Move all zeroes to end of array

Given an array of random numbers, Push all the zero's of a given array to the end of the array. For example, if the given arrays is {1, 9, 8, 4, 0, 0, 2, 7, 0, 6, 0}, it should be changed to {1, 9, 8, 4, 2, 7, 6, 0, 0, 0, 0}. The order of all other elements should be same. Expected time complexity is O(n) and extra space is O(1).

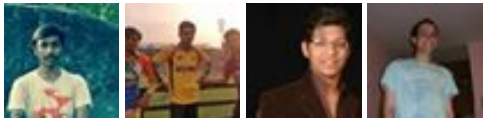
There can be many ways to solve this problem. Following is a simple and interesting way to solve this problem.
Traverse the given array 'arr' from left to right. While traversing, maintain count of non-zero elements in array. Let the count be 'count'. For every non-zero element arr[i], put the element at 'arr[count]' and increment 'count'. After complete traversal, all non-zero elements have already been shifted to front end and 'count' is set as index of first 0. Now all we need to do is that run a loop which makes all elements zero from 'count' till end of the array.

Below is C++ implementation of the above approach.

```
// A C++ program to move all zeroes at the end of array
#include <iostream>
using namespace std;

// Function which pushes all zeros to end of an array.
void pushZerosToEnd(int arr[], int n)
{
    int count = 0;  // Count of non-zero elements

    // Traverse the array. If element encountered is non-zero, then
    // replace the element at index 'count' with this element
    for (int i = 0; i < n; i++)
        if (arr[i] != 0)
            arr[count++] = arr[i]; // here count is incremented
```

```
    // Now all non-zero elements have been shifted to front and 'count
    // set as index of first 0. Make all elements 0 from count to end.
    while (count < n)
        arr[count++] = 0;
}

// Driver program to test above function
int main()
{
    int arr[] = {1, 9, 8, 4, 0, 0, 2, 7, 0, 6, 0, 9};
    int n = sizeof(arr) / sizeof(arr[0]);
    pushZerosToEnd(arr, n);
    cout << "Array after pushing all zeros to end of array :\n";
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";
    return 0;
}
```

Output:

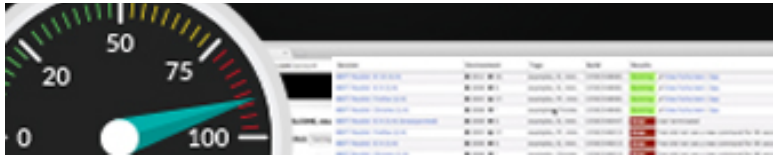
```
Array after pushing all zeros to end of array :
1 9 8 4 2 7 6 9 0 0 0 0
```

Time Complexity: O(n) where n is number of elements in input array.

Auxiliary Space: O(1)

This article is contributed by **Chandra Prakash**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Popular Posts



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.

74 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

Lohith Ravi • 18 days ago

public static void moveZerostoEnd(int array[]) {

int NumberPtr = 0;

int ZeroPtr = 0;

```
while (array[NumberPtr] == 0 && NumberPtr < array.length - 1 )

NumberPtr++;

while (array[ZeroPtr] != 0 && ZeroPtr < array.length - 1)

ZeroPtr++;

if(ZeroPtr < NumberPtr)

{

swap(array, ZeroPtr, NumberPtr);
```

see more

• Reply • Share ›

abc • a month ago

```
#include<iostream>

using namespace std;

#define swap(a,b,c) {c=a;a=b;b=c;}

int main()

{

int a[]={1,0,5,0,6,3,0,4,7,0};

int len=sizeof(a)/sizeof(a[0]);

int low=0,high=len-1;
```


Recent Comments

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

Write a C program to Delete a Tree. · 4 minutes ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 8 minutes ago

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

Root to leaf path sum equal to a given number · 32 minutes ago

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

Count trailing zeroes in factorial of a number · 34 minutes ago

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

Find subarray with given sum · 59 minutes ago

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

Find subarray with given sum · 1 hour ago

int i=0,temp=0;

while(low<high) {="" if(a[high]="=0)" {="" high--;="" }="" if(a[low]="=0&&an
"<<high<<endl;="" swap(a[low],a[high],temp);="" low++;="" high--;="" }="" low+
cout<<a[i]<<"="" ";="" system("pause");="" }="">

