Q1 – [Asymptotic Relations - 1 point] For each of the following pairs of functions f (n) and g(n), determine the most appropriate symbol in the set O, o, θ, Ω, ω. (lg n = log to the base 2 of n).

1.

$$f(n) = 1005n^2 + 10n + 11 \qquad g(n) = \frac{n^3}{1000}$$

$$\approx n^2 \qquad = \frac{1}{1000}n^3$$

$$\approx n^3$$

$$\because n^2 \ll n^3$$

$$\therefore f(n) = o(g(n))$$

2.

$$\begin{aligned} f(n) &= \lg^7(n^7) \\ &= 7\lg^7 n \\ &= \lg^7 n \\ &\because \lg^7 n \gg \sqrt[4]{n} \\ &\therefore f(n) = \omega(g(n)) \end{aligned}$$

$$\begin{aligned} g(n) &= \left(n^{1/2}\right)^{1/2} \\ &= n^{1/4} \\ &= \sqrt[4]{n} \end{aligned}$$

3.

$$\begin{aligned} f(n) &= (n^2-1)(n^2+1)\lg n \\ &= (n^4-1)\lg n \\ &= n^4 \lg n - \lg n \end{aligned} \qquad \begin{aligned} g(n) &= n^4 \lg n^{1001} \\ &= 1001 n^4 \lg n \end{aligned}$$

As there is no polynomial difference it is not immediately clear after reduction which function has a higher growth rate but by taking the quotient of the limit as n approaches infinity:

$$\begin{aligned} \lim_{x\to\infty} \frac{f(n)}{g(n)} &= \lim_{x\to\infty} \frac{n^4 \lg n - \lg n}{1001 n^4 \lg n} \\ &= \lim_{x\to\infty} \frac{\lg n (n^4 - 1)}{1001 n^4 \lg n} \\ &= \lim_{x\to\infty} \frac{n^4 - 1}{1001 n^4} \\ &= \lim_{x\to\infty} \left(\frac{n^4}{1001 n^4}\right) - \lim_{x\to\infty} \left(\frac{1}{1001 n^4}\right) \\ &= \lim_{x\to\infty} \left(\frac{1}{1001}\right) - \lim_{x\to\infty} \left(\frac{1}{1001 n^4}\right) \\ &= \frac{1}{1001} - 0 \\ &= \frac{1}{1001} \\ &\because \frac{1}{1001} < 1 \\ &\therefore f(n) = O(g(n)) \end{aligned}$$

4.

$$
\begin{aligned}
f(n) &= 32^{\lg \sqrt{n}} \\
&= 32^{\lg n^{1/2}} \\
&= 32^{\frac{1}{2}\lg n} \\
&= \left(\sqrt{32}\right)^{\lg n} \\
&= n^{\lg \sqrt{32}} \\
&= n^{2.5} \\
&\because n^{2.5} < n^3 \\
&\therefore f(n) = O(g(n))
\end{aligned}
\qquad\qquad
g(n) = n^3
$$

Q2 – [Step count analysis - 1 point] Analyze the following pseudocode and give a tight bound on the running time as a function of n. You can assume that all individual instructions take O(1) time Show your work.

```
l ← 0
i ← 1
while i ≤ n do
    for j ← 1, i do
        l ← l + 2 * n + 3 * j
    end for
    i = 2 * i
end while
```

In the above pseudocode denotes a function which we will call $P(n)$. $P(n)$ contains 3 separate code blocks, $O(n)$ the assignment block, $W(n)$ the *while* loop and $F(n)$ the for loop. We can perform a step count analysis by considering the 3 code blocks separately and then adding them together.

For convenience we denote $s(B)$ as the number of steps in a code block $B(n)$. To set up our blocks for analysis we let $O(n)$ denote the entry block where i and l are initialized and set to 0. Since these two assignments are atomic operations which only occur once

$$s(O) = 2$$

The next block to consider is the *while* loop which we will denote $W(n)$. Since $W(n)$ depends on i being smaller than n and $W(n)$ exponentially increments i with the assignment $i = i * 2$ we can deduce that $W(n)$ executes roughly $\lg(n)$ times. Because $n \in \mathbb{N}$ W will execute $\lfloor \lg(n) \rfloor + 1$ times. Letting w be the number of atomic operations in $W(n)$, and seeing that the atomic operations are the comparison of $i \leq n$, the execution of the for loop and the assignment to i we see $w = 3$ and:

$$\begin{aligned} s(W) &= (\lfloor \lg(n) \rfloor + 1)w \\ &= 3(\lfloor \lg(n) \rfloor + 1) \end{aligned} \tag{1}$$

