Segregate 0s and 1s in an array

Asked by kapil.

You are given an array of 0s and 1s in random order. Segregate 0s on left side and 1s on right side of the array. Traverse array only once.

```
Input array   =  [0, 1, 0, 1, 0, 0, 1, 1, 1, 0]
Output array =  [0, 0, 0, 0, 0, 1, 1, 1, 1, 1]
```

Method 1 (Count 0s or 1s)

Thanks to Naveen for suggesting this method.

1) Count the number of 0s. Let count be C.

2) Once we have count, we can put C 0s at the beginning and 1s at the remaining n – C positions in array.

Time Complexity: O(n)

The method 1 traverses the array two times. Method 2 does the same in a single pass.

Method 2 (Use two indexes to traverse)

Maintain two indexes. Initialize first index *left* as 0 and second index *right* as n-1.

Do following while *left* < *right*

a) Keep incrementing index *left* while there are 0s at it

b) Keep decrementing index *right* while there are 1s at it

c) If left < right then exchange arr[left] and arr[right] Implementation:

```
// C program to sort a binary array in one pass
#include<stdio.h>

/*Function to put all 0s on left and all 1s on right*/
void segregate0and1(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] == 0 && left < right)
            left++;

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

        /* If left is smaller than right then there is a 1 at left
          and a 0 at right.  Exchange arr[left] and arr[right]*/
        if (left < right)
        {
            arr[left] = 0;
            arr[right] = 1;
            left++;
            right--;
        }
    }
}

/* driver program to test */
int main()
{
    int arr[] = {0, 1, 0, 1, 1, 1};
    int i, arr_size = sizeof(arr)/sizeof(arr[0]);

    segregate0and1(arr, arr_size);

    printf("array after segregation ");
    for (i = 0; i < 6; i++)
        printf("%d ", arr[i]);

    getchar();
    return 0;
}
```

Run on IDE

Time Complexity: O(n)

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

115 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 **8** 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.

115 Comments GeeksforGeeks 1 Login

Recommend 2 Share Sort by Newest

Join the discussion

Amrendra • 3 days ago

```
// Using Recursion (If only needed to print)

#include<bits stdc++.h="">

using namespace std;

void fun(int arr[],int index, const int arrLen)

{

if(index == arrLen)

return;

if(arr[index] == 0)

cout<<"0 ";

fun(arr,index+1, arrLen);

if(arr[index] == 1)
```

see more

• Reply • Share ›

Amrendra • 3 days ago

```
// Using Recursion

#include<bits stdc++.h="">

using namespace std;

void fun(int arr[],int l, int r)

{

if(l>=r)

return;

if(arr[l] == 0)

l++;

if(arr[r] == 1)

r--;
```

see more

• Reply • Share ›

Sai Yashodhar Vinay • 17 days ago

here in pyhton code:

a=[0, 1, 0, 1, 1, 1]
n=len(a)
i=0
j=n-1
def swap(a,i,j):
if i!=j:
t=a[i]
a[i]=a[j]
a[j]=t
while i<j: while="" a[i]="=0:" i+="1" while="" a[j]="=1:" j-="1" swap(a,i,j)="" i+="1" j-="1" print="" a="">

• Reply • Share ›

Sushma • 2 months ago

It can be solved using one index.

```
public void doSeg(int[] arr) {

int index = 0;

for(int i=0 ; i<=arr.length-1; i++) {

if(arr[i] == 0) {

swap(arr,i,index);

index = index + 1;

}

}

public void swap(int arr[], int i, int index) {

int temp = arr[i];

arr[i] = arr[index];

arr[index] = temp;

}
```

• Reply • Share ›

RAJ GOPAL • 2 months ago

```
public class segregate
{
public static void main(String[] args)
{
int arr[]={0, 1, 0, 1, 0, 0, 1, 1, 1, 0};
int right = arr.length-1,left=0;
while(left!=right)
{
if(arr[left]==0)
{
++left;
continue;
}
else if(arr[left]==1 && arr[right]==0)
{
arr[right]=1;
arr[left]=0;
++left;
--right;
}
else if(arr[left]==1 && arr[right]==1)
{
--right;
}
}
for(left=0;left<arr.length;left++){ system.out.print(arr[left]+"="" ");="" }="" }="" }="">
```

