

# Day 20: Sorting

## Objective

Today, we're discussing a simple sorting algorithm called *Bubble Sort*. Check out the [Tutorial](#) tab for learning materials and an instructional video!

Consider the following version of Bubble Sort:

```
for (int i = 0; i < n; i++) {
    // Track number of elements swapped during a single array traversal
    int numberOfSwaps = 0;

    for (int j = 0; j < n - 1; j++) {
        // Swap adjacent elements if they are in decreasing order
        if (a[j] > a[j + 1]) {
            swap(a[j], a[j + 1]);
            numberOfSwaps++;
        }
    }

    // If no elements were swapped during a traversal, array is sorted
    if (numberOfSwaps == 0) {
        break;
    }
}
```

## Task

Given an array,  $a$ , of size  $n$  distinct elements, sort the array in *ascending* order using the *Bubble Sort* algorithm above. Once sorted, print the following **3** lines:

1. `Array is sorted in numSwaps swaps.`  
where *numSwaps* is the number of swaps that took place.
2. `First Element: firstElement`  
where *firstElement* is the *first* element in the sorted array.
3. `Last Element: lastElement`  
where *lastElement* is the *last* element in the sorted array.

**Hint:** To complete this challenge, you will need to add a variable that keeps a running tally of *all* swaps that occur during execution.

## Example

$a = [4, 3, 1, 2]$

```
original a: 4 3 1 2
round 1  a: 3 1 2 4 swaps this round: 3
round 2  a: 1 2 3 4 swaps this round: 2
round 3  a: 1 2 3 4 swaps this round: 0
```

In the first round, the **4** is swapped at each of the **3** comparisons, ending in the last position. In the second round, the **3** is swapped at **2** of the **3** comparisons. Finally, in the third round, no swaps are made so the iterations stop. The output

is the following:

```
Array is sorted in 5 swaps.  
First Element: 1  
Last Element: 4
```

### Input Format

The first line contains an integer,  $n$ , the number of elements in array  $a$ .

The second line contains  $n$  space-separated integers that describe  $a[0], a[1], \dots, a[n - 1]$ .

### Constraints

- $2 \leq n \leq 600$
- $1 \leq a[i] \leq 2 \times 10^6$ , where  $0 \leq i < n$ .

### Output Format

Print the following three lines of output:

1. `Array is sorted in numSwaps swaps.`  
where *numSwaps* is the number of swaps that took place.
2. `First Element: firstElement`  
where *firstElement* is the *first* element in the sorted array.
3. `Last Element: lastElement`  
where *lastElement* is the *last* element in the sorted array.

### Sample Input 0

```
3  
1 2 3
```

### Sample Output 0

```
Array is sorted in 0 swaps.  
First Element: 1  
Last Element: 3
```

### Explanation 0

The array is already sorted, so **0** swaps take place and we print the necessary **3** lines of output shown above.

### Sample Input 1

```
3  
3 2 1
```

### Sample Output 1

```
Array is sorted in 3 swaps.  
First Element: 1  
Last Element: 3
```

### Explanation 1

The array  $a = [3, 2, 1]$  is *not sorted*, so we perform the following **3** swaps. Each line shows  $a$  after each single element is swapped.

1.  $[3, 2, 1] \rightarrow [2, 3, 1]$
2.  $[2, 3, 1] \rightarrow [2, 1, 3]$
3.  $[2, 1, 3] \rightarrow [1, 2, 3]$

After **3** swaps, the array is sorted.