Page No.:
Date:
YOUVA

* Greedy Algorithm ~~or Dynamic Programming~~
  - based on past choices
  - optimization can be solved
  - cannot prove correctness of greedy
  - running time is less compared to divide & conquer
  - Most Algo ↑ Do not lead to optimal sol^n, but we can be sure we ~~can~~ are near to optimal solution
* Divide & Conquer
  - Independent of previous choice
  - optimization cannot be solved
  - Easy to prove correctness
  - running time is more

$B_3\ B_2\ B_1\ B_0$
1 0 1 1

⇒ Knapsack problem $(W = 60\,kg)$

Item	Weight	Value	$(v/w)_i$	$(v/w)_i$, desc
1	5	30	6	6 (1)
2	10	40	4	4 (2)
3	15	45	3	3.6 (5)
4	22	77	3.5	3.5 (4)
5	25	90	3.6	3 (3)

Weight selected → $5 + 10 + 25 + \left(\frac{60 - 40}{22}\right)$

Profit → $5 \times 6 + 10 \times 4 + 25 \times 3.6 + \left(\frac{60 - 40}{22}\right) \times 77$

$= 230$

Page No.:
Date:

Capacity = 50

Q

Item	Weight	Value	
	10	60	6
	20	80	4
	40	100	2.5

Weight → 10 + 20 + $\left(\frac{50-30}{40}\right)$ → Here multiply value

Profit → $10 \times 6 + 20 \times 4 + \frac{1}{2} \times 100$

= 140 + 50 = 190

Capacity = 10

Q

Item	Weight	Value	$\left(\frac{\text{Value}}{\text{weight}}\right)$	(V/w) des
1	4	12	3	50 - (5)
2	8	32	4	20 - (3)
3	2	40	20	5 - (4)
4	6	30	5	4 - (2)
5	1	50	50	3 - (1)

Weight = 1 + 2 + 6 + $\left(\frac{10-9}{8}\right)$

Profit = $1 \times 50 + 2 \times 20 + 6 \times 5 + \frac{10-9}{8} \times 32$

= 120 + 4

= 124

→ Flyood $O(n^3)$

⇒ Dynamic Programming.

Similarities bet^n Dynamic & Greedy approach.

1) Both are applicable to the problem which have optimal sub-structure property
2) Both ~~have~~ are used for optimisation

Optimal Sub-Structure propery :-

→ The optimal sol^n to the problem always contains optimal sol^{n}s to the subproblems.

Dynamic	Greedy
→ Time complexity is more	→ Time complexity is less.

⇒ Techniques used to design DP problem

1) Tabulation → ~~Bot~~
2) Memoization

Tabulation	Memoization
→ Bottom up approach	→ Top-down approach
→ Efficient more because we start from basic and will move to final goal	→ Less efficient because we start from advance & then will see what is needed consequently i.e. decision-making is not perfect

⇒ Matrix - chain multiplication

→ Let there is chain of matrices ABCDE... we need to put the parenthesis i.e. multiply in such a way that number of scalar multiplication will be as minimum as possible.

→ This process is known as characterization.

→ See chap 15 of book.

(A(B(C(D E))))

~~A (B(C D))E~~

A B C D

→ Formula is $\frac{(2n)!}{(n+1)!n!}$ = $\frac{8!}{5!4!}$ =

This is the formula of catalan numbers

Let P(n) is the parenthesization for n matrices

$$P(n) = \begin{cases} 1 \\ \sum_{k=1}^{n-1} P(k)\, P(n-k) \end{cases}$$

$$P(4) = P(1)P(3) + P(2)P(2) + P(3)P(1)$$

$$P(3) = P(1)P(2) + P(2)P(1)$$

$$P(2) = P(1)P(1) = 1$$

$$\therefore P(3) = 1 + 1 = 2$$
$$P(4) = 2 + 1 + 2 = 5$$
~~$P(5)$~~

$$P(5) =$$

$$m[i,j] = \begin{cases} 0 & \text{if } i=j \\ \min_{i \le k < j} \{ m[i,k] + m[k+1,j] + P_{(i-1)} P_{(k)} P_{(j)} \} & \text{if } i<j \end{cases}$$

- $m[1,2] = P(0) \cdot P(1) \cdot P(2)$ (You can cross check by formula)

  $m[3,4] = P(2) \cdot P(3) \cdot P(4)$. (For consecutive numbers.

⇒ Matrix chain order (code) from textbook page 375
(Yaad rakhvano algo exam mate 2 vaar puchayeda)

Matrix chain order accepts sequence of dimensions as. ~~as~~ input, also number of matrices is (total length of dimensions - 1).

