Solution 1

```java
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

class Program {
  // Best: O(nlog(n)) time | O(log(n)) space
  // Average: O(nlog(n)) time | O(log(n)) space
  // Worst: O(n^2) time | O(log(n)) space
  public static int[] quickSort(int[] array) {
    quickSort(array, 0, array.length - 1);
    return array;
  }

  public static void quickSort(int[] array, int startIdx, int endIdx) {
    if (startIdx >= endIdx) {
      return;
    }
    int pivotIdx = startIdx;
    int leftIdx = startIdx + 1;
    int rightIdx = endIdx;
    while (rightIdx >= leftIdx) {
      if (array[leftIdx] > array[pivotIdx] && array[rightIdx] < array[pivotIdx]) {
        swap(leftIdx, rightIdx, array);
      }
      if (array[leftIdx] <= array[pivotIdx]) {
        leftIdx += 1;
      }
      if (array[rightIdx] >= array[pivotIdx]) {
        rightIdx -= 1;
      }
    }
    swap(pivotIdx, rightIdx, array);
    boolean leftSubarrayIsSmaller = rightIdx - 1 - startIdx < endIdx - (rightIdx + 1);
    if (leftSubarrayIsSmaller) {
      quickSort(array, startIdx, rightIdx - 1);
      quickSort(array, rightIdx + 1, endIdx);
    } else {
      quickSort(array, rightIdx + 1, endIdx);
      quickSort(array, startIdx, rightIdx - 1);
    }
  }

  public static void swap(int i, int j, int[] array) {
    int temp = array[j];
    array[j] = array[i];
    array[i] = temp;
  }
}

```