• Reply • Share ›

sumit gaur • 2 months ago

Your code's complexity will be O(n+cn) as upto some extend you are traversir
here's with O(n) traverse only once

void rearrange(int a[],int size)

{

int end=0;

for(int i=0;i<size;i++) {="" if(a[i])="" {="" int="" tmp="a[end];" a[end]="a[i];" a[i]="

• Reply • Share ›

Guest • 2 months ago

Your code's complexity will be O(n+cn) as upto some extend you are traversir

here's with O(n) traverse only once

void rearrange(int a[],int size)

{

int end=0;

for(int i=0;i<size;i++) {="" if(a[i])="" {="" int="" tmp="a[end];" a[end]="a[i];" a[i]="

• Reply • Share ›

Guest • 2 months ago

Your code's complexity will be O(n+cn) as upto some extend you are traversir here's with O(n) traverse only once

void rearrange(int a[],int size)

{

int end=0;

for(int i=0;i<size;i++) {="" if(a[i])="" {="" int="" tmp="a[end];" a[end]="a[i];" a[i]="

 • Reply • Share ›

Ramesh • 3 months ago

private void move(int[] arr) {

int countZeroes = 0;

for(int i=0; i < arr.length; i++) {

if(arr[i] == 0) {

countZeroes++;

} else {

if(countZeroes != 0) {

arr[i-countZeroes] = arr[i];

arr[i] = 0;

}

}

}

}

3 ^ | v • Reply • Share ›

swap • 3 months ago

public class MoveZerosToEnd {

public static void main(String... args) {

MoveZerosToEnd m = new MoveZerosToEnd();

System.out.println(Arrays.toString(m.moveZero(new int[]{0, 1, 2, 3, 0, 9, 4})));

}

public int[] moveZero(int[] arr) {

int k = 1;

int i = 0;

while (i < arr.length) {

if (arr[i] == 0) {

for (int j = i + 1; j < arr.length; j++) {

see more

^ | v • Reply • Share ›

Nodirbek • 3 months ago

How about my algorithm:

For each element of array from right to left check if it's zero then swap it with l
of last non-zero element, and shift to left as we swap. Since the problem state

(not actually shifting non-zero numbers to left) this is correct solution. Below c

http://ideone.com/12oEWe

2 ^ | v • Reply • Share ›

rr • 4 months ago

import java.util.Arrays;

class test {

public static void main(String[] args) {

int[] x = {0,1,2,3};

int count = 0;

if(x == null){

return;

}

for(int i = 0; i<x.length; i++){="" if(x[i]="" !="0){" x[count]="x[i];" count++;="" }="" j<x.length;j++){="" x[j]="0;" }="" system.out.println(arrays.tostring(x));="" }="" }=

^ | v • Reply • Share ›

ravi jadhav • 4 months ago

Here is a simple fact to exploit. Multiplying zero with any element produces a z element, just keep looking whether the multiplication of successive pairs is ze whether both the element are zeroes. You can use subtraction to check if both out which is zero by comparing.

Here is the java code i wrote. I believe when the number of zeroes are lesser c lesser number of comparison. On average n/2.

http://ideone.com/llYYsr

4 ^ | v • Reply • Share ›

Rahul Ramesh ravi jadhav • 2 months ago

But multiplying causes overhead right????
I am not very sure....

^ | v • Reply • Share ›

rihansh • 4 months ago

this can also be done by selecting 0 as pivot element as in case of quick sort l
elements

^ | v • Reply • Share ›

rihansh • 4 months ago

BUT THIS WILL TAKE TWO PASSES WHEN ALL ELEMENT IN THE ARRAY
AN BETTER OPTION

START iterating from right side kept one pointer at end and other one is iterain
element we replace it with the pointer position and decrementing hope u get m

1 ^ | v • Reply • Share ›

rj rihansh • 4 months ago

@rihansh but in this case, order of non-zero elements will not retain

1 ^ | v • Reply • Share ›

bikash • 5 months ago

I think we can approach this problem by keeping two pointers indicating the cu
position, and swapping them. This continues till all zeros are moved to the enc
http://ideone.com/UMHryG

3 ^ | v • Reply • Share ›

That way the order won't be maintained, which is a requirement here.

