

Let us analyze the insertion algorithm's time efficiency. The characteristic operations are copies. Let $n = right - left + 1$ be the length of $a[left...right]$. Step 1 could copy any number from 0 through $n - 1$ components (depending on the value of ins). On average, step 1 performs $(n - 1)/2$ copies. Step 2 performs one further copy. In total:

$$\text{Average no. of copies} = (n - 1)/2 + 1 = (n + 1)/2 \quad (3.1)$$

The fastest-growing term is $n/2$, so the insertion algorithm has time complexity $O(n)$.

Program 3.8 shows how the insertion algorithm would be implemented in Java. The algorithm itself works for any array, but the Java code assumes an array of objects. It must be modified slightly if required to handle an array of primitive values (see Exercise 3.5). Note also that step 1 requires careful coding (see Exercise 3.6).

3.3 Deletion

Suppose that we are required to delete the value at a given position in an array. For instance, we might be required to delete the value at position 3 in the array $a[0...8]$. This entails copying the values in components $a[4...8]$ into their respective left neighbors $a[3...7]$. The rightmost component $a[8]$ will become unoccupied.

Algorithm 3.9 deletes the value at position del of the (sub)array $a[left...right]$. Figure 3.10 illustrates the algorithm's behavior on an array of words.

Let us analyze the deletion algorithm's time efficiency. The characteristic operations are again copies. Let $n = right - left + 1$ be the length of $a[left...right]$. Step 1 could copy any number from 0 through $n - 1$ components (depending on the value of del), therefore:

$$\text{Average no. of copies} = (n - 1)/2 \quad (3.2)$$

The fastest-growing term is $n/2$, so the deletion algorithm has time complexity $O(n)$.

Program 3.11 shows how the deletion algorithm would be implemented in Java.

To delete the value at position del of $a[left...right]$ (where $left \leq del \leq right$):

1. Copy $a[del+1...right]$ into $a[del...right-1]$.
2. Make $a[right]$ unoccupied.
3. Terminate.

Algorithm 3.9 Array deletion algorithm.

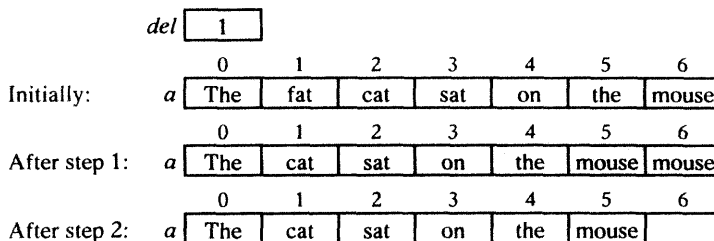


Figure 3.10 Illustration of the array deletion algorithm.