Name: Pathri Vidya Praveen
Roll No.: CS24BTECH11047

Theory Assignment 1 : CS2233

1. For each of these questions, briefly explain your answer.

- If I prove that an algorithm takes $O(n^2)$ worst case time, is it possible that it takes $O(n)$ on some inputs?

Sol: Yes, If an algorithm takes $O(n^2)$ worst case time, it is possible that it takes $O(n)$ on some inputs. For example, in insertion sort, worst case time is $O(n^2)$ when inputs are in reverse sorted order. But, best case time is $O(n)$ for input arrays that are already sorted.

- If I prove that an algorithm takes $O(n^2)$ worst case time, is it possible that it takes $O(n)$ for all inputs?

Sol: Yes, if I prove that an algorithm takes $O(n^2)$ worst case time, it is possible that it takes $O(n)$ for all inputs. But generally, if an algorithm takes $O(n)$ for all input, we can say that worst case time is $O(n)$. $O(n^2)$ is also true but it is weaker bound and so less informative as generally in asymptotic analysis of algorithms, our main focus is on greatest lower bound and least upper bound of running times.

- If I prove that an algorithm takes $\Theta(n^2)$ worst-case time, is it possible that it takes $O(n)$ on some inputs?

Sol: Yes, if I prove that an algorithm takes $\Theta(n^2)$ worst case time, it is possible that it takes $O(n)$ on some inputs. For example, lets' take insertion sort. It has worst case time of $\Theta(n^2)$ (This is basically an asymptotic tight bound). But in best case, it takes $O(n)$ on inputs that are already sorted.

2. Let $f(n)$, $g(n)$ are 2 non-negative functions over non-negative input. Prove or disprove the following and justify your claims via formal mathematical arguments.

- $f(n) + o(f(n)) = \Theta(f(n))$

Sol: $\Rightarrow o(f(n)) = \{ g(n)$: for any +ve constant $c > 0$, $\exists$ constant $n_0 > 0$ such that $0 \le g(n) < c f(n)$ $\forall\ n \ge n_0 \}$

$\Rightarrow$ let $g(n) = o(f(n))$ and $h(n) = f(n) + g(n)$

$\Rightarrow$ Given $g(n) \ge 0$. So, $\boxed{h(n) \ge 1 f(n)}$ — ①

Let $\boxed{c_1 = 1}$

$\Rightarrow$ As $g(n) = o(f(n))$, ~~we take $0 \le f(n) < c g(n)$~~

$g(n) < c f(n)$ for any +ve constant $c > 0$

Let $c = 1$ here. So, $g(n) < f(n)$ $\forall\ n \ge n_0$ (say)

$h(n) = f(n) + g(n)$ $\boxed{h(n) \le 2 f(n)}$ $\forall\ n \ge n_0$ — ②

Let $\boxed{c_2 = 2}$

$\Rightarrow$ Combine both equations ① and ②

$\Rightarrow 0 \leq \frac{1}{2} f(n) \leq h(n) \leq 2f(n) \quad \forall\ n \geq n_0$

$\Rightarrow$ By definition,

$\Theta(g(n)) = \{p(n) : \exists$ +ve constants c_1, c_2, n_0 such that $0 \leq c_1 g(n) \leq p(n) \leq c_2 g(n) \quad \forall\ n \geq n_0\}$

$\Rightarrow$ So, $h(n) = f(n) + g(n) = \Theta(f(n))$

$\Rightarrow$ $\boxed{f(n) + o(f(n)) = \Theta(f(n))}$

- $f(n) = O(g(n))$ implies $g(n) = \Omega(f(n))$

Sol: $\Rightarrow$ Given $f(n) = O(g(n))$

$\Rightarrow O(g(n)) = \{f(n) : \exists$ +ve constants c, n_0 such that $0 \leq f(n) \leq cg(n) \quad \forall\ n \geq n_0\}$

