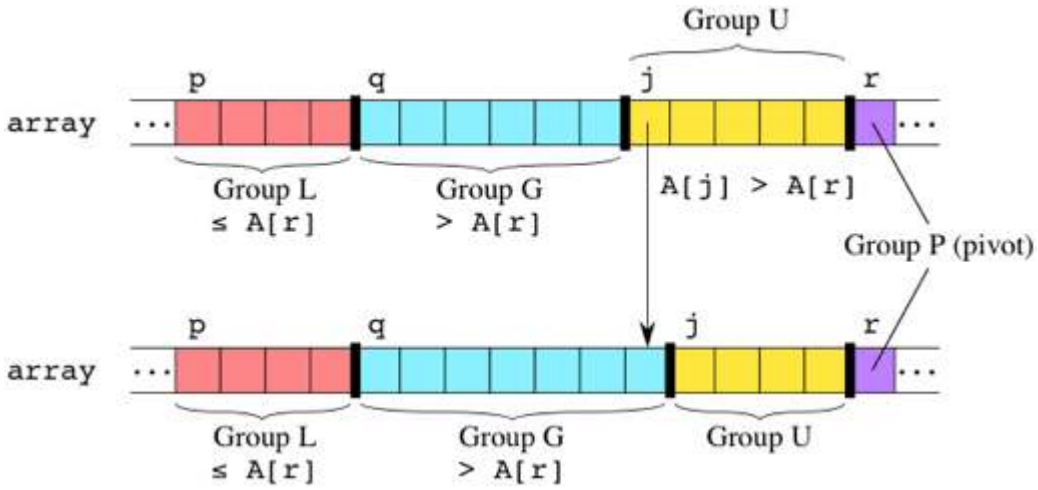
Linear-time Partitioning

The real work of quicksort happens during the divide step, which partitions subarray array[p..r] around a pivot drawn from the subarray. Although we can choose any element in the subarray as the pivot, it's easy to implement partitioning if we choose the rightmost element of the subarray, A[r], as the pivot.

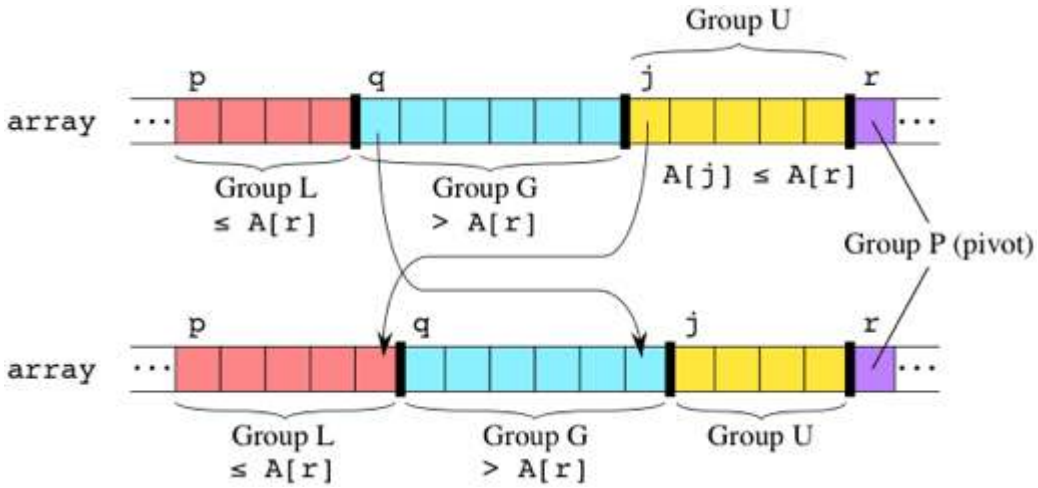
Having chosen a pivot, we partition the subarray by going through it, left to right, comparing each element with the pivot. We maintain two indices q and j into the subarray that divide it up into four groups. We use the variable name q because that index will eventually point at our pivot. We use j because it's a common counter variable name, and the variable will be discarded once we're done.

- The elements in array[p..q-1] are "group L," consisting of elements known to be less than or equal to the pivot.
- The elements in array[q..j-1] are "group G," consisting of elements known to be greater than the pivot.
- The elements in array[j..r-1] are "group U," consisting of elements whose relationship to the pivot is unknown, because they have not yet been compared.
- Finally, array[r] is "group P," the pivot.

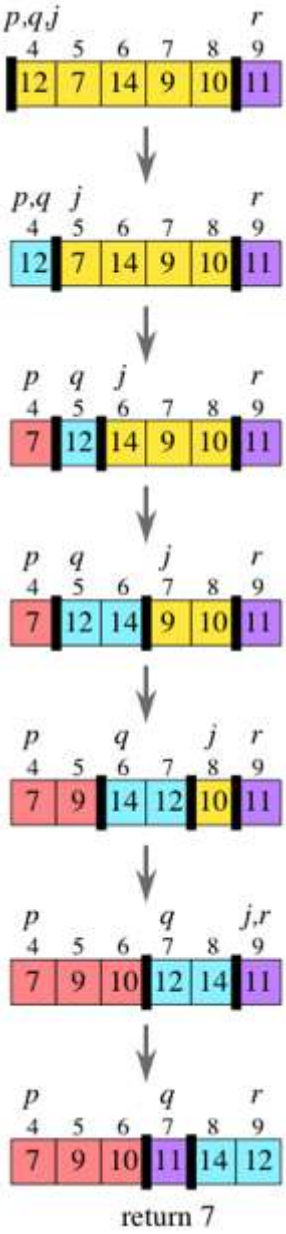
Initially, both q and j equal p. At each step, we compare A[j], the leftmost element in group U, with the pivot. If A[j] is greater than the pivot, then we just increment j, so that the line dividing groups G and U slides over one position to the right:



If instead A[j] is less than or equal to the pivot, then we swap A[j] with A[q] (the leftmost element in group G), increment j, and increment q, thereby sliding the line dividing groups L and G and the line dividing groups G and U over one position to the right:



Once we get to the pivot, group U is empty. We swap the pivot with the leftmost element in group G: swap A[r] with A[q]. This swap puts the pivot between groups L and G, and it does the right thing even if group L or group G is empty. (If group L is empty, then q never increased from its initial value of p, and so the pivot moves to the leftmost position in the subarray. If group G is empty, then q was incremented every time j was, and so once j reaches the index r of the pivot, q equals r, and the swap leaves the pivot in the rightmost position of the subarray.) The partition function that implements this idea also returns the index q where it ended up putting the pivot, so that the quicksort function, which calls it, knows where the partitions are. Here are the steps in partitioning the subarray [12, 7, 14, 9, 10, 11] in array[4..9]:



Partitioning a subarray with n elements takes **Θ(n)** time. Each element A[j] is compared once with the pivot. A[j] may or may not be swapped with A[q], q may or may not be incremented, and j is always incremented. The total number of lines executed per element of the subarray is a constant. Since the subarray has **n** elements, the time to partition is **Θ(n)**: linear-time partitioning.