The final block is the for loop, which is a linear step from 1 to i. Since i is always equal to a multiple of $2^k \quad k \in \mathbb{N}$ we see that on each iteration (k) of the while block the for loop will execute $2^k - 1$ times.

$$\begin{aligned} s(F) &= \left[(2^1 - 1) + (2^2 - 1) + \ \dots \ + (2^{\lfloor \lg n \rfloor - 1})\right] \\ &= \sum_{i=1}^{\lg n} (2 - 1)^i \\ \because &\sum_{k=1}^{n} (2^k - 1) = 2^{n+1} - n - 1 \\ \therefore &\ s(F) = 2^{\lfloor \lg n \rfloor + 1} - \lfloor \lg n \rfloor - 1 \end{aligned} \tag{2}$$

Finding the total steps in the program which we will denote $s(P)$ is a simple process of summing the results of the step functions $s(O)$, $s(W)$ and $s(F)$

$$\begin{aligned} s(P) &= s(O) + s(W) + s(F) \\ &= 2 + 3(\lfloor \lg(n) \rfloor + 1) + \left(2^{\lfloor \lg n \rfloor + 1} - \lfloor \lg n \rfloor - 1\right) \\ &= 2 + 3\lfloor \lg n \rfloor + 3 + 2^{\lfloor \lg n \rfloor + 1} - \lfloor \lg n \rfloor - 1 \\ &= 2^{\lfloor \lg n \rfloor + 1} + 2\lfloor \lg n \rfloor + 4 \end{aligned} \tag{3}$$

Assessing the above, a tight upper-bound and a tight lower-bound can be found by considering the floor operation on $\lg n$, $\lfloor \lg n \rfloor$. Then for an upper-bound:

$$\begin{aligned} 2^{\lfloor \lg n \rfloor + 1} &\approx 2^{\lg n + 1} \\ &= 2\dot{n}^{\lg 2} \\ &= 2n \end{aligned} \tag{4}$$

Similarly for a lower bound,

$$\begin{aligned} 2^{\lfloor \lg n \rfloor} &\approx 2^{\lg n} \\ &= n^{\lg 2} \\ &= n \end{aligned} \tag{5}$$

This completes our step count analysis and we can confidently say that the above algorithm has a complexity of

$$P(n) = \Theta(n)$$

Q3 – [Logarithms – 1 point] Prove that $a^{\log_b x} = x^{\log_b x}$. Do not assume the statement to be true. Deduce your answer by applying logarithm principles.

$$
\begin{aligned}
a^{\log_b x} &\stackrel{?}{=} x^{\log_b x} \\
\log_x a^{\log_b x} &= \log_x \left(x^{\log_b x}\right) \\
\log_b x \log_x a &= \log_b a \\
\log_x a &= \frac{log_b a}{log_b x} \\
\because \log_b a &= \frac{\log_c a}{\log_c b} && \text{(from class notes)} \\
\therefore a^{\log_b x} &= x^{\log_b x}
\end{aligned}
$$

Q4 – [Recursive Relations - 2 points] Please solve the following recurrences.

1. $T(n) = 4T\left(\frac{n}{3}\right) + n$

$$\begin{aligned}
T(n) &= 4T\left(\frac{n}{3}\right) + n \\
T\left(\frac{n}{3}\right) &= 4T\left(\frac{n}{9}\right) + \left(\frac{n}{3}\right) \\
T^1(n) &= 4\left(4T\left(\frac{n}{9}\right) + \left(\frac{n}{3}\right)\right) + n \\
&= 16T\left(\frac{n}{9}\right) + \left(\frac{4n}{3}\right) + n \\
T\left(\frac{n}{9}\right) &= 4T\left(\frac{n}{27}\right) + \left(\frac{n}{9}\right) \\
T^2(n) &= 16\left(4T\left(\frac{n}{27}\right) + \left(\frac{n}{9}\right)\right) + \frac{4n}{3} + n \\
&= 64T\left(\frac{n}{27}\right) + \left(\frac{16n}{9}\right) + \frac{4n}{3} + n
\end{aligned} \tag{6}$$

As a pattern emerges we can generalize for $T^k(n)$ where k is the maximum depth of the recursion before the base case is reached. Then we have

$$\begin{aligned}
T^k(n) &= 4T^k\left(\frac{n}{3^k}\right) + n \cdot \left[\left(\frac{4}{3}\right)^0 + \left(\frac{4}{3}\right)^1 + \ \dots \ + \left(\frac{4}{3}\right)^{k-1}\right] \\
&= 4^kT\left(\frac{n}{3^k}\right) + n \cdot \sum_{i=0}^{k-1}\left(\frac{4}{3}\right)^i \\
\sum_{i=0}^{k-1}\left(\frac{4}{3}\right)^i &= \frac{\left(1 - \left(\frac{4}{3}\right)^{k-1}\right)}{1 - \left(\frac{4}{3}\right)} \\
T^k(n) &= 4^kT\left(\frac{n}{3^k}\right) + n \cdot \frac{\left(1 - \left(\frac{4}{3}\right)^{k-1}\right)}{1 - \left(\frac{4}{3}\right)}
\end{aligned} \tag{7}$$

As k reaches the base case $T(1) = 1 = \frac{n}{3^k}$. Solving for k we can see that $k = \log_3 n$.

