GeeksforGeeks
A computer science portal for geeks

Find the maximum repeating number in O(n) time and O(1) extra space

Given an array of size *n*, the array contains numbers in range from 0 to *k-1* where *k* is a positive integer and *k* <= *n.* Find the maximum repeating number in this array. For example, let *k* be 10 the given array be *arr[]* = {1, 2, 2, 2, 0, 2, 0, 2, 3, 8, 0, 9, 2, 3}, the maximum repeating number would be 2. Expected time complexity is *O(n)* and extra space allowed is *O(1)*. Modifications to array are allowed.

The **naive approach** is to run two loops, the outer loop picks an element one by one, the inner loop counts number of occurrences of the picked element. Finally return the element with maximum count. Time complexity of this approach is *O(n^2).*

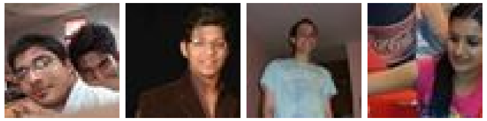
A **better approach** is to create a count array of size k and initialize all elements of *count[]* as 0. Iterate through all elements of input array, and for every element *arr[i]*, increment *count[arr[i]]*. Finally, iterate through *count[]* and return the index with maximum value. This approach takes O(n) time, but requires O(k) space.

Following is the ***O(n)* time and *O(1)* extra space** approach. Let us understand the approach with a simple example where *arr[]* = {2, 3, 3, 5, 3, 4, 1, 7}, *k* = 8, *n* = 8 (number of elements in arr[]).

1) Iterate though input array *arr[]*, for every element *arr[i]*, increment *arr[arr[i]%k]* by *k* (*arr[]* becomes {2, 11, 11, 29, 11, 12, 1, 15 })

2) Find the maximum value in the modified array (maximum value is 29). Index of the maximum value is the maximum repeating element (index of 29 is 3).

3) If we want to get the original array back, we can iterate through the array one more time and



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

do *arr[i] = arr[i] % k* where *i* varies from 0 to *n-1*.

How does the above algorithm work? Since we use *arr[i]%k* as index and add value *k* at the index *arr[i]%k*, the index which is equal to maximum repeating element will have the maximum value in the end. Note that *k* is added maximum number of times at the index equal to maximum repeating element and all array elements are smaller than *k*.

Following is C++ implementation of the above algorithm.

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

// Returns maximum repeating element in arr[0..n-1].
// The array elements are in range from 0 to k-1
int maxRepeating(int* arr, int n, int k)
{
    // Iterate though input array, for every element
    // arr[i], increment arr[arr[i]%k] by k
    for (int i = 0; i< n; i++)
        arr[arr[i]%k] += k;

    // Find index of the maximum repeating element
    int max = arr[0], result = 0;
    for (int i = 1; i < n; i++)
    {
        if (arr[i] > max)
        {
            max = arr[i];
            result = i;
        }
    }

    /* Uncomment this code to get the original array back
       for (int i = 0; i< n; i++)
          arr[i] = arr[i]%k; */

    // Return index of the maximum element
    return result;
}

// Driver program to test above function
int main()
{
    int arr[] = {2, 3, 3, 5, 3, 4, 1, 7};
    int n = sizeof(arr)/sizeof(arr[0]);
```

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

```
    int k = 8;

    cout << "The maximum repeating number is " <<
         maxRepeating(arr, n, k) << endl;

    return 0;
}
```

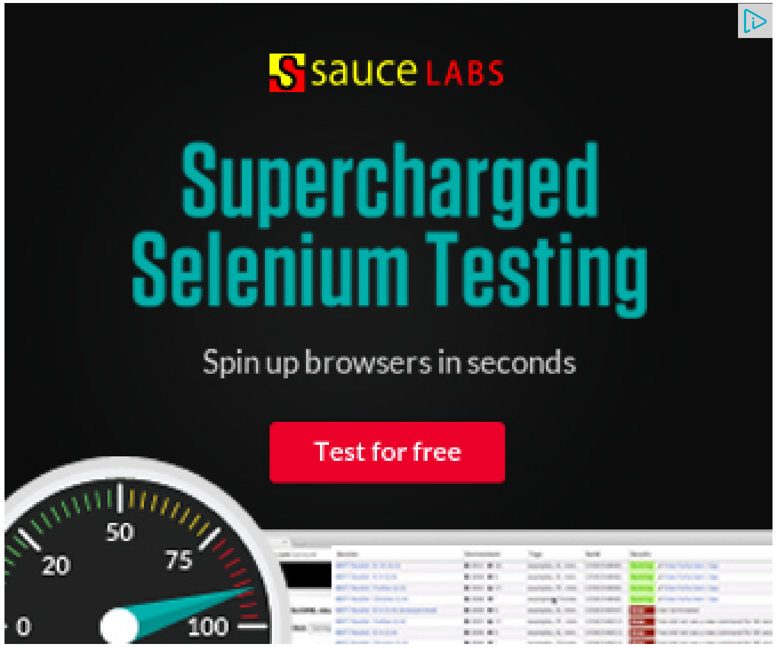
Output:

