Find the two repeating elements in a given array

You are given an array of n+2 elements. All elements of the array are in range 1 to n. And all elements occur once except two numbers which occur twice. Find the two repeating numbers.

For example, array = {4, 2, 4, 5, 2, 3, 1} and n = 5

The above array has n + 2 = 7 elements with all elements occurring once except 2 and 4 which occur twice. So the output should be 4 2.

Method 1 (Basic)

Use two loops. In the outer loop, pick elements one by one and count the number of occurrences of the picked element in the inner loop.

This method doesn't use the other useful data provided in questions like range of numbers is between 1 to n and there are only two repeating elements.

```
#include<stdio.h>
#include<stdlib.h>
void printRepeating(int arr[], int size)
{
  int i, j;
  printf(" Repeating elements are ");
  for(i = 0; i < size; i++)
    for(j = i+1; j < size; j++)
      if(arr[i] == arr[j])
        printf(" %d ", arr[i]);
}

int main()
{
  int arr[] = {4, 2, 4, 5, 2, 3, 1};
  int arr_size = sizeof(arr)/sizeof(arr[0]);
  printRepeating(arr, arr_size);
  getchar();
  return 0;
}
```

Run on IDE

Time Complexity: O(n*n)

Auxiliary Space: O(1)

Method 2 (Use Count array)

Traverse the array once. While traversing, keep track of count of all elements in the array using a temp array count[] of size n, when you see an element whose count is already set, print it as duplicate.

This method uses the range given in the question to restrict the size of count[], but doesn't use the data that there are only two repeating elements.

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

void printRepeating(int arr[], int size)
{
  int *count = (int *)calloc(sizeof(int), (size - 2));
  int i;

  printf(" Repeating elements are ");
  for(i = 0; i < size; i++)
  {
    if(count[arr[i]] == 1)
      printf(" %d ", arr[i]);
    else
     count[arr[i]]++;
  }
}

int main()
{
  int arr[] = {4, 2, 4, 5, 2, 3, 1};
  int arr_size = sizeof(arr)/sizeof(arr[0]);
  printRepeating(arr, arr_size);
  getchar();
  return 0;
}
```

Run on IDE

Time Complexity: O(n)

Auxiliary Space: O(n)

Method 3 (Make two equations)

Let the numbers which are being repeated are X and Y. We make two equations for X and Y and the simple task left is to solve the two equations.

We know the sum of integers from 1 to n is n(n+1)/2 and product is n!. We calculate the sum of input array, when this sum is subtracted from n(n+1)/2, we get X + Y because X and Y are the two numbers missing from set [1..n]. Similarly calculate product of input array, when this product is divided from n!, we get X*Y. Given sum and product of X and Y, we can find easily out X and Y.

Let summation of all numbers in array be S and product be P

X + Y = S – n(n+1)/2

XY = P/n!

Using above two equations, we can find out X and Y. For array = 4 2 4 5 2 3 1, we get S = 21 and P as 960.

X + Y = 21 – 15 = 6

XY = 960/5! = 8

X – Y = sqrt((X+Y)^2 – 4*XY) = sqrt(4) = 2

Using below two equations, we easily get X = (6 + 2)/2 and Y = (6-2)/2

X + Y = 6

X – Y = 2

Thanks to geek4u for suggesting this method. As pointed by Beginer , there can be addition and multiplication overflow problem with this approach.

The methods 3 and 4 use all useful information given in the question 🙂

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

/* function to get factorial of n */
int fact(int n);

void printRepeating(int arr[], int size)
{
  int S = 0;  /* S is for sum of elements in arr[] */
  int P = 1;  /* P is for product of elements in arr[] */
  int x,  y;   /* x and y are two repeating elements */
  int D;      /* D is for difference of x and y, i.e., x-y*/
  int n = size - 2,  i;

  /* Calculate Sum and Product of all elements in arr[] */
  for(i = 0; i < size; i++)
  {
    S = S + arr[i];
    P = P*arr[i];
  }

  S = S - n*(n+1)/2;  /* S is x + y now */
  P = P/fact(n);         /* P is x*y now */

  D = sqrt(S*S - 4*P); /* D is x - y now */

  x = (D + S)/2;
  y = (S - D)/2;

  printf("The two Repeating elements are %d & %d", x, y);
}

int fact(int n)
{
   return (n == 0)? 1 : n*fact(n-1);
}

int main()
{
  int arr[] = {4, 2, 4, 5, 2, 3, 1};
  int arr_size = sizeof(arr)/sizeof(arr[0]);
  printRepeating(arr, arr_size);
  getchar();
  return 0;
}
```

Run on IDE

Time Complexity: O(n)

Auxiliary Space: O(1)

Method 4 (Use XOR)

Thanks to neophyte for suggesting this method.

