Practice Arena

Practice problems aimed to improve your coding skills.

Bubble Sort

LAB-PRAC-14_SORT-MISC

Bubble Sort [20 marks]

Problem Statement
There are several sorting algorithms that have been developed over the years. We saw some during the lectures and will explore others here. One simple algorithm is called Bubble Sort. As the name suggests, bubble sort causes elements to "bubble" up to their correct position in the array.

Bubble Sort maintains the following invariant - after the i-th iteration of the algorithm is over, the i-th largest element in the array must reach its correct location in the array. Thus, if the array has n elements in total, the i-th largest element would be found in the index n - i i.e. the largest element would be found at index n-1 after iteration 1, the 2nd largest element would be found in index n - 2 after iteration 2 and so on. Thus, if there are n elements in the array, bubble sort would run for n-1 iterations since after the (n-1)-th iteration, all elements would be in their correct position.

The way bubble sort accomplishes the above is actually very cute - it goes from left to right, comparing adjacent elements. If it finds an element that is strictly larger than the element to its immediate right (i.e. the elements are out of order), it swaps the two elements. Verify that if we do this on an array from left to right, we would end up transporting the largest element of the array to the last position in the array. The first line of the input will give you n, a strictly positive integer greater than 1 (i.e. n can be 2 or larger) and the second line will give you n integers, separated by a space. Store these numbers in an array of size n.

In your output, you have to print the array after each iteration of bubble sort, on a separate line. Print the array by printing each number in the array from left to right with two numbers separated by a single space. However, there should be no space at the end of a line.

The animation below shows you how bubble sort works. Notice that at step i, the i-th largest element reaches its correct location in the array.

Image courtesy: wikipedia.org

Caution

1. Bubble sort goes over the array from left to right, starting from the index 0, comparing adjacent locations and swapping them if they are out of order, in every iteration. However, you should be able to deduce that it does not need to go all the way till the last index in later iterations.
2. There may be iterations where nothing needs to be done. However you still have to print the array after that iteration. Your output must contain n-1 lines.

3. Please do not try to cheat by using library functions like qsort(). These will not sort the array in the order bubble sort will and hence you will not get partial marks for printing the intermediate steps of the algorithm.
4. The n numbers we give you may be positive, negative or zero. The same number may occur twice in the list too.
5. The number of elements n can be any strictly positive number greater than 1. Your output must have exactly n-1 lines.
6. Be careful about extra/missing lines and extra/missing spaces in your output. There should be no space at the end of any line in your output, nor should there be any extra newlines at the end of your output.

EXAMPLE:
INPUT
```
4
3 5 4 2
```

OUTPUT:
```
3 4 2 5
3 2 4 5
2 3 4 5
```

Explanation: Initial state of the array is 3 5 4 2
Iteration 1:

1. 3 and 5 get compared. They are in correct order so no change
2. 5 and 4 get compared. They are out of order so they get swapped. 3 4 5 2
3. 5 and 2 get compared. They are out of order so they get swapped. 3 4 2 5

State of array after iteration 1: 3 4 2 5
Iteration 2:

1. 3 and 4 get compared. They are in correct order so no change
2. 4 and 2 get compared. They are out of order so they get swapped. 3 2 4 5
3. 4 and 5 get compared. They are in correct order so no change

State of array after iteration 2: 3 2 4 5
Iteration 3:

1. 3 and 2 get compared. They are out of order so they get swapped. 2 3 4 5
2. 3 and 4 get compared. They are in correct order so no change
3. 4 and 5 get compared. They are in correct order so no change

State of array after iteration 2: 2 3 4 5

Notice that after iteration 1, 5 (the largest element in the array) reaches its correct position in the sorted array. After iteration 2, 4 (the second largest element in the array) reaches its correct position in the sorted array and so on.

Grading Scheme:
Total marks: **[20 Points]**

There will be partial grading in this question. There are several lines in your output. Printing each line correctly, in the correct order, carries equal weightage. Each visible test case is worth 2 points and each hidden test case is worth 4 points. There are 2 visible and 4 hidden test cases.

Please remember, however, that when you press Submit/Evaluate, you will get a green bar only if all parts of your answer are correct. Thus, if your answer is only partly correct, Prutor will say that you have not passed that test case completely, but when we do autograding afterwards, you will get partial marks.

Start Solving! (/editor/practice/6289)