

Find the two non-repeating elements in an array of repeating elements

Asked by SG

Given an array in which all numbers except two are repeated once. (i.e. we have $2n+2$ numbers and n numbers are occurring twice and remaining two have occurred once). Find those two numbers in the most efficient way.

Method 1(Use Sorting)

First sort all the elements. In the sorted array, by comparing adjacent elements we can easily get the non-repeating elements. Time complexity of this method is $O(n\log n)$

Method 2(Use XOR)

Let x and y be the non-repeating elements we are looking for and $arr[]$ be the input array. First calculate the XOR of all the array elements.

```
xor = arr[0]^arr[1]^arr[2].....arr[n-1]
```

All the bits that are set in xor will be set in one non-repeating element (x or y) and not in other. So if we take any set bit of xor and divide the elements of the array in two sets – one set of elements with same bit set and other set with same bit not set. By doing so, we will get x in one set and y in another set. Now if we do XOR of all the elements in first set, we will get first non-repeating element, and by doing same in other set we will get the second non-repeating element.

Let us see an example.

```
arr[] = {2, 4, 7, 9, 2, 4}
```

1) Get the XOR of all the elements.

```
xor = 2^4^7^9^2^4 = 14 (1110)
```

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](#)



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](#)

2) Get a number which has only one set bit of the xor.

Since we can easily get the rightmost set bit, let us use it.

$\text{set_bit_no} = \text{xor} \ \& \ \sim(\text{xor}-1) = (1110) \ \& \ \sim(1101) = 0010$

Now set_bit_no will have only set as rightmost set bit of xor.

3) Now divide the elements in two sets and do xor of elements in each set, and we get the non-repeating elements 7 and 9. Please see implementation for this step.

Implementation:

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

/* This function sets the values of *x and *y to nonr-epeating
elements in an array arr[] of size n*/
void get2NonRepeatingNos(int arr[], int n, int *x, int *y)
{
    int xor = arr[0]; /* Will hold xor of all elements */
    int set_bit_no; /* Will have only single set bit of xor */
    int i;
    *x = 0;
    *y = 0;

    /* Get the xor of all elements */
    for(i = 1; i < n; i++)
        xor ^= arr[i];

    /* Get the rightmost set bit in set_bit_no */
    set_bit_no = xor & ~(xor-1);

    /* Now divide elements in two sets by comparing rightmost set
bit of xor with bit at same position in each element. */
    for(i = 0; i < n; i++)
    {
        if(arr[i] & set_bit_no)
            *x = *x ^ arr[i]; /*XOR of first set */
        else
            *y = *y ^ arr[i]; /*XOR of second set*/
    }
}

/* Driver program to test above function */
int main()
```

```
{  
    int arr[] = {2, 3, 7, 9, 11, 2, 3, 11};  
    int *x = (int *)malloc(sizeof(int));  
    int *y = (int *)malloc(sizeof(int));  
    get2NonRepeatingNos(arr, 8, x, y);  
    printf("The non-repeating elements are %d and %d", *x, *y);  
    getchar();  
}
```

Time Complexity: $O(n)$

Auxiliary Space: $O(1)$

Find + Remove Duplicates

 dataladder.com/RemoveDuplicates

\$0 License and free support Clean up your duplicate data today!



Related Topics:

- 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

Market research
that's fast and
accurate.

Get \$75 off

 Google consumer surveys

- Binary representation of a given number



2



Tweet

0



1

705



Subscribe

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

23 Comments

GeeksforGeeks

Sort by Newest ▾



Join the discussion...



rohit_90 • 5 months ago

By using method1(using sorting), we can also find more than two non repeating (not repeating) that's not possible with XOR method. Am I correct...?
Time complexity of method1 is $O(n \log n)$, so to reduce it to $O(n)$ we can solve cost extra memory of $O(n)$... but in sorting if we use merge sort then it will also

^ | ▾ • Reply • Share ›



Mihir Sathe • 8 months ago

We can also use a hashtable and get $O(n)$

1 ^ | ▾ • Reply • Share ›



bhavya ➔ Mihir Sathe • 4 months ago

In order to obtain the 1st non repeating we will need a LinkedHashMap

^ | ▾ • Reply • Share ›



wasseyपुरiyan ➔ Mihir Sathe • 7 months ago

But in that case memory used will also be $O(n)$

3 ^ | ▾ • Reply • Share ›

Recent Comments

Abhi You live US or India?

[Google \(Mountain View\) interview](#) · 27 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](#) · 2 hours ago

AdChoices ▶

▶ [C++ Code](#)

▶ [Java Array](#)

▶ [C++ Array](#)



Guest → wasseypuriyan · 3 months ago

Maybe we can use a HashSet and add an element hen we see we see it for the second time. I'm not sure if it will change the w better on average case.

^ | v · Reply · Share ›

AdChoices ▶

▶ [An Array](#)

▶ [Repeating](#)

▶ [Math Array](#)

AdChoices ▶

▶ [Java Elements](#)

▶ [Array Elements](#)

▶ [Elements Time](#)



cammie · a year ago

Why is this $O(n)$ time complexity when both for loops run up to the size n time

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

^ | v · Reply · Share ›



ajay → cammie · a year ago

this is $O(n)+O(n)$,second loop runs after the first ,it is not an inner loop

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

1 ^ | v · Reply · Share ›



Varadh · 2 years ago

Can someone please give me a clear explanation of this???

1 ^ | v · Reply · Share ›



kartik → Varadh · 2 years ago

See [this](#) comment.

^ | v · Reply · Share ›



Algorithmus · 3 years ago

Hi all



nn all,

just check the output of above code with the following inputs...

