Sort array after converting elements to their squares

Difficulty Level : Medium • Last Updated : 24 Nov, 2021

Given an array of both positive and negative integers 'arr[]' which are sorted. The task is to sort the square of the numbers of the Array.

Examples:

```
Input  : arr[] =  {-6, -3, -1, 2, 4, 5}
Output : 1, 4, 9, 16, 25, 36

Input  : arr[] = {-5, -4, -2, 0, 1}
Output : 0, 1, 4, 16, 25
```

Recommended: Please try your approach on *{IDE}* first, before moving on to the solution.

Simple solution is to first convert each array element into its square and then apply any "O(nlogn)" sorting algorithm to sort the array elements.

Below is the implementation of the above idea

```
#include <bits/stdc++.h>
using namespace std;

// Function to sort an square array
void sortSquares(int arr[], int n)
{
    // First convert each array elements
    // into its square
    for (int i = 0; i < n; i++)
        arr[i] = arr[i] * arr[i];

    // Sort an array using "sort STL function "
    sort(arr, arr + n);
}

// Driver program to test above function
int main()
{
    int arr[] = { -6, -3, -1, 2, 4, 5 };
    int n = sizeof(arr) / sizeof(arr[0]);

    cout << "Before sort " << endl;
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";
    sortSquares(arr, n);

    cout << "\nAfter Sort " << endl;
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";
```

Java

```java
// Java program to Sort square of the numbers
// of the array
import java.util.*;
import java.io.*;

class GFG {
    // Function to sort an square array
    public static void sortSquares(int arr[])
    {
        int n = arr.length;

        // First convert each array elements
        // into its square
        for (int i = 0; i < n; i++)
            arr[i] = arr[i] * arr[i];

        // Sort an array using "inbuild sort function"
        // in Arrays class.
        Arrays.sort(arr);
    }

    // Driver program to test above function
    public static void main(String[] args)
    {
        int arr[] = { -6, -3, -1, 2, 4, 5 };
        int n = arr.length;
```

```
        System.out.println("After Sort ");
        for (int i = 0; i < n; i++)
            System.out.print(arr[i] + " ");
    }
}
```

Python3

```
# Python program to Sort square
# of the numbers of the array

# Function to sort an square array
def sortSquare(arr, n):

    # First convert each array
    # elements into its square
    for i in range(n):
        arr[i]= arr[i] * arr[i]
    arr.sort()

# Driver code
arr = [-6, -3, -1, 2, 4, 5]
n = len(arr)

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

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

# This code is contributed by
# Shrikant13
```

C#

```
// C# program to Sort square
// of the numbers of the array
using System;

class GFG {

    // Function to sort
    // an square array
    public static void sortSquares(int[] arr)
    {
        int n = arr.Length;

        // First convert each array
        // elements into its square
        for (int i = 0; i < n; i++)
            arr[i] = arr[i] * arr[i];

        // Sort an array using
        // "inbuild sort function"
```

```
    {
        int[] arr = { -6, -3, -1,
                      2, 4, 5 };
        int n = arr.Length;

        Console.WriteLine("Before sort ");
        for (int i = 0; i < n; i++)
            Console.Write(arr[i] + " ");

        sortSquares(arr);
        Console.WriteLine("");
        Console.WriteLine("After Sort ");

        for (int i = 0; i < n; i++)
            Console.Write(arr[i] + " ");
    }
}

// This code is contributed by anuj_67.
```

Javascript

```
<script>

// JavaScript program for the above approach

    // Function to sort an square array
```

```
        for (let i = 0; i < n; i++)
            arr[i] = arr[i] * arr[i];

        // Sort an array using "inbuild sort function"
        // in Arrays class.
        arr.sort();
    }

// Driver Code
    let arr = [ -6, -3, -1, 2, 4, 5 ];
    let n = arr.length;

    document.write("Before sort " + "<br/>");
    for (let i = 0; i < n; i++)
        document.write(arr[i] + " ");

    sortSquares(arr);
    document.write("" + "<br/>");
    document.write("After Sort " + "<br/>");
    for (let i = 0; i < n; i++)
        document.write(arr[i] + " ");

// This code is contributed by chinmoy1997pal.
</script>
```

Output:

Time complexity: O(n log n)

Efficient solution is based on the fact that the given array is already sorted. We do the following two steps.