• Reply • Share ›

Shridhar • 2 months ago

I have a more optimal solution...

```
void segregate0and1(int a[], int size)

{

int temp,atleastonezerofoundonright=0;

for(int i=0;i<size;i++){ if(a[i]="=0)" continue;="" if(a[i]="=1){" for(int="" j="i;j&amp;lt;size;j++)"
if(a[j]="=0)" {="" temp="a[j];" a[j]="a[i];" a[i]="temp;" atleastonezerofoundonright="1;" }=""
if(atleastonezerofoundonright="=0)" {="" break;="" }="" }="" }="" }="">
```

• Reply • Share ›

Jaydatta Nagarkar ➚ Shridhar • 5 days ago

a solution with two nested for's

▲ | ▼ • Reply • Share ›

Challenger36 • 2 months ago

simple and easy code in one loop.

int main()

{

int arr[] = {0, 1, 0, 1, 0, 0, 1, 1, 1, 0};

int i, n;

n = sizeof(arr)/sizeof(arr[0]);

int k=n-1,m=0;

int res[10];

for(i=0;i<n;i++) {="" if(arr[i]="=1)" {="" res[k]="1;" k--;="" }="" else="" {="" res[m]="0;" m++;="" }="" }="" printf("array="" after="" segregation="" ");="" for="" (i="0;" i="" <="" n;="" i++)="" printf("%d="" ",="" res[i]);="" getchar();="" return="" 0;="" }="">

▲ | ▼ • Reply • Share ›

EigenHarsha • 2 months ago

In method 2 :
please check it - > right- be right--;

▲ | ▼ • Reply • Share ›

Prashant Kumar • 3 months ago

Do we need to add the check left < right in the 2 while loops. Using arr[left] == 0 and arr[right] == 1 I am able to get the Segregated array.

▲ | ▼ • Reply • Share ›

Anurag Singh • 3 months ago

What if we have to preserve order of elements (like stable sort)??

▲ | ▼ • Reply • Share ›

TulsiRam • 5 months ago

The second method is exactly like: Arrange odds after evens

▲ | ▼ • Reply • Share ›

Holden ➚ TulsiRam • 4 months ago

But that question is more difficult ...

• Reply • Share ›

coderr • 5 months ago

Simple approach in C++ O(n)

```
void sort01(int arr[],int size)

{

int low=0,high=size-1;

while(low<=high)

{

switch(arr[low])

{

case 0:

low++;

break;
```

see more

• Reply • Share ›

contradiction • 6 months ago

done using partition function... tell if some bug is found
http://ideone.com/bvTxge

1 • Reply • Share ›

Ramendu Shandilya • 6 months ago

Java implementation of the first approach

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

1 • Reply • Share ›

Aman Babu Gautam • 6 months ago

another approach for this problem..
http://ideone.com/9A6Cxq

1 • Reply • Share ›

radioactive_platypus • 7 months ago

Any shortcomings of the following solution or any test cases I have not accounted for?

Any shortcomings of the following solution or any test cases I have not accounted for?

It traverses the array once.

https://ideone.com/NelA89

Much appreciated.

1 • Reply • Share ›

blank space • 7 months ago

https://ideone.com/1p2zSz

• Reply • Share ›

Priyank Khattar • 7 months ago

Java code for the second method.

http://ideone.com/LPA3vy

1 • Reply • Share ›

taru sen • 7 months ago

```
#include<stdio.h>

#include<conio.h>

int main(){

int a[10]={1,1,0,0,0,1,0};

int i;

int j;

int count=0;

int count1=0;

for(i=0;i<7;i++){

if(a[i]==0)

printf("%d",a[i]);
```

see more

• Reply • Share ›

manu agrawal • 7 months ago

@GeeksforGeeks

this is another solution.

http://ideone.com/p5j8oI

1 • Reply • Share ›

Anant Raj manu agrawal • 7 months ago

you are changing again again at same index by sifting 1s to the left one- one position. so accumulating 1s from right is better.

• Reply • Share ›

manu agrawal Anant Raj • 7 months ago

I think both will take same time..becuase if i'm shfting only when element is 0,and also last index of 1 is not same as current 0 index.
thanx.

• Reply • Share ›

GeeksforGeeks Mod • 8 months ago

Ashok Jangir & Hengameh,

Thanks for pointing out the typo in program. We have corrected it now.