The approach used here is similar to method 2 of this post.

Let the repeating numbers be X and Y, if we xor all the elements in the array and all integers from 1 to n, then the result is X xor Y.

The 1's in binary representation of X xor Y is corresponding to the different bits between X and Y. Suppose that the kth bit of X xor Y is 1, we can xor all the elements in the array and all integers from 1 to n, whose kth bits are 1. The result will be one of X and Y.

```
void printRepeating(int arr[], int size)
{
  int xor = arr[0]; /* Will hold xor of all elements */
  int set_bit_no;  /* Will have only single set bit of xor */
  int i;
  int n = size - 2;
  int x = 0, y = 0;

  /* Get the xor of all elements in arr[] and {1, 2 .. n} */
  for(i = 1; i < size; i++)
    xor ^= arr[i];
  for(i = 1; i <= n; i++)
    xor ^= 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 < size; i++)
  {
    if(arr[i] & set_bit_no)
      x = x ^ arr[i]; /*XOR of first set in arr[] */
    else
      y = y ^ arr[i]; /*XOR of second set in arr[] */
  }
  for(i = 1; i <= n; i++)
  {
    if(i & set_bit_no)
      x = x ^ i; /*XOR of first set in arr[] and {1, 2, ...n }*/
    else
      y = y ^ i; /*XOR of second set in arr[] and {1, 2, ...n } */
  }

  printf("\n The two repeating elements are %d & %d ", x, y);
}


int main()
{
  int arr[] = {4, 2, 4, 5, 2, 3, 1};
  int arr_size = sizeof(arr)/sizeof(arr[0]);
  printRepeating(arr, arr_size);
  getchar();
  return 0;
}
```

Run on IDE

Method 5 (Use array elements as index)

Thanks to Manish K. Aasawat for suggesting this method.

```
Traverse the array. Do following for every index i of A[].
{
check for sign of A[abs(A[i])] ;
if positive then
   make it negative by   A[abs(A[i])]=-A[abs(A[i])];
else  // i.e., A[abs(A[i])] is negative
   this   element (ith element of list) is a repetition
}
```

```
Example: A[] =  {1, 1, 2, 3, 2}
i=0;
Check sign of A[abs(A[0])] which is A[1].  A[1] is positive, so make it negative.
Array now becomes {1, -1, 2, 3, 2}

i=1;
Check sign of A[abs(A[1])] which is A[1].  A[1] is negative, so A[1] is a repetition.

i=2;
Check sign of A[abs(A[2])] which is A[2].  A[2] is  positive, so make it negative. '
Array now becomes {1, -1, -2, 3, 2}

i=3;
Check sign of A[abs(A[3])] which is A[3].  A[3] is  positive, so make it negative.
Array now becomes {1, -1, -2, -3, 2}

i=4;
Check sign of A[abs(A[4])] which is A[2].  A[2] is negative, so A[4] is a repetition.
```

Note that this method modifies the original array and may not be a recommended method if we are not allowed to modify the array.

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

void printRepeating(int arr[], int size)
{
  int i;

  printf("\n The repeating elements are");

  for(i = 0; i < size; i++)
  {
    if(arr[abs(arr[i])] > 0)
      arr[abs(arr[i])] = -arr[abs(arr[i])];
    else
      printf(" %d ", abs(arr[i]));
  }
}
```

