Segregate Even and Odd numbers

Given an array A[], write a function that segregates even and odd numbers. The functions should put all even numbers first, and then odd numbers.

Example

```
Input  = {12, 34, 45, 9, 8, 90, 3}
Output = {12, 34, 8, 90, 45, 9, 3}
```

In the output, order of numbers can be changed, i.e., in the above example 34 can come before 12 and 3 can come before 9.

The problem is very similar to our old post Segregate 0s and 1s in an array, and both of these problems are variation of famous Dutch national flag problem.

```
Algorithm: segregateEvenOdd()
1) Initialize two index variables left and right:
            left = 0,  right = size -1
2) Keep incrementing left index until we see an odd number.
3) Keep decrementing right index until we see an even number.
4) If lef < right then swap arr[left] and arr[right]
```

Implementation:

C/C++

```
// C program to segregate even and odd elements of array
#include<stdio.h>

/* Function to swap *a and *b */
void swap(int *a, int *b);

void segregateEvenOdd(int arr[], int size)
{
    /* Initialize left and right indexes */
    int left = 0, right = size-1;
    while (left < right)
```

```
    {
        /* Increment left index while we see 0 at left */
        while (arr[left]%2 == 0 && left < right)
            left++;

        /* Decrement right index while we see 1 at right */
        while (arr[right]%2 == 1 && left < right)
            right--;

        if (left < right)
        {
            /* Swap arr[left] and arr[right]*/
            swap(&arr[left], &arr[right]);
            left++;
            right--;
        }
    }
}

/* UTILITY FUNCTIONS */
void swap(int *a, int *b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}

/* driver program to test */
int main()
{
    int arr[] = {12, 34, 45, 9, 8, 90, 3};
    int arr_size = sizeof(arr)/sizeof(arr[0]);
    int i = 0;

    segregateEvenOdd(arr, arr_size);

    printf("Array after segregation ");
    for (i = 0; i < arr_size; i++)
        printf("%d ", arr[i]);

    return 0;
}
```

Run on IDE

Java

```
// Java program to segregate even and odd elements of array
import java.io.*;

class SegregateOddEven
{
    static void segregateEvenOdd(int arr[])
    {
        /* Initialize left and right indexes */
        int left = 0, right = arr.length - 1;
        while (left < right)
        {
            /* Increment left index while we see 0 at left */
            while (arr[left]%2 == 0 && left < right)
                left++;
```

```
            /* Decrement right index while we see 1 at right */
            while (arr[right]%2 == 1 && left < right)
                right--;

            if (left < right)
            {
                /* Swap arr[left] and arr[right]*/
                int temp = arr[left];
                arr[left] = arr[right];
                arr[right] = temp;
                left++;
                right--;
            }
        }
    }

    /* Driver program to test above functions */
    public static void main (String[] args)
    {
        int arr[] = {12, 34, 45, 9, 8, 90, 3};

        segregateEvenOdd(arr);

        System.out.print("Array after segregation ");
        for (int i = 0; i < arr.length; i++)
            System.out.print(arr[i]+" ");
    }
}
/*This code is contributed by Devesh Agrawal*/
```

Run on IDE

Python

```
# Python program to segregate even and odd elements of array

def segregateEvenOdd(arr):

    # Initialize left and right indexes
    left,right = 0,len(arr)-1

    while left < right:

        # Increment left index while we see 0 at left
        while (arr[left]%2==0 and left < right):
            left += 1

        # Decrement right index while we see 1 at right
        while (arr[right]%2 == 1 and left < right):
            right -= 1

        if (left < right):
              # Swap arr[left] and arr[right]*/
              arr[left],arr[right] = arr[right],arr[left]
              left += 1
              right = right-1


# Driver function to test above function
arr = [12, 34, 45, 9, 8, 90, 3]
segregateEvenOdd(arr)
print ("Array after segregation "),
```

```
for i in range(0,len(arr)):
    print arr[i],
# This code is contributed by Devesh Agrawal
```

Run on IDE

Output:

```
Array after segregation 12 34 90 8 9 45 3
```

Time Complexity: O(n)

References:
http://www.csse.monash.edu.au/~lloyd/tildeAlgDS/Sort/Flag/

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

69 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)

1.7 Average Difficulty : **1.7/5.0**
Based on **11** 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.

69 Comments GeeksforGeeks 1 Login

Recommend 2 Share Sort by Newest

Join the discussion…

SlickHackz • 2 months ago

The post says it is a variant of Dutch National Flag. But why there are loops inside? Cant we maintain the authenticity by following below ?
http://code.geeksforgeeks.org/...

