Tutorial 1 DAA

1.Illustrate the operation of INSERTION SORT on the array A =<31, 41, 59, 26, 41, 58>.

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

3. Consider the searching problem:
Input: A sequence of n numbers A =<a1 , a 2 ; : : : ; a n > 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. Using a loop invariant, prove that your algorithm is correct. Make sure that your loop invariant fulfills the three necessary properties.

4. Consider the problem of adding two n-bit binary integers, stored in two n-element arrays A and B. The sum of the two integers should be stored in binary form in an (n + 1)element array C . State the problem formally and write pseudocode for adding the two integers.

5.

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

6.

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.

7.

Consider linear search again (see Exercise 2.1-3). How many elements of the input sequence need to be checked on the average, assuming that the element being searched for is equally likely to be any element in the array? How about in the worst case? What are the average-case and worst-case running times of linear search in Θ-notation? Justify your answers.

8.

How can we modify almost any algorithm to have a good best-case running time?