Jonathan bikash • a month ago

It could run O(n^2) time. (depends on input)

Suppose we have:

00000011111111111111111111111111111111111111111111

Starting from the back, 'end' will increment one each while loop iteratio
majority of the array each iteration to find the next zero to swap with.

Mona • 5 months ago

This method uses the partition step of the Quicksort algo ...
Proved sol in O(n).

```
#include <iostream>

using namespace std;

void pushZerosToEnd(int arr[],int n)

{

int i=-1;

for(int j=0;j<n;j++) {="" if(arr[j]!="0)" {="" i++;="" arr[i]="arr[j];" }="" else="" contii
arr[j]="0;" }="" int="" main()="" {="" int="" arr[]="{1," 9,="" 8,="" 4,="" 0,="" 0,="
n="sizeof(arr)" sizeof(arr[0]);="" pushzerostoend(arr,="" n);="" cout="" <<="" "
zeros="" to="" end="" of="" array="" :\n";="" for="" (int="" i="0;" i="" <="" n;="" i-
";="" return="" 0;="" }="">
```

trojansmith • 5 months ago

Hi,,this partitions the array into non zero and zero

```
int[] arr = {1,9,8,4,0,0,2,7,0,6,0};
        int temp=0;
        int i=0,j=0;

        //reach first zero
        while(arr[i]!=0)i++;
        j=i;
        //partition array into 0 and non-zero
        while(i<arr.length &&="" j<arr.length){="" if(arr[i]!="0){" su
```

Reply • Share ›

trojansmith • 5 months ago

trojansmith...hi this i have given will partition the array into non zero and zeroes

```
int[] arr = {1,9,8,4,0,0,2,7,0,6,0};
        int temp=0;
        int i=0,j=0;

        //reach first zero
        while(arr[i]!=0)i++;
        j=i;
        //partition array into 0 and non-zero
        while(i<arr.length &&="" j<arr.length){="" if(arr[i]!="0){" su
```

Reply • Share ›

xyz • 6 months ago

```
int i=0,j=0;

while(j!=length)

{

if(array[j]!=0)

{

int temp=array[j];

array[j]=0;

array[i]=temp;

i++;

j++;

}

else

{

j++;

}

}
```


Guest • 6 months ago

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

int main(){

int n;

scanf("%d",&n);

int i,arr[n];

for(i=0;i<n;i++) scanf("%d",&arr[i]);="" int="" j="0;" i="0;" while(i<n="" &&="" j<r
i++;="" if(i<n){="" j="i+1;" while(!arr[j]="" &&="" j<n)="" j++;="" if(j=">=n)

break;

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

---

see more

• Reply • Share ›

Sheetal Garg • 6 months ago

#include<stdio.h>

int main()

{

int flag=0,temp,i,j,arr[10]={1,0,9,8,0,0,6,2,0,1};

i=0;j=i+1;

while(j<10)

`

if(arr[i]!=0&&flag==0)

{

i++;

i++;

see more

1 ^ | v • Reply • Share ›

Pradeep Patel • 7 months ago

Performance can be improved if self replacement can be avoided. In case the

```
for (int i = 0; i < n; i++)
      if (arr[i] != 0)
      {
        if(count != i) //avoid self replacement
          arr[count] = arr[i]; // here count is incremented
        ++count;
      }
```

^ | v • Reply • Share ›

krrish ➦ Pradeep Patel • 6 months ago

You can not bet it to be improvement as perhaps conditional evaluatior assignment(depends on processor).Also you are performing n conditic

3 ^ | v • Reply • Share ›

Harish Kumar • 7 months ago

"Correct me if i m wrong"

writes

Take two pointers i=0 and j=0

do these steps until j is less than length

--> assign 'i' the first zeroth index from left

-->assign 'j' the first non Zeroth index from left after 'i' (greater than i)

