


Theory: Quicksort in Java

 22 minutes

0 / 2 problems solved

Skip this topic

Start practicing

451 users solved this topic. Latest completion was 1 day ago.

Quicksort is an efficient in-place sorting algorithm that is often faster in practice comparing with other sorting algorithms. Its time complexity is $O(n \log n)$ in the average case and $O(n^2)$ in the worst case, but for real datasets, it often works like in the average.

Quicksort can be implemented as a recursive or iterative algorithm. We will consider only the recursive version here.

§1. Implementation in Java

Below is a recursive version of quicksort implemented in Java:

```
1 public static void quickSort(int[] array, int left, int right) {
2     if (left < right) {
3
4         int pivotIndex = partition(array, left, right); // the pivot is already on
its place
5         quickSort(array, left, pivotIndex - 1); // sort the left subarray
6
7         quickSort(array, pivotIndex + 1, right); // sort the right subarray
8     }
9 }
```

The `quickSort` method takes an array of ints and a range of indexes (`left` , `right`) to sort the array between them (inclusive).

The `partition` method reorders the array and returns the index of the pivot to divide the array into two parts. The method called `swap` rearranges two elements in the array. Here is an implementation of these methods:

```
1 private static int partition(int[] array, int left, int right) {
2
3     int pivot = array[right]; // choose the rightmost element as the pivot
4     int partitionIndex = left; // the first element greater than the pivot
5
6     /* move large values into the right side of the array */
7     for (int i = left; i < right; i++) {
8         if (array[i] <= pivot) { // may be used '<' as well
9             swap(array, i, partitionIndex);
10            partitionIndex++;
11        }
12    }
13
14    swap(array, partitionIndex, right); // put the pivot on a suitable position
15
16    return partitionIndex;
17 }
18
19 private static void swap(int[] array, int i, int j) {
20
21     int temp = array[i];
22
23     array[i] = array[j];
24
25     array[j] = temp;
26 }
```

Current topic:

[Quicksort in Java](#) ...

Topic depends on:

- ✓

[Quick sort](#) ...
- ✗

[Algorithms in Java](#) ...
- ✗

[Bubble sort in Java](#) ...

Table of contents:

[↑ Quicksort in Java](#)

[§1. Implementation in Java](#)

[§2. Examples](#)

[§3. Summary](#)

[Feedback & Comments](#)

This implementation of the `partition` method is known as **Lomuto partition scheme**. It chooses the rightmost element as a pivot.

§2. Examples

Here are some examples of how to use the implemented method:

```
1  int[] array1 = { 17, 25, 11, 16, 10, 13, 22, 14 };
2
quickSort(array1, 0, array1.length - 1); // { 10, 11, 13, 14, 16, 17, 22, 25 }
3
4  int[] array2 = { 19, 18, 17, 17, 16, 15 };
5  quickSort(array2, 0, array2.length - 1); // { 15, 16, 17, 17, 18, 19 }
```

§3. Summary

In this topic, we have learned how a recursive version of quicksort can be implemented in Java. Although the implementation is simple, it is not the best solution if we are going to sort large arrays. In the worst cases, the depth of the recursion can be large and it will throw the `StackOverflowError` exception. As an exercise, you may try to implement this algorithm using loops rather than recursion.

 Report a typo

37 users liked this theory. **1** didn't like it. What about you?



Start practicing

[Comments \(3\)](#)[Hints \(0\)](#)[Useful links \(0\)](#)[Show discussion](#)