HW1 – Intro to Algorithms  
John Carroll

From Chapter 2 of the textbook, do the following **Exercises**(exercises are at the end of each section)

**1) 2.2-1  
Express the function n3/1000 - 100n2 - 100n + 3 in terms of ‚** Θ**-notation.  
Ans:** Θ(n3)

**2) 2.2-3  
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.  
Ans:**   
The number of elements in the average case would be (n+1)/2 since we know that it is linear and that in the worst case the number of elements would be n+1 for the number of elements checked.

A linear search of an array of size n will require n elements to be examined in the worst case, so worst case would be Θ(n). The average of an array of size n will require n/2 elements to be examined in the average case; however, after excluding lower order variables, leading coefficients and constants, the average case is also deduced to be Θ(n).

**3) 2.3-1  
Using Figure 2.4 as a model, illustrate the operation of merge sort on the array A = {3, 41, 52, 26, 38, 57, 9, 49).  
Ans:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **3** | **9** | **26** | **38** | **41** | **49** | **52** | **57** |

|  |  |  |  |
| --- | --- | --- | --- |
| **9** | **38** | **49** | **57** |

|  |  |  |  |
| --- | --- | --- | --- |
| **3** | **26** | **41** | **52** |

|  |  |
| --- | --- |
| **3** | **41** |

|  |  |
| --- | --- |
| **52** | **26** |

|  |  |
| --- | --- |
| **38** | **57** |

|  |  |
| --- | --- |
| **9** | **49** |

|  |
| --- |
| **3** |

|  |
| --- |
| **41** |

|  |
| --- |
| **52** |

|  |
| --- |
| **26** |

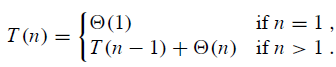
|  |
| --- |
| **38** |

|  |
| --- |
| **57** |

|  |
| --- |
| **9** |

|  |
| --- |
| **49** |

**4) 2.3-4  
We can express insertion sort as a recursive procedure as follows. In order to sort A[1..n-1],we recursively sort A[1..n-1] and then insert A[n]into the sorted array A[1..n-1]. Write a recurrence for the running time of this recursive version of insertion sort.  
Ans:**

It takes Θ (n) time in the worst case to insert A[n] into the sorted array A[1 . . n − 1], thus we get the recurrence:  
  
The solution to this recurrence is *T(n) = Θ(n2).*

And the following **Problems**(at the end of the chapter)

**5) 2.2-d   
What is the worst-case running time of bubblesort? How does it compare to the running time of insertion sort?  
Ans:** Bubblesort has Θ(n2) running time. Insertion sort has the same worst-case running time.

**6) 2.3-a  
Ans:** This code fragment has Θ(n) because the condition for the loop is from n to 0

|  |  |  |
| --- | --- | --- |
|  | Cost | Time |
| 1) | C1 | n |
| 2) | C2 |  |
| 3) | 0 |  |
| 4) | C4 | n - 1 |

**7) 2.3-b  
Ans:**

y = 0 **for** i = n **downto** 0  
 z = 1  
 **for** j = n **downto** 1  
 z = x·z  
 y = y + ai·z  
return y

T(n) = C1n + C2(+ C4(n - 1) = n(n – 1)  
T(n) = C1n + C2n2 – C2n + C4(n – 1)  
T(n) = C2n2  = Θ(n2)

The running time of this algorithm is Θ(n2). This is because the last execution of the inner-loop must iterate n times and the outer-loop will always iterate n times. Horner’s is more efficient because it has only one loop whereas this one has two.