Merge-sort 2

Problem Submissions Leaderboard Discussions

Sort a given set of n integer elements using Merge Sort method and compute its time complexity. Run the program for varied values of n> 5000 and record the time taken to sort. Plot a graph of the time taken versus non graph sheet.The elements can be read from a file or can be generated using the random number generator. Demonstrate using Java how the divide - and - conquer method works along with its time complexity analysis: worst case, average case and best case.

Input Format

5 0 0 4 3 1

Constraints

Size of the array should be always positive

Output Format

Before Sort: 0 0 4 3 1 After sort: 0 0 1 3 4

Sample Input 0

```
5
0
0
4
3
1
```

Sample Output 0

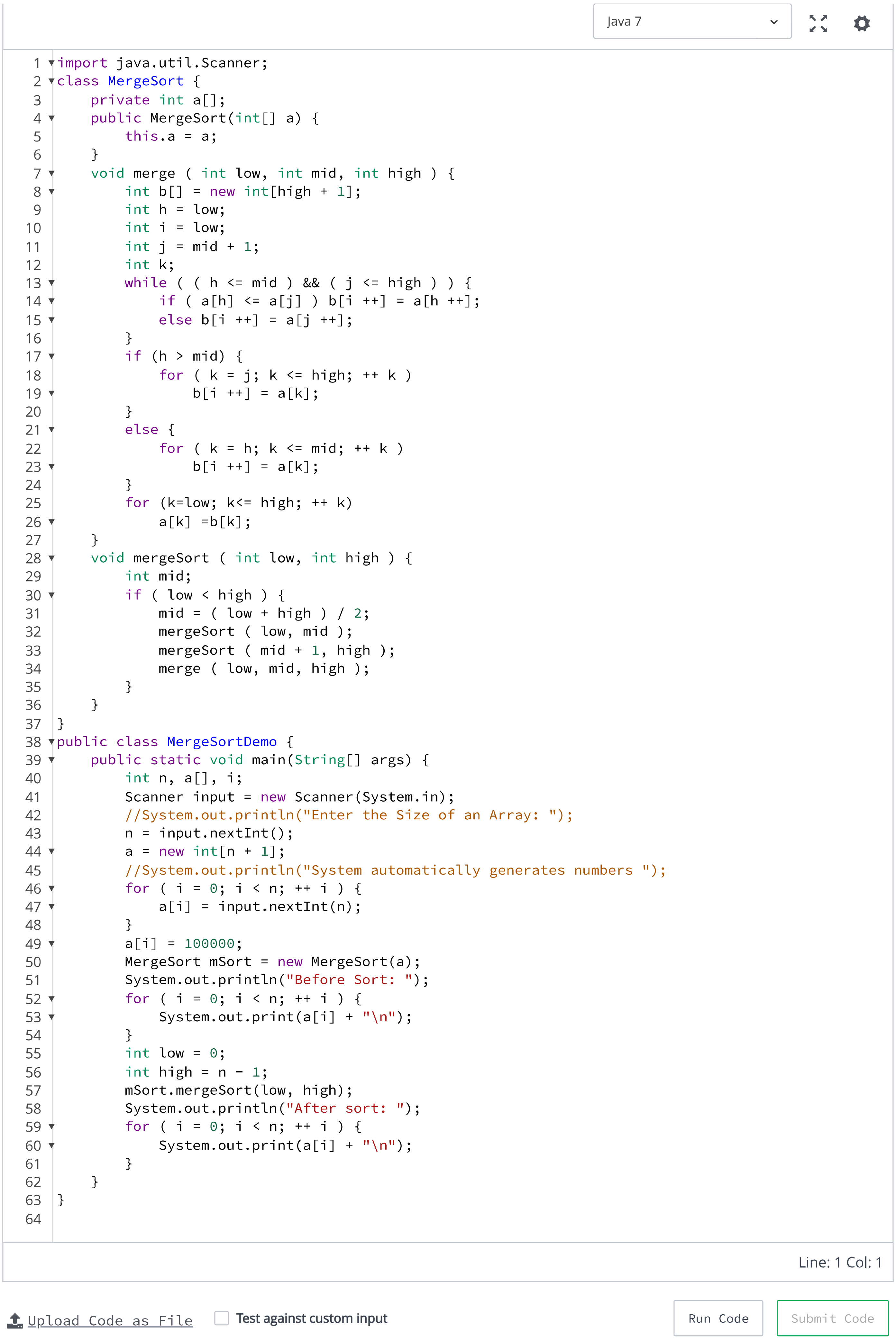
```
Before Sort:
0
0
4
3
1
After sort:
0
0
1
3
4
```

Contest ends in 9 days

Submissions: 103
Max Score: 10
Difficulty: Medium

Rate This Challenge:
☆☆☆☆☆

More


Java 7

```java
import java.util.Scanner;
class MergeSort {
    private int a[];
    public MergeSort(int[] a) {
        this.a = a;
    }
    void merge ( int low, int mid, int high ) {
        int b[] = new int[high + 1];
        int h = low;
        int i = low;
        int j = mid + 1;
        int k;
        while ( ( h <= mid ) && ( j <= high ) ) {
            if ( a[h] <= a[j] ) b[i ++] = a[h ++];
            else b[i ++] = a[j ++];
        }
        if (h > mid) {
            for ( k = j; k <= high; ++ k )
                b[i ++] = a[k];
        }
        else {
            for ( k = h; k <= mid; ++ k )
                b[i ++] = a[k];
        }
        for (k=low; k<= high; ++ k)
            a[k] =b[k];
    }
    void mergeSort ( int low, int high ) {
        int mid;
        if ( low < high ) {
            mid = ( low + high ) / 2;
            mergeSort ( low, mid );
            mergeSort ( mid + 1, high );
            merge ( low, mid, high );
        }
    }
}
public class MergeSortDemo {
    public static void main(String[] args) {
        int n, a[], i;
        Scanner input = new Scanner(System.in);
        //System.out.println("Enter the Size of an Array: ");
        n = input.nextInt();
        a = new int[n + 1];
        //System.out.println("System automatically generates numbers ");
        for ( i = 0; i < n; ++ i ) {
            a[i] = input.nextInt(n);
        }
        a[i] = 100000;
        MergeSort mSort = new MergeSort(a);
        System.out.println("Before Sort: ");
        for ( i = 0; i < n; ++ i ) {
            System.out.print(a[i] + "\n");
        }
        int low = 0;
        int high = n - 1;
        mSort.mergeSort(low, high);
        System.out.println("After sort: ");
        for ( i = 0; i < n; ++ i ) {
            System.out.print(a[i] + "\n");
        }
    }
}
```

Testcase 0 ✓

Congratulations, you passed the sample test case.

Click the **Submit Code** button to run your code against all the test cases.

Input (stdin)

```
5
0
0
4
3
1
```

Your Output (stdout)

```
Before Sort:
0
0
4
3
1
After sort:
0
0
1
3
4
```

Expected Output

```
Before Sort:
0
0
4
3
1
After sort:
0
0
1
3
4
```