

Algorithms
Homework#1

Exercises

2.1-1

Using Figure 2.2 as a model, illustrate the operation of INSERTION-SORT on the array $A = \langle 31, 41, 59, 26, 41, 58 \rangle$.

2.1-2

Rewrite the INSERTION-SORT procedure to sort into nonincreasing instead of non-decreasing order.

2.1-3

Consider the *searching problem*:

Input: A sequence of n numbers $A = \langle a_1, a_2, \dots, a_n \rangle$ and a value v .

Output: An index i such that $v = A[i]$ or the special value NIL if v does not appear in A .

Write pseudocode for *linear search*, which scans through the sequence, looking for v .

2.2-1

Express the function $n^3/1000 - 100n^2 - 100n + 3$ in terms of Θ -notation.

2.2-2

Consider sorting n numbers stored in array A by first finding the smallest element of A and exchanging it with the element in $A[1]$. Then find the second smallest element of A , and exchange it with $A[2]$. Continue in this manner for the first $n - 1$ elements of A . Write pseudocode for this algorithm, which is known as *selection sort*. What loop invariant does this algorithm maintain? Why does it need to run for only the first $n - 1$ elements, rather than for all n elements? Give the best-case and worst-case running times of selection sort in Θ -notation.

2.3-5

Referring back to the searching problem (see Exercise 2.1-3), observe that if the sequence A is sorted, we can check the midpoint of the sequence against v and eliminate half of the sequence from further consideration. *Binary search* is an algorithm that repeats this procedure, halving the size of the remaining portion of the sequence each time. Write pseudocode, either iterative or recursive, for binary search. Argue that the worst-case running time of binary search is $\Theta(\lg n)$.