```
int main()
{
  int arr[] = {1, 3, 2, 2, 1};
  int arr_size = sizeof(arr)/sizeof(arr[0]);
  printRepeating(arr, arr_size);
  getchar();
  return 0;
}
```

Run on IDE

The problem can be solved in linear time using other method also, please see this and this comments

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

166 Comments Category: Arrays

Related Posts:

- Longest Span with same Sum in two Binary arrays
- Count Inversions of size three in a give array
- Find the subarray with least average
- Count triplets with sum smaller than a given value
- Find zeroes to be flipped so that number of consecutive 1's is maximized
- Reorder an array according to given indexes
- Find maximum value of Sum(i*arr[i]) with only rotations on given array allowed
- Find maximum average subarray of k length

(Login to Rate and Mark)

3.3 Average Difficulty : **3.3/5.0**
Based on **10** vote(s)

Add to TODO List
Mark as DONE

Writing code in comment? Please use code.geeksforgeeks.org, generate link and share the link here.

164 Comments GeeksforGeeks 1 Login

Recommend 6 Share Sort by Newest

Join the discussion…

tanuja pandian • 16 days ago

we can use hash map
int main()
{
int n=5;
int a[]={4,2,4,5,3,2,1};
int arrsize=sizeof(a)/sizeof(a[0]);
int hool[5]={0};
for(int i=0;i<arrsize;i++) hool[a[i]]++;="" for(int="" i="0;i<n;i++)" if((hool[i])="=2)"
cout<<i<<="" ";="">

• Reply • Share ›

Darshil • a month ago

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

This line will not work for xor=4,8,16,32,...

• Reply • Share ›

Sachin Vashistha • 2 months ago

Another method
We can sort the array then traversal will do the job
please find the below code

#include <iostream>
using namespace std;
void quicksort(int *arr, int left, int right);

