Compsci 220	The University of Auckland
Assignment 1	
Due: Monday, August 8th, by 11:59pm.	*Semester 2, 2022*

Do not give students "half a point" at any time — this may cause students who do not get such points to complain that their marker is meaner!

The solutions to each part are boxed underneath; at the bottom of the box is the point allocation.

1. (10 marks) Suppose you are given 3 algorithms A_1, A_2 and A_3 solving the same problem. You know that in the worst case the running times are

$$T_1(n) = n5^n, \qquad T_2(n) = 2^{10}n^2 + n^3, \qquad T_3(n) = 10^5 \log_{10}(n^n)$$

(a) Which algorithm is the fastest for very large inputs? Which algorithm is the slowest for very large inputs? (Justify your answer.)

Ideally students will solve this with limits (as below). There are a few different ways to do this, as long as each step is sensible and they are doing reasonable simplifications and are using correct notions, they can have full marks. Alternatively, if students correctly identify where on the hierarchy of functions with respect to asymptotic grows each T_i sits and have good justifications for such claims, they can have full marks. For example T_1 is a linear times exponential [obvious], which is faster than an exponential. T_2 is a qubic [as the other term is a quadratic and is slower, so can be ignored for large inputs], and T_3 is $n \cdot log(n)$ [we simplify the log using log rules to see this].

$$\lim_{n\to\infty} \frac{T_2}{T_1} = \lim_{n\to\infty} \frac{2^{10}n^2 + n^3}{n5^n} = \lim_{n\to\infty} \frac{2^{10}n + n^2}{5^n} =$$

$$\lim_{n\to\infty} \frac{2^{10}n}{5^n} + \lim_{n\to\infty} \frac{n^2}{5^n} = 0 + 0 = 0$$

T_1 grows faster than T_2. So A_1 is slower than A_2

$$\lim_{n\to\infty} \frac{T_2}{T_3} = \lim_{n\to\infty} \frac{2^{10}n^2 + n^3}{10^5 \log_{10}(n^n)} = \lim_{n\to\infty} \frac{2^{10}n^2 + n^3}{10^5 n \log_{10}(n)} =$$

$$\lim_{n\to\infty} \frac{2^{10}n + n^2}{10^5 \log_{10}(n)} = \lim_{n\to\infty} \frac{2^{10}n}{10^5 \log_{10}(n)} + \lim_{n\to\infty} \frac{n^2}{10^5 \log_{10}(n)} = \infty$$

T_2 grows faster than T_3. Hence A_2 is slower than A_3

$$\lim_{n\to\infty} \frac{T_3}{T_1} = \lim_{n\to\infty} \frac{10^5 \log_{10}(n^n)}{n5^n} = \lim_{n\to\infty} \frac{10^5 n \log_{10}(n)}{n5^n} =$$

$$\lim_{n\to\infty} \frac{10^5 \log_{10}(n)}{5^n} = 0$$

T_1 grows faster than T_3. Thus, A_1 is slower than A_3.

Finally, you can conclude that A_1 is the slowest algorithm and A_3 is the fastest algorithm.

Marking scheme:

- **Total: 4 points**
- **1 point** for each limit. They do not have to give much details about the way they found out the limit. Note they may do $\lim_{n\to\infty} \frac{T_1}{T_2}$ instead of the one I did.
- **1 point** For correctly identifying where in the asymptotic hierarchy of functions each T_i is located. For T_1 they don't need much/any justification, for the other two, they need some reasoning to identify that they are respectively $O(n^3)$ and $O(n \cdot log(n))$. If what they say makes sense, give them the marks, be generous.
- **1 point** for the correct conclusion.

(b) For which problem sizes is A_2 the better than A_1? (Justify your answer.)