1 • Reply • Share ›

Holden • 9 months ago

Java Version: http://ideone.com/Q6pjaS

• Reply • Share ›

Holden • 9 months ago

There is 2 typos:
The "if" should be:
if(arr[left] < arr[right])

also:
right--;

• Reply • Share ›

GeeksforGeeks Mod Holden • 8 months ago

Thanks for pointing out the typo in program. We have corrected the first one. The second one doesn't seem to be a typo. Please let us know if you think otherwise.

1 • Reply • Share ›

Holden GeeksforGeeks • 8 months ago

Thank you for your reply.

There are two "right-" in the code, which should be "right--".

while (arr[right] == 1 && left < right)

```
right–;

/* If left is smaller than right then there is a 1 at left

and a 0 at right. Exchange arr[left] and arr[right]*/

if (left < right)

{

arr[left] = 0;

arr[right] = 1;

left++;

right–;

}
```

• Reply • Share ›

GeeksforGeeks Mod ➚ Holden • 8 months ago

Thanks for the reply. There seems to be a bug in syntax highlighting script. The scrip replaces right-- with right-

We will soon fix this post.

1 • Reply • Share ›

Ashok Jangir • 9 months ago

Slight typo: it will be right--; instead of right-;

• Reply • Share ›

Mr. Lazy • 9 months ago

http://ideone.com/MW7Fz3

1 • Reply • Share ›

Amar Vashishth • 9 months ago

Using Doubly-Linked List:
https://ideone.com/BPwilK

• Reply • Share ›

Guest • 9 months ago

NICE

• Reply • Share ›

Parikshit • a year ago

given a array of random 0's and 1's ,convert it to an array such that contiguous indexes of resulting array holds groups of o's and 1's such that their count is always in a arithmetic progression in O(n) time.

• Reply • Share ›

abhijith7 • a year ago

Isn't the following is a simple approach

```
#include<stdio.h>

void rearrange(int a[], int size)

{
int temp, i, j=0;

for( i=0; i<size; i++)="" {="" if(a[i]="=0" &&="" i="">0 )
{
if( a[i-1] != 0 )
{
temp = a[i];
a[i] = a[j];
a[j] = temp;
j++;
}
}
}

//print the array
for( i=0; i<size; i++)="" {="" printf("%d="" ",a[i]);="" }="" printf("\n");="" }="" *="" driver=""
program="" to="" test="" rearrange="" *="" int="" main()="" {="" int="" a[]="
{1,0,1,1,1,0,0,0,1,1,1,1,0};" int="" n="sizeof(a)/sizeof(a[0]);" rearrange(a,="" n);="
getchar();="" return="" 0;="" }="">
```

• Reply • Share ›

Danish • a year ago

```
#include<stdio.h>

#include<stdlib.h>

void segregate1andzero(int a[],int n){

int i=0,j=n-1;

int temp;

while(i!=j){
```

if(a[i]==0&&a[j]==0)

i++;

else if(a[i]==1&&a[j]==0){

temp=a[i];

a[i]=a[j];

see more

• Reply • Share ›

praveen • a year ago

#include <iostream>

using namespace std;

void binaryDigitSort(char arr[], int size){

int j=0, temp;
for(int i=0; i<size; i++){="" if(arr[i]!="1"){="" if(i!="j){" temp="arr[j];" arr[j]="arr[i];" arr[i]="temp;" }="" j++;="" }="" for(int="" i="0;" i<size;i++){="" cout<<arr[i];="" }="" cout<<endl;="" int="" main(){="" char="" arr[]="10101010111111100001111000100" ;="" int="" size="sizeof(arr)/sizeof(arr[0]);" binarydigitsort(arr,="" size);="" return="" 0;="" }="">

• Reply • Share ›

chand • a year ago

how to segregate -1, 0 and 1 in an array in an effective way

• Reply • Share ›

<HoldOnLife!#> chand • a year ago

we can assume a index say 3 and keep a count[3]++ whenevever a[i]=-1 , all we need is count that would work
any better approach for signed numbers?

