

Logic Flaws

Contents

1	Problem Statement	1
2	Counting Sort	2
3	Insertion Sort	3
4	Quick Sort	4

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):

- $\{3, 1, 4\} \rightarrow \{1, 3, 4\}$
- $\{8, 3, 7, 2\} \rightarrow \{2, 3, 7, 8\}$

2 Counting Sort

```
1  #include <vector>
2
3  std::vector<int> CountingSort(std::vector<int> V) {
4      std::vector<int> counting_array(V.size()+10, 0);
5      for (int i = 0; i < static_cast<int>(V.size()); ++i) {
6          ++counting_array[V[i]];
7      }
8
9      std::vector<int> sorted_array;
10     for (int i = 0; i < static_cast<int>(counting_array.size()); ++i) {
11         for (int j = 0; j < counting_array[i]; ++j) {
12             sorted_array.push_back(i);
13         }
14     }
15
16     return sorted_array;
17 }
```

3 Insertion Sort

```
1  #include <vector>
2
3  std::vector<int> InsertionSort(std::vector<int> V) {
4      bool is_array_sorted = true;
5      for (int i = 0; i < static_cast<int>(V.size()) - 1; ++i) {
6          if (V[i] > V[i+1]) {
7              is_array_sorted = false;
8              break;
9          }
10     }
11     if (is_array_sorted) {
12         return V;
13     }
14
15     for (int i = 0; i < static_cast<int>(V.size()) - 1; ++i) {
16         int min_index = 0;
17         for (int j = i; j < static_cast<int>(V.size()); ++j) {
18             if (V[min_index] > V[j]) {
19                 min_index = j;
20             }
21         }
22         int temp = V[min_index];
23         V[min_index] = V[i];
24         V[i] = temp;
25     }
26
27     return V;
28 }
```

4 Quick Sort

```
1  #include <vector>
2
3  std::vector<int> FilterLessThan(std::vector<int> V, int pivot) {
4      std::vector<int> result;
5
6      for (int elem : V) {
7          if (elem < pivot) {
8              result.push_back(elem);
9          }
10     }
11
12     return result;
13 }
14
15 std::vector<int> FilterGreaterThan(std::vector<int> V, int pivot) {
16     std::vector<int> result;
17
18     for (int elem : V) {
19         if (elem > pivot) {
20             result.push_back(elem);
21         }
22     }
23
24     return result;
25 }
26
27 std::vector<int> QuickSort(std::vector<int> V) {
28     if (static_cast<int>(V.size()) <= 1) {
29         return V;
30     }
31
32     int pivot = V.back();
33     V.pop_back();
34
35     std::vector<int> result;
36     for (int elem : QuickSort(FilterLessThan(V, pivot))) {
37         result.push_back(elem);
38     }
39     result.push_back(pivot);
40     for (int elem : QuickSort(FilterGreaterThan(V, pivot))) {
41         result.push_back(elem);
42     }
43
44     return result;
45 }
```