You need to find all values of n, such that $T_2 < T_1$:. This is a hard inequality to solve by hand, but it can be simplified and plugged into some computational engine to produce a plot, or give a solution. First we observe the following simplifications:

$$2^{10}n^2 + n^3 < n5^n$$
$$2^{10}n + n^2 < 5^n$$
$$n(2^{10} + n) < 5^n$$

Plugging this into a solver, we get that n has to be greater than 5.66466, but since we are dealing with $n \in \mathbb{N}$, it only makes sense that $n \geq 6$ for the above inequality to be true, hence we conclude that for all values of n such that $1 \leq n \leq 5$, T_1 is going to run faster than T_2. This conclusion can also be arrived at by plugging numbers into the two algorithms (or the simplified inequality) and observing the resulting behaviour. Further, a plot is also fine presuming the conclusion is made correctly.

Marking scheme:

- **Total: 3 points**
- **1 point** for the correct answer.
- **2 points** for justification. Here they can use the following ways of justification:
  - theoretical, if they manage to do this;
  - write a code to calculate the thing (they should include their code in this case);
  - writing a table and just by trial and error checking all numbers in the appropriate range.
  - using Wolfram Alpha (or similar) to find rational points of intersections and write their integer approximation.

  If the explanation is incomplete, then give just 1 point.

(c) For which problem sizes is A_2 better than A_3? (Justify your answer.)

You need to find all values of n, such that $T_2 < T_3$. Similar to above, a sensical way to approach this is to do some simplifications on the inequality, then use a computational engine to either plot or solve for n, and use the given info in an appropriate way to formulate the conclusion. Taking the inequality and simplifying, we get:

$$2^{10}n^2 + n^3 < 10^5 log(n^n)$$
$$2^{10}n^2 + n^3 < 10^5 n \cdot log(n)$$
$$n(2^{10} + n) < 10^5 log(n)$$

Solving for n, we get $1.02447 < n < 187.623$, so the appropriate interpretation of this is that for all values of n in the range $2 \leq n \leq 187$, T_2 is going to be smaller than T_3.

Marking scheme:

- **Total: 3 points**
- **1 point** for the correct answer.
- **2 points** for justification. Here they can use the following ways of justification:
  - theoretical, if they manage to do this;
  - write a code to calculate the thing (they should include their code in this case);
  - writing a table and just by trial and error checking all numbers *close* to appropriate n.
  - using Wolfram Alpha (or similar) to find rational points of intersections and write their integer approximation.

  If the explanation is incomplete, then give just 1 point.

2. (10 marks) **Asymptotic Notations:**

(a) Prove using definitions only that $n^n log(n)$ is not $O(n^n)$. (HINT: proof by contradiction)

Suppose for the sake of contradiction that $n^n log(n)$ is $O(n^n)$. Cashing out on the definition, we have it that there must exist some c and n_0 such that for all $n \geq n_0$ we have it that $n^n log(n) \leq cn^n$. Suppose WLOG that $c > n_0 \geq 1$ (if it is not, we choose a new c s.t. $c > n_0$ as well as $n_0 \geq 1$, and the inequality (and consequently the definition) still holds). Now we can choose $n = 10^{2c} > c > n_0$, and then we plug this into the already presented inequality and observe:

$$(10^2c)^{10^{2c}} \log(10^{2c}) \leq c \cdot (10^{2c})^{10^{2c}}$$
$$\log(10^{2c}) \leq c$$
$$2c \leq c$$

Behold, a contradiction!

- **Total: 3 points**
- **1 point** For using (or trying to use) definitions only (so no limit rules or other rules).
- **1 point** for correctly structuring the proof: ie assuming for a contradiction that f is $O(g)$, and correctly using this to take some c and n_0 s.t. the inequality holds.
- **1 points** For the actual proof. It is OK if some of the steps in their inequalities aren't the most rudimentary or obvious. But they get this point only if there are no important logical mistakes.

(b) Prove that $n^3 - 50$ is $\Omega(n^2)$, either with limit laws or definitions.

Below we provide a proof using definitions. However a proof using Limit laws is also acceptable.
Take $c = \frac{1}{2}$ and $n_0 = 50$, then observe:

$$n^3 - 50 = \frac{1}{2}n^3 + (\frac{1}{2}n^3 - 50)$$

Then, as $n \geq 50$, we have it that $(\frac{1}{2}n^3 - 50) > 0$, so we can observe the following:

$$\frac{1}{2}n^3 + (\frac{1}{2}n^3 - 50) \geq \frac{1}{2}n^3 \geq \frac{1}{2}n^2 \cdot 1 = \frac{1}{2}n^2$$

Ergo, there exist a c and an n_0 such that $\forall n > n_0$ we have it that $n^3 - 50 \geq c \cdot n^2$.