--> Swap the values at those indices

1 ^ | v • Reply • Share ›

Pushkar ➦ Harish Kumar • 5 months ago

Worst Case your algo might take O(n2)

^ | v • Reply • Share ›

Guest • 7 months ago

"Correct me if i m wrong"

Well we can do it by two pointers and we traverse the whole array only once a writes

Take two pointers i=0 and j=0

do these steps until j<n --=""> assign 'i' the first zeroth index from left

-->assign 'j' the first non Zeroth index from left after 'i' (greater than i)

--> Swap the values at those indices

The code in c is as follows

void shiftZeros(int array[],int n)

```
int i=0,j=0;
i = getNextZeroIndex(i,n,array);
if( i == -1)
return;
j = i+1;
while(j<n) {="" if(array[j])="" {="" array[i]="array[j];" array[j]="0;" i++;="" i="getNe
return;="" j="i;" }="" j++;="" }="" printarray(array,n);="" }="" int="" getnextzeroinc
while(i<n="" &&="" array[i]="" !="0)" {="" if(="" i="=" n)="" return="" -1;="" i++;=
```


Pravin Guest • 5 months ago

Please forgive my ignorance.. wats the programming language that is

Himanshu Mishra • 7 months ago

My in python. Surprisingly same Algorithm :)

```python
import time
def main():
  start = time.time()
  a = [1, 9, 8, 4, 0, 0, 2, 7, 0, 6, 0]
  b = []
  count = 0
  for i in a:
    if i != 0:b.append(i)
    else:count += 1
  c = [0] * count
  answer = b + c
  print(answer)
  print("time taken : ",time.time() -start)
  return 0
```

```
main()
```

• Reply • Share ›

Dinesh Panchananam Himanshu Mishra • 5 months ago

Pyth 2.7 Shorter code.

```
f= [1,9,8,4,0,0,2,7,0,6,0,23,32]
g = filter(lambda x:x!=0 ,f)
print g+(len(f)-len(g))*[0]
```

1 • Reply • Share ›

sathish • 7 months ago

I tried a Simple one.If anything wrong ,please advice.The code is working 100%

no extra space used.

```
void movee(int *a,int n)

{

int i,c=n-1;

for(i=0;i<=c;i++)

{

if(*(a+i)==0 && *(a+c)==0)

{

c--;

while(c>i)
```

see more

1 ^ | v • Reply • Share ›

shashi ➦ sathish • 7 months ago

dude, maintain order

1 ^ | v • Reply • Share ›

Ruhi Sharma • 7 months ago

We can also do this by taking two variables i and j, initialize i as index 0 and j a
Now, keep incrementing i till we find a 0 and then keep decrementing j till we fi
Continue this until i >= j

C code for the above approach is :

```
front_index=0;
end_index=no_of_elem-1;
while(front_index < end_index)
{
if(arr[front_index]==0)
{
if(arr[end_index]!=0){
arr[front_index]= arr[end_index];
arr[end_index]=0;
front_index++;
end_index--;
}
else{
end_index--;
}
}
else{
front_index++;
```

}
}

Reply • Share ›

abhisek ➚ Ruhi Sharma • 7 months ago

we cannot use dutch flag here as the order has to be maintained

Reply • Share ›

guest ➚ Ruhi Sharma • 7 months ago

The order of non-zero elements are not preserved by this method.

Reply • Share ›

atiq • 7 months ago

@GeeksforGeeks ....Good exploitation but
No need to do this part.....
// Now all non-zero elements have been shifted to front and 'count' is
// set as index of first 0. Make all elements 0 from count to end.
while (count < n)
arr[count++] = 0

just do replace a[i] with 0 at the same moment
for (int i = 0; i < n; i++)
if (arr[i] != 0)
{ arr[count++] = arr[i];

arr[i]=0;
}

That's how in one pass.... ;)

Reply • Share ›

ssd ➚ atiq • 7 months ago

this will not work. try it on the array with one element [1]