Subbing this value into recurrence $T^k(n)$ we see:

$$\begin{aligned}
T^{\log_3 n}(n) &= 4^{\log_3 n} T\left(1\right) + \frac{\left(n - \left(\frac{4}{3}\right)^{\log_3 n - 1}\right)}{1 - \left(\frac{4}{3}\right)} \\
&= 4^{\log_3 n} T\left(1\right) + \frac{n - 3\dot{4}^{\log_3 n - 1}}{-\frac{1}{3}} \\
&= 4^{\log_3 n} T\left(1\right) - 3n + 9\dot{4}^{\log_3 n - 1} \\
&= 4^{\log_3 n} T\left(1\right) - 3n + \frac{9}{4} n^{\log_3 4} \\
&= n^{\log_3 4}\left(1\right) - 3n + \frac{9}{4} n^{\log_3 4} \\
&= n^{\log_3 4} + \frac{9}{4} n^{\log_3 4} - 3n \\
&= \frac{13}{4} n^{\log_3 4} - 3n \\
&= \frac{13}{4} n^{1.26} - 3n \\
&\because n^{1.26} \gg n \\
&\therefore f(n) = \Theta(n^{1.26})
\end{aligned} \tag{8}$$

2. $T(n) = 3T\left(\frac{n}{3}\right) + \frac{n}{2}$

$$\begin{aligned}
T\left(n\right) &= 3T\left(\frac{n}{3}\right) + \frac{n}{2} \\
T\left(\frac{n}{3}\right) &= 3T\left(\frac{n}{9}\right) + \frac{2n}{3} \\
T^2\left(n\right) &= 3\left(3T\left(\frac{n}{9}\right) + \frac{2n}{3}\right) + \frac{n}{2} \\
&= 9T\left(\frac{n}{9}\right) + 2n + \frac{n}{2} \\
T\left(\frac{n}{9}\right) &= 3T\left(\frac{n}{27}\right) + \frac{2n}{9} \\
T^3\left(n\right) &= 9\left(3T\left(\frac{n}{27}\right) + \frac{2n}{9}\right) + 2n + \frac{n}{2} \\
&= 27T\left(\frac{n}{27}\right) + 2n + 2n + \frac{n}{2} \\
&= 3^3T\left(\frac{n}{3^3}\right) + 2n(k-1) + \frac{n}{2}
\end{aligned} \tag{9}$$

Analyzing the above and letting the maximum depth be denoted by k we see,

$$T^k(n) = 3^kT\left(\frac{n}{3^k}\right) + 2n(k-1) + \frac{n}{2}$$

Then as we reach the base case $T(1) = 1 = \frac{n}{3^k}$ which we can simplify to $k = \log_3 n$ subbing that into $T^k(n)$ we can solve the recurrence.

$$\begin{aligned}
T^k(n) &= 3^{\log_3 n}T(1) + 2n(\log_3 n - 1) \\
&= n + 2n\log_3 n - 2n \\
&\approx 2n\log n - n
\end{aligned} \tag{10}$$

Therefore the growth rate expressed by $T(n)$ is

$$T(n) = O(n \log n)$$

3. $T(n) = 4T\left(\frac{n}{2}\right) + n^{2.5}$

$$\begin{aligned}
T(n) &= 4T\left(\frac{n}{2}\right) + n^{2.5} \\
T\left(\frac{n}{2}\right) &= 4T\left(\frac{n}{4}\right) + \left(\frac{n}{2}\right)^{2.5} \\
T^1(n) &= 4\left(4T\left(\frac{n}{4}\right) + \left(\frac{n}{2}\right)^{2.5}\right) + n^{2.5} \\
&= 16T\left(\frac{n}{4}\right) + 4\left(\frac{n^{2.5}}{4^{2.5}}\right) + n^{2.5} \\
&= 16T\left(\frac{n}{4}\right) + 4\left(\frac{n^{2.5}}{32}\right) + n^{2.5} \\
T\left(\frac{n}{4}\right) &= 4T\left(\frac{n}{8}\right) + \left(\frac{n}{4}\right)^{2.5} \\
T^2(n) &= 16T\left(4T\left(\frac{n}{8}\right) + \left(\frac{n}{4}\right)^{2.5}\right) + 4\left(\frac{n^{2.5}}{32}\right) + n^{2.5} \\
&= 64T\left(\frac{n}{8}\right) + 16\left(\frac{n^{2.5}}{32}\right) + 4\left(\frac{n^{2.5}}{32}\right) + n^{2.5} \\
T^k(n) &= 4^kT\left(\frac{n}{2^k}\right) + 4^k\left(\frac{n^{2.5}}{32}\right) + n^{2.5} \\
&= 4^kT\left(\frac{n}{2^k}\right) + 2^{2k}\left(2^{-5}\right)n^{2.5} + n^{2.5}
\end{aligned} \tag{11}$$

