Name - Rohan Ghosh   Batch - BCS2B
Id - 181001001122

: Algo - Assignment

Q.1) Matrix Chain multiplication.

a) $<5, 10, 3, 12, 5, 50, 6>$

A_1	A_2	A_3	A_4	A_5	A_6
5x10	10x3	3x12	12x5	5x50	50x6

$$m[1,3] = \min_{1 \le s < 3} \begin{cases} s=1: & m[1,1] + m[2,3] + 5\times10\times12 \\ & = 0 + 360 + 600 = 960 \\ s=2: & m[1,2] + m[3,3] + 5\times3\times12 \\ & = 150 + 0 + 180 = 330 \checkmark \end{cases}$$

$$m[1,4] = \min_{1 \le s \le 4} \begin{cases} s=1: & m[1,1] + m[2,4] + 5\times10\times5 \\ & = 0 + 330 + 250 = 580 \\ s=2: & m[1,2] + m[3,4] + 5\times3\times5 \\ & = 150 + 180 + 75 = 405 \checkmark \\ s=3: & m[1,3] + m[4,4] + 5\times12\times5 \\ & = 330 + 0 + 300 = 630 \end{cases}$$

$$m[2,4] = \min_{2 \le s \le 4} \begin{cases} s=2: & m[2,2] + m[3,4] + 10\times3\times5 \\ & = 0 + 180 + 150 = 330 \checkmark \\ s=3: & m[2,3] + m[4,4] + 10\times12\times5 \\ & = 360 + 0 + 600 = 900 \end{cases}$$

$m[4,6] = \min_{4 \le s \le 6}$

$$
\begin{cases}
s=4 & m[4,4] + m[5,6] + 12\times5\times6 \\
 & = 0 + 1500 + 360 = 1860 \checkmark \\
s=5 & m[4,5] + m[6,6] + 12\times50\times6 \\
 & = 3000 + 0 + 3600 = 6600
\end{cases}
$$

$m[3,5] = \min_{3 \le s \le 5}$

$$
\begin{cases}
s=3 & m[3,3] + m[4,5] + 3\times12\times50 \\
 & = 0 + 3000 + 1800 = 4800 \\
s=4 & m[3,4] + m[5,5] + 3\times5\times50 \\
 & = 180 + 0 + 750 = 930 \checkmark
\end{cases}
$$

$m[1,5] = \min_{1 \le s \le 5}$

$$
\begin{cases}
s=1 & m[1,1] + m[2,5] + 5\times10\times50 \\
 & = 0 + 2430 + 2500 = 4930 \\
s=2 & m[1,2] + m[3,5] + 5\times3\times50 \\
 & = 150 + 930 + 750 = 1830 \\
s=3 & m[1,3] + m[4,5] + 5\times12\times50 \\
 & = 330 + 3000 + 3000 = 6330 \\
s=4 & m[1,4] + m[5,5] + 5\times5\times50 \\
 & = 405 + 0 + 1250 = 1655
\end{cases}
$$

$m[1,6] =$

$$
\begin{cases}
s=1 & m[1,1] + m[2,6] + 5\times10\times6 \\
 & = 0 + 1950 + 300 = 2250 \\
s=2 & m[1,2] + m[3,6] + 5\times3\times6 \\
 & = 150 + 1770 + 90 = 2010 \\
s=3 & m[1,3] + m[4,6] + 5\times12\times6 \\
 & = 330 + 1860 + 360 = 2550 \\
s=4 & m[1,4] + m[5,6] + 5\times5\times6 \\
 & = 405 + 1500 + 150 = 2055 \\
s=5 & m[1,5] + m[6,6] + 5\times50\times6 \\
 & = 1655 + 0 + 1500 = 3155
\end{cases}
$$

$$m[2,5] = \min_{2 \le s \le 5} \begin{cases} s=2: & m[2,2] + m[3,5] + 10\times3\times50 \\ & = 0 + 930 + 1500 = 2430 \\ s=3: & m[2,3] + m[4,5] + 10\times12\times50 \\ & = 360 + 3000 + 6000 = 9360 \\ s=4: & m[2,4] + m[5,5] + 5\times10\times50 \\ & = 330 + 0 + 2500 = 2830 \end{cases}$$

$$m[3,6] = \min_{3 \le s \le 6} \begin{cases} s=3: & m[3,3] + m[4,6] + 12\times3\times6 \\ & = 0 + 1960 + 216 = 2076 \\ s=4: & m[3,4] + m[5,6] + 3\times5\times6 \\ & = 150 + 1500 + 90 = 1770 \\ s=5: & m[3,5] + m[6,6] + 3\times50\times6 \\ & = 930 + 0 + 900 = 1830 \end{cases}$$

$$m[2,6] = \min_{2 \le s < 6} \begin{cases} s=2: & m[2,2] + m[3,6] + 10\times3\times6 \\ & = 0 + 1770 + 180 = 1950 \\ s=3: & m[2,3] + m[4,6] + 10\times12\times6 \\ & = 360 + 1860 + 720 = 2940 \\ s=4: & m[2,4] + m[5,6] + 10\times5\times6 \\ & = 330 + 1500 + 300 = 2130 \\ s=5: & m[2,5] + m[6,6] + 10\times50\times6 \\ & = 2430 + 0 + 3000 = 5430 \end{cases}$$