```
int arr[] = {2, 3, 2, 7, 9, 11, 2, 3, 11};
```

```
int arr[] = {2, 3, 3, 7, 9, 11, 2, 3, 11};
```

This give wrong results...

1 ^ | v • Reply • Share ›



Sandeep → Algorithmus • 3 years ago

Please take a closer look at the problem statement. It says "all number your example1, 2 is repeated twice and in example2, 3 is repeated twice"

^ | v • Reply • Share ›



nn • 4 years ago

can some1 explain me the logical part of how it works...i find it difficult to analyze

^ | v • Reply • Share ›



Sandeep → nn • 4 years ago

XOR of two same numbers results in 0(000..00)

XOR of two different numbers x and y results in a number which contains bits where x and y differ. So if x and y are 10...0100 and 11...1001, then result would be 01...1101

So the idea is to XOR all the elements in set. In the result xor, all repeating elements will be 0. The result would contain the set bits where two non-repeating elements are present.

Now, if we take any set bit of the result xor and again do XOR of the set with the original array, we will get the one non-repeating element. And for other non-repeating elements, the particular bit is not set.

We have chosen the rightmost set bit of the xor as it is easy to find out

5 ^ | v • Reply • Share ›



Vikram N → Sandeep · 6 months ago

Nice explanation!!!

Understood by reading this.

1 ^ | v · Reply · Share ›



ultimate_coder → Sandeep · 11 months ago

Excellent !!!

This explanation should be in the main article..

^ | v · Reply · Share ›



Zongjun → Sandeep · a year ago

```
/* Get the rightmost set bit in set_bit_no */  
set_bit_no = xor & ~(xor-1);
```

Now, if we take any set bit of the result xor and again do XOR c
set, we get the one non-repeating element. And for other non-re
subset where that particular bit is not set.

How come by XOR on one bit and we can get the whole number



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

^ | v · Reply · Share ›



aygul → Zongjun · a year ago

It does not XOR on one bit, but it does an AND on one b
numbers. So some of the same numbers will go to if pa
go to else part BUT one of the non repeating number wi
part. Now in if part and else part seperately you XOR sc
repeating number, which gives the non-repeating numb

^ | v · Reply · Share ›



saurabh → Sandeep · 2 years ago

That's a clear explanation ...Thank's

^ | v · Reply · Share ›



dd · 4 years ago

what about the following prob:

Given an array of integers where some numbers repeat 1 time, some number repeats 3 times, how do you find the number that repeat 3 times

^ | v · Reply · Share ›



Nikhil Agrawal → dd · 11 months ago

@dd Since XORing same number odd number of times yield that number of times yield 0, so answer to ur question is:

Step1: Find XOR of all elements

Step2: Try to find any bit position in the XOR sum calculated above

Step3: Maintain one set in which all element would be having same bit having position bit value at set bit position.

Step4: Take any value as calculated above and browse the array. If correct then the other value would be ur answer otherwise vice-versa.

Answer: One set will contain element repeated 1 time and other set ha

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

^ | v · Reply · Share ›



Pankaj Chopra → dd · 3 years ago

Hi,

we can easily find it by

1. Sorting the elements in an array. Assume the sorted array {3,3,4,4,6
 2. Then comparing each element with 3rd next element, as there is only one
- This will give the solution in $n-3$ comparisons.

^ | v • Reply • Share ›



SG ... • 5 years ago

awesome buddy ... :) ... It works perfectly .

^ | v • Reply • Share ›



KSK → SG ... • 4 years ago

Excellent logic !!

^ | v • Reply • Share ›



Subscribe



Add Disqus to your site

@geeksforgeeks, **Some rights reserved**

Contact Us!

Powered by **WordPress** & **MooTools**, customized by geeksforgeeks team