# Homework 1: Analysis and design of algorithms

1. (a) What is meant by the **time complexity** of an algorithm? [2]

The time taken for an algorithm to run on average as the problem gets larger

(b) The time complexity of an algorithm can be expressed in Big-O notation. Why is this measure of interest to computer scientists? [2]

Computer scientists need a way to measure how fast an algorithm is and how long it takes to run but with actual time like seconds for example, the time will depend on how large the problem is, like an algorithm may run faster if the processor is faster or the problem is slower, so with big o notation, an average time can be measured of how fast an algorithm itself is just based on how many instructions or operations are carried out.

(c) Time complexity can be roughly calculated by counting the number of steps involved in processing n items.

If the number of steps in an algorithm is computed as 3n3 + n2 + 10n + 6, what is the Big-0 notation that expresses the time complexity of the algorithm? Explain how you arrive at your answer. [3]

O(n^3) because as a problema gets larger, or the input of n gets greater, the non dominant terms and constants become less and less relevant because they will end up being so small to n^3 which is the dominant term to the point they are negligible, therefore 6 is a constant so it can be ignored, 10n is a non dominant term so ignored, and so is n^2, the coefficient of n^3 is 3 which is also negligible when input becomes large so the ignored, therefore final term is just n^3 which is the only one that’s not negligible so O(n^3)

2. (a) Arrange the following in ascending order of time complexity:

O(n2) O(log n) O(n!) O(n) O(2n)

O(logn), O(n), O(n^2), O(2^n). O(n!) [3]

(b) For each one, state whether it is constant time, exponential time, logarithmic time, polynomial time or linear time. [5]

1. O(n2)

polynomial

1. O(log n)

logarithmic

1. O(n!)

factorial

1. O(n)

Linear

1. O(2n)

Exponential

3. The algorithm for a bubble sort of n items is given below.

for i = 1 to n-1

for j = 1 to (n-i)

if numbers [j] > numbers[j+1]

# Swap the names in the array

temp = numbers[j]

numbers[count] = numbers[count+1]

numbers[count+1] = temp

endif

next j

next i

(a) Calculate the number of steps needed to sort a list of 5 items.   
Count the IF statement in the inner loop as one statement. [2]

For I = 1 to 4 (4 times), for j = 1 to 4 (4 times), 1

4 \* 1 = 4, 4 \* 4 = 16

16 steps

(b) Simplify the number of steps to sort a list of n items, given that

1 + 2 +… + n-1 = n(n-1)/2 [2]

n^2

(c) What is the time complexity of the algorithm expressed in Big-O notation?

O(n^2) [1]

Total 20 marks