```
The maximum repeating number is 3
```

Exercise:
The above solution prints only one repeating element and doesn't work if we want to print all maximum repeating elements. For example, if the input array is {2, 3, 2, 3}, the above solution will print only 3. What if we need to print both of 2 and 3 as both of them occur maximum number of times. Write a O(n) time and O(1) extra space function that prints all maximum repeating elements. (Hint: We can use maximum quotient arr[i]/n instead of maximum value in step 2).

Note that the above solutions may cause overflow if adding k repeatedly makes the value more than INT_MAX.

This article is compiled by Ashish Anand 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:

- 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

20 Tweet 5 2

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

58 Comments GeeksforGeeks

Sort by Newest ▾

705 Subscribe

Recent Comments

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

Write a C program to Delete a Tree. · 2 minutes ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 5 minutes ago

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

Root to leaf path sum equal to a given number · 30 minutes ago

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

Count trailing zeroes in factorial of a number · 32 minutes ago

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

Find subarray with given sum · 57 minutes ago

Join the discussion…

Suganya · 2 months ago

If size of array is less than max element in the array.

For example,a = [6,6,3]
k=7
In first iteration,
a[6%7] += 7; ----> a[6] +=7
But we have only upto a[2].

How to handle this case?

^ | ˅ · Reply · Share ›

Ashish · 2 months ago

What can be done if the array contains negative numbers ?

^ | ˅ · Reply · Share ›

void · 3 months ago

considering arr[] = {0,0,5,1,3,3,3,7,2,2,2}, though arr[2] and arr[3] both added k
initial value is larger than arr[3], then 2 is returned incorrectly. We need to test

Here is the python code:

```
k = 8
a = [0,0,5,1,3,3,3,7,2,2,2]

for i in range(len(a)):
a[a[i]%k] += k

print a.index(max(a))

j = -1
```

newCoder3006 Code without using while loop. We can do it...

Find subarray with given sum · 1 hour ago

```
for i in range(len(a)):
j = max(j, a[i]/k)
a[i] = a[i]%k

print j
```

Reply • Share ›

vinod • 5 months ago

The problem statement doesn't state that k is given. The following code is for

http://ideone.com/sljbDI

Reply • Share ›

icefrog • 5 months ago

nonsense. count[] doesn't account for memory space?

Reply • Share ›

zzer icefrog • 18 days ago

actually it is input, don't need any other space at all

Reply • Share ›

Ajinkya • 6 months ago

doesnt work !!Please check before posting stuff .

int arr[] = {2, 3, 3, 3, 3, 4, 1, 7};

1 • Reply • Share ›

tczf1128 Ajinkya • 6 months ago

it can work

Reply • Share ›

Sriharsha g.r.v • 7 months ago

say if 11 is part of the above sequence and repaeted 4 times..then arr[3] gets than k>=max value of(array)..pls correct me if i am wrong!!

1 ^ | ⌄ • Reply • Share ›

prakash • 7 months ago

if n<k this="" algorithm="" will="" crash="" since="" arr[arr[i]%k]="" may="" be= scenario="" is="" n="5;k=9" arr="{1,2,2,5,8}" and="" arr[8%9]="arr[8]" which=""

^ | ⌄ • Reply • Share ›

DRAGONWARRIOR • 8 months ago

CAN WE USE MOORE VOTING ALGO

1 ^ | ⌄ • Reply • Share ›

Krishna ➦ DRAGONWARRIOR • 4 months ago

NO, It works only if the element repeats itself more than n/2 times..

^ | ⌄ • Reply • Share ›

Prakash Verma • 9 months ago

what happens in case of k>>n.

1 ^ | ⌄ • Reply • Share ›

KeSha Shah • 9 months ago

k <=n is teh constraint .. read solution carefully

^ | ⌄ • Reply • Share ›

Sandeep Srivastava • 9 months ago

Sandeep Jain :

Can you explain why have you used arr[i]%k. If k=maximum(arr)+1,arr[i] is sar addition. If I am wrong please correct me..

1 ^ | ⌄ • Reply • Share ›

aman • 11 months ago

The above program will fail if we put 1000 instead of 7.The output will 7 instead

^ | ⌄ • Reply • Share ›

Ronny ➦ aman • 11 months ago

@aman

Please read the problem statement carefully.

"Given an array of size n, the array contains numbers in range from 0 t
k <= n."

1 ^ | ⌄ • Reply • Share ›

illuminati • 11 months ago

This is why i love geeks...
perfect explanation!!...

^ | ⌄ • Reply • Share ›

hbandi • 11 months ago

[sourcecode language="java"]

Can you see this,may be simple way

