Segregate 0s and 1s in an array

Difficulty Level : Easy • Last Updated : 02 Aug, 2021

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 [Basically you have to sort 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]
```

Recommended: Please solve it on "***PRACTICE***" first, before moving on to the solution.

Thanks to Naveen for suggesting this method.
1) Count the number of 0s. So let's understand with an example we have an array **arr = [0, 1, 0, 1, 0, 0, 1]** the size of the array is **7** now we will traverse the entire array and find out the number of zeros in the array, In this case the number of zeros is **4** so now we can easily get the number of **Ones** in the array by **Array Length – Number Of Zeros**.

2) Once we have counted, we can fill the array first we will put the **zeros** and then **ones** (we can get number of ones by using above formula).
Time Complexity : O(n)

C++

```
// C++ code to Segregate 0s and 1s in an array
#include <bits/stdc++.h>
using namespace std;

// Function to segregate 0s and 1s
void segregate0and1(int arr[], int n)
```

```
{
    int count = 0; // Counts the no of zeros in arr

    for (int i = 0; i < n; i++) {
        if (arr[i] == 0)
            count++;
    }

    // Loop fills the arr with 0 until count
    for (int i = 0; i < count; i++)
        arr[i] = 0;

    // Loop fills remaining arr space with 1
    for (int i = count; i < n; i++)
        arr[i] = 1;
}

// Function to print segregated array
void print(int arr[], int n)
{
    cout << "Array after segregation is ";

    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";
}
```

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

    segregate0and1(arr, n);
    print(arr, n);

    return 0;
}

// This code is contributed by Sahil_Bansall
```

Java

```
// Java code to Segregate 0s and 1s in an array
class GFG {

    // function to segregate 0s and 1s
    static void segregate0and1(int arr[], int n)
    {
        int count = 0; // counts the no of zeros in arr
        for (int i = 0; i < n; i++) {
            if (arr[i] == 0)
                count++;
        }

        // loop fills the arr with 0 until count
        for (int i = 0; i < count; i++)
            arr[i] = 0;

        // loop fills remaining arr space with 1
        for (int i = count; i < n; i++)
            arr[i] = 1;
    }

    // function to print segregated array
    static void print(int arr[], int n)
    {
        System.out.print("Array after segregation is ");
        for (int i = 0; i < n; i++)
            System.out.print(arr[i] + " ");
    }

    // driver function
    public static void main(String[] args)
    {
        int arr[] = new int[]{ 0, 1, 0, 1, 1, 1 };
    }
}

// This code is contributed by Kamal Rawal
```

Python3

```
# Python 3 code to Segregate
# 0s and 1s in an array

# Function to segregate 0s and 1s
def segregate0and1(arr, n) :

    # Counts the no of zeros in arr
    count = 0

    for i in range(0, n) :
        if (arr[i] == 0) :
            count = count + 1
    # Loop fills the arr with 0 until count
    for i in range(0, count) :
        arr[i] = 0

    # Loop fills remaining arr space with 1
    for i in range(count, n) :
        arr[i] = 1


# Function to print segregated array
def print_arr(arr , n) :
    print( "Array after segregation is ",end = "")

    for i in range(0, n) :
        print(arr[i] , end = " ")


# Driver function
arr = [ 0, 1, 0, 1, 1, 1 ]
n = len(arr)

segregate0and1(arr, n)
print_arr(arr, n)
```

C#

```
// C# code to Segregate 0s and 1s in an array
using System;

class GFG {

    // function to segregate 0s and 1s
    static void segregate0and1(int []arr, int n)
    {
        // counts the no of zeros in arr
        int count = 0;

        for (int i = 0; i < n; i++) {
            if (arr[i] == 0)
                count++;
        }

        // loop fills the arr with 0 until count
        for (int i = 0; i < count; i++)
            arr[i] = 0;
        // loop fills remaining arr space with 1
        for (int i = count; i < n; i++)
            arr[i] = 1;
    }

    // function to print segregated array
    static void print(int []arr, int n)
    {
        Console.WriteLine("Array after segregation is ");
        for (int i = 0; i < n; i++)
            Console.Write(arr[i] + " ");
    }

    // driver function
    public static void Main()
    {
        int []arr = new int[]{ 0, 1, 0, 1, 1, 1 };
        int n = arr.Length;

        segregate0and1(arr, n);
        print(arr, n);

    }
}
```

