof(arr[0]);
    printf("The element with single occurrence is %d ",
            getSingle(arr, n));
    return 0;
}
```

Output:

```
2
```

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

Following is another O(n) time complexity and O(1) extra space method suggested by *aj*. We can sum the bits in same positions for all the numbers and take modulo with 3. The bits for which sum is not multiple of 3, are the bits of number with single occurrence.
Let us consider the example array {5, 5, 5, 8}. The 101, 101, 101, 1000
Sum of first bits%3 = (1 + 1 + 1 + 0)%3 = 0;
Sum of second bits%3 = (0 + 0 + 0 + 0)%0 = 0;
Sum of third bits%3 = (1 + 1 + 1 + 0)%3 = 0;
Sum of fourth bits%3 = (1)%3 = 1;

Hence number which appears once is 1000

```
#include <stdio.h>
#define INT_SIZE 32

int getSingle(int arr[], int n)
{
    // Initialize result
    int result = 0;

    int x, sum;

    // Iterate through every bit
    for (int i = 0; i < INT_SIZE; i++)
    {
      // Find sum of set bits at ith position in all
      // array elements
      sum = 0;
      x = (1 << i);
      for (int j=0; j< n; j++ )
      {
          if (arr[j] & x)
            sum++;
      }

      // The bits with sum not multiple of 3, are the
      // bits of element with single occurrence.
      if (sum % 3)
        result |= x;
    }

    return result;
}

// Driver program to test above function
int main()
{
    int arr[] = {12, 1, 12, 3, 12, 1, 1, 2, 3, 2, 2, 3, 7};
    int n = sizeof(arr) / sizeof(arr[0]);
    printf("The element with single occurrence is %d ",
            getSingle(arr, n));
    return 0;
}
```

```
7
```

705 Subscribe

Recent Comments

Abhi You live US or India?

Google (Mountain View) interview · 23 minutes ago

Aman Hi, Why arent we checking for conditions...

Write a C program to Delete a Tree. · 1 hour ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 1 hour ago

Sanjay Agarwal bool tree::Root_to_leaf_path_given_sum(tree...

Root to leaf path sum equal to a given number · 1 hour ago

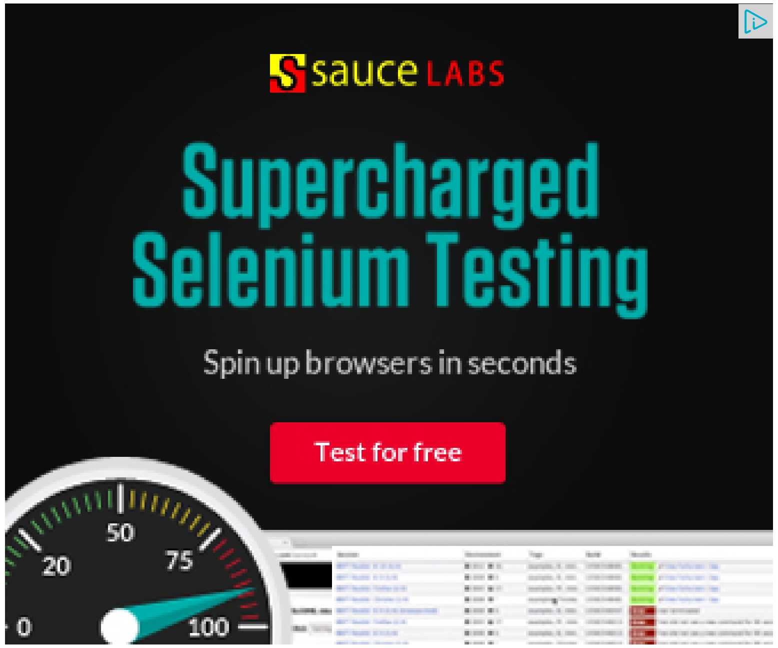
GOPI GOPINATH @admin Highlight this sentence "We can easily...

Count trailing zeroes in factorial of a number · 1 hour ago

