Quicksort is based on the three-step process of divide-and-conquer.

- To sort the subarray $A[p \,..\, r]$:

  Divide: Partition $A[p \,..\, r]$, into two (possibly empty) subarrays $A[p \,..\, q-1]$ and $A[q+1 \,..\, r]$, such that each element in the first subarray $A[p \,..\, q-1]$ is $\leq A[q]$ and $A[q]$ is $\leq$ each element in the second subarray $A[q+1 \,..\, r]$.

  Conquer: Sort the two subarrays by recursive calls to QUICKSORT.

  Combine: No work is needed to combine the subarrays, because they are sorted in place.

- Perform the divide step by a procedure PARTITION, which returns the index q that marks the position separating the subarrays.

```
QUICKSORT($A, p, r$)
1  if $p < r$
2     then $q \leftarrow$ PARTITION($A, p, r$)
3          QUICKSORT($A, p, q-1$)
4          QUICKSORT($A, q+1, r$)
```

Initial call is QUICKSORT$(A, 1, n)$.

Partitioning

Partition subarray $A[p \,..\, r]$ by the following procedure:

```
PARTITION($A, p, r$)
1  $x \leftarrow A[r]$
2  $i \leftarrow p-1$
3  for $j \leftarrow p$ to $r-1$
4       do if $A[j] \leq x$
5             then $i \leftarrow i+1$
6                  exchange $A[i] \leftrightarrow A[j]$
7  exchange $A[i+1] \leftrightarrow A[r]$
8  return $i+1$
```

- PARTITION always selects the last element $A[r]$ in the subarray $A[p \,..\, r]$ as the pivot – the element around which to partition.
- As the procedure executes, the array is partitioned into four regions, some of which may be empty:
  Loop invariant

  1. All entries in $A[p \,..\, i]$ are $\leq$ pivot.
  2. All entries in $A[i+1 \,..\, j-1]$ are $>$ pivot.
  3. $A[r] =$ pivot.

  It's not needed as part of the loop invariant, but the fourth region is $A[j \,..\, r-1]$, whose entries have not yet been examined, and so we don't know how they compare to the pivot.

Correctness Use the loop invariant to prove correctness of PARTITION:

initializion Before the loop starts, all the conditions of the loop invariant are satisfied, because r is the pivot and the subarrays $A[p \,..\, i]$ and $A[i+1 \,..\, j-1]$ are empty.

maintainance While the loop is running, if $A[j] \leq$ pivot, then $A[j]$ and $A[i+1]$ are swapped and then i and j are incremented. If $A[j] >$ pivot, then increment only j.

termination When the loop terminates, $j = r$, so all elements in A are partitioned into one of the three cases: $A[p \,.\,.\, i] \leq$ pivot, $A[i+1 \,.\,.\, r-1] >$ pivot, and $A[r] =$ pivot.

The last two lines of Partition move the pivot element from the end of the array to between the two subarrays. This is done by swapping the pivot and the first element of the second subarray, i.e., by swapping $A[i+1]$ and $A[r]$.