Solution 1

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

class Program {
  // O(n) time | O(1) space
  public static int[] subarraySort(int[] array) {
    int minOutOfOrder = Integer.MAX_VALUE;
    int maxOutOfOrder = Integer.MIN_VALUE;
    for (int i = 0; i < array.length; i++) {
      int num = array[i];
      if (isOutOfOrder(i, num, array)) {
        minOutOfOrder = Math.min(minOutOfOrder, num);
        maxOutOfOrder = Math.max(maxOutOfOrder, num);
      }
    }
    if (minOutOfOrder == Integer.MAX_VALUE) {
      return new int[] {-1, -1};
    }
    int subarrayLeftIdx = 0;
    while (minOutOfOrder >= array[subarrayLeftIdx]) {
      subarrayLeftIdx++;
    }
    int subarrayRightIdx = array.length - 1;
    while (maxOutOfOrder <= array[subarrayRightIdx]) {
      subarrayRightIdx--;
    }
    return new int[] {subarrayLeftIdx, subarrayRightIdx};
  }

  public static boolean isOutOfOrder(int i, int num, int[] array) {
    if (i == 0) {
      return num > array[i + 1];
    }
    if (i == array.length - 1) {
      return num < array[i - 1];
    }
    return num > array[i + 1] || num < array[i - 1];
  }
}

```