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.

C

```
// 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
    // 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;
}
```

Run on IDE

Java

```
// A JAVA program to put positive numbers at even indexes
// (0, 2, 4,..) and negative numbers at odd indexes (1, 3,
// 5, ..)
import java.io.*;

class Alternate {

    // 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, ..).
    static 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 divide the array around it.
```

```
        int i = -1, temp = 0;
        for (int j = 0; j < n; j++)
        {
            if (arr[j] < 0)
            {
                i++;
                temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }

        // 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)
        {
            temp = arr[neg];
            arr[neg] = arr[pos];
            arr[pos] = temp;
            pos++;
            neg += 2;
        }
    }

    // A utility function to print an array
    static void printArray(int arr[], int n)
    {
        for (int i = 0; i < n; i++)
            System.out.print(arr[i] + "   ");
    }

    /*Driver function to check for above functions*/
    public static void main (String[] args)
    {
        int arr[] = {-1, 2, -3, 4, 5, 6, -7, 8, 9};
        int n = arr.length;
        rearrange(arr,n);
        System.out.println("Array after rearranging: ");
        printArray(arr,n);
    }
}
/*This code is contributed by Devesh Agrawal*/
```

Run on IDE

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.

83 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)

3.6 Average Difficulty : **3.6/5.0**
Based on **3** 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.