M

	1	2	3	4	5	6
1	0	150	330	405	1655	2010
2	—	0	360	330	2430	1950
3	—	—	0	180	930	1770
4	—	—	—	0	3000	1860
5	—	—	—	—	0	1500

	1	2	3	4	5	6
1	–	1	2	2	4	2
2	–	–	2	2	2	2
3	–	–	–	3	4	4
4	–	–	–	–	4	4
5	–	–	–	–	–	5
6	–	–	–	–	–	–

$\therefore$ Cost $= 2010$

$\therefore$ Parenthesis sol^n

$= (A.A_2).(A_3.A_4).(A_5.A_6)$

b) $\langle 3,5,4,6 \rangle$

A_1	A_2	A_3
3×5	5×4	4×6

M

	1	2	3
1	0	60	132
2	–	0	120
3	–	–	0

$m[1,2] = 60$

$m[2,3] = 5\times4\times6 = 120$

$$m[1,3] = \min_{1\leq s<3} \begin{cases} s=1: & m[1,1] + m[2,3] + 3\times5\times6 \\ & = 0+120+90 = 210 \\ s=2: & m[1,2] + m[3,3] + 3\times4\times6 \\ & = 60+0+72 \\ & = 132 \end{cases}$$

S

	1	2	3
1	–	1	2
2	–	–	2
3	–	–	–

$\therefore$ Cost $= 132$

$\therefore$ Parenthesis soln: $(A_1.A_2).(A_3)$

d) $\langle 10,100,5,50 \rangle$

A_1	A_2	A_3
10×100	100×5	5×50

$m[1,2] = 5000$

$m[2,3] = 2500$

M

	1	2	3
1	0	5000	7500
2	–	0	25000
3	–	–	0

Scanned with CamScanner

$m[1,3] = \min_{1 \le s < 3}$
$s=1$: $m[1,1] + m[2,3] + 10 \times 100 \times 50$
$= 0 + 25000 + 50000 = 75000$
$s=2$: $m[1,2] + m[3,3] + 10 \times 5 \times 50$
$= 5000 + 0 + 2500$
$= 7500$

S

	1	2	3
1	–	1	2
2	–	–	2
3	–	–	–

$\therefore$ Cost $= 7500$

$\therefore$ Parenthesis soln: $(A_1 \cdot A_2) \cdot (A_3)$

d) $\langle 5, 4, 6, 2 \rangle$

A_1	A_2	A_3
5×4	4×6	6×2

$m[1,2] = 120$
$m[2,3] = 48$

m

	1	2	3
1	0	120	88
2	–	0	48
3	–	–	0

$m[1,3] = \min_{1 \le s < 3}$
$s=1$: $m[1,1] + m[2,3] + 5 \times 4 \times 2$
$= 0 + 48 + 40 = 88$
$s=2$: $m[1,2] + m[3,3] + 5 \times 6 \times 2$
$= 120 + 0 + 60 = 180$

S

	1	2	3
1	–	1	1
2	–	–	2
3	–	–	–

$\therefore$ Cost $= 88$

$\therefore$ Parenthesis soln:
$(A_1) \cdot (A_2 \cdot A_3)$

Q.2) Greedy algorithm for fractional Knapsack

a) $n = 4$
$m = 5$
$(w_i, P_i) : (2,3), (3,4), (4,5), (5,6)$

	1	2	3	4
P_i :	3	4	5	6
w_i :	2	3	4	5
P_i/w_i =	1.50	1.33	1.25	1.20

Priority of selection : i = | 1 | 2 | 3 | 4 |

Sol^n: 2 of P_1 + 3 of P_2
= 3 + 4
= 7.

b) $n = 4$
$m = 10$
$(w_i, P_i) = (2,10), (4,40), (6,30), (3,50)$

	1	2	3	4
P_i :	10	40	30	50
w_i :	2	4	6	3
P_i/w_i :	5	10	5	16.6

Priority of selection: i = | 4 | 2 | 1 | 3 |

Sol^n = 3 of P_4 + 4 of P_2 + 3 of P_3
= 50 + 40 + (3 × 5)
= 105

$m = 20$

(w_i, P_i) : (18,25), (15,24), (10,15)

	(1)	(2)	(3)
P_i :	25	24	15
w_i :	18	15	10
P_i/w_i :	1.38	1.6	1.5

Priority of selection : i = | 2 | 3 | 1 |

$\therefore$ Soln : 15 of P_2 + 5 of P_3

$= 24 + 5 \times 1.5$

$= 24 + 7.5$

$= 31.50$

Q.3) Tradeoff b/w Dynamic, greedy & Divide & conquer algorithm.

-> For divide & conquer, each problem is independent. So we can follow any order to solve them.

