Logic Flaws

Contents

1 Problem Statement

You will be given 3 implemented sorting algorithm. Each algorithm sorts an array `std::vector<int> V` and returns sorted Array. It is guaranteed that:

- `V.size() <= 100`
- `0 <= V[i] && V[i] <= 100` for all `i` with `0 <= i && i < V.size()`

Examples (All of the given codes works in following example):

- {3, 1, 4} → {1, 3, 4}
- {8, 3, 7, 2} → {2, 3, 7, 8}

2 Counting Sort

```cpp
#include <vector>

std::vector<int> CountingSort(std::vector<int> V) {
  std::vector<int> counting_array(V.size()+10, 0);
  for (int i = 0; i < static_cast<int>(V.size()); ++i) {
    ++counting_array[V[i]];
  }

  std::vector<int> sorted_array;
  for (int i = 0; i < static_cast<int>(counting_array.size()); ++i) {
    for (int j = 0; j < counting_array[i]; ++j) {
      sorted_array.push_back(i);
    }
  }

  return sorted_array;
}
```

3 Insertion Sort

```cpp
#include <vector>

std::vector<int> InsertionSort(std::vector<int> V) {
  bool is_array_sorted = true;
  for (int i = 0; i < static_cast<int>(V.size()) - 1; ++i) {
    if (V[i] > V[i+1]) {
      is_array_sorted = false;
      break;
    }
  }
  if (is_array_sorted) {
    return V;
  }

  for (int i = 0; i < static_cast<int>(V.size()) - 1; ++i) {
    int min_index = 0;
    for (int j = i; j < static_cast<int>(V.size()); ++j) {
      if (V[min_index] > V[j]) {
        min_index = j;
      }
    }
    int temp = V[min_index];
    V[min_index] = V[i];
    V[i] = temp;
  }

  return V;
}
```

4 Quick Sort

```cpp
#include <vector>

std::vector<int> FilterLessThan(std::vector<int> V, int pivot) {
  std::vector<int> result;

  for (int elem : V) {
    if (elem < pivot) {
      result.push_back(elem);
    }
  }

  return result;
}

std::vector<int> FilterGreaterThan(std::vector<int> V, int pivot) {
  std::vector<int> result;

  for (int elem : V) {
    if (elem > pivot) {
      result.push_back(elem);
    }
  }

  return result;
}

std::vector<int> QuickSort(std::vector<int> V) {
  if (static_cast<int>(V.size()) <= 1) {
    return V;
  }

  int pivot = V.back();
  V.pop_back();

  std::vector<int> result;
  for (int elem : QuickSort(FilterLessThan(V, pivot))) {
    result.push_back(elem);
  }
  result.push_back(pivot);
  for (int elem : QuickSort(FilterGreaterThan(V, pivot))) {
    result.push_back(elem);
  }

  return result;
}
```