

3.12 Hand-test the merging algorithm (Algorithm 3.23) to merge the following arrays of names:

James, Tolstoy, Wells
Curie, Einstein, Kelvin, Maxwell

How many name comparisons are required?

3.13 A simple way to represent a *set* of words is by a sorted array of words with no duplicates. You are given two sets of words, s_1 and s_2 , represented in this way. By modifying the merging algorithm (Algorithm 3.23), devise algorithms for the following problems:

- Compute the *union* of s_1 and s_2 . The union is the set of those words found in s_1 or s_2 or both.
- Compute the *intersection* of s_1 and s_2 . The intersection is the set of those words found in both s_1 and s_2 .

3.14 Consider the selection sort algorithm given in Algorithm 3.27.

- Hand-test this algorithm with the following array of words:

red, orange, yellow, green, blue, indigo, violet

How many word comparisons and movements are required?

- Repeat with the insertion sort algorithm given in Algorithm 3.31.

3.15 Consider the problem of reading a file of (unsorted) values into an array, where the array must be sorted. There are n values in the file.

- Write an algorithm to read all of the unsorted values into the array, and then sort the array using the selection sort algorithm given in Algorithm 3.27. What is the time efficiency of your algorithm?
- Write an algorithm to read each value in turn, and insert it into a sorted array (initially empty). What is the time efficiency of your algorithm? How does this compare with your answer to part (a)?
- Implement your algorithms from parts (a) and (b), and compare them by timing their execution on files with a range of sizes.

3.16 The Dutch national flag problem is as follows. You are given an array of colors (reds, whites, and blues), in no particular order. Sort them into the order of the Dutch national flag (reds followed by whites followed by blues).

- Devise an efficient algorithm to solve this problem.
- What is your algorithm's time complexity?

3.17 You are given two unsorted arrays of values. You are required to obtain a sorted array containing all these values. Suggest *two* different ways of achieving this. Compare their time efficiency. (*Note:* Assume that suitable merging and sorting algorithms are already available.)

3.18 Devise an algorithm to solve the following problem. Given an array $a[0..n-1]$, and a shorter array $b[0..m-1]$, find the position of the leftmost subarray of a whose elements equal (pairwise) all the elements of b . In other words, if the answer is p , then $a[p]$ must equal $b[0]$, ..., and $a[p+m-1]$ must equal $b[m-1]$.