merge sort assignmnet

code

class Tester {

public static void mergeSort(int[] elements, int size) {

if (size < 2) {

return; // Base case: array is already sorted if it has 0 or 1 element

}

int mid = size / 2;

int[] left = new int[mid];

int[] right = new int[size - mid];

// Fill left and right subarrays

System.arraycopy(elements, 0, left, 0, mid);

System.arraycopy(elements, mid, right, 0, size - mid);

// Recursively sort left and right subarrays

mergeSort(left, mid);

mergeSort(right, size - mid);

// Merge sorted left and right subarrays into elements

merge(elements, left, right, mid, size - mid);

}

public static void merge(int[] elements, int[] left, int[] right, int leftMerge, int rightMerge) {

int i = 0, j = 0, k = 0;

// Merge left and right arrays into elements

while (i < leftMerge && j < rightMerge) {

if (left[i] <= right[j]) {

elements[k++] = left[i++];

} else {

elements[k++] = right[j++];

}

}

// Copy remaining elements of left array, if any

while (i < leftMerge) {

elements[k++] = left[i++];

}

// Copy remaining elements of right array, if any

while (j < rightMerge) {

elements[k++] = right[j++];

}

}

public static double findMedian(int elements[]) {

int n = elements.length;

if (n % 2 == 1) {

// Odd number of elements: median is the middle element

return elements[n / 2];

} else {

// Even number of elements: median is the average of middle two elements

int middleIndex = n / 2;

return (elements[middleIndex - 1] + elements[middleIndex]) / 2.0;

}

}

public static void main(String[] args) {

int elements[] = { 64, 34, 25, 12, 22, 11, 90 };

mergeSort(elements, elements.length);

System.out.println("Sorted Array:");

for (int element : elements) {

System.out.print(element + " ");

}

System.out.println();

System.out.println("Median: " + findMedian(elements));

}

}