$\Rightarrow$ let $0 \leq f(n) \leq c_1 g(n) \quad \forall\ n \geq n_0$

$\Rightarrow 0 \leq \frac{1}{c_1} f(n) \leq g(n) \quad \forall\ n \geq n_0$

$\Rightarrow$ let $c_2 = 1/c_1$

$\Rightarrow 0 \leq c_2 f(n) \leq g(n) \quad \forall\ n \geq n_0$

$\Rightarrow$ By definition of Ω notation,

$\Omega(f(n)) = \{g(n) : \exists$ +ve constants c, n_0 such that $0 \leq cf(n) \leq g(n) \quad \forall\ n \geq n_0\}$

$\Rightarrow g(n) = \Omega(f(n))$

$\Rightarrow$ $\boxed{f(n) = O(g(n)) \rightarrow g(n) = \Omega(f(n))}$

- $f(n) + g(n) = \Theta(\min\{f(n), g(n)\})$

Sol: $\Rightarrow$ By definition,

$\Theta(g(n)) = \{f(n) : \exists$ +ve constants c_1, c_2, n_0 such that $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \quad \forall\ n \geq n_0\}$

$\Rightarrow$ This statement given is false and can be shown as false with a counter example of polynomial running times.

⇒ Let $f(n) = n^{k_1}$, $g(n) = n^{k_2}$ where $k_1 > k_2$

$\min\{f(n), g(n)\} = \min\{n^{k_1}, n^{k_2}\} = n^{k_2} = g(n)$

$h(n) = f(n) + g(n) = n^{k_1} + n^{k_2}$

As $k_1 > k_2$, $h(n)$ is $\Theta(n^{k_1})$ as we know that n^{k_1} is the term with highest degree.

But as per statement $\Theta(\min\{f(n), g(n)\})$ ~~$\Theta(n^{k_2})$~~

i.e $h(n)$ is $\Theta(n^{k_2})$. This is false

⇒ $\boxed{f(n) + g(n) \neq \Theta(\min\{f(n), g(n)\})}$

- $n! = o(n^n)$ and $n! = \omega(2^n)$

Sol: Part 1: Prove that $n! = o(n^n)$

⇒ $o(g(n)) = f(n)$ iff $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$.

⇒ $L = \lim_{n\to\infty} \frac{n!}{n^n}$ ⇒ ~~$L = 0$~~ —① $\boxed{\frac{n!}{n^n} > 0}$ —①

⇒ ~~$L = \lim_{n\to\infty} \frac{n(n-1)(n-2)\cdots 1}{n \cdot n \cdots n}$~~ $< \lim_{n\to\infty}$ ($n-1$ times)

⇒ $\frac{n!}{n^n} = \frac{n(n-1)\cdots 1}{n \cdot n \cdots n} < \frac{n \cdot n \cdots n \cdot 1}{n \cdot n \cdots n \cdot n} < \frac{1}{n}$

$\boxed{\frac{n!}{n^n} < \frac{1}{n}}$ —②

⇒ By ① and ②, $0 < \frac{n!}{n^n} < \frac{1}{n}$.

$\lim_{n\to\infty} 0 = \lim_{n\to\infty} \frac{1}{n} = 0$. By Sandwich theorem,

$\boxed{L = 0}$. $\boxed{n! = o(n^n)}$

Part 2: Prove that $n! = \omega(2^n)$.

⇒ $\omega(g(n)) = f(n)$ iff $\lim_{n\to\infty} \frac{g(n)}{f(n)} = 0$

⇒ Let $L = \lim_{n\to\infty} \frac{2^n}{n!}$. ⇒ $\boxed{\frac{2^n}{n!} > 0}$ —①

⇒ ~~$\frac{2^n}{n!} = \frac{2 \cdot 2 \cdots 2^n}{2 \cdot}$~~

$\Rightarrow n! = 1 \cdot 2 \cdot \cdots \cdot n \geq 1 \cdot 2 \cdot 3 \cdot 3 \cdot \cdots \cdot 3$ ($n-2$ terms)

