

values known to be greater than or equal to $a[p]$. In this state, if $a[r]$ contains a value less than the pivot, step 2.1.1 rotates the values in $a[p]$, $a[p + 1]$, and $a[r]$. The result of this rotation is that $a[p]$ now contains the value less than the pivot, $a[p + 1]$ contains the pivot itself, and $a[r]$ contains a value greater than the pivot. Step 2.1.2 restores the loop invariant by incrementing p (and the loop itself increments r).

There is a subtle point here: how should the rotation behave when $r = p + 1$, i.e., no values greater than the pivot have yet been found? We could make the algorithm test for this situation and simply swap $a[p]$ with $a[r]$. But this is unnecessary: Figure 3.40 fortuitously works even in this situation (at the expense of a redundant copy).

Figure 3.43 illustrates the algorithm's behavior when partitioning the same array as in Figure 3.40. Figure 3.43 may be seen as an expansion of the first step in Figure 3.40.

Program 3.44 shows implementations of the quick-sort and partitioning algorithms as Java methods.

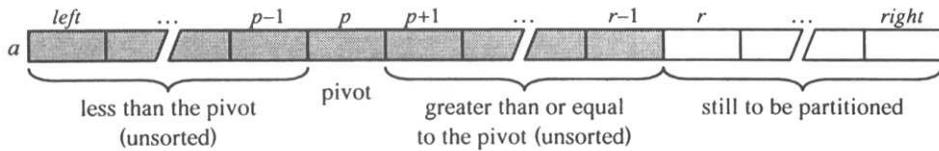


Figure 3.42 Loop invariant for the array partitioning algorithm.

		0	1	2	3	4	5	6	7	8
Initially:	a	fox	cow	pig	cat	rat	lion	tiger	goat	dog
		$p=0$	$r=1$	2	3	4	5	6	7	8
Just before step 2:	a	fox	cow	pig	cat	rat	lion	tiger	goat	dog
		0	$p=1$	$r=2$	3	4	5	6	7	8
After 1 iteration of step 2:	a	cow	fox	pig	cat	rat	lion	tiger	goat	dog
		0	$p=1$	2	$r=3$	4	5	6	7	8
After 2 iterations of step 2:	a	cow	fox	pig	cat	rat	lion	tiger	goat	dog
		0	1	$p=2$	3	$r=4$	5	6	7	8
After 3 iterations of step 2:	a	cow	cat	fox	pig	rat	lion	tiger	goat	dog
		0	1	$p=2$	3	4	$r=5$	6	7	8
After 4 iterations of step 2:	a	cow	cat	fox	pig	rat	lion	tiger	goat	dog
		0	1	$p=2$	3	4	5	$r=6$	7	8
After 5 iterations of step 2:	a	cow	cat	fox	pig	rat	lion	tiger	goat	dog
		0	1	$p=2$	3	4	5	6	$r=7$	8
After 6 iterations of step 2:	a	cow	cat	fox	pig	rat	lion	tiger	goat	dog
		0	1	$p=2$	3	4	5	6	7	$r=8$
After 7 iterations of step 2:	a	cow	cat	fox	pig	rat	lion	tiger	goat	dog
		0	1	2	$p=3$	4	5	6	7	8
After 8 iterations of step 2:	a	cow	cat	dog	fox	rat	lion	tiger	goat	pig

Figure 3.43 Illustration of the array partitioning algorithm.