

Solution 1Solution 2

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 #include <vector>
4 using namespace std;
5
6 void mergeSortHelper(vector<int> *mainArray, int startIdx, int endIdx,
7                     vector<int> *auxiliaryArray);
8 void doMerge(vector<int> *mainArray, int startIdx, int middleIdx, int endIdx,
9             vector<int> *auxiliaryArray);
10
11 // Best: O(nlog(n)) time | O(n) space
12 // Average: O(nlog(n)) time | O(n) space
13 // Worst: O(nlog(n)) time | O(n) space
14 vector<int> mergeSort(vector<int> array) {
15     if (array.size() <= 1) {
16         return array;
17     }
18     vector<int> auxiliaryArray = array;
19     mergeSortHelper(&array, 0, array.size() - 1, &auxiliaryArray);
20     return array;
21 }
22
23 void mergeSortHelper(vector<int> *mainArray, int startIdx, int endIdx,
24                     vector<int> *auxiliaryArray) {
25     if (startIdx == endIdx) {
26         return;
27     }
28     int middleIdx = (startIdx + endIdx) / 2;
29     mergeSortHelper(auxiliaryArray, startIdx, middleIdx, mainArray);
30     mergeSortHelper(auxiliaryArray, middleIdx + 1, endIdx, mainArray);
31     doMerge(mainArray, startIdx, middleIdx, endIdx, auxiliaryArray);
32 }
33
34 void doMerge(vector<int> *mainArray, int startIdx, int middleIdx, int endIdx,
35             vector<int> *auxiliaryArray) {
36     int k = startIdx;
37     int i = startIdx;
38     int j = middleIdx + 1;
39     while (i <= middleIdx && j <= endIdx) {
40         if (auxiliaryArray->at(i) <= auxiliaryArray->at(j)) {
41             mainArray->at(k++) = auxiliaryArray->at(i++);
42         } else {
43             mainArray->at(k++) = auxiliaryArray->at(j++);
44         }
45     }
46     while (i <= middleIdx) {
47         mainArray->at(k++) = auxiliaryArray->at(i++);
48     }
49     while (j <= endIdx) {
50         mainArray->at(k++) = auxiliaryArray->at(j++);
51     }
52 }
53
```

