

```

1 void dualPivotQuicksort(array, leftPivotIndex, rightPivotIndex, numDivisions) {
2
3     length = rightPivotIndex - leftPivotIndex
4     if (length < maximum_threshold_for_quicksort) { // insertion sort for tiny array
5         insertionSort(array)
6         return;
7     }
8
9     oneDivison = length / numDivisons
10
11     median1 = leftPivotIndex + third
12     median2 = rightPivotIndex - third
13
14     if (median1 <= leftPivotIndex)
15         median1 = leftPivotIndex + 1
16
17     if (median2 >= rightPivotIndex)
18         median2 = rightPivotIndex - 1
19
20     if (array[median1] < array[median2]) {
21         swap(array, median1, leftPivotIndex)
22         swap(array, median2, rightPivotIndex)
23     }
24
25     else {
26         swap(array, median1, rightPivotIndex)
27         swap(array, median2, leftPivotIndex)
28     }
29
30     pivot1 = array[leftPivotIndex]
31     pivot2 = array[rightPivotIndex]
32
33     firstElementOfMiddlePartitionIndex = leftPivotIndex + 1;
34     lastElementOfMiddlePartitionIndex = rightPivotIndex - 1;

```

```

35
36 //Sorting, finally
37 for ( currentIndex = firstElementOfMiddlePartitionIndex; currentIndex <= lastElementOfMidd
38     if (array[currentIndex] < pivot1){
39         swap(array, currentIndex, firstElementOfMiddlePartitionIndex)
40         increment firstElementOfMiddlePartitionIndex
41     }
42
43     if (array[currentIndex] > pivot2) {
44         while (currentIndex < lastElementOfMiddlePartitionIndex && array[lastElementOfMidd
45             decrement lastElementOfMiddlePartitionIndex
46         }
47
48         swap(array, currentIndex, lastElementOfMiddlePartitionIndex)
49         decrement lastElementOfMiddlePartitionIndex
50
51         if (array[currentIndex] < pivot1){
52             swap(array, currentIndex, firstElementOfMiddlePartitionIndex)
53             firstElementOfMiddlePartitionIndex
54         }
55     }
56 }
57
58 if (lastElementOfMiddlePartitionIndex - firstElementOfMiddlePartitionIndex < 13) {
59     increment numDivisions
60 }
61
62 swap(array, firstElementOfMiddlePartitionIndex - 1, leftPivotIndex)
63 swap(array, lastElementOfMiddlePartitionIndex + 1, rightPivotIndex)
64
65 dualPivotQuicksort(array, leftPivotIndex, firstElementOfMiddlePartitionIndex - 2, numDivis
66 dualPivotQuicksort(array, lastElementOfMiddlePartitionIndex + 2, rightPivotIndex, numDivis
67
68
69 if (dist > len - 13 && pivot1 != pivot2) { //elements are equal

```

```

70     for ( currentIndex = firstElementOfMiddlePartitionIndex; currentIndex <= lastElementOfMiddlePartitionIndex; currentIndex++)
71         if (array[currentIndex] == pivot1) {
72             swap(array, currentIndex, firstElementOfMiddlePartitionIndex++)
73         }
74         if (array[currentIndex] == pivot2) {
75             swap(array, currentIndex, lastElementOfMiddlePartitionIndex--)
76             if (array[currentIndex] == pivot1) {
77                 swap(array, currentIndex, firstElementOfMiddlePartitionIndex++)
78             }
79         }
80     }
81 }
82
83 if (pivot1 < pivot2) {
84     dualPivotQuicksort(array, firstElementOfMiddlePartitionIndex, lastElementOfMiddlePartitionIndex)
85 }
86 }

```