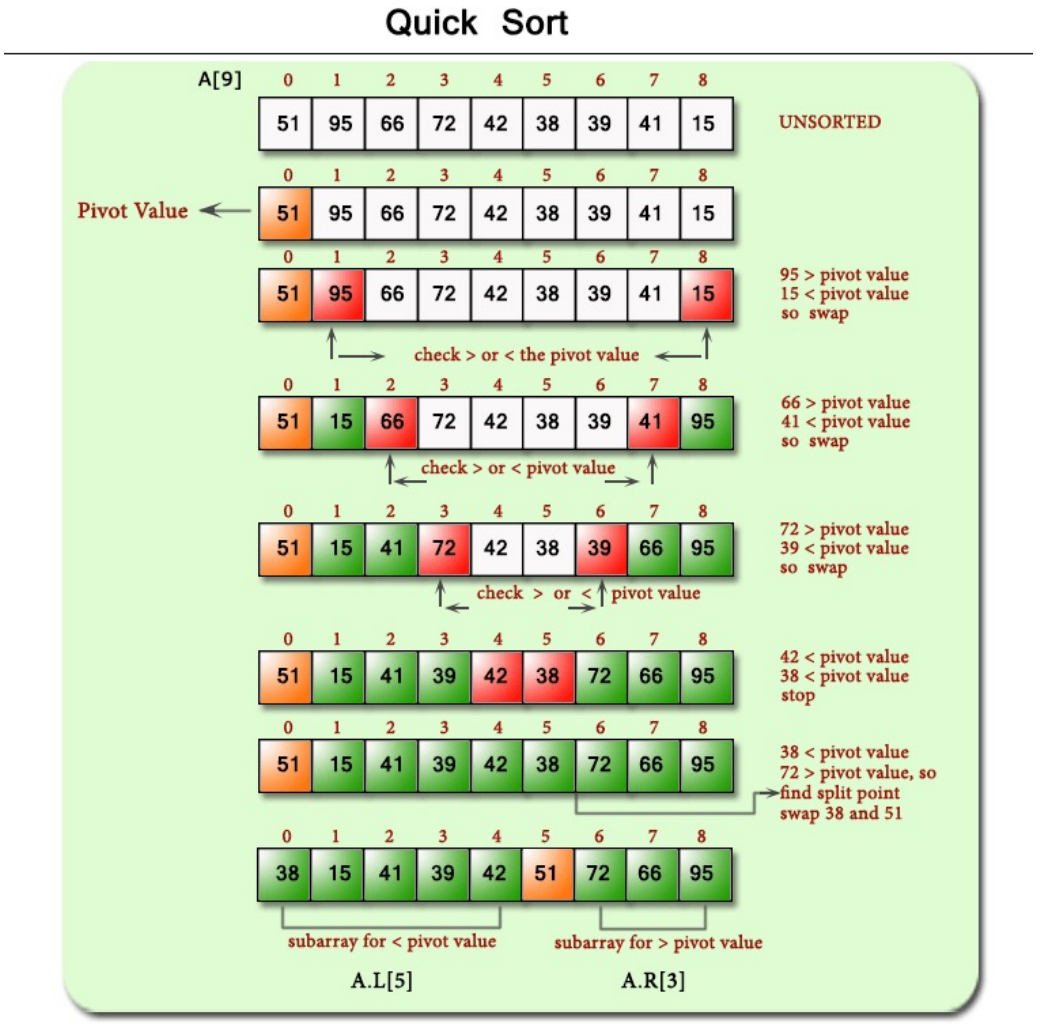
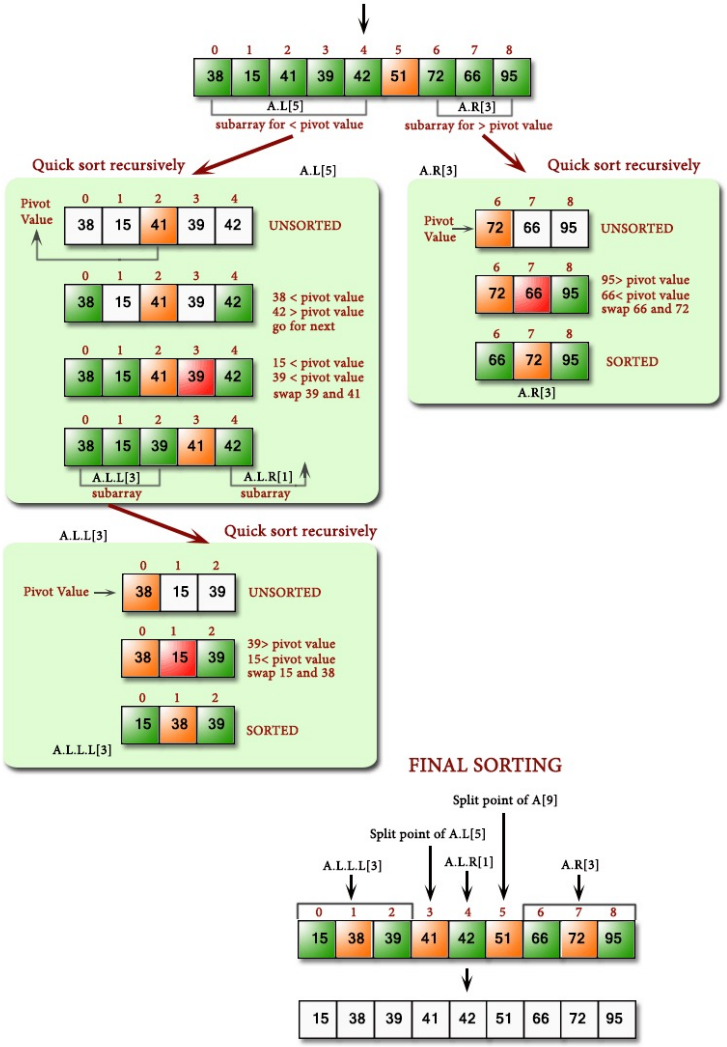
Quick Sort