1. Divide the array into two-part "Negative and positive ".
2. Use merge function to merge two sorted arrays into a single sorted array.

Below is the implementation of the above idea.

C++

```
// C++ program to Sort square of the numbers of the array
#include <bits/stdc++.h>
using namespace std;

// function to sort array after doing squares of elements
void sortSquares(int arr[], int n)
{
    // first divide array into negative and positive part
    int K = 0;
    for (K = 0; K < n; K++)
        if (arr[K] >= 0)
            break;
```

```
int j = K; // Initial index of second half
int ind = 0; // Initial index of temp array

// store sorted array
int temp[n];
while (i >= 0 && j < n) {
    if (arr[i] * arr[i] < arr[j] * arr[j]) {
        temp[ind] = arr[i] * arr[i];
        i--;
    }
    else {
        temp[ind] = arr[j] * arr[j];
        j++;
    }
    ind++;
}

/* Copy the remaining elements of first half */
while (i >= 0) {
    temp[ind] = arr[i] * arr[i];
    i--;
    ind++;
}

/* Copy the remaining elements of second half */
while (j < n) {
    temp[ind] = arr[j] * arr[j];
    j++;
    ind++;
```

```
}

// Driver program to test above function
int main()
{
    int arr[] = { -6, -3, -1, 2, 4, 5 };
    int n = sizeof(arr) / sizeof(arr[0]);

    cout << "Before sort " << endl;
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";
    sortSquares(arr, n);

    cout << "\nAfter Sort " << endl;
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";

    return 0;
}
```

Java

```
// Java program to Sort square of the numbers
// of the array
import java.util.*;
import java.io.*;

class GFG {
```

```
for (k = 0; k < n; k++) {
    if (arr[k] >= 0)
        break;
}

// Now do the same process that we learnt
// in merge sort to merge two sorted arrays
// here both two halves are sorted and we traverse
// first half in reverse manner because
// first half contains negative elements
int i = k - 1; // Initial index of first half
int j = k; // Initial index of second half
int ind = 0; // Initial index of temp array

int[] temp = new int[n];
while (i >= 0 && j < n) {
    if (arr[i] * arr[i] < arr[j] * arr[j]) {
        temp[ind] = arr[i] * arr[i];
        i--;
    }
    else {

        temp[ind] = arr[j] * arr[j];
        j++;
    }
    ind++;
}

while (i >= 0) {
```

```
        }

        // copy 'temp' array into original array
        for (int x = 0; x < n; x++)
            arr[x] = temp[x];
    }

    // Driver program to test above function
    public static void main(String[] args)
    {
        int arr[] = { -6, -3, -1, 2, 4, 5 };
        int n = arr.length;

        System.out.println("Before sort ");
        for (int i = 0; i < n; i++)
            System.out.print(arr[i] + " ");

        sortSquares(arr);
        System.out.println("");
        System.out.println("After Sort ");
        for (int i = 0; i < n; i++)
            System.out.print(arr[i] + " ");
    }
}
```

Python3

```
# first divide array into negative and positive part
K = 0
for K in range(n):
    if (arr[K] >= 0 ):
        break

# Now do the same process that we learnt
# in merge sort to merge to two sorted array
# here both two halves are sorted and we traverse
# first half in reverse manner because
# first half contains negative elements
i = K - 1 # Initial index of first half
j = K # Initial index of second half
ind = 0 # Initial index of temp array

# store sorted array
temp = [0]*n
while (i >= 0 and j < n):

    if (arr[i] * arr[i] < arr[j] * arr[j]):
        temp[ind] = arr[i] * arr[i]
        i -= 1

    else:

        temp[ind] = arr[j] * arr[j]
        j += 1

    ind += 1
```

```
    ''' Copy the remaining elements of second half '''
    while (j < n):
        temp[ind] = arr[j] * arr[j]
        j += 1
        ind += 1

    # copy 'temp' array into original array
    for i in range(n):
        arr[i] = temp[i]

# Driver code
arr = [-6, -3, -1, 2, 4, 5 ]
n = len(arr)

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

sortSquares(arr, n)
print("\nAfter Sort ")
for i in range(n):
    print(arr[i], end =" " )

# This code is contributed by shubhamsingh10
```