• Reply • Share ›

Ashutosh Dwivedi • 4 months ago

http://ideone.com/1KLJPQ

To segregate all elements in the range [1,n] in a single pass of the array, using modified Dutch National Flag algo.
Please tell me if you find any mistakes/ test cases that dont run.

• Reply • Share ›

Lokesh • 5 months ago

http://code.geeksforgeeks.org/...

• Reply • Share ›

coderr • 5 months ago

Can anybody explain why is the complexity O(n)? For every check in main while loop, there runs an inner while loop. So it should be O(n*n).

• Reply • Share ›

TulsiRam coderr • 5 months ago

Any element is compared once only bhai

• Reply • Share ›

Abhishek Jatram coderr • 5 months ago

what is the complexity of the code given below:
int i=1,j;

while(i<5){
for(j=0;j<5;j++){
i++;
}
}

• Reply • Share ›

coderr Abhishek Jatram • 5 months ago

It will be O(n). And this clears my doubt. Thank you so much. :)

• Reply • Share ›

Anubhav Arora • 5 months ago

Java Implementation

https://ideone.com/w0PMNc

• Reply • Share ›

Abhishek Gupta • 6 months ago

The output of above example is 12 34 90 8 9 45 3
not 12, 34, 8, 90, 45, 9, 3

2 • Reply • Share ›

coolk • 6 months ago

Java version

http://ideone.com/wj81As

• Reply • Share ›

ss • 6 months ago

we can also use the partition algorithm

• Reply • Share ›

Ashish ss • 6 months ago

a variant of hoare's partition algorithm is used above ..

• Reply • Share ›

Aaryan Singh • 7 months ago

#include<stdio.h>

```
#include<stdio.h>
#include<conio.h>
int main()
{
int arr[]={1,7,44,51,54,14,23,25,48,41};
int i,j,temp;
int l=0; int e=9;
for(i=l;i<=e;i++)
{
if(arr[i]%2==0)
l++;
else
{

if(arr[e]%2==0)
{
temp=arr[i];
arr[i]=arr[e];
```

see more

• Reply • Share ›

sid • a year ago

```
void segregateEvenOdd(int num[])
//
int nStart <- 0
int nEnd <- length of num array - 1
while nStart < nEnd
do
- if num[nStart] is even
- then
- - nStart++
- else
- - swap num[nStart] and num[nEnd] by reference
- - nEnd--
```

• Reply • Share ›

Gopal Modi • a year ago

A simple solution by dutch algo , in O(n)

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

{

int low=0,high=n-1,mid=0,ch;
```

```
while(low<=high)

{

ch=a[low]%2;

switch(ch)

{

case 0:

