Solution 1

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

#include <vector>
using namespace std;

void quickSortHelper(vector<int> &array, int startIdx, int endIdx);

// 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
vector<int> quickSort(vector<int> array) {
  quickSortHelper(array, 0, array.size() - 1);
  return array;
}

void quickSortHelper(vector<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.at(leftIdx) > array.at(pivotIdx) &&
        array.at(rightIdx) < array.at(pivotIdx)) {
      swap(array[leftIdx], array[rightIdx]);
    }
    if (array.at(leftIdx) <= array.at(pivotIdx)) {
      leftIdx += 1;
    }
    if (array.at(rightIdx) >= array.at(pivotIdx)) {
      rightIdx -= 1;
    }
  }
  swap(array[pivotIdx], array[rightIdx]);
  bool leftSubarrayIsSmaller =
      rightIdx - 1 - startIdx < endIdx - (rightIdx + 1);
  if (leftSubarrayIsSmaller) {
    quickSortHelper(array, startIdx, rightIdx - 1);
    quickSortHelper(array, rightIdx + 1, endIdx);
  } else {
    quickSortHelper(array, rightIdx + 1, endIdx);
    quickSortHelper(array, startIdx, rightIdx - 1);
  }
}

```