PHP

```
<?php
// PHP code to Segregate
// 0s and 1s in an array

// Function to segregate
// 0s and 1s
function segregate0and1(&$arr, $n)
{
    $count = 0; // Counts the no
                // of zeros in arr

    for ($i = 0; $i < $n; $i++)
    {
        if ($arr[$i] == 0)
            $count++;
    }

    // Loop fills the arr
    // with 0 until count
    for ($i = 0; $i < $count; $i++)
```

```
        $arr[$i] = 0;

    // Loop fills remaining
    // arr space with 1
    for ($i = $count; $i < $n; $i++)
        $arr[$i] = 1;
}

// Function to print
// segregated array
function toprint(&$arr , $n)
{
    echo ("Array after segregation is ");

    for ($i = 0; $i < $n; $i++)
        echo ( $arr[$i] . " ");
}

// Driver Code
$arr = array(0, 1, 0, 1, 1, 1 );
$n = sizeof($arr);

segregate0and1($arr, $n);
toprint($arr, $n);

// This code is contributed
```

Javascript

```
<script>
// JavaScript code to Segregate 0s and 1s in an array

// Function to segregate 0s and 1s
function segregate0and1(arr, n)
{
    let count = 0; // Counts the no of zeros in arr

    for (let i = 0; i < n; i++) {
        if (arr[i] == 0)
            count++;
    }

    // Loop fills the arr with 0 until count
    for (let i = 0; i < count; i++)
        arr[i] = 0;

    // Loop fills remaining arr space with 1
    for (let i = count; i < n; i++)
```

```
            arr[i] = 1;
    }

    // Function to print segregated array
    function print(arr, n)
    {
        document.write("Array after segregation is ");

        for (let i = 0; i < n; i++)
            document.write(arr[i] + " ");
    }

    // Driver function

        let arr = [ 0, 1, 0, 1, 1, 1 ];
        let n = arr.length;

        segregate0and1(arr, n);
        print(arr, n);


    // This code is contributed by Surbhi Tyagi

    </script>
```

```
Array after segregation is 0 0 1 1 1 1
```

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 the 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++

```
// C++ program to sort a binary array in one pass
#include <bits/stdc++.h>
using namespace std;

/*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;
```

```
}

/* Driver code */
int main()
{
    int arr[] = {0, 1, 0, 1, 1, 1};
    int i, arr_size = sizeof(arr)/sizeof(arr[0]);

    segregate0and1(arr, arr_size);

    cout << "Array after segregation ";
    for (i = 0; i < 6; i++)
        cout << arr[i] << " ";
    return 0;
}

// This is code is contributed by rathbhupendra
```

C

```
// 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++;
```

```
/* 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;
}
```

ava

```
class Segregate
```

```
{
    /*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--;
            }
```

```
    public static void main(String[] args)
    {
        Segregate seg = new Segregate();
        int arr[] = new int[]{0, 1, 0, 1, 1, 1};
        int i, arr_size = arr.length;

        seg.segregate0and1(arr, arr_size);

        System.out.print("Array after segregation is ");
        for (i = 0; i < 6; i++)
            System.out.print(arr[i] + " ");
    }
}
```

## Python

```
  Python program to sort a binary array in one pass

# Function to put all 0s on left and all 1s on right
def segregate0and1(arr, size):
    # Initialize left and right indexes
    left, right = 0, size-1

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

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

        # 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 += 1
            right -= 1

    return arr

# driver program to test
arr = [0, 1, 0, 1, 1, 1]
arr_size = len(arr)
print("Array after segregation")
```

C#

```
// C# program to sort a binary array in one pass
using System;

class Segregate
{
    /*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 above functions */
    public static void Main()
    {
        Segregate seg = new Segregate();
        int []arr = new int[]{0, 1, 0, 1, 1, 1};
        int i, arr_size = arr.Length;
```

```
            Console.Write(arr[i] + " ");
    }
}

//This code is contributed by vt_m.
```