```
package com.pg.code;

public class MaximumRepatingNumber {

public static void main(String[] args) {
int arr[] = {2, 2, 2, 5, 7, 7,7, 7};

MaximumRepatingNumber mRN=new MaximumRepatingNumber();
System.out.println("max repeated number is :: "+mRN.maxRepeateNumber(a

}
private int maxRepeateNumber(int[] array) {

int localArray[] = new int[array.length];
```

```
for (int i = 0; i < localArray.length; i++) {
localArray[i] = 0;
```

see more

^ | ⌄ • Reply • Share ›

mukesh2009mit • a year ago

```
 /* FIND OUT MAXIMUM TIME REPEATING NUMBER IN THE RANGE OF 0 TO N-1 *

#include<stdio.h>
#include<conio.h>
void main()
{
clrscr();
int a[20],n,i,max;
printf("\n Enter the size of Array :");
scanf("%d",&n);
printf("\n Enter the array elements in the range 0 to n-1 :");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
for(i=0;i<n;i++)
        a[a[i]%n]+=n;

max=a[0]/n;
for(i=1;i<n;i++)
```

see more

3 ^ | ⌄ • Reply • Share ›

mukesh2009mit • a year ago

```
/* FIND OUT MAXIMUM TIME REPEATING NUMBER IN THE RANGE OF 0 TC

#include<stdio.h>
#include<conio.h>
void main()
{
clrscr();
int a[20],n,i,max;
printf("\n Enter the size of Array :");
scanf("%d",&n)
printf("\n Enter the array elements in the range 0 to n-1 :");
for(i=0;i<n;i++) scanf("%d",&amp;a[i])="" for(i="0;i&lt;n;i++)" a[a[i]%n]+="n;" m
n="">max)
max=a[i]/n;

printf("\n Maximum repeating Elements : ");
for(i=0;i<n;i++) if(a[i]="" n="=max)" printf("="" %d="" ",i);="" *="" reconstruct=""
a[i]%="n;" getch();="" }="">
```

Reply · Share ›

NIRAJ · a year ago

```
int maxRepeatNum(int arr[],int n,int k)
{
int count[14]={0};
for(int i=0;i<n;i++)
{
++count[arr[i]];
}

int max=count[0],num=0;
for(int i=1;imax)
{
```

max=count[i];
num=i;
}
}
return num;
}

⌃ | ⌄ • Reply • Share ›

Anshul Goel • a year ago

What if k and n are different(k<n), because in such cases,the elements at pos

[sourcecode language="C"]
/* Paste your code here (You may delete these lines if not writing code) */

⌃ | ⌄ • Reply • Share ›

Gaurav • a year ago

I think the above algorithm (given as an exercise) would fail if we use the follov

```
    int arr[] = {2,2,2,3,3,3,4,4,4};

  Modified array becomes:
  arr[] = {2,2,17,18,18,3,4,4,4}
```

n = 9, k = 5.
When you use arr[i]/n approach, it gives max as 2 (for both elements 3 and 4)

1 ⌃ | ⌄ • Reply • Share ›

Gaurav ➦ Gaurav • a year ago

*Continuing my previous comment

arr[2]/n = 1;
arr[3]/n = 2;

arr[4]/n = 2;

Hence, only 3 and 4 are maximum repeating. 2 is ignored.
Please correct me if I am wrong.

⌃ | ⌄ • Reply • Share ›

AMIT ➦ Gaurav • 11 months ago

its not arr[i]/n...its arr[i]/k..

⌃ | ⌄ • Reply • Share ›

pradheep • a year ago

we can use arr[arr[i]] instead if arr[arr[i]%k] since here k is always less than or calculation of mod operator

[sourcecode language="C"]
/* Paste your code here (You may delete these lines if not writing code) */

1 ⌃ | ⌄ • Reply • Share ›

Prashanth ➦ pradheep • 8 months ago

Ya you are right

⌃ | ⌄ • Reply • Share ›

Ankit Chaudhary ➦ Prashanth • 7 months ago

we r adding k to a[i], therefore a[i] can be greater than k.
See given example.

⌃ | ⌄ • Reply • Share ›

Anon • a year ago

Why is %k required?? How would simply incrementing arr[arr[i]] by k hurt?
Anyways since a[i]<k, a[i]%k=a[i].

⌃ | ⌄ • Reply • Share ›

Setu · a year ago

The case when there are more than one number which is repeating same nur

⌃ | ⌄ · Reply · Share ›

GeeksforGeeks ➜ Setu · a year ago

@Setu: This is given in the exercise. Please see the hint to print all ma

⌃ | ⌄ · Reply · Share ›

Chandra Prakash · a year ago

Solution of Exercise question:

```
#include<iostream.h>
//using namespace std;.

