

Combine: Because the subarrays are already sorted, no work is needed to combine them: the entire array $A[p \dots r]$ is now sorted.

The following procedure implements quicksort:

```

QUICKSORT( $A, p, r$ )
1  if  $p < r$ 
2       $q = \text{PARTITION}(A, p, r)$ 
3      QUICKSORT( $A, p, q - 1$ )
4      QUICKSORT( $A, q + 1, r$ )

```

To sort an entire array A , the initial call is $\text{QUICKSORT}(A, 1, A.\text{length})$.

Partitioning the array

The key to the algorithm is the **PARTITION** procedure, which rearranges the subarray $A[p \dots r]$ in place.

```

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

```

Figure 7.1 shows how **PARTITION** works on an 8-element array. **PARTITION** always selects an element $x = A[r]$ as a *pivot* element around which to partition the subarray $A[p \dots r]$. As the procedure runs, it partitions the array into four (possibly empty) regions. At the start of each iteration of the **for** loop in lines 3–6, the regions satisfy certain properties, shown in Figure 7.2. We state these properties as a loop invariant:

At the beginning of each iteration of the loop of lines 3–6, for any array index k ,

1. If $p \leq k \leq i$, then $A[k] \leq x$.
2. If $i + 1 \leq k \leq j - 1$, then $A[k] > x$.
3. If $k = r$, then $A[k] = x$.