PHP

```
<?php
// PHP program to sort a
// binary array in one pass

// Function to put all 0s on
// left and all 1s on right
function segregate0and1(&$arr, $size)

    // Initialize left and
    // right indexes
    $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--;
        }
```

```
$arr = array(0, 1, 0, 1, 1, 1);
$arr_size = sizeof($arr);

segregate0and1($arr, $arr_size);

printf("Array after segregation is ");
for ($i = 0; $i < 6; $i++)
    echo ($arr[$i]. " ");

// This code is contributed
// by Shivi_Aggarwal
?>
```

Javascript

```
script>

// Javascript program to sort a binary array in one pass

/*Function to put all 0s on left and all 1s on right*/
```

```
function segregate0and1(arr, size)
{
    /* Initialize left and right indexes */
    let 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--;
        }
    }
}
```

```
segregate0and1(arr, arr_size);

document.write("Array after segregation ");
for (i = 0; i < 6; i++)
    document.write(arr[i] + " ");

</script>
```

Output:

```
Array after segregation is 0 0 1 1 1 1
```

Time Complexity: O(n)

nother approach :

1. Take two pointer type0(for element 0) starting from beginning (index = 0) and type1(for element 1) starting from end (index = array.length-1).

Initialize type0 = 0 and type1 = array.length-1

2. It is intended to Put 1 to the right side of the array. Once it is done, then 0 will definitely towards the left side of the array.

C++

```
// C++ program to sort a
// binary array in one pass
#include <bits/stdc++.h>
using namespace std;

/*Function to put all 0s on
left and all 1s on right*/
void segregate0and1(int arr[],
                    int size)
{
    int type0 = 0;
    int type1 = size - 1;

    while(type0 < type1)
    {
        if(arr[type0] == 1)
        {
            swap(arr[type0],
```

```
        else
            type0++;
    }
}

// Driver Code
int main()
{
    int arr[] = {0, 1, 0, 1, 1, 1};
    int i, arr_size = sizeof(arr) /
                      sizeof(arr[0]);

    segregate0and1(arr, arr_size);

    cout << "Array after segregation is ";
    for (i = 0; i < arr_size; i++)
        cout << arr[i] << " ";

    return 0;
```

Java

```java
// Java code to segregate 0 and 1
import java.util.*;

class GFG{
/**
Method for segregation 0 and 1 given input array
*/
static void segregate0and1(int arr[]) {
        int type0 = 0;
        int type1 = arr.length - 1;

        while (type0 < type1) {
            if (arr[type0] == 1) {
                // swap
                arr[type1] = arr[type1]+ arr[type0];
                arr[type0] = arr[type1]-arr[type0];
                arr[type1] = arr[type1]-arr[type0];
                type1--;
            } else {
                type0++;
            }
        }
```

```java
public static void main(String[] args) {

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

        segregate0and1(array);

        for(int a : array){
            System.out.print(a+" ");
        }
    }
}
```

Python3

```python
# Python program to sort a
# binary array in one pass

# Function to put all 0s on
# left and all 1s on right
def segregate0and1(arr, size):
    type0 = 0
    type1 = size - 1

    while(type0 < type1):
        if(arr[type0] == 1):
            (arr[type0],
             arr[type1]) = (arr[type1],
                            arr[type0])
            type1 -= 1
        else:
            type0 += 1

# Driver Code
arr = [0, 1, 0, 1, 1, 1]
arr_size = len(arr)
segregate0and1(arr, arr_size)
print("Array after segregation is",
                         end = " ")
for i in range(0, arr_size):
        print(arr[i], end = " ")

# This code is contributed
# by Shivi_Aggarwal
```

```
using System;

class GFG {

// Method for segregation 0
// and 1 given input array
static void segregate0and1(int[] arr)
{
    int type0 = 0;
    int type1 = arr.Length - 1;

    while (type0 < type1)
    {
        if (arr[type0] == 1)
        {
            // swap
            arr[type1] = arr[type1] + arr[type0];
            arr[type0] = arr[type1] - arr[type0];
            arr[type1] = arr[type1] - arr[type0];
            type1--;
        }
        else
        {
            type0++;
        }
    }

}

// Driver Code
public static void Main(string[] args)
{

    int[] array = new int[] {0, 1, 0, 1, 1, 1};
    segregate0and1(array);

    Console.Write("Array after segregation is ");
    foreach (int a in array)
    {
        Console.Write(a + " ");
    }
}
}

// This code is contributed by Shrikant13
```

```
// PHP program to sort a
// binary array in one pass

// Function to put all 0s on
// left and all 1s on right
function segregate0and1(&$arr , $size)
{
    $type0 = 0;
    $type1 = $size - 1;

    while($type0 < $type1)
    {
        if($arr[$type0] == 1)
        {
            $temp = $arr[$type0];
            $arr[$type0] = $arr[$type1];
            $arr[$type1] = $temp;
            $type1--;
        }
        else
        $type0++;
    }
}

// Driver Code
$arr = array(0, 1, 0, 1, 1, 1);
$arr_size = sizeof($arr);

segregate0and1($arr, $arr_size);

echo ("Array after segregation is ");
for ($i = 0; $i < $arr_size; $i++)
    echo ($arr[$i] . " ");

// This code is contributed
// by Shivi_Aggarwal
?>
```