From the base case $T(1) = 1$ so $1 = \frac{n}{2^k}$ solving for k we find the maximum value of k would be $k = \log_2 n$, Then

$$\begin{aligned}
T(n) &= 2^{2\log_2 n}T(1) + 2^{2\log_2 n}(2^{-5})n^{2.5} + n^{2.5} \\
&= 2^2 * 2^{\log_2 n} + 2^2 * 2^{\log_2 n} * 2^{-5} * n^{2.5} + n^{2.5} \\
&= 4n + 4n\left(\frac{1}{32}\right)n^{2.5} + n^{2.5} \\
&= \frac{1}{8}n^{3.5} + n^{2.5} + 4n
\end{aligned} \tag{12}$$

Therefore looking at the n term with the highest polynomial growth rate we see that

$$T(n) = O(n^{3.5})$$

4. $T(n) = T(\frac{n}{2}) + T(\frac{n}{4}) + T(\frac{n}{8}) + 1$

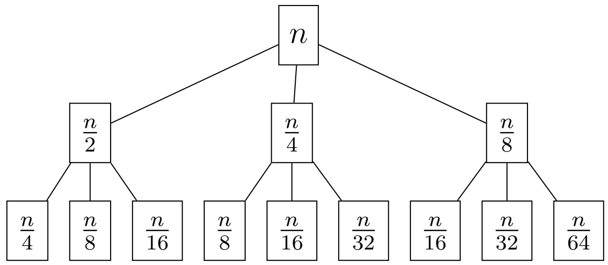

Figure 1: Recursive Tree for $T(n) = T(\frac{n}{2}) + T(\frac{n}{4}) + T(\frac{n}{8}) + 1$

Going through the above recurrence tree we start at $k = 0$ where k is the current level in the tree. at $k = 0$ We have a single element and $T^0(n) = O(1)$. At $k = 1$ we can simply add the nodes at this level

$$T^1(n) = \frac{n}{2} + \frac{n}{4} + \frac{n}{8} = \frac{7n}{8}$$

Moving to the next level where $k = 2$

$$\begin{aligned} T^2(n) &= \frac{n}{4} + \frac{n}{8} + \frac{n}{16} + \frac{n}{8} + \frac{n}{16} + \frac{n}{32} + \frac{n}{16} + \frac{n}{32} + \frac{n}{64} \\ &= \frac{16n}{64} + \frac{8n}{64} + \frac{4n}{64} + \frac{8n}{64} + \frac{4n}{64} + \frac{2n}{64} + \frac{4n}{64} + \frac{2n}{64} + \frac{n}{64} \\ &= \frac{47n}{64} \\ &= \left(\frac{7}{8}\right)^2 n \end{aligned} \tag{13}$$

We can see a pattern begin to emerge where

$$T^k(n) = \left(\frac{7}{8}\right)^k n$$

Since $T(1) = 1 = \frac{n}{2^k}$ we see that $k = \log_2 n$, subbing the value of k into $T^k(n)$ as the total depth and summing the levels we have:

$$\begin{aligned} T(n) &= \left[\left(\frac{7}{8}\right)^{0} + \left(\frac{7}{8}\right)^{1} + \ ... \ + \left(\frac{7}{8}\right)^{\log_2 n}\right] n \\ &= \sum_{m=0}^{\log_2 n} \left(\frac{7}{8}\right)^{m} n \\ &= \left(\frac{1-\left(\frac{7}{8}\right)^{\log_2 n+1}}{1-\left(\frac{7}{8}\right)}\right) n \\ &= \left(8 - 8\left(\frac{7}{8}\right)\left(\frac{7}{8}\right)^{\log_2 n}\right) n \\ &= \left(8 - 7\left(\frac{n^{\log_2 7}}{n^3}\right)\right) n \\ &= 8n - 7\left(\frac{n^{2.81}}{n^2}\right) \\ &= 8n - 7n^{0.81} \end{aligned} \tag{14}$$

Therefore after analyzing the recurrence tree above we can see that

$$T(n) = O(n)$$