$\Rightarrow n! \geq 2 \cdot 3^{n-2}$

$\Rightarrow \frac{2^n}{n!} \leq \frac{2^n}{2 \cdot 3^{n-2}} \leq \frac{9}{2}\left(\frac{2}{3}\right)^n \Rightarrow \boxed{\frac{2^n}{n!} \leq \frac{9}{2}\left(\frac{2}{3}\right)^n}$ ——②

$\Rightarrow$ By ① and ②,

$0 \leq \frac{2^n}{n!} \leq \frac{9}{2}\left(\frac{2}{3}\right)^n$

$\lim_{n\to\infty} 0 = \lim_{n\to\infty} \frac{9}{2}\left(\frac{2}{3}\right)^n = 0$. By Sandwich theorem,

$L = \lim_{n\to\infty} \frac{2^n}{n!} = 0$. $\boxed{n! = \omega(2^n)}$

---

3. Is the function $\lceil \ln^* n \rceil !$ polynomially bounded?
Is the function $\lceil \ln \ln n \rceil !$ polynomially bounded?

Sol: Part 1:- $f(n) = \lceil \ln . n \rceil !$

$\Rightarrow$ A function $g(n)$ is said to be polynomially bounded if $\exists$ constants c, k such that $h(n) \leq c \cdot n^k$ for all sufficiently large n.

$\Rightarrow$ By Stirling's approximation,

$n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$

$\ln(n!) = \ln\left(\sqrt{2\pi n}\left(\frac{n}{e}\right)^n\right)$

$= \ln\left(\frac{n}{e}\right)^n + \ln\left(\sqrt{2\pi}\right) + \frac{1}{2}\ln n$

$= n(\ln n - 1) + \ln\left(\sqrt{2\pi}\right) + \frac{1}{2}\ln n$

$\ln n! = n \ln n - n + \frac{1}{2}\ln n + \ln\left(\sqrt{2\pi}\right)$

$\boxed{\ln n! = \Theta(n \ln n)}$ ——①

$\Rightarrow \lceil \ln n \rceil ! \leq (\ln n + 1)!$

$\leq e^{\log((\log n+1)!)}$

$= e^{\log((\log n+1)!)}$

$= \Theta(e^{\log((\log n)!)})$ (we can ignore 1 term)

$= \Theta(e^{\log n \log\log n})$ [By eq^n ①]

$\boxed{\lceil \ln n \rceil! = \Theta(n^{\log\log n})}$

↳ So, $\lceil \ln n \rceil!$ is not polynomially bounded

Part 2: $f(n) = \lceil \ln \ln n \rceil!$

$\Rightarrow \lceil \ln \ln n \rceil! \leq (\ln \ln n)!$ (ceil can be approximated to normal term in order calculation)

$= e^{\log(\log\log n)!}$

$= \Theta(e^{\log(\log\log n)!})$

$= \Theta(e^{\log\log n})$

$\Rightarrow$ Take $\log f(n)$ for easy calculation.

$\log(\lceil \ln \ln n \rceil!) = \log((\log\log n)!)$

$= \Theta(\log((\log\log n)!))$

$= \Theta(\log\log n \cdot \log\log\log n)$

$= O(\log\log n \cdot \log\log n)$ [Take $\log\log\log n \leq \log\log n$]

$= \Theta(1 \cdot \log n)$ (Even weaker bound)

$= O(\log n)$

⇒ So, $\log . f(n)$ is of order $O(\log n)$.
⇒ $f(n)$ is polynomially bounded
⇒ $\lceil \ln \ln n \rceil !$ is polynomially bounded

---

4. Solve the following recursions. Mention any assumptions you might be making.

• $T(n^2) = 7T(n^2/4) + cn^2$ and $T(1) = 1$

Sol: ⇒ let $x = n^2$.