In greedy algorithms, choose the best sub-problem of a ~~greedy~~ given problem.

In case of dynamic programming, we solve many sub problems and only select the optimal sub problems which will contribute to solving larger problems, not all are selected for purpose.

Q. 4) (0-1) Knapsack problem

a) $n = 4$, $M = 8$

v_i	15	10	9	5
w_i	1	5	3	4

→

	1	2	3	4
v_i :	15	10	9	5
w_i :	1	5	3	4

← w →

i	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1	0	15	15	15	15	15	15	15	15
2	0	15	15	15	15	15	25	25	25
3	0	15	15	15	24	24	25	25	25
4	0	15	15	15	24	24	25	25	25

⇒ 25 from 2nd row $[P_2 = 10]$

$25 - 10 = 15$

15 from 1st row $[P_1 = 15]$

$15 - 15 = 0$

∴ sequential decision answer →

$$\begin{Bmatrix} x_1 & x_2 & x_3 & x_4 \\ 1, & 1, & 0, & 0 \end{Bmatrix}$$

Scanned with CamScanner

3) Value = [20, 5, 10, 40, 15, 25]

weight = [1, 2, 3, 8, 7, 4]

M = 10

→

	1	2	3	4	5	6
v_i :	20	5	10	40	15	25
w_i :	1	2	3	8	7	4

w →

i	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	20	20	20	20	20	30	20	20	20	20
2	0	20	20	25	25	25	25	25	25	25	25
3	0	20	20	25	30	30	35	35	35	35	35
4	0	20	20	25	30	30	35	35	40	60	60
5	0	20	20	25	30	30	35	35	40	60	60
6	0	20	20	25	30	45	45	50	55	60	60

⇒ 60 from 4th row [$v_4 = 40$]

60 − 40 = 20

⇒ 20 from 1st row [$v_1 = 20$]

20 − 20 = 0

∴ Sequential decision answer:

$$\left\{ \begin{array}{cccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\ 1, & 0, & 0, & 1, & 0, & 0 \end{array} \right\}$$

c) $n = 4$; $M = 5$

v_i	3	4	5	6
w_i	2	3	4	5

⟵ w ⟶

i \ w	0	1	2	3	4
0	0	0	0	0	0
1	0	3	3	3	3
2	0	3	4	4	7
3	0	3	4	5	7
4	0	3	4	5	7

=> 7 from 2nd row $[v_2 = 4]$
$7 - 4 = 3$

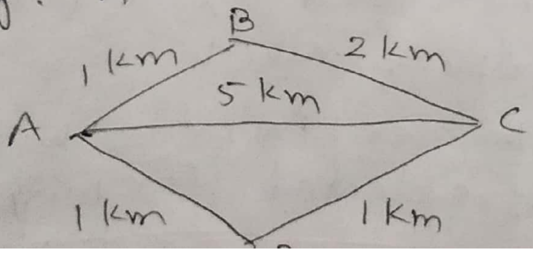
=> 3 from 1st row $[v_1 = 3]$
$3 - 3 = 0$

Answer: $\begin{Bmatrix} x_1 & x_2 & x_3 & x_4 \\ 1, & 1, & 0, & 0 \end{Bmatrix}$

Q.5) Describe main ideas behind greedy algorithm with an example other than knapsack algorithm

=> Main idea behind greedy algorithm is to make best use of memory / time and take the least option available to execute the task. One example is to follow shortest route b/w 2 cities.

Eg: A ⟶ C



we will follow A → D → C route to reach from city A to city C as per greedy algorithm.

* Example when greedy algorithm does not yield an optimal soln.

→ Suppose I have Rs.100. I want to buy toffees. Shop A sells packets of 50 toffees for Rs. 50/- and shop B sells packets of 70 toffees for Rs. 60/- and now if I go by greedy algorithm I will go by shop B and have 70 toffees with Rs. 40/- left but it was optimal if I buy 2 packets from shop A to have 100 ~~tc~~ toffees and no left money. this is more optimal in terms of toffee acquired.

* Algorithm for min^m no. of denominations for given value of V, with supply of {1, 2, 5, 10, 20, 50, 100, 500, 2000} currencies.

Eg: V = 70

Output: 1, 50/- note and 1 20/- note.

⇒ (1) Create an array ARR with all given denominations in descending order.

(2) Check the largest available denomination which is less than or equal to given value of V.

(3) Create an counter array COUNT of same size as ARR and initialize all elements with 0.

(4) At step 2, when we get the largest denominator, we add 1 to the same index in COUNT array as in ARR.

(5) Subtract the min^{m} denomination found from V and the value of V is updated with it.

(6) Same step is repeated from step 2, until V is 0.

(7) The count array is consulted for positive counter of adjoining element/ denomination in ARR array.

As, we are manually creating the array without any sorting technique i.e. will be involved, the time complexity

$O(cn)$ → where c is the time for each iteration of loop for the work inside it, n is no. of times the loop will run, the total value of denomination.

So, $T(n) = O(n)$.

Scanned with CamScanner