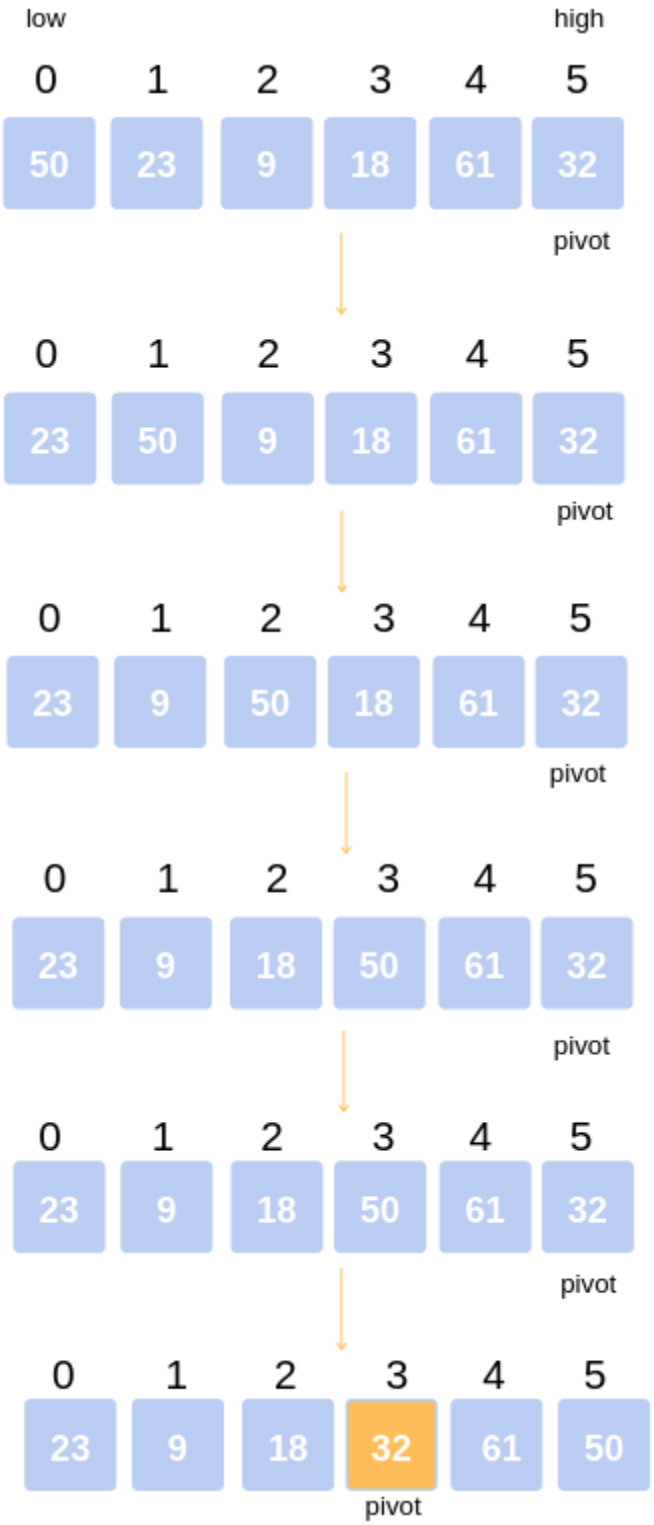
QUICK SORT:

QuickSort is one of the most efficient sorting algorithms and is based on the splitting of an array into smaller ones. The name comes from the fact that, quick sort is capable of sorting a list of data elements significantly faster than any of the common sorting algorithms.

Consider the following array: 50, 23, 9, 18, 61, 32



Step 1: Decide any value to be the pivot from the list (generally the last value). Suppose for two values “Low” and “High” corresponding to the first index, and last index.
In our case low is 0 and high is 5.

Values at low and high are 50 and 32 and Value of pivot is 32.

Hence, call for partitioning, rearranging the array in such a way that pivot (32) comes to its actual position. And to the left of the pivot, the array has all the elements less than it, and to the right greater than it.

In the partition function, we start from the first element and compare it with the pivot. Since 50 is greater than 32, we don’t make any change and move on to the next element 23.

Compare again with the pivot. Since 23 is less than 32, we swap 50 and 23. The array becomes 23, 50, 9, 18, 61, 32.

We move on to the next element 9 which is again less than pivot (32) thus swapping it with 50 makes our array as

23, 9, 50, 18, 61, 32.

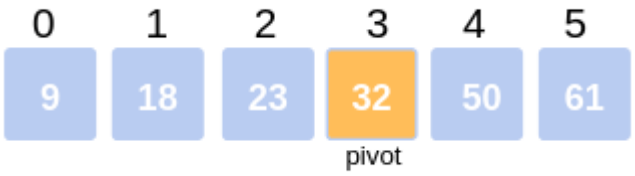
Similarly, for next element 18 which is less than 32, the array becomes

23, 9, 18, 50, 61, 32 Now 61 is greater than pivot (32), hence no changes.

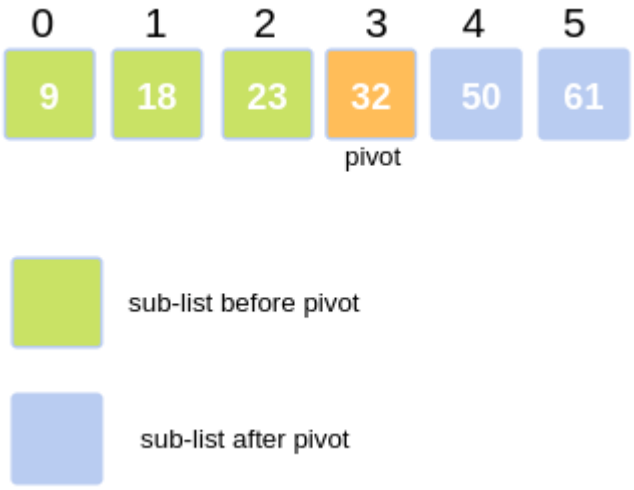
Lastly, we swap our pivot with 50 so that it comes to the correct position.

Thus the pivot (32) comes at its actual position and all elements to its left are lesser, and all elements to the right are greater than itself.

Step 2: Hence the array after the first step becomes
23, 9, 18, **32**, 61, 50



Step 3: Now the list is divided into two parts:
1. Sublist before pivot
2. Sublist after pivot



Step 4: Repeat the steps for these sublists again.
The final array thus becomes 9, 18, 23, 32, 50, 61.

TIME COMPLEXITIES FOR QUICK SORT:

1. Best: Ω(nlogn)
2. Average: θ(nlogn)
3. Worst: O(n^2)