Assignment - 1

Sujoy Roy

D21CO003

1 Part - B

1.1 a) If, $f(n) = 100 * 2^n + 8n^2$ prove that f(n) = O(2n) Can you claim that f(n) = O(2n) IF so, prove the same.

Show that $f(n) = 1002^n + 8n^2$ is O(2n).
Assuming, $n > 1$ then,

$$\frac{f(n)}{g(n)} = \frac{100 * 2^n + 8n^2}{2^n} \quad < \quad \frac{100 * 2^n + 8 * 2^n}{2^n}$$

Choose C = 108, note that $n^2 < 2^n$. Thus, $100 * 2^n + 8n^2$ is $O(2^n)$ because $100 * 2^n + 8n^2 \leq 108 * 2^n$ whenever $n > 1$.

1.2 b) Is it correct to say that f(n) = 3n + 8 = Ω(1)?. Given the facts that f(n) = 3n + 3 = Ω(n) and f(n) = 3n + 3 = Ω(1), which one is correct ? Which one would you choose to prescribe the growth rate of f(n) ?

Defination:- f(n) = Ω(g(n)) :
When we say f(n) = Ω(g(n)) where f and g be real valued functions, n a real number. f(n) = Ω(g(n)), i.e., f = Ω(g) if and only if there are positive real constants C and n_0 such that $f(n) \geq C.(g(n))$ for all $n > n_0$. Thus in the function, f(n) = 3n + 8 = Ω(1) we say that $3n + 8 \geq C.1$ Note that C.1 where C = any constant value.
Similarly, f(n) = 3n + 3 = Ω(n) as f(n) = 3n + 3 then $3n + 3 \geq C.n$ for all $C > 1$ and $n > n_0$.
Now we wish to find C and n_0 , that satisfy the conditions of the definition of Ω for f(n) = 3n + 3 = Ω(n). Let C = 1 and solve : $3n + 3 \geq 1.n$. This holds if $n \geq 1$. Let $n_0 = 1$. Then C = 1 and $n_0 = 1$ guarantee that $3n + 3 \geq C.n$ for all $n > n_0$.
f(n) = 3n + 3 = Ω(1) is similar as the function f(n) = 3n + 8 = Ω(1). Therefore we say f(n) = 3n + 3 = Ω(1)
Thus, we conclude that both the function f(n) = 3n + 3 = Ω(n) and f(n) = 3n + 3 = Ω(1) are correct. When we have to choose between two function that are given as f(n) = 3n + 3 = Ω(n) and f(n) = 3n + 3 = Ω(1) to prescribe the growth rate of f(n).

We choose the function which gives us the TIGHT BOUND i.e function that provides nearest growth rate.

1.3 c) Consider the two functions viz. $f(n) = n^2$ and $g(n) = 2n^2$. Which functions growth rate is higher ? Use appropriate asymptotic notation to specify the time complexity of the two functions.

We are given two function $f(n) = n^2$ $g(n) = 2n^2$.
When we say Asymptotic Analysis, our goal is to simplify analysis of running time by getting rid of "details", which may be affected. In Asymptotic Notation we have:
O-Notation , Ω-Notation , θ-Notation.
For the above functions we use -Notation.As both functions satisfies O-Notation Ω-Notation ,
For f(n) = n^2 if we take, C = 1 and $n_0 = 1$ it guarantee that $n2 \leq C.n^2$ for all $n > n_0$. With this we satisfy the condition of the definition of O implying $f(n) = O(n^2)$.
For $g(n) = 2n^2$ if we take, C = 2 and $n_0 = 1$ it guarantee that $n2 \leq C.n^2$ for all $n > n_0$. With this we satisfy the condition of the definition of O implying $g(n) = O(n^2)$.
For $f(n) = n^2$ if we take, C = 1 and $n_0 = 1$ it guarantee that $n^2 \leq C.n^2$ for all $n > n_0$. With this we satisfy the condition of the definition of Ω implying $f(n) = \Omega(n^2)$.
For $g(n) = 2n^2$ if we take, C = 2 and $n_0 = 1$ it guarantee that $2n^2 \geq C.n^2$ for all $n > n_0$. With this we satisfy the condition of the definition of Ω implying $g(n) = \Omega(n^2)$.
We say that f(n) is $\theta(g(n))$ if there exists constants c1, c2, and n_0, such that $c1.g(n) \leq f(n) \leq c2.g(n)$ for $n > n_0$. We also say f(n) is $\theta(g(n))$ if and only if f(n) is O(g(n)) and f(n) is Ω(g(n)).
With above statements, both the functions $f(n) = n^2$ and $g(n) = 2n^2$ satisfies the condition for O and Ω implying that θ is also satisfied. As both the function have same polynomial value in θ - notation i.e $f(n) = \theta(n^2)$ $g(n) = \theta(n^2)$, with this we can infer that both the functions have same growth rate

1.4 d) Prove the following: For any two functions f(n) and g(n), f(n) = (g(n)) only if f(n)=O(g(n) and f(n)=$\Omega(g(n))$.