- **Total: 2 points**
- **1 point** for a correct framework of establishing this (ie if they are proving with definitions, they take some c and some n_0 such that the inequality holds, or if they are using limit laws, they are using the right limit law in a correct way.)
- **1 point** If there are no major issues/problems with the logic that they are using. In other words (-1) points for minor mistakes.

(c) Prove that $\sqrt(15)n^2\log(5n) + 0.4n + \sqrt{n}$ is $\Theta(n^2\log(n))$ using definitions only.

Call $\sqrt{15}n^2\log(5n) + 0.4n + \sqrt{n}$ f. We establish that f is $\Theta(n^2\log(n))$ by showing that it is both $\Omega(n^2\log(n))$ and $O(n^2\log(n))$.
First, we show that f is $\Omega(n^2 log(n))$ (this is the easy direction!). Take $c_1 = \sqrt{15}$ and $n_0 = 1$, and observe that:

$$\sqrt{15}n^2\log(5n) + 0.4n + \sqrt{n} \geq \sqrt{15}n^2\log(5n) = \sqrt{15}n^2(\log(n) + \log(5)) =$$
$$\sqrt{15}n^2\log(n) + \sqrt{15}n^2\log(5) \geq \sqrt{15}n^2\log(n)$$

Hence we have it that f is $\Omega(n^2\log(n))$.
Now we will show that f is $O(n^2\log(n))$. We take $c_2 = 6\cdot\sqrt{15}$ and $n_0' = 5$. We first note that for all $n \geq n_0' > 1$, $\sqrt{15}n^2\log(5n) > g$ for $g = 0.4n$ or $g = \sqrt{n}$. Hence:

$$\sqrt{15}n^2\log(5n) + 0.4n + \sqrt{n} \geq \sqrt{15}n^2\log(5n) + \sqrt{15}n^2\log(5n) + \sqrt{15}n^2\log(5n) =$$
$$3\cdot\sqrt{15}n^2\log(5n) = 3\cdot\sqrt{15}n^2(\log(n) + \log(5)) =$$
$$3\cdot\sqrt{15}n^2\log(n) + 3\cdot\sqrt{15}n^2\log(5)$$

We also note that for all $n \geq 5$, $3\cdot\sqrt{15}n^2\log(n) \geq 3\cdot\sqrt{15}n^2\log(5)$, so:

$$3\cdot\sqrt{15}n^2\log(n) + 3\cdot\sqrt{15}n^2\log(5) \geq 3\cdot\sqrt{15}n^2\log(n) + 3\cdot\sqrt{15}n^2\log(n) = 6\cdot\sqrt{15}n^2\log(n)$$

This demonstrates that indeed f is $O(n^2\log(n))$, and concludes our proof that f is $\Theta(n^2\log(n))$.

- **Total: 2 points**
- **1 point** for a correct framework of establishing this: ie they correctly identify that we need to show f is both $O(g)$ and $\Omega(g)$, and show understanding that they need to present some such c_1 and c_2 and corresponding n_0, and work with inequalities.
- **1 point** If there are no major issues/problems with the logic that they are using. In other words (-1) points for minor mistakes.

(d) Suppose that f, h, g, k are functions and that $f = h \cdot g + k$. Suppose further, that h is $O(n^2)$, g is $\Theta(\log(n))$ and k is $\Omega(n)$. What do we know about f? State everything we can determine for full marks.

