Assign 3/

Q1) a) $1^5 + 2^5 + 3^5 \dots\dots n^5 = \theta(n^6)$.

If $f(n) = \Theta\, g(n)$, then

$f(n) = O\, g(n)$ & $g(n) = O(f(n))$

Considering $f(n) = 1^5 + 2^5 + 3^5 \dots n^5$ and $g(n) = n^6$

for $f(n) = O\, g(n)$;

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

$$\therefore \lim_{n \to \infty} \frac{1^5 + 2^5 + 3^5 \dots n^5}{n^6}$$

$$\Rightarrow \lim_{n \to \infty} \frac{1}{n^6} + \lim_{n \to \infty} \frac{2^5}{n^6} + \dots\dots + \lim_{n \to \infty} \frac{\cancel{n^8}}{n^6}$$

$$\Rightarrow 0 + 0 + 0 \dots\dots + 0 \qquad \left(\because \frac{c}{\infty} = 0\right)$$

$= 0_{//}$

$\therefore 1^5 + 2^5 + 3^5 \dots n^5 = O(n^6)$

Now considering $f(n) = n^6$ & $g(n) = 1^5 + 2^5 + 3^5 \ldots n^{65}$

We know $\sum_{k=1}^{n} f(k) \geqslant \int_0^n f(k)\, dk.$

taking $f(k) = n^5.$

$$\sum_{n=1}^{n} n^5 \geqslant \int_0^n n^5\, dn$$

$$\sum n^5 \geqslant \frac{1}{6} n^6.$$

$$\therefore \quad 6. \sum n^5 \geqslant n^6.$$

$$\therefore \quad n^6 \leq 6.\left(1^5 + 2^5 + 3^5 \ldots n^5\right)$$

$$\therefore \quad f(n) \leqslant c. \, g(n),$$

$$\therefore \quad n^6 = O\left(1^5 + 2^5 + 3^5 \ldots n^5\right)$$

$$\because \quad n^6 = O(1^5 + 2^5 + 3^5 \ldots n^5) \;\&\; (1^5 + 2^5 + 3^5 \ldots n^5) = O(n^6)$$

$$1^5 + 2^5 + 3^5 \ldots n^5 = \Theta(n^6)$$

b) Prove $n^3 + 2n$ is divisible by 3.

<u>Induction</u>

let $n=0$.
$n^3 + 2n = 0$ which is divisible by 3.

len $n=1$.

$1 + 2(1) = 3$ which is divisible by 3

Now Assuming $n^3 + 2n$ is divisible by 3.

$\therefore$ $f(k) = k^3 + 2k$ which is divisible by 3

for $f(k+1)$:-

$f(k+1) = (k+1)^3 + 2(k+1)$

$\Rightarrow$ $f(k+1) = k^3 + 3k^2 + 3k + 1 + 2k + 2$.

$\Rightarrow$ $f(k+1) = k^3 + 2k + 3k^2 + 3k + 3$

$\because$ we know $k^3 + 2k$ is divisible by 3
let $k^3 + 2k = 3.c$ which c is some constant.

$\therefore$ $f(k+1) = 3c + 3k^2 + 3k + 3$
$= 3(c + k^2 + k + 1)$

$\therefore$ $f(k+1)$ is divisible by 3

Thus by induction method $n^3 + 2n$ is divisible by 3.

Q2) Problem:-

To make changes for 'k' cents using denominations $\{s_1, s_2, \ldots s_m\}$.

To compute this, we define

$T[i,j]$ = ~~Minimum~~ number of ways to give 'k' cents

where j = The number of cents whose change is required.

i = The denominations

$$T[0,j] = 0 \quad ; \quad \text{if } j \neq 0$$

$$T[i,0] = 1 \quad ;$$

$$T[i,j] = T[i-1,j] + q[i, w[j] - s[i]]$$

where $w[j]$:- Weight at 'j' column
$s[i]$:- Denomination at 'i' row.

$$T[i,j] = T[i-1,j] \quad \text{if } w[j] < s[i]$$

Taking an example of Total cents = 6.
and denomination = {1, 2, 4, 6}

		0	1	2	3	4	5	6
(0)	0	1	0	0	0	0	0	0
(1)	1	1	1	1	1	1	1	1
(2)	2	1	1	2	2	3	3	4
(3)	4	1	1	2	2	4	4	6
(4)	6	1	1	2	2	4	4	7 → final Answer.

Considering T[2,4].

$T[2,4] = T[1,4] + T[2,(4-2)]$

$= 1 + T[2,2] = 1 + 2 = 3$

Considering T[6,6]

$T[6,6] = T[5,6] + T[6,(6-6)]$

$= 6 + T[6,0] = 6 + 1 = 7$

Time Complexity.

Since a 2-D matrix is formed, there will be 2 for loops. Considering 'n' cents and there are 'm' denominations

$\therefore T(n) = O(n \cdot m)$

Q3) Placing paranthesis for the maximum result in an equation.

For this problem we take 2 arrays; One array stores all the variables and the 2nd array stores all the operations.

eg if Equation => $1 + 5 - 8 + 4$

$A = \{1, 5, 8, 4\}$
$B = \{+, -, +\}$.

To place braces, we introduce another variable 'k' which iterates between i and j to traverse through all possibilities of brackets.

$T[i, j]$ = Maximum amount computed between 'i' and 'j' numbers.

'k' being the position where brackets are to be placed.

$$T[i, j] = \max \{ T[i, k] \text{ operation}(k) \; T[k+1, j]$$

where $i \leq k \leq j$

$$T[i, i] = A[i]$$

Taking Example of the above equation.

$$1 + 5 - 8 + 4$$

	1(1)	2(5)	3(8)	4(4)
(1) 1	1	6	-2	2
(5) 2		5	-3	1
(8) 3			8	12
(4) 4				4

$\longrightarrow$ Final Answer.

$$T[2,4] = \max \begin{cases} T[2,2] - T[3,4] & ; \ k=1 \\ T[2,3] + T[4,4] & ; \ k=2. \end{cases}$$

$$= \max(-6, -3+4)$$

$$= 1$$

$$T[1,4] = \max \begin{cases} T[1,1] + T[2,4] & ; \ k=1 \\ T[1,2] - T[3,4] & ; \ k=2 \\ T[1,3] + T[4,4] & ; \ k=3. \end{cases}$$

$$= \max(2, -6, 2)$$

$$= 2$$

Time Complexity.

The algorithm forms 2-D matrix and also contains k which iterates through i to j. Hence there are 3 for loops

$\therefore \quad T(n) = O(n^3)$

Q4) a) Given Algorithm:-

Pick the highest value of coin less than the given cents and recurse ~~xx~~ the remaining amount.

eg if cents:- 50
coins = {10, 15, 20, 25, 30}

Algorithm first picks 30 since its the highest coin and $30 \leq 50$.

Remaining = 20. (Now recurse on 20)

Algorithm next selects 20 since its the highest coin and $20 \leq 20$.

$\therefore$ Coins = 30 & 20 = 2 no. of coins.

But this algorithm doesn't hold true for all case.

eg cents = 50.
coins = {1, 10, 25, 45}

According to Algorithm, it chooses 45 coin first and then chooses '5' '1' coins
$\therefore$ Total No. of coins = 5+1 = 6.

When the change of 50 can be given by 2 '25' coins
$\therefore$ Total No. of coins = 2,
Hence the algorithm is not optimal

CamScanner