Javascript

```
    <script>

    // Javascript program to sort a
    // binary array in one pass

    // Function to put all 0s on
```

```
        let type1 = size - 1;

        while (type0 < type1)
        {
            if (arr[type0] == 1)
            {

                // Swap
                arr[type1] = arr[type1] + arr[type0];
                arr[type0] = arr[type1] - arr[type0];
                arr[type1] = arr[type1] - arr[type0];
                type1--;
            }
            else
                type0++;
        }
    }

     / Driver Code
    let arr = [ 0, 1, 0, 1, 1, 1 ];
    let i, arr_size = arr.length;
```

```
segregate0and1(arr, arr_size);

document.write("Array after segregation is ");
for(i = 0; i < arr_size; i++)
    document.write(arr[i] + " ");

// This code is contributed by subhammahato348

</script>
```

Output:

```
 Array after segregation is 0 0 1 1 1 1
```

Time complexity: O(n)

// Thanks san4net for suggesting this method.

Segregate 0s and 1s in an array | GeeksforGeeks

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

Attention reader! Don't stop learning now. Get hold of all the important DSA concepts with the **DSA Self Paced Course** at a student-friendly price and become industry ready. To complete your preparation from learning a language to DS Algo and many more, please refer **Complete Interview Preparation Course**.

In case you wish to attend **live classes** with experts, please refer **DSA Live Classes for orking Professionals** and **Competitive Programming Live for Students**.

Like 77

Previous Next

RECOMMENDED ARTICLES

Page : 1 2 3

01 **Segregate Prime and Non-Prime Numbers in an array**
30, Apr 21

02 **Segregate Even and Odd numbers**
07, Jun 10

05 **Segregate Even and Odd numbers | Set 2**
01, Nov 20

06 **Segregate even and odd numbers | Set 3**
13, May 18

03 **Segregate even and odd nodes in a Linked List**
22, Apr 11

04 **Segregate even and odd numbers | Set 2**
01, Feb 18

07 **Modify array to another given array by replacing array elements with the sum of the array**
12, Feb 21

08 **Minimize swaps required to place largest and smallest array elements at first and last array indices**
28, Jan 21

Article Contributed By :

GeeksforGeeks

Easy Normal Medium Hard Expert

Improved By : nik1996, Shivi_Aggarwal, shrikanth13, rathbhupendra, surbhityagi15, subham348, subhammahato348, namant744

Article Tags : Accolite, Amazon, Fab.com, MakeMyTrip, Paytm, Zoho, Arrays, Sorting

Practice Tags : Paytm, Zoho, Accolite, Amazon, MakeMyTrip, Fab.com, Arrays, Sorting

Improve Article Report Issue

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

Load Comments

GeeksforGeeks

5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org

Company

- About Us
- Careers
- Privacy Policy
- Contact Us
- Copyright Policy

Learn

- Algorithms
- Data Structures
- Languages
- CS Subjects
- Video Tutorials

Web Development

- HTML
- CSS
- JavaScript
- Bootstrap

Contribute

- Write an Article
- Write Interview Experience
- Internships
- Videos