7SENG010W Data Structures & Algorithms
Week 1 Tutorial Exercises Solutions

These exercises cover: Abstract Data Types (ADTs), Big-O Complexity, Timing an Algorithm

Exercise 1 Solution
See the `TubeStation` and `Testing` class code for a sample solution.

Exercise 2 Solution
Complete the table for the quadratic growth rate running time equation $T(N) = 2N^2 + 3N + 4$. This gives an idea why when calculating the Big-O for a T(N) that only the value of the *dominant term*, i.e. N^2, determines its corder of complexity.

Values of N	**$T(N) = 2N^2 + 3N + 4$**			
	$2N^2$	**3N**	**4**	**T(N)**
1000	2,000,000	3,000	4	2,003,004
2000	8,000,000	6,000	4	8,006,004
4000	32,000,000	12,000	4	32,012,004
8,000	128,000,000	24,000	4	128,024,004
16,000	512,000,000	48,000	4	512,048,004

Note that when N = 16,000 that the percentage of T(16000) accounted for by the $2N^2$ term is (512,000,000 / 512,048,004) * 100 = 99.99%.

So this illustrates why when calculating the Big-O for an algorithm's T(N), it ignores all but the "dominant term" in the T(N) function.

Exercise 3 Solution
Completed the B-g-O values table.

Big-O	Values of N				
	20	**50**	**100**	**1000**	**100,000**
O(1)	1	1	1	1	1
O(N)	20	50	100	1000	100,000
$O(N^2)$	400	2500	10,000	1,000,000	10,000,000,000
$O(N^3)$	8000	125,000	1,000,000	1,000,000,000	1×10^{15}
$O(\log_2(N))$	5 (4.3219280948874)	6	7	10	17
$O(N \log_2(N))$	100	300	700	10,000	1,700,000
$O(2^N)$	1048576	1.125899907×10^{15}	1.2676506×10^{30}	$1.071508607\times10^{301}$	Largest N = 1342: 2^{1342} = $9.599623077\times10^{403}$
O(N!)	2.432902008×10^{18}	3.04140932×10^{64}	$9.332621544\times10^{157}$	Largest N = 212: 212! = $4.733702183\times10^{402}$	Too big!

Note that when we deal with order of complexity expressions that involve $\log_2(N)$ the result is very rarely a whole number, so the standard practice is to take the smallest whole number that is greater than the fractional value, this is called the "ceiling" in maths e.g. ceiling(4.3219280948874) = 5.

The comparison given for the rough age of the Universe as approximately 13.5 billion years, & in seconds that is: $(13.5 \times 10^9) \times (365\times24\times3600) = 4.25736\times10^{17}$.)

Again for comparison if for a particular algorithm it has an order of complexity of $O(2^N)$, then if the algorithm was applied to 59 data items, i.e. N = 59 then its expected "execution time" in time units would be:

$O(2^N)$: 2^59 = 5.764607523×10^{17}

which is longer than the age of the Universe!

Exercise 4 Solution
See the linear searching program class code for a sample solution.

On my laptop with a quite good intel i7 processor it too well into the 10s of millions to get even close to a second.