$m[1,3] = \{ m[1,1] + m[2,3] + p(0)p(1)p(3)$
$\{ m[1,2] + m[3,3] + p(0)p(2)p(3)$

$m[2,3] = p(1) \cdot p(2) \cdot p(3)$

$p(0)\ p(1)\ p(2)\ p(3) + p(0)\ p(1)\ p(3)$

Don't write this term.

$p(0)\ p(2)\ p(3).$

$(i \le k < j-1)$

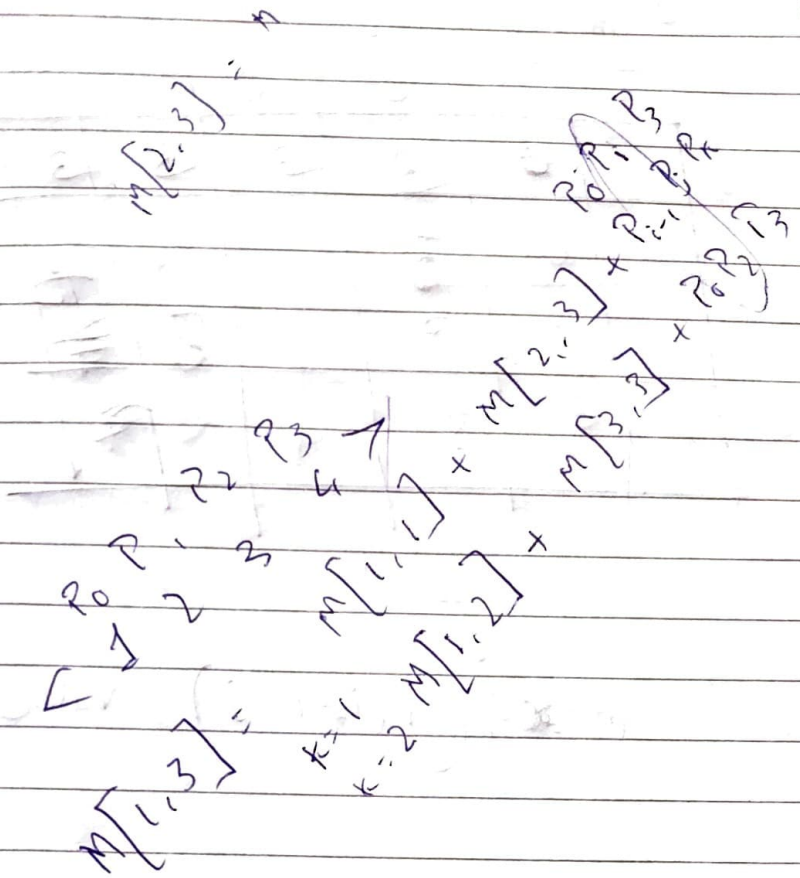
$$m[1,4] = \begin{cases} m(1,1) + m(2,4) + p_0 p_1 p_4 \\ m[1,2] + m(3,4) + p_0 p_2 p_4 \\ m[1,3] + m[4,4] + p_0 p_3 p_4 \end{cases}$$

$m(1,2) = p_0 p_1 p_2$

$m(1,0)$

$p_1 p_2 p_4.$

$$m(2,4) = \{ m(2,2) + m(3,4) + p_1 p_2 p_4$$



⇒ 0-1 knapsack problem.

knapsack cap = 5

Item	Weight	Value
1	2	3
2	3	4
3	4	5
4	5	6

Item vs knapsack capacity

Page No.
Date.

j = Capacity

weight	Item i	0	1	2	3	4	5
	0	0	0	0	0	0	0
3	1	0	0	0	3	3	3
4	2	0	0	0	3	4	4
5	3	0	0	0	3	4	5
6	4	0	0	0	3	4	5

$$T(i,j) = \max\{T(i-1,j),\ value + T(i-1, j-weight)\}$$

$T(1,5)$

$T(0,5)$

$3 + 0$

$4 + T(1,2)$

weight 1, 2, 3

value 10, 15, 40

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	10	10	10	10	10	10
2	0	10	15	25	25	25	25
3	0	10	15	40	50	55	65

40

Page No.
Date:

Items → 5
Weight → 2, 5, 3, 8, 1
Value → 3, 5, 5, 9, 3

	j: 0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1	0	0	3	3	3	3	3	3	3
2	0	0	3	3	3	5	5	8	8
3	0	0	3	5	5	8	8	8	10
4	0	0	3	5	5	8	8	8	10
5	0	3	3	6	8	8	11	11	13

	j: 0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1	0	3	3	3	3	3	3	3	3
2	0	3	3	6	6	6	6	6	6
3	0	3	3	6	8	8	11	11	11
4	0	3	3	6	8	8	11	11	11
5	0	3	3	6	8	8			14