

WORKSHEET, WEEK 07: SORTING

7.1 QuickSort

7.1.1 Pseudocode for Quicksort

This variant of Quicksort uses the leftmost element of the input area as a pivot. It is the same as we have in the lecture slides, but may differ from some other Quicksort flavors (randomized etc.) that you may encounter in other sources.

```

QUICKSORT( $A[\ell \dots r]$ ) :
1  if  $\ell < r$  :
2       $i = \ell$       ( $i$  increases from the left and searches elements  $\geq$  than pivot)
3       $j = r + 1$     ( $j$  decreases from the right and searches elements  $\leq$  than pivot.)
4       $v = A[\ell]$     ( $v$  is the pivot.)
5      while  $i < j$  :
6           $i = i + 1$ 
7          while  $i < r$  and  $A[i] < v$  :
8               $i = i + 1$ 
9           $j = j - 1$ 
10         while  $j > \ell$  and  $A[j] > v$  :
11              $j = j - 1$ 
12          $A[i] \leftrightarrow A[j]$  (Undo the extra swap at the end)
13      $A[i] \leftrightarrow A[j]$  (Undo the extra swap at the end)
14      $A[j] \leftrightarrow A[\ell]$  (Move pivot to its proper place)
15     QUICKSORT( $A[\ell \dots j - 1]$ )
16     QUICKSORT( $A[j + 1 \dots r]$ )

```

(A) Run this pseudocode for one invocation $\text{QUICKSORT}(A[0..11])$, where the table to sort is the following:

13, 0, 23, 1, 8, 9, 29, 16, 8, 24, 6, 11.

Draw the state of the array every time you swap two elements (i.e. execute $A[k_1] \leftrightarrow A[k_2]$ for any k_1, k_2).

(B) Continue with the first recursive call of $\text{QUICKSORT}()$ (the original call $\text{QUICKSORT}(A[0..11])$ is assumed to be the 0th call of this function). Draw the state of the array every time you swap two elements.

(C) Decide which is the second recursive call of $\text{QUICKSORT}()$ and draw the state of the array every time you swap two elements. Show the end-result after this second recursive call at the very end.

7.1.2 Solution

Your answer can be simple lists of numbers (without any grid lines or additional markings). Just try to keep the lists of numbers aligned.

(A) Swaps during the 0th call:

	0	1	2	3	4	5	6	7	8	9	10	11
	13	0	23	1	8 ₂	9	29	16	8 ₁	24	6	11
after Line 12	13	0	11	1	8 ₂	9	29	16	8 ₁	24	6	23
	13	0	11	1	8 ₂	9	6	16	8 ₁	24	29	23
	13	0	11	1	8 ₂	9	6	8 ₁	16	24	29	23
	13	0	11	1	8 ₂	9	6	16	8 ₁	24	29	23
Line 13	13	0	11	1	8 ₂	9	6	8 ₁	16	24	29	23
Line 14	8 ₁	0	11	1	8 ₂	9	6	13	16	24	29	23

(B) Since this example contains two elements equal to 8, we added subscripts to them (to show clearly, where every one is being swapped). As integer numbers they are fully identical to the Quicksort algorithm. (Still, the Quicksort algorithm does redundant swaps on them.)

Swaps during the first recursive call.

	8 ₁	0	11	1	8 ₂	9	6	13	16	24	29	23
	8 ₁	0	6	1	8 ₂	9	11	13	16	24	29	23
	8 ₁	0	6	1	8 ₂	9	11	13	16	24	29	23
Line 13	8 ₁	0	6	1	8 ₂	9	11	13	16	24	29	23
Line 14	8 ₂	0	6	1	8 ₁	9	11	13	16	24	29	23

(C) Notice that the second recursive call happens within the first recursive call (sorting the left side of the left half).

Swaps during the second recursive call:

	8 ₂	0	6	1	8 ₁	9	11	13	16	24	29	23
	8 ₂	0	6	1	8 ₁	9	11	13	16	24	29	23
	8 ₂	0	6	1	8 ₁	9	11	13	16	24	29	23
Line 13	8 ₂	0	6	1	8 ₁	9	11	13	16	24	29	23
Line 14	1	0	6	8 ₂	8 ₁	9	11	13	16	24	29	23

7.2 Problems

Question 1: You are given an array:

$a + 10$	c	$a + 20$	a	$c + 5$	b	$b + 20$	$a + 15$	$b + 1$	$b + 15$	$2 \cdot c$	$b + 2$
----------	-----	----------	-----	---------	-----	----------	----------	---------	----------	-------------	---------

Here a, b, c are the last three digits of your Student ID. The pseudocode (same as in the sample) is used to sort it. Pivot is the leftmost element.

- (A) Run the initial call of `QUICKSORT($A[0..11]$)`. Draw the state of the array every time you swap two elements.
- (B) Draw the content of the array immediately *before* the second recursive call of `QUICKSORT()`. (The original call `QUICKSORT($A[0..11]$)` is assumed to be the 0th call of this function).