```
void quicksort(int *arr, int left, int right);
void PrintDuplicates(int arr[],int size)
{
int j=1;
int count=0;
quicksort(arr,0,size-1);
for(int i=0;i<size;i++) {="" if(arr[i]!="j)" {="" count++;="" cout<<arr[i];="" if="" (count="=2)"
break;="" }="" else="" j++;="" }="" }="" void="" swap(int="" i,int="" j,="" int="" *a){="" int="
temp="a[i];" a[i]="a[j];" a[j]="temp;" }="" void="" quicksort(int="" *arr,="" int="" left,="" int="
right){="" int="" min="(left+right)/2;" cout<<"qs:"<<left<<","<<right<<"\n";="" int="" i="left;"
int="" j="right;" int="" pivot="arr[min];" while(left<j="" ||="" i<right)="" {="" while(arr[i]
<pivot)="" i++;="" while(arr[j]="">pivot)
```

see more

• Reply • Share ›

Ajcoo • 5 months ago

The xor method doesn't work if the array contains 0 in it. How to fix that?

• Reply • Share ›

Irfan ➚ Ajcoo • 3 months ago

You're right! You can check for base cases.

• Reply • Share ›

Saurabh ➚ Ajcoo • 5 months ago

Please read the question carefully. Array contains integers from 1 to n.

• Reply • Share ›

sanjeev • 6 months ago

I think method 5 will not work , if we have 0 as a repeated element in the array.. plz correct me, if i m wrong ... :)

• Reply • Share ›

Shaan123 ➚ sanjeev • 6 months ago

its given..nos are in d range 1 to n
if its extended to zero, just maintain a count for 0..if it occurs.

1 • Reply • Share ›

sasha • 6 months ago

Plz explain xor one with example

• Reply • Share ›

Avata This comment was deleted.

sasha Guest • 6 months ago
I didn't get the set bit part.
• Reply • Share ›

Ishupreet Singh • 7 months ago
I am very new to g4g, so i dont know the protocol for submitting a new method here. Still, i wld put it over here. Hope it helps someone.

We can use block sort to sort the array. Then we can traverse the array and keep on subtracting the i+1 element from i element, wherever we get 0 as the result, that is the repetition.
• Reply • Share ›

redblood • 7 months ago
@GeeksforGeeks -> In the 4th XOR method , it's gives the wrong output if i take input array as {4,5, 2, 4, 5, 2, 3, 1} it's produce the output 5 and 0 which is wrong.
• Reply • Share ›

Mr. Lazy redblood • 7 months ago
Yeah.. it'll definitely produce wrong output when you give wrong input. Only two elements are repeated in the input array and are the only ones to be detected. I guess you misunderstood the question.
4 • Reply • Share ›

Redblood Mr. Lazy • 7 months ago
got it !
• Reply • Share ›

Amit • 8 months ago
For 4th XOR , output is not coming correct for { 2 , 5 ,7 ,8 , 1, 6,3,4 ,3, 5} , N = 8 , please tell me if i am doing something wrong or ?
• Reply • Share ›

himanshu bindal Amit • 7 months ago
check results are absolutely correct
• Reply • Share ›

Holden • 8 months ago
In Java, we can use HashMap or even HashSet, easily :)
http://ideone.com/V5VSiT
1 • Reply • Share ›

Holden • 8 months ago

these 2 links which you are referring by 'this' and 'this' are not working:

> The problem can be solved in linear time using other method also,
> please see this and this comments.

please correct them, thank you

• Reply • Share ›

bhai • 9 months ago

what is the work of set_bit i am not getting plz explain anybody..
with example

• Reply • Share ›

debugger ➚ bhai • 9 months ago

- Conisder the array to be a[]= {1,3,2,2,1}

- Now XOR all the elements of a[] and again with numbers 1 to 3.

- So, as per the property of XOR, same elements will cancel out and we will be left with 1 XOR 2 = 3.

- As per above program, we are here : xor = 3 (0011).

- But we don't know the exact numbers,let them be X and Y.

- A bit is set in xor only if corresponding bits in X and Y are different.This is the crucial step to understand.

- Let's consider the rightmost set bit in xor: 0011

- Now again if we XOR all the elements of a[] and 1 to n that have rightmost bit set we will get one of the repeating numbers,say X.

Ex: Elements in a[] with bit set: {1,3,1}
Elements from 1 to n with bit set {1,3}
Result of XOR'ing all these is X = 1.

- Similarly , if we XOR all the elements of a[] and 1 to n that have rightmost bit not set,we will get the other element, Y.
Ex: Elements in a[] with bit not set: {2,2}
Elements from 1 to n with bit not set {2}
Result of XOR'ing all these is Y = 2

3 • Reply • Share ›

blackbox • 10 months ago

Looks like calloc syntax is wrong method 2
int *count = (int *)calloc(sizeof(int), (size - 2));

Shouldn't be like this instead
int *count = (int *)calloc((size-2), sizeof(int));

• Reply • Share ›

Dewendra K Pandey • a year ago

int arr[] = {1,2,3,2,4,3,6}; Try with Sum and Prod method.... Complete failed.. not even correct for 2 No.s

• Reply • Share ›

sachin ➚ Dewendra K Pandey • 9 months ago

i/p taken by you is wrong .....array size =7 then i/p lies b/w 1 to 5 .so 6 is not in i/p array

• Reply • Share ›

Abhinav Gupta • a year ago

In the method 2, count has be freed to avoid memory leak.
Please find the code in the following link:
http://ideone.com/pAXTuR

• Reply • Share ›

Sheldon • a year ago

If the input array is {1,2,1,3,3} the method returns 2 and 3 instead of 1 and 3. Someone please explain where I am going wrong.

• Reply • Share ›

Sheldon ➚ Sheldon • a year ago

The above comment is for the last method. method 5.

• Reply • Share ›

Prabhath • a year ago

In method 3 : instead of multipling all the elements , we can
calculate another equation pow(a1,2) + pow(a2,2) --------- +pow(an,2)
as we know 1+4+9+16------+ pow(n,2) is n*(n+1)*(2n+1)/6

we can get pow(x,2) + pow(y,2) and x+y hence we can calculate x and y using these two formula .

• Reply • Share ›

sandeep sandhu • a year ago

plz tell me the most efficient way to find different no. of elements in a matrix

size of (10000x10000)

Reply • Share ›

Guest • a year ago

i guess method 5 will not work for : 1 2 3 1 3
it will give output 2 and 1
instead of 1 and 3

Reply • Share ›

Hassan Masood ➚ Guest • 7 months ago

Method 5 will not work, if the element value in array is bigger than array size. e.g {1,3,2,15,2,1}. As when it will pick number 15 and will go look at index 15 it will be out of memory, something garbage.

Reply • Share ›

ramboww ➚ Guest • a year ago

no ,it works ,you can run the program.

Reply • Share ›

Brajesh Kumar • a year ago

Reagrding method 2, you can't take count[] of size n(i.e the size of given array), it will cause the segmentation fault if the repeated elements in the array are a big integer values.

suppose your array is: int arr[] = {44444, 2, 44444, 5, 2, 3, 1};

then you will end up assigning a count value to count[44444], but you have created the array of size 5 only !
Solution could be having the count array as big as INT_MAX size which is not good IMO.

Reply • Share ›

Brajesh Kumar ➚ Brajesh Kumar • a year ago

sorry I missed one imp statement:

"All elements of the array are in range 1 to n."

The solution goes well !

1 Reply • Share ›

chandra ➚ Brajesh Kumar • a year ago

better to use hash function

1 Reply • Share ›

psahdev • a year ago

for xor method:

for xor method:
ar={1,1,2,2,3,4,5}
after first step: 1xor2= 011
since lsb is 1:
then xor(1,1,2,2,3,4,5,1,3,5)=1xor4 which is not correct?
please someone explain

• Reply • Share ›

psahdev ➚ psahdev • a year ago
got it in below descriptions. Thanks to Gopinath
1 • Reply • Share ›

typing.. • a year ago
In method of XOR, after having x^y, does we calculate x and y by running a loop from 1 to n and checking following condition--
let x^y be p and loop runs on i that is, for i=1 to n, then
int j=i^p;
if(j^p==i)
then i and j are the two numbers.
• Reply • Share ›

Preethi • a year ago
About method-5:-
if may array contains 18 1 18 15 28 576 576... How this methos work???
• Reply • Share ›

Hassan Masood ➚ Preethi • 7 months ago
I believe the method 5 will not work in you example, as the value in array is bigger then array size.
• Reply • Share ›

vipinkaushal ➚ Preethi • a year ago
all n+2 number are in the range 1 to n and only two are repeating
• Reply • Share ›

Preethi ➚ vipinkaushal • a year ago
Sorry can u explain with example...
• Reply • Share ›

typing.. ➚ Preethi • a year ago
Numbers are in range of 1 to n, 0 is not included in this question, for 0, you have to modify solutions..
• Reply • Share ›

Preethi typing.. • a year ago
Ya ok:-)
• Reply • Share ›

Ashish Thakran • a year ago
Below link can be useful to find out the algorithm to find duplicate or repeated elements in an array in java

Find out duplicate or repeated elements in an array in java

http://newtechnobuzzz.blogspot...
1 • Reply • Share ›

srinivas devaki • a year ago
in method 2: it is efficient to allocate a bool rather than int
• Reply • Share ›

Sakshet • a year ago
I have a O(logn) solution to this problem that uses Binary search tree.
Here is the code.

