20

Searching and Sorting

Objectives

In this chapter you'll:

- Search for a given value in an `array` using linear search and binary search.
- Sort an `array` using insertion sort, selection sort and the recursive merge sort algorithms.
- Use Big O notation to express the efficiency of searching and sorting algorithms and to compare their performance.
- Understand the nature of algorithms of constant, linear and quadratic runtime.

Self-Review Exercises

20.1 Fill in the blanks in each of the following statements:

a) A selection sort application would take approximately ________ times as long to run on a 128-element `array` as on a 32-element `array`.

ANS: 16, because an $O(n^2)$ algorithm takes 16 times as long to sort four times as much information.

b) The efficiency of merge sort is ________.

ANS: $O(n \log n)$.

20.2 What key aspect of both the binary search and the merge sort accounts for the logarithmic portion of their respective Big Os?

ANS: Both of these algorithms incorporate "halving"—somehow reducing something by half. The binary search eliminates from consideration half of the `array` after each comparison. The merge sort splits the `array` in half each time it's called.

20.3 In what sense is the insertion sort superior to the merge sort? In what sense is the merge sort superior to the insertion sort?

ANS: The insertion sort is easier to understand and to implement than the merge sort. The merge sort is far more efficient ($O(n \log n)$) than the insertion sort ($O(n^2)$).

20.4 In the text, we say that after the merge sort splits the `array` into two sub-`array`s, it then sorts these two sub-`array`s and merges them. Why might someone be puzzled by our statement that "it then sorts these two sub-`array`s"?

ANS: In a sense, it does not really sort these two sub-`array`s. It simply keeps splitting the original `array` in half until it provides a one-element sub-`array`, which is, of course, sorted. It then builds up the original two sub-`array`s by merging these one-element `array`s to form larger sub-`array`s, which are then merged, and so on.

Exercises

NOTE: Solutions to the programming exercises are located in the `ch20solutions` *folder.*