newCoder3006 If the array contains negative numbers also. We...

Find subarray with given sum · 1 hour ago

This article is compiled by **Sumit Jain** and reviewed by GeeksforGeeks team. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.



Related Tpoics:

- Check if a number is multiple of 9 using bitwise operators
- How to swap two numbers without using a temporary variable?
- Divide and Conquer | Set 4 (Karatsuba algorithm for fast multiplication)
- Find position of the only set bit
- Swap all odd and even bits
- Add two bit strings
- Write your own strcmp that ignores cases
- Binary representation of a given number

23 Tweet 1 0

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

52 Comments **GeeksforGeeks**

Sort by Newest ▾

Join the discussion…

AN · a month ago
can u explain the logic behind 2nd solution?
^ | ⌄ · Reply · Share ›

Naveen · 3 months ago
Second implementation is incorrect, it will pass only when non-unique elemen
Failure Case:
1. input[] = { 5, 5, 1}

Sum of first bits = { 1 + 1 + 1}%3 = 0
Sum of second bits = { 0 + 0 + 0)%3 = 0
Sum of third bits = {1 + 1 + 0}%3 = 2

ans = 4;
^ | ⌄ · Reply · Share ›

wliao ➔ Naveen · 23 days ago
The problem statement, "Given an array where every element occurs t
^ | ⌄ · Reply · Share ›

lallu · 3 months ago
but the compleity is not O(n) it is O(nlog(MAXelement in the array))
am i right or correct me if i am wrong
^ | ⌄ · Reply · Share ›

Guest · 3 months ago
if in first method ,in given example suppose there is no occurence of 4th integ

2 ..... correct me

Reply • Share ›

rihansh • 3 months ago

int allsum,allXOR;
for i<-0 to n
allsum+=arr[i]';
allxor^=arr[i];

then result =(3*allxor-allsum)/2;
hope everybody got it if i am wrong then correct me and if anyone is having an

Reply • Share ›

rihansh • 3 months ago

hey kindl look at the following solution this one is intersting one hope every boc

Reply • Share ›

rihansh • 3 months ago

this can be done n the following manner in o(n)
take the sum of all numbers and allso XOR of all number the multiply the xor b
will give the answer

3 • Reply • Share ›

Rajesh Pal • 4 months ago