$$\Rightarrow T(x) = 7T\left(\frac{x}{4}\right) + cx$$

$$= 7\left(7T\left(\frac{x}{16}\right) + \frac{cx}{4}\right) + cx$$

$$= 7^2 T\left(\frac{x}{4^2}\right) + cx + \frac{7}{4}cx$$

$$= 7^3 T\left(\frac{x}{4^3}\right) + cx + \frac{7}{4}cx + \left(\frac{7}{4}\right)^2 cx$$

- - - - continue

$$T(x) = 7^k T\left(\frac{x}{4^k}\right) + cx \sum_{i=0}^{k-1} \left(\frac{7}{4}\right)^i$$

Take $\frac{x}{4^k} = 1 \Rightarrow \boxed{x = 4^k} \Rightarrow \boxed{k = \log_4 x}$

$$\Rightarrow T(x) = 7^k T(1) + cx\left(\frac{\left(\frac{7}{4}\right)^k - 1}{\frac{7}{4} - 1}\right)$$

Assumption here: choose k to get to base case, so $x = 4^k$.

$$\Rightarrow 7^k = 7^{\log_4 x} = x^{\log_4 7}$$

$$\Rightarrow \sum_{i=0}^{k-1} \left(\frac{7}{4}\right)^i = \frac{\left(\frac{7}{4}\right)^{\log_4 x} - 1}{\frac{3}{4}} = \frac{4}{3}\left(x^{\log_4 \left(\frac{7}{4}\right)} - 1\right)$$

$\Rightarrow T(x) = x^{\log_4^7} + \frac{4}{3}cx\left(x^{\log_4^7 - 1} - 1\right)$

$T(x) = x^{\log_4^7} + \frac{4}{3}c\left(x^{\log_4^7} - \frac{4}{3}cx\right.$

$\Rightarrow \boxed{T(x) = \Theta\left(x^{\log_4^7}\right)}$ $\left(\because \log_4^7 > 1\right)$

$\Rightarrow T(n^2) = \Theta\left(n^{2\log_4^7}\right) = \Theta\left(n^{\log_4^{49}}\right)$

$\boxed{T(n) = \Theta\left(n^{\log_4^7}\right)}$

$T(n) = \left(n^{\log_4^7}\right)\left(1 + \frac{4c}{3}\right) - \frac{4cx}{3}$

- $T(n) = nT(\sqrt{n})$ and $T(2) = 4$.

Assume n to be of the form 2^{2^i}

Sol: $\Rightarrow T\left(2^{2^i}\right) = 2^{2^i} T\left(2^{2^{i-1}}\right)$

$= 2^{2^i} \cdot 2^{2^{i-1}} \cdot 2^{2^{i-2}} \cdots T\left(2^{2^{i-2}}\right)$

continue

$= 2^{2^i} \cdot 2^{2^{i-1}} \cdot 2^{2^{i-2}} \cdots 2^{2^1} \cdot T\left(2^{2^0}\right)$

$= 2^{2^i + 2^{i-1} + \cdots + 2^1}$ → Solve GP $\quad T(2) = 4$

$= 2^{2(2^i - 1)} \cdot T(2)$

$= 2^{2^{i+1} - 2} \cdot 4$

$= 2^{2^{i+1}} \quad T(2) = 2^2$

$= 2^{2^i + 2^i} = 2^{2^i} \cdot 2^{2^i}$

$\Rightarrow T\left(2^{2^i}\right) = \left(2^{2^i}\right)^2$

$\Rightarrow \boxed{T(n) = n^2}$ $\quad \boxed{T(n) = \Theta(n^2)}$

• $T(n) = T(\frac{n}{2}) + 2T(\frac{n}{4}) + \frac{3n}{2}$ (whenever $n > 3$) and $T(1) = 0$, $T(2) = 2$.