For a function g(n), (g(n)) is given by the relation:
$\theta(g(n)) = f(n)$: there exist positive constants c1, c2 and n_0 such that 0 c1.g(n) f(n) c2.g(n) for all n n0 The above expression can be described as a function f(n) belongs to the set (g(n)) if there exist positive constants c1 and c2 such that it can be sandwiched between c1.g(n) and c2.g(n), for sufficiently large n. If a function f(n) lies anywhere in between c1.g(n) and c2.g(n) for all $n \geq n_0$, then f(n) is said to be asymptotically tight bound.
For O(g1(n)) = f(n), we say there exist positive constants c and n$_0 such that 0 \leq f(n) \leq c.g_1(n)$ for all $n \geq n_0$. The above expression can be described as a function f(n) belongs to the set O(g1(n)) if there exists a positive constant c such that it lies between 0 and $c.g_1(n)$, for sufficiently large n.
For $\Omega(g2(n)) = f(n)$, we say there exist positive constants c and n_0 such that $0 \leq c.g_2(n) \leq f(n)$ for all $n \geq n_0$. The above expression can be described as a function f(n) belongs to the set $\Omega(g2(n))$ if

there exists a positive constant c such that it lies above $c.g_2(n)$, for sufficiently large n.
As from above statements we assume that for $\theta g(n)$ the function f(n) lie between $c_1.g_1(n) and c_2.g_2(n)$, for all $n > n_0$ where c1 and c2 are two positive real constants and g(n) = g1(n) = g2(n). Then we say the function f(n) satisfies the definition of $O(g(n)) and \Omega(g(n))$ both.
Thus we conclude that for any two functions f(n) and g(n), f(n) = $\theta(g(n))$ only if f(n) = O(g(n)) and f(n) = $\Omega(g(n))$.

1.5 e) Solve the following problems: i) Show that $T(n) = 1 + 2 + 3 + ..n = \theta(n^2)$ ii) Prove that $2n^3n^2 = O(n^3)$ iii) Prove that $7n^2logn + 25000n = O(n^2logn)$

i) $T(n) = 1 + 2 + 3 + ..n = \Theta(n^2)$ The above function T(n) that is given is sum of series where the end of series is n^{th} number.
We solve the series using sum of arithmetic series
Sum = n.(a1 + a2)/2, where a_1 is the first number in the series, a_2 is the last number of the series and n is the total number of elements in the series.
Substituting the values of the series given we get
Sum = n.(1 + n)/2, 1=starting element; n=last element in series; n= total number of element.
We now know the value of T(n) i.e Sum.

$$T(n) = \frac{n \cdot (1 + n)}{2}$$

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

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

When we make Asymptotic Analysis of the function T(n), we take the higher polynomial value of n. We do so because for higher value of n, $\frac{n}{2} \to 0$ as compared to $\frac{n^2}{2}$. Hence, from above statements we conclude that the given series $T(n) = 1 + 2 + 3 + ..n = \theta(n^2)$

ii) $2n^3n^2 = O(n^3)$
When we say f(n) = O(g(n)), we say there exist positive constants c and n_0 such that $0 \leq f(n) \leq c.g_1(n)$ for all $n \geq n_0$.
Let us assume that $f(n) = 2n^3n^2$ and $g(n) = n^3$, now according to the statements $f(n) \leq g(n)$ but instead $f(n) > g(n)$.
Thus the above expression cannot be described as a function f(n) belongs to the set O(g(n)) .

iii) Prove that $7n^2logn + 25000n = O(n^2logn)$
Sum Rule: Given T1(n) = O(f(n)) and T2(n) = O(g(n)), we say

$T_1(n) + T_2(n) = O(max(f(n), g(n))$
Now, let say $T1 = 7n^2logn$ and T2 = 25000n
$\frac{7n^2logn}{25000n} = \frac{7nlogn}{2500} = nlogn$ (neglecting the constants)
From the above statement we conclude that T1 is n log n times greater than T2. Thus, from sum rule we conclude that:
$7n^2logn + 25000n = O(n^2logn)$
we neglect 7 as its a constant and for big values of n it won't make much difference.