For this question we are going to abuse notations in order to convey the point across and make the process more comprehensive. Rewriting f in our abuse of notations, we get the following:

$$f = O(n^2)\Theta(\log(n)) + \Omega(n)$$

So we note that f is a sum of two functions, namely $O(n^2)\Theta(\log(n))$ and $\Omega(n)$. We first begin discussing what we can about each one of these functions, and then we conclude what we know about f. $O(n^2)\Theta(\log(n))$ is clearly $\Omega(\log(n))$ and $O(n^2\log(n))$, because we have the tight bound for g, but h can be anything from a constant to a quadratic. Further, the only thing that we can tell about $\Omega(n)$ is that it is bound below by a linear, and we do not know the upper bound for k. Putting all this together, we can deduce that f is $\Omega(n)$ (as the lower bound of the right sub-function is bigger than the lower-bound of the left sub-function), and we cannot determine anything about the upper bound, again because the left sub-function's upper bound is unknown, and it could potentially be anything larger than $O(n^2 \log(n))$.

- **Total: 3 points**
- **1 point** For correctly establishing that $h \cdot k$ is $O(n^2 \log(n)$ and $\Omega(\log(n))$.
- **1 point** For correctly reasoning about the implication of the fact that f is the sum of two functions, of which we both know a lower bound, but only know the upper bound of one of these.
- **1 point** For correctly reasoning that we can only ever know the lower bound of f due to one of the sub-functions that it is a sum of being of an unknown upper bound.

3. (10 marks) **Elementary operations and algorithms:**

(a) Work out the number of elementary operations in the following algorithm. For this exercise only count the constant number C of elementary operations as executed on lines 2 and 4.

```
0: function Blah (positive integer n)
1:   for i ← 2 to n do
2:       Constant number C of elementary operations.
3:     for j ← 1 to i do
4:         Constant number C of elementary operations.
```

We note that the outer loop ranges over $i \in [2, n]$, and for every iteration, on line 2 we have a constant number C of elementary operations. There is also what happens inside the second loop. Thus the function $T(n)$ is going to look like $T(n) = C(n - 1) + g(n)$, where $g(n)$ counts the number of elementary operations inside the second loop. To determine the behaviour of this loop, we note that it ranges from $j = 1$ to i. i always starts at 2 and increments by 1 on every iteration of the outer loop. Thus we can observe that for the first iteration of the outer loop, there are going to be two iterations of the inner loop, for the second iteration of the outer loop there are going to be 3 iterations of the inner loop, and in general, for the $k - th$ iteration of the outer loop (where k is an arbitrary value in the range $[2, n]$), i is going to be $k + 1$ and so there are going to be $k + 1$ iterations of the inner loop. If we look at the sequential number of iterations of the inner loop, we see $2, 3, 4, ..., k + 1, ..., n$ iterations, and summing this up we get $\frac{n(n+1)}{2} - 1$. Finally, we note that on each iteration of the inner loop a constant number C of elementary operations occurs, thus we have it that $g(n) = C \cdot \frac{n(n+1)}{2} - 1$. Putting all of this together, we have it that $T(n) = C((n + \frac{n(n+1)}{2} - 2) = \frac{C}{2}(n(n + 3) - 2)$.

- **Total: 5 points**
- **1 point** For correctly identifying that the outer loop runs $n-1$ times. If for whatever reason they interpreted it to run up to but not including $i=n$, this is fine as long as they stated their interpretation.
- **1 point** For correctly realizing that the inner loop runs $2+3+...+n$ many times. Same principle as above if they treated the range to be exclusive of i.
- **1 point** Correctly treating the elementary operations C in their final equation.
- **2 point** For a correct answer. Give 1 point if there are some minor mistakes.
- **2 point** Give 2 to 3points for effort it its a genuine attempt, but wrong answer, depending on how many of the above they got right.

(b) Work out the number of elementary operations the following algorithm uses in an average case. You can assume that the probability of getting a number divisible by 3 is $\frac{1}{3}$.

0: **function** Meh (positive integer n)
1: **if** $n\%3 = 0$ **do**
2:     **for** $i \leftarrow 0$ **to** $n-1$ **do**
3:         **for** $j \leftarrow 0$ **to** $n-1$**,** $j \leftarrow j+2$ **do**
4:             **if** $i = j$ **do**
5:                 **for** $k \leftarrow 0$ **to** n^4**,** $k \leftarrow k+n^2$ **do**
6:                     Constant number C of elementary operations.
7:             **else**
8:                 Constant number C of elementary operations.
9: **else**
10:     Execute Meh($3n^2$)

We first consider what happens when n is divisible by 3. The first step here is to note that the outer loop iterates over $i=0$ to $i=n-1$, which is n iterations. Taking an arbitrary iteration, we then note that the second for loop iterates over $j=0$ to $j=n-1$, but j is incremented by 2, so this for loop will iterate $\lfloor\frac{n}{2}\rfloor$ many times. Inside the second loop is a conditional that is only ever true at most once per iteration of the outer loop, and even then only when i is even. When the conditional is true, the third loop ranges over $k=0$ increasing to $k=n^4$, but every time it increments in blocks of n^2, which means that it iterates a total of n^2 times.