Sol: $\Rightarrow T(n) = T(\frac{n}{2}) + 2T(\frac{n}{4}) + \frac{3n}{2}$

$\Rightarrow (T(n) + T(\frac{n}{2})) = 2(T(\frac{n}{2}) + T(\frac{n}{4})) + \frac{3n}{2}$

$\Rightarrow$ Let $f(n) = T(n) + T(\frac{n}{2})$:

$\Rightarrow f(n) = 2 \cdot f(\frac{n}{2}) + \frac{3n}{2}$, $f(2) = T(2) + T(1)$

$f(2) = 2$

$= 2^1 (2f(\frac{n}{4}) + \frac{3n}{4}) + \frac{3n}{2}$

$= 2^2 f(\frac{n}{2^2}) + \frac{3n}{2} + \frac{3n}{2}$

$= 2^k f(\frac{n}{2^k}) + \frac{3nk}{2}$ — continue

Assumption: Choose k such that $n = 2^{k+1}$.

$k = \log_2^n - 1$ ($\frac{n}{2^k} = 2$ for getting base case)

$\Rightarrow f(n) = \frac{n}{2} f(2) + \frac{3n}{2} \log_2^n - \frac{3n}{2}$

$\boxed{f(n) = \frac{3n}{2} \log_2^n - \frac{n}{2}}$

$\Rightarrow T(n) + T(\frac{n}{2}) = \frac{3n}{2} \log_2^n - \frac{n}{2}$

$= n \log_2^n + \frac{n}{2} \log_2^n - \frac{n}{2}$

$= (n \log_2^n) + \frac{n}{2}(\log_2^n - 1)$

$= (n \log_2^n) + (\frac{n}{2} \log_2 \frac{n}{2})$

$\Rightarrow$ By comparing terms, $\boxed{T(n) = n \log_2^n}$

$\Rightarrow$ $T(n) = n \log_2^n$, $T(n) = \Theta(n \log n)$

$T(1) = 0$, $T(2) = 1$

- $T(n) = 4T\left(\frac{n}{2}\right) + n^3$ and $T(1) = 1$

Sol: $\Rightarrow T(n) = 4T\left(\frac{n}{2}\right) + n^3$

$= 4\left(4T\left(\frac{n}{4}\right) + \frac{n^3}{8}\right) + n^3$

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

$= - - - -$ continue

$= 4^k T\left(\frac{n}{2^k}\right) + n^3\left(1 + \frac{1}{2} + \cdots + \left(\frac{1}{2}\right)^{k-1}\right)$

Assumption: To get to base case, choose

$\frac{n}{2^k} = 1$. $n = 2^k$. $k = \log_2^n$

$\Rightarrow 4^k = 4^{\log_2^n} = 4^{\log_4^{n^2}} = n^2$

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

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

$= n^2 + 2n^3 - 2n^2$

$T(n) = 2n^3 - n^2$ $\quad$ $T(n) = \Theta(n^3)$

- $T(n) = T\left(\frac{n}{2}\right) + c \log n$

Sol: $\Rightarrow T(n) = T\left(\frac{n}{2}\right) + c \log n$

$= T\left(\frac{n}{4}\right) + c \log n + c \log \frac{n}{2}$

$= T\left(\frac{n}{4}\right) + 2c \log n - c(\log 1 + \log 2)$

$= - - - -$ continue

$$\Rightarrow T(n) = T\left(\frac{n}{2^2}\right) + 2c\log n - c(\log 2^0 + \log 2^1)$$

$$= \quad - \quad - \quad - \quad \text{continue}$$

$$\Rightarrow T(n) = T\left(\frac{n}{2^k}\right) + kc\log n - c(\log(2^0 \cdot 2^1 \cdot \ldots \cdot 2^{k-1}))$$

$$= T\left(\frac{n}{2^k}\right) + kc\log n - c\log\left(2^{\frac{k(k-1)}{2}}\right)$$

$$= T\left(\frac{n}{2^k}\right) + kc\log n - \frac{ck(k-1)}{2}\log 2$$

Assumption: Let $T(1) = T_1$, be the base case. To get to base case, choose $\frac{n}{2^k} = 1$.