1.6 f) If T1(n) = O(f(n)) and T2(n) = O(g(n)) then show that (a) T1(n) + T2(n) = max(O(g(n), O(f(n)) (b) T1(n) T2(n) = O((g(n) (f(n))

a) We can assume, without loss of generality, that,
max(O(f(n)), O(g(n))) = O(f(n)), (Because one has to be larger).
Therefore, some positive n$_o exists, such that, g(n) \leq f(n), when n \geq n_o$

Now, T1(n) = O(f(n)) and T2(n) = O(g(n))
It follows from the definition of "Big Oh" notation, that there must be positive constants c1, c2, n1 and n2 such that,
T1(n) $\leq c1f(n), whenn \geq n1$
$T2(n) \leq c2g(n), whenn \geq n2$

$$Letn = max(n_0, n_1, n_2)$$

. This is valid because n will always be the highest n, preserving the inequality. Therefore

$$T_1(n) \leq c_1 f(n), whenn \geq n$$

$$T_2(n) \leq c_2 g(n), whenn \geq n$$

$$T_1(n) + T_2(n) \leq c_1 f(n) + c_2 g(n), whenn \geq n$$

But $g(n) \leq f(n), whenn \geq n$ So, again using the (highly useful) inequality, we can say that,

$$T_1(n) + T_2(n) \leq c_1 f(n) + c_2 f(n), whenn \geq n$$

$$T1(n) + T2(n) \leq (c1 + c2)f(n), whenn \geq n$$

Let $c = c1 + c2$. This is valid because it doesn't matter for the purposes of this calculation what the value of c is, just that there is one.

$$T1(n) + T2(n) \leq cf(n), whenn \geq n$$

$$T1(n) + T2(n) = O(f(n))$$

But, from our general assumption above, we can say that O(f(n)) = max(O(f(n)), O(g(n))) so,

$$T1(n) + T2(n) = max(O(f(n)), O(g(n)))$$

$$T(n) = max(O(f(n)), O(g(n)))$$

b) By definition, there is a positive constants c1, c2, n1 and n2 such that,

$$T_1(n) \leq c_1 f(n), when n \geq n_1$$

$$T_2(n) \leq c_2 g(n), when n \geq n_2$$

Let n = $max(n1, n2). Therefore, T_1(n) \leq c_1 f(n), when n \geq n$

$$T_2(n) \leq c_2 g(n), when n \geq n$$

Therefore,

$$T_1(n) \cdot T_2(n) \leq (c_1 f(n)) \cdot (c_2 g(n)), when n \geq n$$

$$T1(n) \cdot T2(n) \leq c1 \cdot c2(f(n) \cdot g(n)), when n \geq n$$

Let c = $c1 \cdot c2. Therefore, T_1(n) \cdot T_2(n) \leq c_{(f(n) \cdot g(n)), when n \geq n}$

$$T_1(n) \cdot T_2(n) = O(f(n) \cdot g(n))$$

$$(n) = O(f(n) \cdot g(n))$$

1.7 g) Show that maxf(n), g(n) = (f(n) + g(n))

1. $f(n) \leq f(n) + g(n)$ and $g(n)(n) + g(n)$. Hence, max(f(n),g(n))$\in O(f(n) + g(n))$
2.f(n)+g(n)$\leq 2max(f(n), g(n))$. Hence, max(f(n),g(n))$\in \Omega(f(n) + g(n))$

Therefore, we get that max(f(n),g(n)) $\in \Theta(f(n) + g(n))$

$$max(f(n), g(n)) = \begin{cases} f(n), & \text{if } f(n) \geq g(n) \\ g(n), & g(n) \geq f(n) \end{cases}$$

For instance, if f(n) = 10n and g(n) = n$^2, we get that, max(f(n), g(n)) = \begin{cases} 10n, & \text{if } n \leq 10 \\ n^2, & n \geq 10 \end{cases}$

1.8 i) Prove that if T(x) is a polynomial of degree n, then T(x) = (x^n).

Let, $T(x) = a_nx^n + a_{n-1}x^{n-1} + ... + a_1x + a_0$ be a polynomial in a of degree with $a_n > 0$.

