

Rearrange positive and negative numbers in $O(n)$ time and $O(1)$ extra space

An array contains both positive and negative numbers in random order. Rearrange the array elements so that positive and negative numbers are placed alternatively. Number of positive and negative numbers need not be equal. If there are more positive numbers they appear at the end of the array. If there are more negative numbers, they too appear in the end of the array.

For example, if the input array is [-1, 2, -3, 4, 5, 6, -7, 8, 9], then the output should be [9, -7, 8, -3, 5, -1, 2, 4, 6]

The solution is to first separate positive and negative numbers using partition process of QuickSort. In the partition process, consider 0 as value of pivot element so that all negative numbers are placed before positive numbers. Once negative and positive numbers are separated, we start from the first negative number and first positive number, and swap every alternate negative number with next positive number.

```
// A C++ program to put positive numbers at even indexes (0, 2, 4,...)
// and negative numbers at odd indexes (1, 3, 5, ...)
#include <stdio.h>

// prototype for swap
void swap(int *a, int *b);

// The main function that rearranges elements of given array. It puts
// positive elements at even indexes (0, 2, ..) and negative numbers at
// odd indexes (1, 3, ..).
void rearrange(int arr[], int n)
{
    // The following few lines are similar to partition process
    // of QuickSort. The idea is to consider 0 as pivot and
```

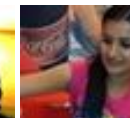
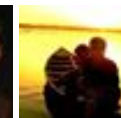
Google™ Custom Search



GeeksforGeeks



53,519 people like [GeeksforGeeks](#).



Interview Experiences

Advanced Data Structures

Dynamic Programming

Greedy Algorithms

Backtracking

Pattern Searching

Divide & Conquer

Mathematical Algorithms

Recursion

RemoteScan Is Now Dell

 software.dell.com/RemoteScan

Same Trusted RemoteScan Solutions. Get Your Free 10-Day Trial Now!

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](#)

```
// divide the array around it.
int i = -1;
for (int j = 0; j < n; j++)
{
    if (arr[j] < 0)
    {
        i++;
        swap(&arr[i], &arr[j]);
    }
}

// Now all positive numbers are at end and negative numbers at
// the beginning of array. Initialize indexes for starting point
// of positive and negative numbers to be swapped
int pos = i+1, neg = 0;

// Increment the negative index by 2 and positive index by 1, i.e.
// swap every alternate negative number with next positive number
while (pos < n && neg < pos && arr[neg] < 0)
{
    swap(&arr[neg], &arr[pos]);
    pos++;
    neg += 2;
}
}
```

```
// A utility function to swap two elements
```

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

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

```
// A utility function to print an array
```

```
void printArray(int arr[], int n)
```

```
{
    for (int i = 0; i < n; i++)
        printf("%4d ", arr[i]);
}
```

```
// Driver program to test above functions
```

```
int main()
```

```
{
    int arr[] = {-1, 2, -3, 4, 5, 6, -7, 8, 9};
    int n = sizeof(arr)/sizeof(arr[0]);
    rearrange(arr, n);
}
```

```

    printArray(arr, n);
    return 0;
}

```

Output:

4 -3 5 -1 6 -7 2 8 9

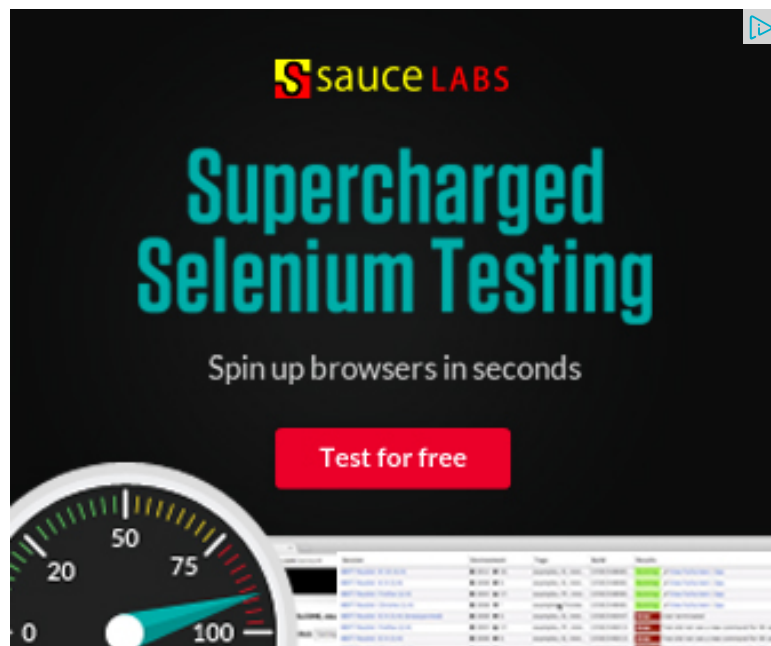
Time Complexity: $O(n)$ where n is number of elements in given array.

Auxiliary Space: $O(1)$

Note that the partition process changes relative order of elements.

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

If you like GeeksforGeeks and would like to contribute, you can also write an article and mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.



Related Topics:



- Remove minimum elements from either side such that $2 \times \text{min}$ 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



44



Tweet

2



2

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

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

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

Find subarray with given sum · 1 hour ago

AdChoices

► [JavaScript Array](#)

► [Negative Numbers](#)


► [Java Array](#)

AdChoices 

► [Java Array](#)

► [Convert Numbers](#)

► [Numbers Number](#)

AdChoices 

► [Adding Numbers](#)

► [Counting Numbers](#)

► [Odd Numbers](#)

@geeksforgeeks, **Some rights reserved**

Contact Us!

Powered by **WordPress** & **MooTools**, customized by geeksforgeeks team