1 ^ | v • Reply • Share ›

rk ssd • 7 months ago

We can set arr[i]=0 only when (i != count-1) . Is there any other

^ | v • Reply • Share ›

pavansrinivas • 7 months ago

Code in JAVA (similar to partition in Quicksort)...order is not presrved...

```
void arrange(){
        int[]a = {1, 9, 8, 4, 0, 0, 2, 7, 0, 6, 0};
        int l = 0;
        int r = a.length-1;
        while(l<r){ while(a[l]!="0&amp;amp;&amp;amp;l&amp;lt;r){" l++;=""
                        r--;
                }
                a[l] = a[r];
                a[r] = 0;
                l++;
                r--;
        }
        for(int i=0;i<a.length;i++){ system.out.print(a[i]+="
```

^ | v • Reply • Share ›

Rashik • 7 months ago

#define N 15

void main()

{

int ar[N]={0,0,20,30,0,0,0,40,0,0,50,0,60,0};

```
int j=0,i=0;

while(1)

{

while(ar[i]!=0)

i++;

j=i ;

while(ar[j]==0)
```

see more

Reply • Share ›

Satyendra Kumar Singh • 7 months ago

```
void ShiftZeroToEnd(int arr[],int n)
{
int cur=-1;
for(int i=0;i<n;i++) {="" if(arr[i]="=0)" {="" if(cur="=-1)" cur="i;" }="" else="" if(cu
cur++;="" }="" }="" }="" the="" above="" program="" will="" push="" all="" zeros
```

1 • Reply • Share ›

sijayaraman • 7 months ago

```
void move_zero_to_end(int arr[],int len)
{
int j = -1;
for(int i=0;i<len;i++) {="" if(arr[i]!="0" )="" {="" j++;="" swap(&arr[i],&arr[j]);="" }=
*b)="" {="" int="" t="*a;" *a="*b;" *b="t;" }="">
```

Reply • Share ›

guest • 7 months ago

```
#include<iostream>
using namespace std;
int main()
{
int a[]={1, 9, 8, 4, 0, 0, 2, 7, 0, 6, 0, 9};
int n=sizeof(a)/sizeof(a[0]);
int m=n,k;
for(int i=0;i<n;i++) {="" if(a[i]="=0)" for(int="" j="i+1;j&amp;lt;n;j++)" if(a[j]!="0)" {="" k
}="" for(int="" i="0;i&amp;lt;n;i++)" cout<<a[i]<<"="" ";="" }="">
```

1 ^ | v • Reply • Share ›

Guest • 7 months ago

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

int main() {
while (1) {
int array[30];
int arraySize;
cin >> arraySize;
for (int i = 0; i < arraySize; i++) {
cin >> array[i];
}
int writePtr = 0;
int temp;
for (int readPtr = 0; readPtr < arraySize; readPtr++) {
temp = array[readPtr];
if (temp != 0) {
array[writePtr] = temp;
writePtr++;
```

Reply • Share ›

REYAZ • 7 months ago

another simpe method is to maintain two indices i and j, traverse the array till \
point to this element and then we increment j till the end of the array while doin
if element pointed to by j is 0 then j++,
else if element pointed to by j is other than 0 , then swap the elements of posit
and j.
COMPLEXITY: O(n).
this is a modified version of DNF algo, in which while segregating three types,
one type of objects with their order preserved.. plz correct me if i am wrong

6 Reply • Share ›

Vijay Kumar ➦ REYAZ • 7 months ago

```
int i = 0;
for( i=0; i<n; i++){="" if(a[i]="=0){break;}" }="" for(int="" ;
```

Reply • Share ›

Hector ➦ REYAZ • 7 months ago

I think this is correct!!!!

1 Reply • Share ›

its_dark ➦ REYAZ • 7 months ago

OR

you can also start i as index 0 and j as index 'n-1'.
Now, keep incrementing i till we find a 0 and then keep decrementing j
them.
continue till i>j.

1 Reply • Share ›

Jekin ➚ its_dark • 7 months ago

Yes. But then it won't preserve original order.

▲ | ▼ • Reply • Share ›

Load more comments

Subscribe Add Disqus to your site