$$n = 2^k \quad k = \log_2 n$$

$$\Rightarrow T(n) = T_1 + \frac{c}{\log 2}(\log n)^2 - \frac{ck(k-1)}{2}\log 2$$

$$\Rightarrow \boxed{T(n) = T_1 + \frac{c}{\log 2}(\log n)^2 - \frac{c\log 2}{2}(\log n)^2 + \frac{c\log 2}{2}\log n}$$

$$\Rightarrow \boxed{T(n) = \Theta(c(\log n)^2)}$$

$\hookrightarrow T(1) = T_1$ is base case

5. Arrange the following function by order of growth. That is, find an ordering among the following functions $f_1, f_2, f_3, \ldots, f_6$ such that $f_1 = \Omega(f_2)$, $f_2 = \Omega(f_3)$, $\ldots$ $f_5 = \Omega(f_6)$. Partition the list into equivalence class such that functions $f(n)$ and $g(n)$ are in the same class (i.e they are of same order), if and only if $f(n) = \Theta(g(n))$. Justify your claims.

Sol: ~~$\Rightarrow f_1(n) = \frac{n^{1.2}}{\log n}$~~ ~~$\Rightarrow f_2(n) = n^2$~~

~~$\Rightarrow f_3(n) = n \log n$~~ ~~$\Rightarrow f_4(n) = 1.1^n$~~

~~$\Rightarrow f_5(n) = 0.9^n$~~ $\Rightarrow f_6(n) = (\log n)^3$

$\Rightarrow$ To make ordering as per question, from f_1 to f_6, functions must be ordered from fastest to slowest growth rate.

$f_1(n) = 1.1^n$ $\longrightarrow$ exponentially $\uparrow$ as $n \to \infty$ because of base > 1

$f_2(n) = n^2$ $\longrightarrow$ polynomial

$f_3(n) = \frac{n^{1.2}}{\log n}$ $\longrightarrow$ $\frac{\text{polynomial}}{\text{logarithmic}}$

$f_4(n) = n \log n$ $\longrightarrow$ ~~polynomial~~ linear $\times$ logarithmic

$f_5(n) = (\log n)^3$ $\longrightarrow$ poly logarithmic

$f_6(n) = (0.9)^n$ $\longrightarrow$ exponentially $\downarrow$ as $n \to \infty$ because of base < 1

$\Rightarrow f_1(n) = 1.1^n = \Omega(n^2) = \Omega(f_2(n))$

$\Rightarrow f_2(n) = n^2 = \Omega\left(\frac{n^{1.2}}{\log n}\right) = \Omega(f_3(n))$

$\Rightarrow f_3(n) = \frac{n^{1.2}}{\log n} = \Omega\left(\frac{n^{1.2}}{\log} n \log n\right) = \Omega(f_4(n))$

$\Rightarrow f_4(n) = n \log n = \Omega((\log n)^3) = \Omega(f_5(n))$

$\Rightarrow f_5(n) = (\log n)^3 = \Omega(0.9^n) = \Omega(f_6(n))$

⇒ Each function above has to be placed in its own equivalence class and so there are 6 equivalence classes. — 1 function in each. This is because no 2 functions above grow with same rate asymptotically i.e no pair $f(n) = \Theta(g(n))$ is possible.

⇒ $\{1.1^n\}$, $\{n^2\}$, $\{\frac{n^{1.2}}{\log n}\}$, $\{n \log n\}$, $\{(\log^8 n)^3\}$, $\{0.9^n\}$ are 6 classes.

