

## Summary

In this chapter:

- We have reviewed the general properties of arrays, and their specific properties in Java.
- We have studied and analyzed array insertion and deletion algorithms.
- We have studied and analyzed the array linear search and binary search algorithms. We found that binary search is significantly faster than linear search.
- We have studied and analyzed the array merging algorithm.
- We have studied and analyzed the array selection sort, insertion sort, merge-sort, and quick-sort algorithms. We found that merge-sort and quick-sort are significantly faster than the others, although quick-sort's performance is variable.

## Exercises

- 3.1 (This exercise is drill for readers unfamiliar with arrays.)  
 Consider an integer array  $a[\textit{left}..\textit{right}]$ , where  $\textit{right} \geq \textit{left}$ .
- Write an algorithm to return the greatest integer in the array.
  - Write an algorithm to return the position of the greatest integer in the array.
  - Write an algorithm to sum the integers in the array.
  - Write an algorithm to count the number of odd and even integers in the array.
  - Write an algorithm to reverse the order of the integers in the array.
- 3.2 Write an algorithm to test whether an array  $a[\textit{left}..\textit{right}]$  is sorted in ascending order. In terms of the number of comparisons required, determine the time efficiency of your algorithm: in the best case; in the worst case; and on average.  
 Implement your algorithm as a Java method, assuming that the array elements are `Comparable` objects.
- 3.3 A *palindrome* is a sequence that is identical to the reverse of itself. For example, the sequence «'m', 'a', 'd', 'a', 'm'» is a palindrome.  
 Write an algorithm to test whether a character array  $a[\textit{left}..\textit{right}]$  is a palindrome. What are the time efficiency and space efficiency of your algorithm?  
 Implement your algorithm as a Java method.
- 3.4 Modify the algorithm of Exercise 3.3 to ignore spaces and punctuation. For example, the sentence "Madam, I'm Adam." is a palindrome if we ignore spaces and punctuation.
- 3.5 Modify Program 3.8 to insert a value `val` of type `double` into an array `a` of type `double [ ]`. Repeat this exercise for other Java primitive types: `int` and `char`.  
 How many methods would you need to cover all possible types in Java?
- 3.6 Consider Algorithm 3.6 and its implementation in Program 3.8. What would go wrong if step 1 was instead implemented by a `for` statement that scanned *from left to right* copying each element?  
 Can you write an algorithm that scans from left to right, but does not have this error? What are its space efficiency and time efficiency?