Case 1: T(n) ∈O(x^n)

$T(x) = a_nx^n + a_{n-1}x^{n-1} + ... + a_0$

$T(x) = a_n \cdot x^n \cdot (1 + \frac{a_{n-1}}{a_n} \cdot \frac{1}{x} + \frac{a_o}{a_n} \cdot \frac{1}{x^n})$

$T(x) \leq a_n \cdot x^n \cdot (1 + \frac{|a_{n-1}|}{a_n} \cdot \frac{1}{x} + ... + \frac{|a_o|}{a_n} \cdot \frac{1}{x^n})$

$T(x) \leq a_n \cdot x^n \cdot (1 + \frac{|a_{n-1}|}{a_n} + ... + \frac{|a_o|}{a_n})$

$c = a_n \cdot x^n \cdot (1 + \frac{|a_{n-1}|}{a_n} + ... + \frac{|a_o|}{a_n})$

$T(n) \leq c.x^n$, c $\in Nholds$

Therefore, $T(n) \in O(x^n)$ – eq. 1

Case 2: $T(n) \in \Omega(x^n)$

$T(n) = a_nx^n + a_{n-1}x^{n-1} + ... + a_o$

$T(n) = a_nx^n \cdot (1 + \frac{a_{n-1}}{a^n} \cdot \frac{1}{n} + ... + \frac{a_o}{a_n} \cdot \frac{1}{x^n})$

Expression in parenthesis has limit 1 for x $\to \infty$

$Therefore, Thereis$x$_o \in N$, such that expression in parenthesis is $\geq \frac{1}{2}, x \geq x_o$

Therefore, $c = \frac{1}{2}a_n$ and $x \geq x_o$

$T(n) \geq \frac{1}{2}a_nx^n = c \cdot x^n$ holds

Therefore, $T(n) \in \Omega(x^n)$ – eq. 2

Therefore, From eq. 1 and eq. 2, we can say that $T(x) = \Theta(x^n)$

1.9 j) If P(n) is any polynomial of degree m or less then show that $P(n) = a^0 + a_1n + a_2n^2 + .. + a_mn^m$ then P(n) = O(nm)

Given P(n) is a polynomial of degree m or less

Therefore, Let $P(n) = a_mn^m + a_{m-1}n^{m-1} + ... + a_1m$, be the polynomial with $a_m > 0$.

Therefore,

$P(n) = a_mn^m + a_{m-1}n^{m-1} + ... + a_o$

$P(n) = a_mn^m(1 + \frac{a_{m-1}}{a^m} \cdot \frac{1}{n} + ... + \frac{a_o}{a_m} \cdot \frac{1}{n^m})$

$P(n) \leq a_mn^m(1 + \frac{|a_{m-1}|}{a^m} \cdot \frac{1}{n} + ... + \frac{|a_o|}{a_m} \cdot \frac{1}{n^m})$

$P(n) \leq a_m n^m (1 + \frac{|a_{m-1}|}{a^m} + ... + \frac{|a_o|}{a_m})$
Therefore, $n \geq 1$, with
$c = a_m(1 + \frac{|a_{m-1}|}{a_m} + ... + \frac{|a_o|}{a_m})$
$P(n) \leq c \cdot n^m$, n $\in Nholds$
$Therefore$,P(n) $\in O(n^m)$, Hence Proved

1.10 l) Let A and B be two programs that perform the same task. Let tA(n) and tB(n) respectively denote their values. For each of the following pairs, find the range of n value for which program A is faster than program B:

i. $\mathbf{t}_A(n) = 1000nandt_B(n) = 10n^2$
$ii.t_A(n) = 1000nlog_2nandt_B(n) = n^2$
$iii.t_A(n) = 2n^2andt_B(n) = n^3$
$iv.t_A(n) = 2nandt_B(n) = 100n$

iv) Less then 50

1.11 m) Consider an input array A of n elements. Each element is an n-bit integer except 0. Which sorting algorithm would you recommend for sorting the array ? Why ? What will be the complexity your sorting algorithm ? [Hint: What is the range in which each array value (i.e. a number) i.e. an integer falls into ?]

Radix sort would be recommended in order to sort the array of n elements,because numbers with the same value will appear in the output array in the same order as they do appear in the input array. Therefore the the complexity will be order of N^2times.

1.12 n) Given the following statement viz. Consider an input array a[1..n] of arbitrary numbers. It is given that the array has only O(1) distinct elements. What does the statement imply?

Essentially, it means no matter what the size of the array is (namely, no matter what n is), it will have no more than a fixed constant number of distinct elements. This assumes that you know what the term O(1) means. If not, ask for clarification.