

blems separately, and combine their answers. The divide-and-conquer strategy is not suitable for all problems, but it does work very well for sorting.

If we are required to sort an array, we can divide the array into two subarrays of about equal length, sort each subarray separately, and finally merge the two subarrays. This is the basis of the *merge-sort* algorithm, which is shown as Algorithm 3.35. Step 1.2 sorts the left subarray; this step could use any sorting algorithm, but it might as well call the merge-sort algorithm recursively. Step 1.3 similarly sorts the right subarray. Step 1.4 then merges the two subarrays into an auxiliary array; this step could use Algorithm 3.23. Finally, step 1.5 overwrites the original array.

As usual with a recursive algorithm, we must ensure that we have at least one easy (non-recursive) case as well as at least one hard (recursive) case. In the merge-sort algorithm, the easy case is $left \geq right$, when there is no sorting to do at all; the harder case is $left < right$, when there are at least two values to be sorted.

Figure 3.36 illustrates the merge-sort algorithm's behavior as it sorts an array of words.

Let us analyze the merge-sort algorithm's time efficiency. Let $n = right - left + 1$ be the length of $a[left...right]$, and let $comps(n)$ be the number of comparisons required to sort n components. Step 1.1 divides the array into two subarrays each of about $n/2$ components. Thus step 1.2 will have about $n/2$ components to sort, performing $comps(n/2)$ comparisons. Step 1.3 will likewise perform $comps(n/2)$ comparisons. We know from Section 3.5 that the merge in step 1.4 will perform about n comparisons.

To sort $a[left...right]$:

1. If $left < right$:
 - 1.1. Let m be an integer about midway between $left$ and $right$ (such that $left \leq m < right$).
 - 1.2. Sort $a[left...m]$.
 - 1.3. Sort $a[m+1...right]$.
 - 1.4. Merge $a[left...m]$ and $a[m+1...right]$ into an auxiliary array b .
 - 1.5. Copy all components of b into $a[left...right]$.
2. Terminate.

Algorithm 3.35 Array merge-sort algorithm.

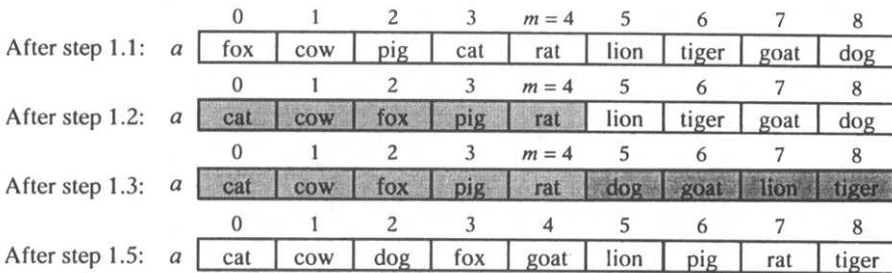


Figure 3.36 Illustration of the array merge-sort algorithm.