C#

```
// Function to sort an square array
public static void sortSquares(int[] arr)
{
    int n = arr.Length;

    // first divide array into negative and positive part
    int k;
    for (k = 0; k < n; k++) {
        if (arr[k] >= 0)
            break;
    }

    // Now do the same process that we learnt
    // in merge sort to merge to two sorted array
    // here both two halves are sorted and we traverse
    // first half in reverse manner because
    // first half contains negative elements
    int i = k - 1; // Initial index of first half
    int j = k; // Initial index of second half
    int ind = 0; // Initial index of temp array

    int[] temp = new int[n];
    while (i >= 0 && j < n) {
        if (arr[i] * arr[i] < arr[j] * arr[j]) {
            temp[ind] = arr[i] * arr[i];
            i--;
        }
        else {
```

```
    while (i >= 0) {
        temp[ind++] = arr[i] * arr[i];
        i--;
    }
    while (j < n) {
        temp[ind++] = arr[j] * arr[j];
        j++;
    }

    // copy 'temp' array into original array
    for (int x = 0; x < n; x++)
        arr[x] = temp[x];
}

// Driver code
public static void Main(String[] args)
{
    int[] arr = { -6, -3, -1, 2, 4, 5 };
    int n = arr.Length;

    Console.WriteLine("Before sort ");
    for (int i = 0; i < n; i++)
        Console.Write(arr[i] + " ");

    sortSquares(arr);
    Console.WriteLine("");
    Console.WriteLine("After Sort ");
    for (int i = 0; i < n; i++)
        Console.Write(arr[i] + " ");
```

Javascript

```
<script>

// Javascript program to Sort
// square of the numbers
// of the array

    // Function to sort an square array
    function sortSquares(arr)
    {
        let n = arr.length;
        // first dived array into part
        // negative and positive
        let k;
        for (k = 0; k < n; k++) {
            if (arr[k] >= 0)
                break;
        }

        // Now do the same process that we learn
        // in merge sort to merge to two sorted array
        // here both two half are sorted and we traverse
        // first half in reverse meaner because
        // first half contain negative element


        let i = k - 1; // Initial index of first half
```

```
            temp[ind] = arr[i] * arr[i];
            i--;
        }
        else {

            temp[ind] = arr[j] * arr[j];
            j++;
        }
        ind++;
    }

    while (i >= 0) {
        temp[ind++] = arr[i] * arr[i];
        i--;
    }
    while (j < n) {
        temp[ind++] = arr[j] * arr[j];
        j++;
    }

    // copy 'temp' array into original array
    for (let x = 0; x < n; x++)
        arr[x] = temp[x];
}

// Driver program to test above function
let arr=[ -6, -3, -1, 2, 4, 5 ];
let n = arr.length;
document.write("Before sort <br>");
```

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

    // This code is contributed by rag2127

</script>
```

Output

```
Before sort
-6 -3 -1 2 4 5
After Sort
1 4 9 16 25 36
```

Time complexity: O(n)

space complexity: O(n)

Method 3 –

Another efficient solution is based on the two-pointer method as the array is already sorted we can compare the first and last element to check which is bigger and proceed with the result.

Algorithm –

- Initialize left=0 and right=n-1

C++

```
// CPP code for the above approach
#include <bits/stdc++.h>
using namespace std;

// Function to sort an square array
void sortSquares(vector<int>& arr, int n)
{
    int left = 0, right = n - 1;
    int result[n];

    // Iterate from n - 1 to 0
    for (int index = n - 1; index >= 0; index--) {

        // Check if abs(arr[left]) is greater
        // than arr[right]
        if (abs(arr[left]) > arr[right]) {
            result[index] = arr[left] * arr[left];
            left++;
        }
        else {
            result[index] = arr[right] * arr[right];
            right--;
        }
    }
```

```
{
    vector<int> arr;
    arr.push_back(-6);
    arr.push_back(-3);
    arr.push_back(-1);
    arr.push_back(2);
    arr.push_back(4);
    arr.push_back(5);

    int n = 6;

    cout << "Before sort " << endl;
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";

    sortSquares(arr, n);
    cout << endl;
    cout << "After Sort " << endl;
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";
    return 0;
}

// this code is contributed by Manu Pathria
```

Java

```
// Function to sort an square array
public static void sortSquares(int arr[])
{
    int n = arr.length, left = 0,
        right = n - 1;

    int result[] = new int[n];

    for(int index = n - 1; index >= 0; index--)
    {
        if (Math.abs(arr[left]) > arr[right])
        {
            result[index] = arr[left] * arr[left];
            left++;
        }
        else
        {
            result[index] = arr[right] * arr[right];
            right--;
        }
    }
    for(int i = 0; i < n; i++)
        arr[i] = result[i];
}