// Returns maximum repeating element in arr[0..n-1].
// The array elements are in range from 0 to k-1.
void maxRepeating(int* arr, int n, int k).
{

// Iterate though input array, for every element.
// arr[i], increment arr[arr[i]%k] by k.

int i;.
for (i = 0; i< n; i++).
arr[arr[i]%k] += k;.
for (i = 0; i< n; i++).
cout << arr[i]<< " ";.

cout<<endl;.
```

see more

⌃ | ⌄ · Reply · Share ›

Rishit · a year ago

Why are we doing mod k since it is guaranteed that all the numbers are in the

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

^ | ˅ • Reply • Share ›

kartik ➜ Rishit • a year ago

We may add k multiple times at an index, that is why mod is required.

1 ^ | ˅ • Reply • Share ›

Palash • a year ago

This solution works without any dependency on k, i.e. given an array of size n
can find the elements present maximum number of time in O(N) time and O(1
Here's the algorithm -
For i=0 to n-1

Start from i=0, go to a[i]th element, store it in a temp variable, make it -1 (if it a
decrement it. Also make a[i]=n.

Go in a while loop with temp element and loop till you return to original index (i)
or if the element is equal to n.

Find the minimum of the array. The index of minimum is the answer.

Example:
arr[] = {2, 3, 3, 5, 3, 4, 1, 7}, n=8
modified arr[] = {8,-1,-1,-3,-1,-1,8,-1}
Ans = 3.

^ | ˅ • Reply • Share ›

chandu ➜ Palash • 10 months ago

Can you explain your logic ?
I m not able to get it.

Can you do a dry run of any example ?

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

^ | v • Reply • Share ›

Akshat Gupta • a year ago

In your question description it is said that k<=n but in your solution description whuch is not satisfying the above condition

[sourcecode language="C"]
/* Paste your code here (You may delete these lines if not writing code) */

^ | v • Reply • Share ›

GeeksforGeeks ➜ Akshat Gupta • a year ago

Please refresh the page. We have updated the solution description. Le incorrect.

^ | v • Reply • Share ›

Sandeep Jain • a year ago

Saransh: The given solution prints only one number. This is given as an exerc

^ | v • Reply • Share ›

GeeksforGeeks • a year ago

@all: The sample input used for algorithm was not a valid input. We have corr the inconvenience.

@atul: The solution may not work if there is overflow. Thanks for bringing up tl the original post.

^ | v • Reply • Share ›

geekcomp ➜ GeeksforGeeks • 9 months ago

@GeekforGeeks:

Can you explain why have you used arr[i]%k. If k=maximum(arr)+1,arr|
each addition. If I am wrong please correct me..

Reply • Share ›

Satendra • a year ago

i can see the hint in the exercise working, but not getting the logic

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

Reply • Share ›

Saransh Sharma • a year ago

but suppose if arr[]={3,4,1,2,5,6,4,3}, n=8, k=10.
then the final array will be arr[]={3,14,11,22,25,16,4,3}.
this shows that the 4 is the maximum repeating number..
but here 3 and 4 both occur twice?

Reply • Share ›

aayush • a year ago

For modified version of this problem
for all maximum repeating elements
arr[i]/k will be same
correct me if i m wrong..?????

This method as same as method4 for finding repeating element in array

Reply • Share ›

GeeksforGeeks ➦ aayush • a year ago

@ayush: Yes, the value of arr[i]/k will be same for all maximum repeati

Reply • Share ›

sam ➦ GeeksforGeeks • a year ago

even max value for more than one maximum repeated element more than one maximum repeated elements with arr[i]/n logic

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

• Reply • Share ›

WihE • a year ago

The first sentence states that it has to be k<=n.
In your example k<=n is not given.

I converted your example into Java and tried some more tests:

```
[sourcecode language="Java"]public int maxRepeating(int[] arr, int n, int k) {
// Iterate though input array, for every element
// arr[i], increment arr[arr[i]%k] by k
for (int i = 0; i < n; i++)
arr[arr[i] % k] += k;

// Find index of the maximum repeating element
int max = arr[0], result = 0;
for (int i = 1; i < n; i++) {
if (arr[i] > max) {
max = arr[i];
result = i;
}
}
```

see more

• Reply • Share ›

Palash • a year ago

Is K=N, then, array elements may fall outside the array index boundary.

I think, it only works for K=N.

∧ | ∨ • Reply • Share ›

Palash ➦ Palash • a year ago

Sorry, pay no attention to the comment above, something went wrong

∧ | ∨ • Reply • Share ›

Load more comments

✉ Subscribe   Add Disqus to your site