```
#include <iostream>
#include <string>
using namespace std;

struct node
{
int data;
node *lchild, *rchild;
node()
{
lchild=rchild=NULL;
}
};

void create(node *&T, int x)
{
```

see more

• Reply • Share ›

Code_Fun_do Sakshet • a year ago
Thats not O(logn) solution its O(n) only.

worst case : first and last element are same.

3 • Reply • Share ›

shree • 2 years ago

//checkout this method, works in O(n) time, don't have the misconception
that two loops, so O(n^2), Once a bit is set, it doesn't participate in
the swap

#include<stdio.h>

void twoRepeating(int *a, int n)
{
int i,t;
for(i=0;i<n;i++) {="" while(a[a[i]]!="a[i])" {="" t="a[a[i]];" a[a[i]]="a[i];" a[i]="t;" }="" }="" printf("the="" two="" repeating="" numbers="" are="" %d="" &="" %d\n",a[0],a[n-1]);="" }="" void="" main()="" {="" int="" a[]="{4,2,5,1,2,3,3},n=sizeof(a)/sizeof(int);" tworepeating(a,n);="" }="">

• Reply • Share ›

shree • 2 years ago

/*checkout this method, works in O(n) time, don't have the misconception that two loops, so O(n^2), Once a bit is set, it doesn't participate in the swap*/
#include<stdio.h>

void twoRepeating(int *a, int n)
{
int i,t;
for(i=0;i<n;i++) {="" while(a[a[i]]!="a[i])" {="" t="a[a[i]];" a[a[i]]="a[i];" a[i]="t;" }="" }="" printf("the="" two="" repeating="" numbers="" are="" %d="" &="" %d\n",a[0],a[n-1]);="" }="" void="" main()="" {="" int="" a[]="{4,2,5,1,2,3,3},n=sizeof(a)/sizeof(int);" tworepeating(a,n);="" }="">

1 • Reply • Share ›