// Driver code
public static void main(String[] args)
{
```

```
    sortSquares(arr);
    System.out.println("");
    System.out.println("After Sort ");
    for(int i = 0; i < n; i++)
        System.out.print(arr[i] + " ");
}
}

// This code is contributed by jinalparmar2382
```

Python3

```
# Python3 program to Sort square of the numbers of the array

# function to sort array after doing squares of elements
def sortSquares(arr, n):
    left, right = 0, n - 1
    index = n - 1
    result = [0 for x in arr]

    while index >= 0:

        if abs(arr[left]) >= abs(arr[right]):
            result[index] = arr[left] * arr[left]
            left += 1
        else:
```

```
# Driver code
arr = [-6, -3, -1, 2, 4, 5 ]
n = len(arr)

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

sortSquares(arr, n)
print("\nAfter Sort ")
for i in range(n):
    print(arr[i], end =" " )
```

C#

```
// C# program to Sort square of
// the numbers of the array
using System;
class GFG{

// Function to sort an square array
public static void sortSquares(int [] arr)
{
  int n = arr.Length, left = 0,
  right = n - 1;
  int []result = new int[n];
```

```
    {
      result[index] = arr[left] *
                      arr[left];
      left++;
    }
    else
    {
      result[index] = arr[right] *
                      arr[right];
      right--;
    }
  }
  for(int i = 0; i < n; i++)
    arr[i] = result[i];
}

// Driver code
public static void Main(string[] args)
{
  int []arr = {-6, -3, -1, 2, 4, 5};
  int n = arr.Length;
  Console.WriteLine("Before sort ");

  for(int i = 0; i < n; i++)
    Console.Write(arr[i] + " ");

  sortSquares(arr);
  Console.WriteLine("");
  Console.WriteLine("After Sort ");
```

```
// This code is contributed by Chitranayal
```

Javascript

```
<script>
    // Javascript program to Sort square of
    // the numbers of the array

    // Function to sort an square array
    function sortSquares(arr)
    {
      let n = arr.length, left = 0,
      right = n - 1;
      let result = new Array(n);
      result.fill(0);

      for(let index = n - 1; index >= 0; index--)
      {
        if (Math.abs(arr[left]) >
            arr[right])
        {
          result[index] = arr[left] *
                          arr[left];
          left++;
        }
        else
        {
```

```
        arr[i] = result[i];
    }

    let arr = [-6, -3, -1, 2, 4, 5];
    let n = arr.length;
    document.write("Before sort " + "</br>");

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

    sortSquares(arr);
    document.write("</br>");
    document.write("After Sort " + "</br>");

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

 // This code is contributed by rameshtravel07.
</script>
```

Output

```
Before sort
-6 -3 -1 2 4 5
```

Auxiliary Space: O(n)

This article is contributed by **Nishant singh**. If you like GeeksforGeeks and would like to contribute, you can also write an article using write.geeksforgeeks.org or mail your article to review-team@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

before or after it

RECOMMENDED ARTICLES

Page : 1 2 3

01 Sort a linked list after converting elements to their squares
11, Nov 21

02 Replace the odd positioned elements with their cubes and even positioned elements with their squares
26, Mar 19

03 Sort perfect squares in an array at their relative positions
26, Nov 18

04 Modify array by sorting nearest perfect squares of array elements having their digits sorted in

05 Maximize sum of squares of array elements possible by replacing pairs with their Bitwise AND and Bitwise OR
12, Oct 20

06 Sort Array according to modulus of their values with their frequencies
13, Dec 21

07 Count of primes after converting given binary number in base between L to R
10, May 20

08 Sort an array of strings by replacements with

Article Contributed By :

GeeksforGeeks

Vote for difficulty

Current difficulty : Medium

Easy Normal Medium Hard Expert

Improved By : shrikanth13, vt_m, 29AjayKumar, SHUBHAMSINGH10, skrish87, parmarjinal, ukasp, manupathria, chinmoy1997pal, rag2127, rameshtravel07, uppooraprajwal

Article Tags : Arrays, Sorting

Practice Tags : Arrays, Sorting

Improve Article Report Issue

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

feedback@geeksforgeeks.org