Since we only care about the elementary operations C, we need to count the total number of times line 8 and line 6 are executed.

We are going to do this in two cases. First, we are going to consider all such executions when i is not even, which is $\lfloor\frac{n}{2}\rfloor$ many times. For each such iteration, the j loop iterates $\lceil\frac{n}{2}\rceil$ many times, giving us the following total number of elementary operations:

$$\lfloor\frac{n}{2}\rfloor \cdot \lceil\frac{n}{2}\rceil C$$

We now consider what happens when i is even. Namely, this happens $\lceil\frac{n}{2}\rceil$ many times. On every such iteration, the if condition on line 4 is going to be true exactly once, and the else condition is going to be true exactly $\lceil\frac{n}{2}\rceil - 1$ many times (as the jth loop iterates $\lceil\frac{n}{2}\rceil$ many times). When the if condition is met, the third loop iterates

n^2 many times, and on each such iteration, line 6 is executed incuring C elementary operations. Each time the condition on line 4 is not met, line 8 is executed, and we have C elementary operations. Putting all this together, we get:

$$\lceil \frac{n}{2} \rceil \Big((\lceil \frac{n}{2} \rceil - 1)C + n^2 C \Big) = C \lceil \frac{n}{2} \rceil \Big(\lceil \frac{n}{2} \rceil - 1 + n^2 \Big)$$

Putting the above information together, we have it that when $n\%3 = 0$, the number of elementary operations is:

$$C \lceil \frac{n}{2} \rceil \Big(\lceil \frac{n}{2} \rceil - 1 + n^2 \Big) + \lfloor \frac{n}{2} \rfloor \cdot \lceil \frac{n}{2} \rceil C = C \lceil \frac{n}{2} \rceil \left(\Big(\lceil \frac{n}{2} \rceil - 1 + n^2 \Big) + \lfloor \frac{n}{2} \rfloor \right)$$

.

The case when $n\%3 \neq 0$ is easy, we just note that the function is recursively called on itself with the input $3n^2$, which is divisible by 3, so we can use the above formula to get:

$$C \lceil \frac{3n^2}{2} \rceil \left(\Big(\lceil \frac{3n^2}{2} \rceil - 1 + (3n^2)^2 \Big) + \lfloor \frac{3n^2}{2} \rfloor \right)$$

We can now make the claim that given an arbitrary n with $\frac{1}{3}$ probability that it is divisible by 3, the average number of elementary operations is going to be:

$$\frac{1}{3} C \lceil \frac{n}{2} \rceil \left(\Big(\lceil \frac{n}{2} \rceil - 1 + n^2 \Big) + \lfloor \frac{n}{2} \rfloor \right) + \frac{2}{3} C \lceil \frac{3n^2}{2} \rceil \left(\Big(\lceil \frac{3n^2}{2} \rceil - 1 + (3n^2)^2 \Big) + \lfloor \frac{3n^2}{2} \rfloor \right)$$

- **Total: 5 points**
- Note: The students are not required to get the correct floor and ceiling functions. Observing roughly what happens is fine. Don't punish mistakes for incorrect use of floor/ceiling functions.
- **2 point** For correctly getting the formula when n is divisible by three. If the student makes a different assumption about the upper bound of i, j, k, ie assumes it does not include the stated values, this is fine (for instance $i < n - 1$).
- **2 point** For correctly realizing what happens when n is not divisible by 3.
- **1 point** For correctly putting the two equations into a single average number of elementary operations. May look different to mine.
- **2 point** If they get the above wrong, give two points for an honest attempt at working through the loops and attempts at justification.
- **1 point** On top of the above, again, if they don't get the actual formula, give 1 extra point if what they get kind of vaguely resembles a right answer. So if there is an honest but flawed attempt that shows semi-coherent working, the student should get 3 out of 5 for this question.

