```
void dualPivotQuicksort(array,  leftPivotIndex, rightPivotIndex,  numDivisions) {

    length = rightPivotIndex - leftPivotIndex
    if (length < maximum_threshold_for_quicksort) { // insertion sort for tiny array
        insertionSort(array)
        return;
    }

     oneDivison = length / numDivisons

     median1 = leftPivotIndex + third
     median2 = rightPivotIndex - third

    if (median1 <= leftPivotIndex)
        median1 = leftPivotIndex + 1

    if (median2 >= rightPivotIndex)
        median2 = rightPivotIndex - 1

    if (array[median1] < array[median2]) {
        swap(array, median1, leftPivotIndex)
        swap(array, median2, rightPivotIndex)
    }

    else {
        swap(array, median1, rightPivotIndex)
        swap(array, median2, leftPivotIndex)
    }

     pivot1 = array[leftPivotIndex]
     pivot2 = array[rightPivotIndex]

     firstElementOfMiddlePartitionIndex = leftPivotIndex + 1;
     lastElementOfMiddlePartitionIndex = rightPivotIndex - 1;

    //Sorting, finally
    for ( currentIndex = firstElementOfMiddlePartitionIndex; currentIndex <= lastElementOfMidd
        if (array[currentIndex] < pivot1){
            swap(array, currentIndex, firstElementOfMiddlePartitionIndex)
            increment firstElementOfMiddlePartitionIndex
        }

        if (array[currentIndex] > pivot2) {
            while (currentIndex < lastElementOfMiddlePartitionIndex && array[lastElementOfMidd
                decrement lastElementOfMiddlePartitionIndex
            }

            swap(array, currentIndex, lastElementOfMiddlePartitionIndex)
            decrement lastElementOfMiddlePartitionIndex

            if (array[currentIndex] < pivot1){
                swap(array, currentIndex, firstElementOfMiddlePartitionIndex)
                firstElementOfMiddlePartitionIndex
            }
        }
    }

    if (lastElementOfMiddlePartitionIndex - firstElementOfMiddlePartitionIndex < 13) {
        increment numDivisions
    }

    swap(array, firstElementOfMiddlePartitionIndex - 1, leftPivotIndex)
    swap(array, lastElementOfMiddlePartitionIndex + 1, rightPivotIndex)

    dualPivotQuicksort(array, leftPivotIndex, firstElementOfMiddlePartitionIndex - 2, numDivis
    dualPivotQuicksort(array, lastElementOfMiddlePartitionIndex + 2, rightPivotIndex, numDivis


    if (dist > len - 13 && pivot1 != pivot2) { //elements are equal
```

```
        for ( currentIndex = firstElementOfMiddlePartitionIndex; currentIndex <= lastElementOf
            if (array[currentIndex] == pivot1) {
                swap(array, currentIndex, firstElementOfMiddlePartitionIndex++)
            }
            if (array[currentIndex] == pivot2) {
                swap(array, currentIndex, lastElementOfMiddlePartitionIndex--)
                if (array[currentIndex] == pivot1) {
                    swap(array, currentIndex, firstElementOfMiddlePartitionIndex++)
                }
            }
        }
    }

    if (pivot1 < pivot2) {
        dualPivotQuicksort(array, firstElementOfMiddlePartitionIndex, lastElementOfMiddleParti
    }
}
```