```
/*beauty of map (stl container) ,easily can be solved in o(n) time. :P*/
#include <cstdlib>
#include <stdio.h>
#include <vector>
#include <string>

#include<map>
#include<iterator>
```

```
#include <iostream>

using namespace std;

int main()
{
map<int ,int=""> mymap;
vector<int> myvector;
int n;
cin>>n;
```

see more

• Reply • Share ›

Castle Age Rajesh Pal • 3 months ago

O(1) space though.

• Reply • Share ›

Jing Conan Wang • 4 months ago

There is better way to understand the first method.

Ones and twos are two masks.

(1) If the digit i of ones is 1, the total number of 1s appeared on digit i % 3 == 1

(2) If the digit i of twos is 1, the total number of 1s appeared on digit i %3 == 2.

ones stores the bits which have appeared once. arr[i] store the bits that will ap
ones & arr[i] stores the bits that become to satisfies (2).

As a result, the update rule for twos

twos = twos | (ones & arr[i]);

From this analysis, we can conclude that the two methods are equivalent. The
efficient way to calculate the mod.

Reply • Share ›

groomnestle • 5 months ago

2nd answer is more straightforward and clear.

2 • Reply • Share ›

Srinivas • 6 months ago

difference of all numbers in the array gives the answer

4 • Reply • Share ›

Guest • 8 months ago

#include<stdio.h>

int partition(int *a, int s, int e);

void q_sort(int *arr, int s, int e)
{ int q;
if(s<e) {="" q="partition(arr," s,="" e);="" q_sort(arr,="" s,="" q-1);="" q_sort(arr
partition(int="" *a,="" int="" s,="" int="" e)="" {="" int="" x;="" x="a[e];" int="" j="s
e)="" {="" if(a[j]<x)="" {="" i="i+1;" temp="a[j];" a[j]="a[i];" a[i]="temp;" }="" j="j+
a[e]="temp;" return="" i+1;="" }="" int="" find(int="" *a,="" int="" n)="" {="" int=""
1);="" for(i="0;" i<n;i++)="" printf("\t%d",a[i]);="" printf("\n");="" for(i="1;i&lt;n;i++
else="" if(count="=0)" break;="" else="" count="0;" }="" return="" a[i-1];="" }=""
arr[]="{3," 3,="" 2,="" 3};="" int="" n="sizeof(arr)" sizeof(arr[0]);="" printf("the=""
occurrence="" is="" %d="" \n",find(arr,="" n));="" return="" 0;="" }="">

Reply • Share ›

Guest • 8 months ago

void q_sort(int *arr, int s, int e)
{ int q;

if(s<e) {="" q="partition(arr," s,="" e);="" q_sort(arr,="" s,="" q-1);="" q_sort(arr
partition(int="" *a,="" int="" s,="" int="" e)="" {="" int="" x;="" x="a[e];" int="" j="s
e)="" {="" if(a[j]<x)="" {="" i="i+1;" temp="a[j];" a[j]="a[i];" a[i]="temp;" }="" j="j+
a[e]="temp;" return="" i+1;="" }="" int="" find(int="" *a,="" int="" n)="" {="" int="
1);="" for(i="0;" i<n;i++)="" printf("\t%d",a[i]);="" printf("\n");="" for(i="1;i&lt;n;i++
else="" if(count="=0)" break;="" else="" count="0;" }="" return="" a[i-1];="" }="
arr[]="{3," 3,="" 2,="" 3};="" int="" n="sizeof(arr)" sizeof(arr[0]);="" printf("the="
occurrence="" is="" %d="" \n",find(arr,="" n));="" return="" 0;="" }="">

Reply • Share ›

gargsanjay • 10 months ago

the last method is n log(no. of bits)

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

Reply • Share ›

Sandeep Jain • a year ago

Could u provide a sample input for which it failed?

Reply • Share ›

Let's Make a New India • a year ago

2nd method is not a general method....it fails in many cases.

Reply • Share ›

Ajay Kumar • a year ago

2nd method is gd.

Reply • Share ›

Hanish • a year ago

Can we also write
twos = twos ^ (ones & arr[i]);

instead of |
Will it affect the code in any way ??
Since, it is never possible that a bit is set in both ones and twos

Reply • Share ›

Biren Barodia • a year ago

if I am understanding solution1 correctly, shouldn&#039t the algo look like this:

ones: The bits that have appeared 1st time or 3rd time or 5th time.. etc.

twos: The bits that have appeared 2nd time or 4th time or 6th time.. etc.

Reply • Share ›

Lokesh Gopu • a year ago

int getSingle(int arr[], int n).
{

int f = 0, s = 0, t = 0;.

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

{.

int tempf;.

tempf = (t & arr[i]) ;.

t = t & (t ^arr[i]);.

t = t | (s & arr[i]);.

s = s & (s ^arr[i]);.

s = s | (f & arr[i]);.

see more

1 ^ | ⌄ • Reply • Share ›

Lokesh • a year ago

```
 /* Paste your code here (You may delete these lines if not writing c
int getSingle(int arr[], int n)
{
    int f = 0, s = 0, t = 0;
    for(int i=0;i<n;i++)
    {
            int tempf;

            tempf = ( t & arr[i]) ;
            t = t & (t ^ arr[i]);

            t = t | ( s & arr[i]);
            s = s & (s ^ arr[i]);

            s = s | (f & arr[i]);
            f = f & (f ^ arr[i]);

            f = tempf | f;