4. (10 marks) **Elementary operations and algorithms:**

   (a) What is the explicit form of the following recurrence relation:
   $T(n) = 3T(n/3) + 1; \quad T(1) = 1.$

We start by assuming $n = 3^k$ for some positive integer k, then we observe the following:

$$T(3^k) = 3T(3^{k-1}) + 3^0$$
$$T(3^k) = 3^2T(3^{k-2}) + 3^0 + 3^1$$
$$T(3^k) = 3^3T(3^{k-3}) + 3^0 + 3^1 + 3^2$$

We can now observe the general pattern:

$$T(3^k) = 3^iT(3^{k-i}) + 3^0 + 3^1 + 3^2 + \cdots + 3^{i-1}$$

Having gotten this far, we want to get rid of the recursive call, we note that the given intial conditions are for $T(1)$, so we set i to some value such that $k - i = 0$, which just happens to be k itself, and we get the following:

$$T(3^k) = 3^kT(3^{k-k}) + 3^0 + 3^1 + 3^2 + \cdots + 3^{k-1}$$
$$T(3^k) = 3^k + 3^0 + 3^1 + 3^2 + \cdots + 3^{k-1}$$
$$T(3^k) = \sum_{i=0}^{k} 3^i$$

We can now use the following sum from the coursebook (pg6), $\Sigma_{i=m}^{n} a^i = \frac{a^{n+1}-a^m}{a-1}$, to get:

$$T(3^k) = (3^{k+1} - 1)/2$$

Finally, we substitute n back in to get:

$$T(n) = (3n - 1)/2$$

- **Total: 5 points**
- **1 point** For correctly assuming $n = 3^k$ and putting that into the formula.
- **1 point** For correctly writing down the final result in terms of n.
- **3 point** For their working. Read this as -1 point per mistake for a maximum of -2. If it's a simple mistake and carries through to the final result, still give them the points for the final formula. Alternatively if they are completely off give them a maximum of 2 for effort.

(b) What is the explicit form of the following recurrence relation:
$T(n) = T(n-1) + \log_2 n$; $\quad T(0) = 0$. Hint $n!$ is approximately $\sqrt{2\pi n}n^ne^{-n}$.

$$T(n) = T(n-1) + \log_2 n$$
$$T(n) = T(n-2) + \log_2(n) + \log_2(n-1)$$
$$T(n) = T(n-3) + \log_2(n) + \log_2(n-1) + \log_2(n-2)$$

We can now observe the general form to be:

$$T(n) = T(n-i) + \log_2(n) + \log_2(n-1) + \log_2(n-2) + \cdots + \log_2(n-(i+1))$$

Choosing a value of i to be able to use the initial condition $T(0)$, we choose $i = n$, getting:

$$T(n) = T(n-n) + \log_2(n) + \log_2(n-1) + \log_2(n-2) + \cdots + \log_2(1)$$
$$T(n) = 0 + \log_2(n) + \log_2(n-1) + \log_2(n-2) + \cdots + \log_2(1)$$

Now we use log rules to get:

$$T(n) = \log_2(n!)$$

And we can use our hint:

$$T(n) = \log_2(n!) \approx \log_2(\sqrt{2\pi n}n^n e^{-n})$$
$$T(n) \approx \log_2(\sqrt{2\pi n}) + \log_2(n^n) + \log_2(e^{-n})$$
$$T(n) \approx \log_2(\sqrt{2\pi n}) + n\log_2(n) - n * \log_2(e)$$

- **Total: 5 points**
- **1 point** For correctly assuming $n = 3^k$ and putting that into the formula.
- **1 point** For correctly writing down the final result in terms of n.
- **3 point** For their working. Read this as -1 point per mistake for a maximum of -2. If it's a simple mistake and carries through to the final result, still give them the points for the final formula. Alternatively if they are completely off give them a maximum of 2 for effort.