



Figure 3.28 Loop invariant for the array selection sort algorithm.

		$l=0$	1	2	3	4	5	6	7	8
Initially:	a	fox	cow	pig	cat	rat	lion	tiger	goat	dog
		0	$l=1$	2	3	4	5	6	7	8
After 1 iteration:	a	cat	cow	pig	fox	rat	lion	tiger	goat	dog
		0	1	$l=2$	3	4	5	6	7	8
After 2 iterations:	a	cat	cow	pig	fox	rat	lion	tiger	goat	dog
		0	1	2	$l=3$	4	5	6	7	8
After 3 iterations:	a	cat	cow	dog	fox	rat	lion	tiger	goat	pig
		0	1	2	3	$l=4$	5	6	7	8
After 4 iterations:	a	cat	cow	dog	fox	rat	lion	tiger	goat	pig
		0	1	2	3	4	$l=5$	6	7	8
After 5 iterations:	a	cat	cow	dog	fox	goat	lion	tiger	rat	pig
		0	1	2	3	4	5	$l=6$	7	8
After 6 iterations:	a	cat	cow	dog	fox	goat	lion	tiger	rat	pig
		0	1	2	3	4	5	6	$l=7$	8
After 7 iterations:	a	cat	cow	dog	fox	goat	lion	pig	rat	tiger
		0	1	2	3	4	5	6	7	$l=8$
After 8 iterations:	a	cat	cow	dog	fox	goat	lion	pig	rat	tiger

Figure 3.29 Illustration of the array selection sort algorithm.

Figure 3.29 illustrates the selection sort algorithm's behavior as it sorts an array of words. Note that the unsorted subarray shrinks steadily until it is reduced to a single component.

Let us analyze the selection sort algorithm's time efficiency. The characteristic operations in a sorting algorithm are usually held to be comparisons. Let $n = right - left + 1$ be the length of $a[left \dots right]$. It is easy to see that step 1.1 must perform $right - l$ comparisons. The main loop is performed repeatedly, with l taking the values $left, left + 1, \dots, right - 2, right - 1$. Summing these terms we get:

$$\begin{aligned}
 \text{No. of comparisons} &= (right - left) + (right - left - 1) + \dots + 2 + 1 \\
 &= (n - 1) + (n - 2) + \dots + 2 + 1 \\
 &= (n^2 - n)/2
 \end{aligned}
 \tag{3.8}$$