• Reply • Share ›

Ahmed Sheeraz • a year ago

#include<iostream>
using namespace std;
int main()
{
int arr[8];
for(int i=0;i<8;i++)

{
cin>>arr[i];
}
int left =0;
int right=7;
while(left<right) {="" while(arr[left]="0&&left<right)" left++;="" {="" *="" decrement="" right="" index="" while="" we="" see="" 1="" at="" right="" *="" while(arr[right]="=" 1="" &&="" left="" <="" right)="" right--;="" *="" if="" left="" is="" smaller="" than="" right="" then="" there="" is="" a="" 1="" at="" left="" and="" a="" 0="" at="" right.="" exchange="" arr[left]="" and="" arr[right]*="" if(left="" <="" right)="" {="" arr[left]="0;" arr[right]="1;" left++;="" right--;="" }="" }="" }="" for(int="" i="0;i<8;i++)" {="" cout<<arr[i]<<"="" ";="" }="" }="">

1 • Reply • Share ›

Guest • a year ago

method 1 can have a better solution rather than travelling 1+1.. it can be ;done in n/2 + n/2 times.

in the counting phase start one index from 0 and second index from N/2+1 (if N is odd) or N/2 (if N is even). get the count of 0s and 1s.

in the second run add the required number of 0s and 1s.

Solution 2:

Declare a dummy array with the same size as an input array and initialise it to 0.

start traversing the input array from 0th index and N/2th index (N/2+1 for odd number).

if from 0th index 0 is found then make the 0th element of input element to 0. and if 1 is found then make the Nth element of input array to 1.

Note: Before updating input array update the dummy array. as in if input array value was changed from 0 to 1 or 1 to 0 then corresponding index in the dummy array to be updated by value 1 else retain it to 0

see more

• Reply • Share ›

karthik • a year ago

hi this method required half the iteration. But this is same as below mentioned method.
int i=0;
int j=a1.length-1;
for(i=0;i<a1.length&&(j>=a1.length/2);i++)
{
if(a1[i]==1 && a1[j]==0)

```
{
int temp=a1[j];
a1[j]= a1[i];
a1[i]=temp;
j--;
}
if(a1[j]==1)
j--;
}
for (int z1 = 0; z1 < a1.length; z1++) {
System.out.print(a1[z1] + " ");
}
```

• Reply • Share ›

karthik karthik • a year ago

please use(j!=i) instead of (j>=a1.length/2) in for loop

• Reply • Share ›

karthik karthik • a year ago

sorry guys after lots of testing following method seems to be workingfor(int i=0;i<j;i++) while(i<j)="" {="" if(a1[i]="=0" &&="" a1[j]="=1)" {="" int="" temp="a1[j];" a1[j]="a1[i];" a1[i]="temp;" j--;="" }="" if(a1[i]="=0&a1[j]==0)" i--;="" if(a1[j]="=0)" {="" j--;="" }="" }="">

• Reply • Share ›

SomeGuy • a year ago

Please correct the code:
right–; should be right—;

• Reply • Share ›

Prafulla Malviya • a year ago

```
i=0;

j=arr.length-1;

while(i < j){

if(i==0){

i++;

}

if(j==1){
```

```
j--;

}

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

{

arr[i]=0;

arr[j]=1;

}

}
```


Prafulla Malviya • a year ago

```
public void sortArray(int[] array){
int index1, index2, counter=1;
while((index1 < array.length) && ((index1 < (array.length-counter))){
if(index1==1){
index2 = index1+counter;
if(index2 == 0){
array[index2] = 1;
array[index1] = 0;
c=1;
index1++;
}else{

c++;

}
}
}
}
```

complexity (n-1)

Devil evil • a year ago

```
while ( j < n )//..... i=j=0; initially and n is the size of an array.....//

{
if ( 0 == a[j] )
{
temp = a[i];
```

```
a[j] = a[i];
a[i] = temp;
++i;
}
++j;
}
```

• Reply • Share ›

Devil evil • a year ago

```
int sort01( int * a, int n )
{
int i, j, temp;
i = j = 0;
if ( !a ) {
printf("Invalid array parameter\n");
return 0;
}
if ( n < 0 ) {
printf("Invalid array size\n");
return 0;
}
while ( j < n ) {
if ( 0 == a[j] ) {
temp = a[j];
a[j] = a[i];
a[i] = temp;
++i;
}
++j;
}
return 1;
}
```

• Reply • Share ›

Load more comments