Quick sort works on the principle of Divide and Conquer. Where we divide the given array with respect to a particular element, known as **'pivot'** such that the lower partition of the array is less than the pivot and upper partition elements of the array are higher than the pivot. Quick sort is also known as partition-exchange sort.

Quick Sort


Quick Sort
A[9]
0 1 2 3 4 5 6 7 8
51 95 66 72 42 38 39 41 15
UNSORTED
Pivot Value
51 95 66 72 42 38 39 41 15
51 95 66 72 42 38 39 41 15
95 > pivot value
15 < pivot value
so swap
check > or < the pivot value
51 15 66 72 42 38 39 41 95
66 > pivot value
41 < pivot value
so swap
check > or < pivot value
51 15 41 72 42 38 39 66 95
72 > pivot value
39 < pivot value
so swap
check > or < pivot value
51 15 41 39 42 38 72 66 95
42 < pivot value
38 < pivot value
stop
51 15 41 39 42 38 72 66 95
38 < pivot value
72 > pivot value, so
find split point
swap 38 and 51
38 15 41 39 42 51 72 66 95
subarray for < pivot value
subarray for > pivot value
A.L[5]
A.R[3]

Quick Sort


0 1 2 3 4 5 6 7 8
38 15 41 39 42 51 72 66 95
A.L[5]
subarray for < pivot value
A.R[3]
subarray for > pivot value
Quick sort recursively
A.L[5]
Pivot Value
UNSORTED
38 < pivot value
42 > pivot value
go for next
15 < pivot value
39 < pivot value
swap 39 and 41
A.L.L[3]
subarray
A.L.R[1]
subarray
A.R[3]
Quick sort recursively
Pivot Value
UNSORTED
95> pivot value
66< pivot value
swap 66 and 72
SORTED
A.L.L[3]
Quick sort recursively
Pivot Value
UNSORTED
39> pivot value
15< pivot value
swap 15 and 38
SORTED
A.L.L.L[3]
FINAL SORTING
Split point of A[9]
Split point of A.L[5]
A.L.L.L[3]
A.L.R[1]
A.R[3]
15 38 39 41 42 51 66 72 95
15 38 39 41 42 51 66 72 95

Algo for Quick Sort

Here **arr** is the array to be sorted, **length** is the number of elements stored in array, **lower** is having the index of first element of array, **upper** is having the index of last element of array, **mid** will store the index of middle element of array, **left** will store the index of first element of left sub-array, **right** will store the index of first element of right sub-array, **t** is the temporary array and **i** will store the index position from where element to be copied, then the algorithm is:

1. quick_sort(arr,lower, upper)

   Check, if lower < upper, then:

   a) Set pivot = lower.
   b) Set left = lower + 1.
   c) Set right = upper.
   d) Repeat while right ≥ left
      i. Repeat while left ≤ upper and arr[left] ≤ arr[pivot]
         - Set left = left + 1.
      ii. Repeat while right ≥ lower and arr[right] > arr[pivot]
         - Set right = right – 1.
      iii. check if, left < right, then:
         - Set t = arr[left].
         - Set arr[left] = arr[right].
         - Set arr[right] = t.
   e) Set t = arr[pivot].
   f) Set arr[pivot] = arr[right].
   g) Set arr[right] = t.
   h) Set pivot = right.
   i) Call quick_sort(arr, lower, pivot - 1).
   j) Call quick_sort(arr, pivot + 1, upper).

1. Exit