swap((a+low++),(a+mid++));
```

see more

• Reply • Share ›

Devin • a year ago

while loops inside a while loop. isn't tat O(n^2) complexity?

• Reply • Share ›

surbhijain93 ➚ Devin • a year ago

No, because we are traversing the array only once.

• Reply • Share ›

Kim Jong-il • a year ago

@GeeksforGeeks Hope you include this one. O(n)
Algorithm to maintain the order.

1: Get the count of all the even number in the list, let it be evcount.
2: Initialize i = 0, count=evcount;
3: WHILE i < count
-------- IF A[i] is even
---------- Then i++ and continue;
--------- ELSE
---------- swap A[i] with A[evcount++];

Note: In the ELSE we are not incrementing the i, we will increment that only when we get an even number at ith position.

1 • Reply • Share ›

Achin Saxena ➚ Kim Jong-il • a year ago

in the ELSE part, do we have to swap A[i] with A[evcount] and then increment evcount or we have to first increment evcount and then swap A[i] with A[evcount] ?

Kim Jong-il ➚ Achin Saxena • a year ago

@Achin Saxena NO first we have to swap and then increment.

• Reply • Share ›

reddy1925 ➚ Kim Jong-il • a year ago

we have to use another array i think?
we cant swap in the same array

• Reply • Share ›

Kim Jong-il ➚ reddy1925 • a year ago

i do not think their is any problem. Logic is very simple and straight forward.

If you have not noticed this "Note: In the ELSE we are not incrementing the i, we will increment that only when we get an even number at ith position.". then please read it and try to understand.

• Reply • Share ›

voldemort ➚ Kim Jong-il • a year ago

The code doesn't work for { 2, 4, 5, 9, 3, 8, 6 }. It outputs 2 4 8 6 5 3 9 instead of 2 4 8 6 5 9 3. The odd numbers does not maintain order.

1 • Reply • Share ›

Prakhar Gairola • a year ago

One simple approach can be, by trying to place all even numbers on the left side, odd numbers will by default move to other side.

• Reply • Share ›

<HoldOnLife!#> • 2 years ago

http://ideone.com/m7TMIN How is my approach ?

• Reply • Share ›

Kim Jong-il ➚ <HoldOnLife!#> • a year ago

Instead of code if you post your logic then that will be easier to verify.

• Reply • Share ›

Bommisetty Avinash Avinash • 2 years ago

i cant understand the logic when i go through left=0 it gives {12,90,8,9,45,34,3}.if i assign left=1 i am ok with the logic and gives {12 34 90 8 9 45 3} but when i compile it with left=0 or left=1 it gives same o/p: {12 34 90 8 9 45 3}..can any one help me??

• Reply • Share ›

Achin Saxena ➚ Bommisetty Avinash Avinash • a year ago

both the no. at left =0 and left=1 are even no. So swapping will not take place until left becomes left=2 which is an odd no. and hence the positions of the no. at left=0 and left=1 remain same.

• Reply • Share ›

KeshaShah • 2 years ago

The ans is 12 34 90 8 9 45 3 unlike the one given at top ! I guess the order should not be important

• Reply • Share ›

Saurabh • 2 years ago

Similar to segregate 0s and 1s problem.

• Reply • Share ›

NAVEEN PRAJAPATI • 2 years ago

optimized solution in O(n) using bitwise & operator property :
since the order doesn't matter as given in the question only we have to separate the even and odd :

here is the link :

http://ideone.com/jPKCrE

• Reply • Share ›

Loky ➚ NAVEEN PRAJAPATI • a year ago

if order doesnot matter we can simply do it in O(n/2)

The approach wud be
for(i=0, j=n-1; i<n 2;)="" if((arr[i]%2!="0)&&(arr[j]%2==0))" swap="" arr[i]="" with="" arr[j];="" i++;="" j--;="" if(arr[i]%2="=0)" i++;="" if(arr[j]%2!="0)" j--;="">

1 • Reply • Share ›

danny • 2 years ago

A simpler solution with using just single while other then 3 while used in the program...

```
void evenodd(int a[],int n)
{
int low=0,high=n-1,temp;
while(low<=high)
{
if(a[low]%2==0)
```

```
if(a[low]%2==0)
low++;
else if(a[high]%2!=0)
high--;
else
{
a[low]=a[low]+a[high];
a[high]=a[low]-a[high];
a[low]=a[low]-a[high];
}
}
}
```

• Reply • Share ›

sonu431 ➚ danny • 2 years ago

can u explain this last three lines of code

```
else
{
a[low]=a[low]+a[high];
a[high]=a[low]-a[high];
a[low]=a[low]-a[high];
}
```

• Reply • Share ›

GOPI GOPINATH ➚ sonu431 • 2 years ago

Its swaping (without third variable)

• Reply • Share ›

vikash • 2 years ago

To maintain the order v can use this algo...

Get the evencount and so u ll get the index where to insert an odd no.. now traverse through the array and insert the elements in a new array in their respective index.

