

# Quicksort In-Place

## Problem Statement

The previous version of Quicksort was easy to understand, but it was not optimal. It required copying the numbers into other arrays, which takes up space and time. To make things faster, one can create an "in-place" version of Quicksort, where the numbers are all sorted within the array itself.

## Challenge

Create an in-place version of Quicksort. You need to follow [Lomuto Partitioning](#) method.

## Guideline

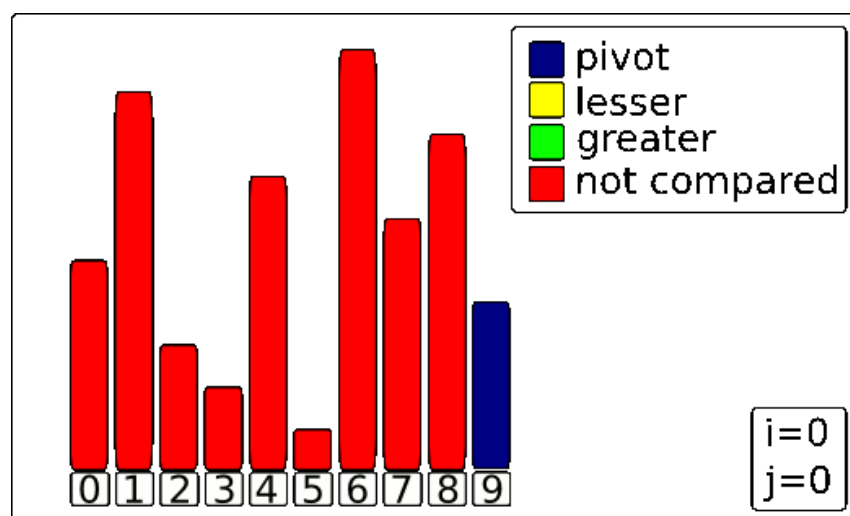
Instead of copying the array into multiple sub-arrays, use indices to keep track of the different sub-arrays. You can pass the indices to a modified *partition* method. The partition method should partition the sub-array and then return the index location where the pivot gets placed, so you can then call partition on each side of the pivot.

- Always select the *last* element in the 'sub-array' as a pivot.
- Partition the left side and then the right side of the sub-array.
- Print out the whole array at the end of every partitioning method.
- An array of length 1 or less will be considered sorted, and there is no need to sort or to print them.

Since you cannot just create new sub-arrays for the elements, partition method will need to use another trick to keep track of which elements are greater and which are lower than the pivot.

## The In-place Trick

- If an element is lower than the pivot, you should swap it with a larger element on the left-side of the sub-array.
- Greater elements should remain where they are.
- At the end of the partitioning method, the pivot should be swapped with the first element of the right partition, consisting of all larger elements, of the sub-array.
- To ensure that you don't swap a small element with another small element, use an index to keep track of the small elements that have already been swapped into their "place".



## Input Format

There will be two lines of input:

- $n$  - the size of the array
- $ar$  - the  $n$  numbers of the array

### Output Format

Print the entire array on a new line at the end of every partitioning method.

### Constraints

$1 \leq n \leq 5000$   
 $-10000 \leq x \leq 10000, x \in ar$   
 There are no duplicate numbers.

### Sample Input

```
7
1 3 9 8 2 7 5
```

### Sample Output

```
1 3 2 5 9 7 8
1 2 3 5 9 7 8
1 2 3 5 7 8 9
```

### Explanation

5 is initially selected as the pivot, and the array is partitioned as shown in the diagram. The left side is partitioned next with 2 as the pivot. Finally, the right side is partitioned with 8 as the pivot. The entire array is now sorted.