Justification for above claim:

⇒ $\lim_{n\to\infty} \frac{1.1^n}{n^2} = \infty = \lim_{n\to\infty} \frac{f_1(n)}{f_2(n)}$ $\boxed{f_1(n) \neq \Theta(f_2(n))}$

⇒ $\lim_{n\to\infty} \frac{n^2}{\frac{n^{1.2}}{\log n}} = \lim_{n\to\infty} \frac{f_2(n)}{f_3(n)} = \lim_{n\to\infty} n^{0.8} \log n$

$= \infty$

⇒ $\boxed{f_2(n) \neq \Theta(f_3(n))}$

⇒ $\lim_{n\to\infty} \frac{\frac{n^{1.2}}{\log n}}{n \log n} = \lim_{n\to\infty} \frac{f_3(n)}{f_4(n)} = \lim_{n\to\infty} \frac{n^{0.2}}{(\log n)^2}$

$= \infty$. $\boxed{f_3(n) \neq \Theta(f_4(n))}$

⇒ $\lim_{n\to\infty} \frac{n \log n}{(\log n)^3} = \lim_{n\to\infty} \frac{f_4(n)}{f_5(n)} = \lim_{n\to\infty} \frac{n}{(\log n)^2}$

$= \infty$ $\boxed{f_4(n) \neq \Theta(f_5(n))}$

$$\Rightarrow \lim_{n\to\infty} \frac{(\log n)^3}{0.9^n} = \lim_{n\to\infty} \frac{f_5(n)}{f_6(n)} = \lim_{n\to\infty} \left(\frac{10}{9}\right)^n (\log n)^3$$

$$= \infty \qquad \boxed{f_5(n) \neq \Theta(f_6(n))}$$

6. What value is returned by the following function? Express your answer as a function of n. Give worst case running time using big-Oh notation.

```
function XYZ(n)
  r := 0
  for i := 1 to n do
    for j := 1 to i do
      for k := j to i+j do
        for l := 1 to i+j-k do
          r := r+1;
  return (r)
```

Sol: Returned value for function:

$$\Rightarrow r = \sum_{i=1}^{n} \sum_{j=1}^{i} \sum_{k=j}^{i+j} \sum_{l=1}^{i+j-k} 1$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{i} \sum_{k=j}^{i+j} \frac{(i+j-k)(i+j-k+1)}{2}$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{i} \sum_{i+j-k=0}^{i} \frac{(i+j-k)(i+j-k+1)}{2}$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{i} \sum_{p=0}^{i} \frac{p(p+1)}{2} \qquad \boxed{p = i+j-k}$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{i} \left(\frac{1}{2} \frac{i(i+1)(2i+1)}{6} + \frac{1}{2} \frac{i(i+1)}{2} \right)$$

$$\Rightarrow r = \sum_{i=1}^{n} \sum_{j=1}^{i} \frac{i(i+1)}{4} \left[\frac{2i+1}{3} + 1 \right]$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{i} \frac{i(i+1)}{4} \frac{2(i+2)}{3}$$

$$= \sum_{i=1}^{n} \frac{i^2(i+1)(i+2)}{6}$$

$$= \frac{1}{6} \sum_{i=1}^{n} (i^4 + 3i^3 + 2i^2)$$

$$\Rightarrow r = \sum_{i=1}^{n} \sum_{j=1}^{i} \sum_{k=j}^{i+j} \sum_{l=1}^{i+j-k} 1$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{i} \sum_{k=j}^{i+j} (i+j-k)$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{i} \sum_{P=0}^{i} P \qquad \boxed{P = i+j-k}$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{i} \frac{P(P+1)}{2} \frac{i(i+1)}{2}$$

$$= \sum_{i=1}^{n} i \cdot \frac{i(i+1)}{2} = \sum_{i=1}^{n} \frac{i^2(i+1)}{2}$$

$$= \frac{1}{2} \sum_{i=1}^{n} (i^3 + i^2)$$

$$\Rightarrow \boxed{r = \frac{n^2(n+1)^2}{8} + \frac{n(n+1)(2n+1)}{12}}$$

Worst case running time with big-Oh:

$\Rightarrow$ As highest degree term of r is $\frac{n^4}{8}$,

$r = \Theta(n^4)$

Worst case run time = $\boxed{O(n^4)}$