University of Nottingham
UK | CHINA | MALAYSIA

Sorting

Exercise 1: In-Place Quick Sort

The in-place quick sort takes an array S of size n and two indices a and b as input, and sort the subarray $S[a..b]$. The pivot is $S[b]$.

Apply in-place quick sort over the array below. Show the sorting process stey by step.

85	24	63	45	17	31	96	50

In-Place Quick Sort

```
1   /** Sort the subarray S[a..b] inclusive. */
2   private static <K> void quickSortInPlace(K[ ] S, Comparator<K> comp,
3                                                              int a, int b) {
4     if (a >= b) return;          // subarray is trivially sorted
5     int left = a;
6     int right = b-1;
7     K pivot = S[b];
8     K temp;                      // temp object used for swapping
9     while (left <= right) {
10      // scan until reaching value equal or larger than pivot (or right marker)
11      while (left <= right && comp.compare(S[left], pivot) < 0) left++;
12      // scan until reaching value equal or smaller than pivot (or left marker)
13      while (left <= right && comp.compare(S[right], pivot) > 0) right--;
14      if (left <= right) {         // indices did not strictly cross
15        // so swap values and shrink range
16        temp = S[left]; S[left] = S[right]; S[right] = temp;
17        left++; right--;
18      }
19    }
20    // put pivot into its final place (currently marked by left index)
21    temp = S[left]; S[left] = S[b]; S[b] = temp;
22    // make recursive calls
23    quickSortInPlace(S, comp, a, left - 1);
24    quickSortInPlace(S, comp, left + 1, b);
25  }
```

Code Fragment 12.6: In-place quick-sort for an array S. The entire array can be sorted as quickSortInPlace(S, comp, 0, S.length−1).

In-Place Quick Sort over $[0, 7]$

0	1	2	3	4	5	6	7
85	24	63	45	17	31	96	50
31	24	63	45	17	85	96	50
31	24	17	45	63	85	96	50
31	24	17	45	50	85	96	63

left= 4

In-Place Quick Sort over $[0, 3]$ and $[5, 7]$

0	1	2	3
31	24	17	45

5	6	7
85	96	63
63	96	85

left=3

left=5

0	1	2	3	4	5	6	7
31	24	17	45	50	63	96	85

In-Place Quick Sort over $[0, 2]$ and $[6, 7]$

0	1	2
31	24	17
17	24	31

6	7
96	85
85	96

left=0

left=6

0	1	2	3	4	5	6	7
17	24	31	45	50	63	85	96

In-Place Quick Sort over $[1, 2]$ and $[7]$

1	2
24	31

7
96

left=2

left=7

0	1	2	3	4	5	6	7
17	24	31	45	50	63	85	96