```
for k -> 0 to n-1
if(a[k]%2==0)
evencount++;
i=0;
j=evencount;
for k -> 0 to n-1
if(a[k]%2==0)
b[i++]=a[k];
else
b[j++]=a[k];
```

b[n] contains the o/p array that also maintains the order. This algo has space complexity o(n)

please tel me if m wrong sumwer...

3 • Reply • Share ›

Loky ➚ vikash • a year ago

what if we are not allowed to use a new array?

1 • Reply • Share ›

Vikash Kumaran ➚ Loky • a year ago

i think there is no o(n) soln using o(1) extra space, that maintains the order too.

1 • Reply • Share ›

deep sutaria • 2 years ago

Does not work with n = 8 (or even number of elements)..Any inputs??

• Reply • Share ›

Venu Gopal • 2 years ago

O(n) approach only but using 1 while only in place of 3 while loops
and I thinks this is simpler too.
http://ideone.com/CIsVKq

• Reply • Share ›

destroyer • 2 years ago

```
#include<stdio.h>

main()
{
int a[]={12,34,45,9,8,90,3};
int n=sizeof(a)/sizeof(a[0]);
cons(a,n);
getch();
}

cons(int a[],int n)
{
int i,j,temp;
for(i=0;i<n;i++) {="" for(j="i+1;j&lt;n;j++)" {="" if(a[i]%2="=1" &="" a[j]%2!="1)" {="" temp="a[i];" a[i]="a[j];" a[j]="temp;" }="" }="" for(i="0;i&lt;n;i++)" printf("%d="" ",a[i]);="" }="">
```

• Reply • Share ›

shelly • 2 years ago

```
void segregateEvenAndOdd( int A[], int N ) {
    int countOfEven = -1, i = 0;
    while ( i < N ) {
        if ( !( A[i] & 1 ) )
            swap ( A[i], A[++countOfEven] );
        i++;
    }
}
```

PS: ordering is also maintained.

• Reply • Share ›

Vinodhini shelly • 2 years ago

doesn't work for this case : 1 3 4 2 5 12 19 10 .
Can you check?

• Reply • Share ›

vivek • 2 years ago

here the order of numbers are not maintained ,
input-{12,34,45,9,8,90,3}
output-{12,34,90,8,9,45,3}
expected -{12,34,8,90,45,9,3}
how can we maintain the order?

1 • Reply • Share ›

me.abhinav • 3 years ago

Yet another solution with O(n) time and O(1) space is as follows:

1) Initialize a variable 'toSwap' = index of first odd number.
2) Traverse the array from there onwards upto end. If current element (i.e. arr[i]) is odd then proceed to next element or else if current element (i.e. arr[i]) is even then SWAP(arr[i], arr[toSwap]) and increment 'toSwap' by 1.

```
#include <iostream>
//#define SIZ 100
//#define MAX(a, b) (a>b)?a:b

using namespace std;

void swap(int *a, int *b){
```

```
        int temp = *a;
        *a = *b;
        *b = temp;
```

see more

1 • Reply • Share ›

akshat gupta • 3 years ago

Analogous to the partition subroutine of quicksort..

except for, instead of comparing '>' or '<' with pivot,
we partion for even and odd..

• Reply • Share ›

Sandeep • 3 years ago

```
void segOddEven(int arr[], int n){
int start = 0;
int end = n-1;
while(start < end){
if(arr[start]%2 == 1 && arr[end]%2 == 0){
int temp = arr[start];
arr[start] = arr[end];
arr[end] = temp;
start++;
end--;
}
else if(arr[start]%2 == 1){
end--;
}
else{
start++;
}
}
}
```

• Reply • Share ›

AKSHAT • 3 years ago

how to maintain the relative ordering(stable ordering) for even's and odd's in context to the input ordering

1 • Reply • Share ›

Balasubramanian.N • 4 years ago

This approach is similar to the Partition algorithm given in CLRS for QuickSort.
This avoids the extra checks that are needed in the normal approach

This avoids the extra checks that are needed in the normal approach.

```
void segregate(int* a,int len)
{
        int i=-1;
        for(int j=0;j<len;++j)
        {
                if(a[j]%2==0)
                {
                        ++i;
                        int temp=a[i];
                        a[i]=a[j];
                        a[j]=temp;
                }
        }
}

Please comment, if you find anything wrong.

Thanks,
Balasubramanian.N
```

• Reply • Share ›

crazy • 4 years ago

how to do above problem if we want to maintain the order of the sequence....
i/p {12, 34, 45, 9, 8, 90, 3}
o/p {12, 34, 8, 90, 45, 9, 3}

• Reply • Share ›

Load more comments

Subscribe Add Disqus to your site Add Disqus Add Privacy