            f = f | (arr[i] & (arr[i] ^ (s | t)));
    }

    return f;
}
```

^ | ⌄ • Reply • Share ›

GeeksforGeeks • a year ago
Please take a closer look at the problem statement. The array {15, 15, 4} does
• Reply • Share ›

Koda Rahul • a year ago
second method does not for for all cases.
eg:{15,15,4}
• Reply • Share ›

Koda Rahul • a year ago
second method doesn&#039t work for evry input.
eg: {15,15,4}
• Reply • Share ›

swati • 2 years ago
can't we just sum numbers and take modulo by 3. Is it necessary to consider

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

• Reply • Share ›

Parminder swati • 2 years ago
Would fail if the number we are looking for has 3 as factor

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

• Reply • Share ›

Kartik swati • 2 years ago
This approach has been discussed earlier and doesn't work. Please se
• Reply • Share ›

HLS ↪ Kartik • 2 years ago

The approach doesn't work.

e.g. 1,1,1,3

sum=6
sum%3=0

what next?

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

Reply • Share ›

Angel ↪ HLS • 8 months ago

If you are referring to the second approach, then that's r
You add the corresponding bits from each number, sum
sum the third bits:
1 1 1 3 result
| | | | |
v v v v v
0+0+0+0=0%3=0
0+0+0+1=1%3=1
1+1+1+1=4%3=1

when we put together the results 011 we get the 3 see?

Reply • Share ›

Sandy • 2 years ago

I doubt the values of common_bit_mask during the iterations.
I think it should be -1,-1,-3,-2

Can you explain me a bit?

Sumit ➦ Sandy • 2 years ago

Only the values of last two bits are mentioned in comments. When you
you get negative value because the sign bit is inverted when ~ operato

^ | ˅ • Reply • Share ›

aj • 2 years ago

Infact there is another approach. You can take the sum of bits in same positio
with 3. The number you get is the number that appears only once.
e.g.
101, 101, 101, 1000
sum of first bits%3 = (1+1+1+0)%3 = 0;
sum of second bits%3 = (0+0+0+0)%0=0;
sum of third bits%3 = (1+1+1+0)%3 = 0;
sum of fourth bits%3 = (1)%3 = 1;
hence number which appears once is 1000

2 ^ | ˅ • Reply • Share ›

a2 ➦ aj • 2 years ago

This approach will not work if the numerals have the digits 3 and above

Will it ?

^ | ˅ • Reply • Share ›

Kartik ➦ a2 • 2 years ago

I think it works for any input. If you have counter example, then
time.

^ | ˅ • Reply • Share ›

a2 ➦ Kartik • 2 years ago

Sry , I was mistaken .. had got confused !!

Kartik ➜ aj • 2 years ago

@aj: I have implemented your approach and it seems to be working fin

```
 #include <stdio.h>
#define INT_SIZE 32
int getSingle(int arr[], int n)
{
    int x = 1;
    int result = 0;

    // Let us take the example of {3, 3, 2, 3} to understand tl
    for (int i = 0; i < INT_SIZE; i++)
    {
      int count = 0;
      for (int j=0; j< n; j++ )
      {
          x = (1 << i);
          if (arr[j] & x)
            count++;
```

see more

^ | ⌄ • Reply • Share ›

rohit ➜ Kartik • 2 years ago

gud code but consider array of size =o(n^2)

^ | ⌄ • Reply • Share ›

Kartik ➜ rohit • 2 years ago

@rohit: I could not understand your point, please elabor

^ | ⌄ • Reply • Share ›

Kartik ➦ Kartik • 2 years ago
Thanks for replying rohit.

When we talk about time complexity, we always talk ab
So it is valid to say the time complexity of the algo is O(

The character for less than operator is considered as a
^ | ⌄ • Reply • Share ›

rohit ➦ Kartik • 2 years ago
again same thing, consider array of size less than and
^ | ⌄ • Reply • Share ›

rohit ➦ Kartik • 2 years ago
@kartik-site error,nly half comment came. i was saying
^ | ⌄ • Reply • Share ›

Kartik ➦ aj • 2 years ago
@aj: This solutions looks good to be added to the original post. Could
^ | ⌄ • Reply • Share ›

aj ➦ Kartik • 2 years ago
I will try to write down the code today if possible...But I have a a

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

1 ^ | ⌄ • Reply • Share ›

angel • 2 years ago
sum of array of integers will always be in form of 3(a+b+c)+d
so sum%3 always return the number which is occuring only once,as per my u

Reply • Share ›

Kartik angel • 2 years ago

@angel: Thanks for sharing your thoughts. I think this will give the requ
required number Please correct me if I am wrong.

Reply • Share ›

angel Kartik • 2 years ago

yaa you r right

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

Reply • Share ›

aj angel • 2 years ago

what if d is divisible by 3??

Reply • Share ›

rohit • 2 years ago

1.add bitwise xor of all elements of the array to a variable
2.do step 1 3 times
3.value of the variable is our answer
correct me if i m wrong :)

Reply • Share ›

Load more comments