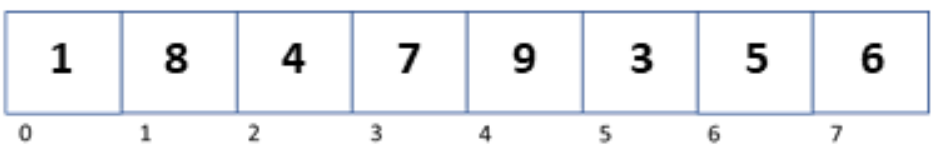
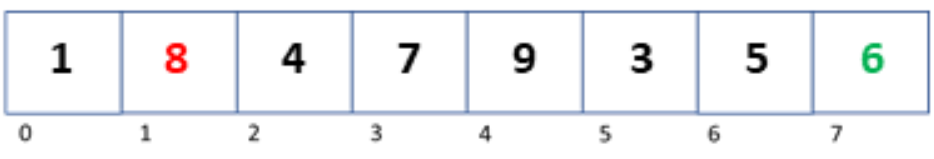
Suppose we call partition(list, left = 0, right = 7), where list is the following:



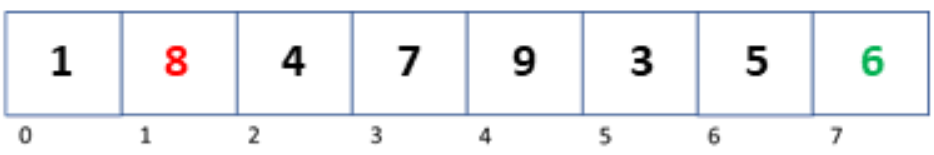
The pivot (shown in green) will be the rightmost element of the subarray we are given, while the swap offset (shown in red) is initially the leftmost element of the subarray.

We set i to 0, the leftmost index in our subarray. Then we check if the value at index i is less than or equal to the pivot, and if it is, we swap the elements at index i and the swap offset (note that this will change nothing if the swap offset is the same as index i) and increment the swap offset by 1. Then we repeat this process for each index in the array until we reach the pivot.

After checking i = 0: (Index 0 and 0 are swapped)

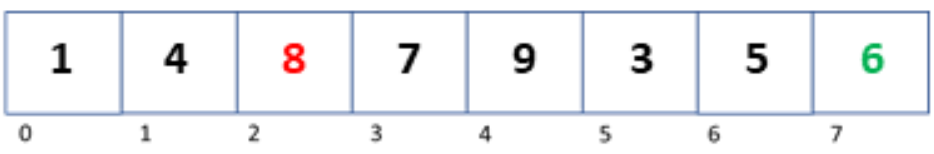


After checking i = 1:

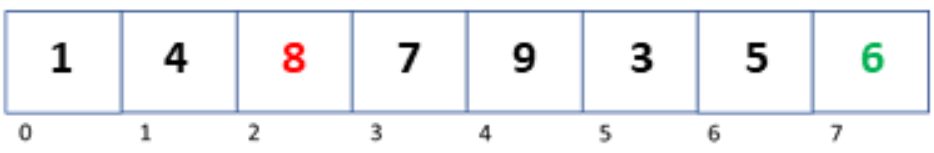


After checking i = 2: (Index 1 and 2 are swapped)

After checking i = 3:



After checking i = 4:



After checking i = 5: (Index 2 and 5 are swapped)

After checking i = 5: (Index 3 and 6 are swapped)

Finally, the pivot will be swapped with the current swap offset.

The result is that the pivot is now in the index it would be in if the subarray were fully sorted, and all elements less than the pivot have been moved to the left of the pivot while all the elements greater than the pivot have moved to the right of the pivot.

Note that the pivot is NOT necessarily the median. The index of the pivot is returned after partition() has finished with the array.

It is up to jSmallest() to determine whether or not additional calls need to be made to locate the median.