**Homework 1**

**ESE 344 [ Spring 2025 ]**

**Prof. Alex Doboli**

**Due date: 02/17/2025 midnight**

**Submit all exercises using Github**

1. **Exercise 2-3 in the textbook**

Correctness of Horner’s rule

You are given the coefﬁcients a0; a1; a2; … ; an of a polynomial

P(x) = ∑nk=0 ak xk = a0 + a1 x1 + a2 x2 + … an xn;

and you want to evaluate this polynomial for a given value of x.

Horner’s rule says to evaluate the polynomial according to this parenthesization:

P(x) = a0 + x (a1 + x (a2 + x (a3 … + x (an-1 + an x)));

The procedure HORNER implements Horner’s rule to evaluate P(x), given the coefﬁcients a0; a1; a2; … ; an in an array A[0:n] and the value of x .

HORNER (A, n, x)

p = 0;

for i = n downto 0

p = A[i] + x \* p

return p

a. In terms of O notation, what is the running time of this procedure?

b. Write pseudocode to implement the naive polynomial-evaluation algorithm that computes each term of the polynomial from scratch. What is the running time of this algorithm? How does it compare with HORNER?

c. Consider the following loop invariant for the procedure HORNER:

At the start of each iteration of the for loop of lines 2–3,

P(x) = ∑n-(i+1)k=0 A[k+i+1] xk

Interpret a summation with no terms as equaling 0. Following the structure of the loop-invariant proof presented in this chapter, use this loop invariant to show that, at termination, P(x) = ∑nk=0 A[k] xk.

1. **Exercise 2-4 in the textbook**

Let A[1 …n] be an array of n distinct numbers. If i < j and A[i] > A[j], then the pair (i, j) called an inversion of A.

a. List the ﬁve inversions of the array [ 2; 3; 8; 6; 1 ].

b. What array with elements from the set {1; 2; … ; n} has the most inversions? How many does it have?

c. What is the relationship between the running time of insertion sort and the number of inversions in the input array? Justify your answer.

d. Give an algorithm that determines the number of inversions in any permutation on n elements in O [n lg n] worst-case time. (Hint: Modify merge sort.)

1. **Coding exercise [ Credit: codeleet ]**

Devise C++ code to solve the following exercise.

Given two arrays arr1 and arr2, the elements of arr2 are distinct, and all elements in arr2 are also in arr1.

Sort the elements of arr1 such that the relative ordering of items in arr1 are the same as in arr2. Elements that do not appear in arr2 should be placed at the end of arr1 in **ascending** order.

**Example 1:**

**Input:** arr1 = [2,3,1,3,2,4,6,7,9,2,19], arr2 = [2,1,4,3,9,6]

**Output:** [2,2,2,1,4,3,3,9,6,7,19]

**Example 2:**

**Input:** arr1 = [28,6,22,8,44,17], arr2 = [22,28,8,6]

**Output:** [22,28,8,6,17,44]