Assembly Line Scheduling

→ An optimization problem is one that has multiple feasible soln, each having a specific cost. Our objective is to find best of them
→ Dynamic Programming is used to solve optimization problems
→ Intro. by Richard Bellman in 1955, word dynamic ~~refers to~~ reflects time-varying aspect of the problems. The word programming refers to tabular method (like linear programming) to solve a problem.
→ Solves every sub-problem just once & saves the sol" in a table. Avoid recomputing the sol". every time the sub-problem is encountered.
→ It's not a algorithm but a ~~techn~~ strategy for designing algorithms.

Divide & Conquer	Dynamic Programming
(1) Solves problems by combining ~~sol~~ soln. to sub-problems	(1)
(1) Sub-problems are independent	(1) Sub-problems are not independent
(2) It solves some sub-problems many times.	(2) ~~Solves~~ Solves every sub-problem only-once

Similarity: Solves sub-problems by combining soln. to sub-problems.

How to know if an optimization ~~sub-problem~~ problem can be solved by applying dynamic programming?

DP works if the problem has the fol. 2 properties:

1) Optimal Sub-structure: An optimal soln. to the problem contains within it, optimal soln. to sub-problems

2) Overlapping sub-problems: When a recursive algorithm ~~wi~~ visits the same problem repeatedly, ~~$~~ then the optimization problem has overlapping sub-problems.

Steps to design a Dynamic Programming algorithm:

1) Characterize the structure of an optimal soln.
2) Recursively define the value of an optimal soln.
3) Compute the value of an optimal soln., typically in a bottom-up fashion
4) Construct an optimal soln. ~~for~~ from computed information.

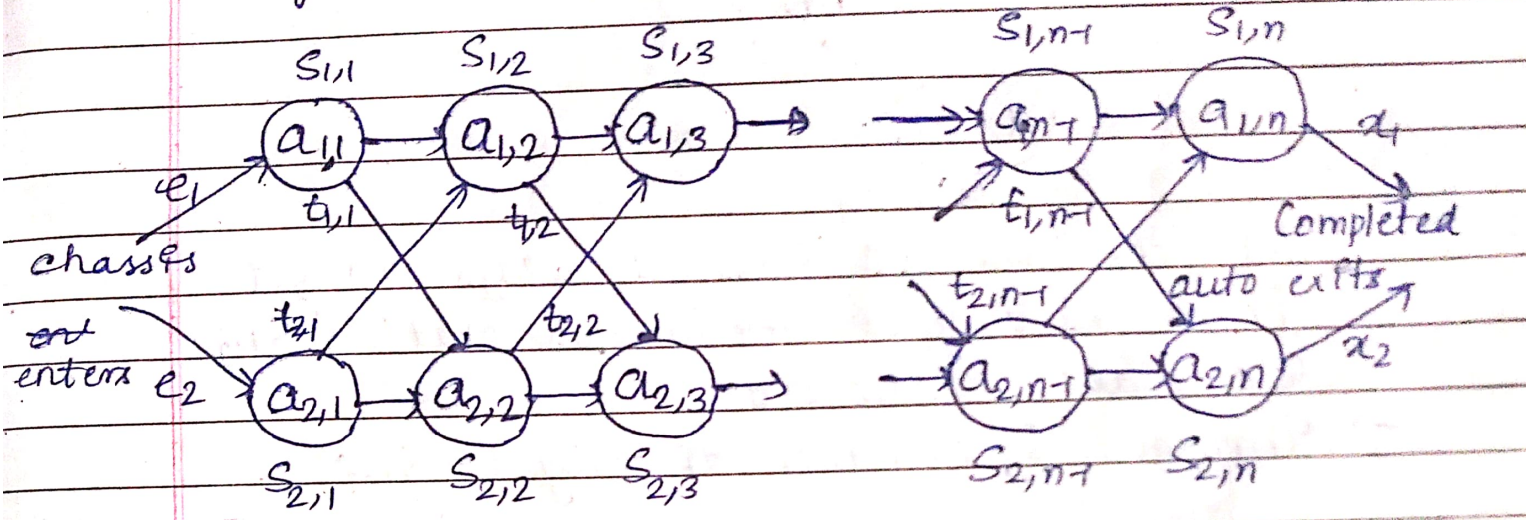
Problem:

Automobile factory has 2 assembly lines
* Each line has n stations:

$S_{1,1}$ ---- $S_{1,n}$ & $S_{2,1}$ ---- $S_{2,n}$

- Corresponding station $S_{1,j}$ & $S_{2,j}$ perform the same func^n but can take diff. amounts of time $a_{1,j}$ & $a_{2,j}$

- Entry times are e_1 & e_2 & exit times are x_1 & x_2



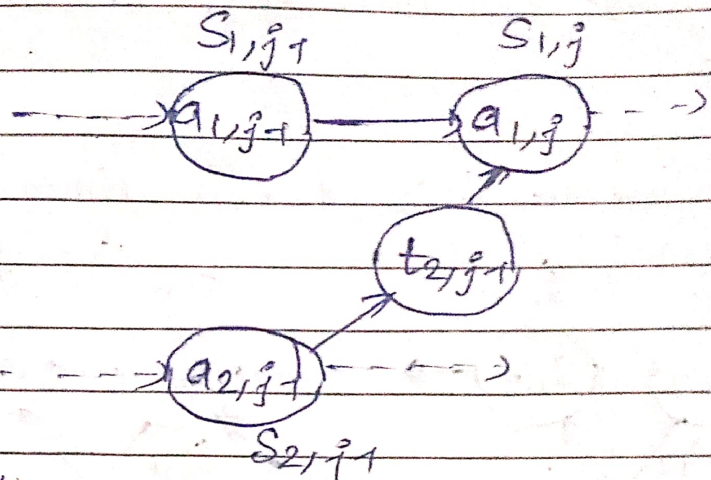
- After going through a station, car can either:
  - → Stay on same line at no cost
  - → Transfer to other line: cost after $S_{i,j}$ is $t_{i,j}$, $j = 1 \cdots n-1$

Problem: what stations should be chosen from line 1 & line 2 in order to minimize total time through the factory for one car.

Brute force soln:

- Enumerate all possibilities of selecting stations
- Compute how long it takes in each case & choose the best one
- There are 2^n possibilities to choose stations
- Infeasible when n is large.

1) Structure of Optimal Soln.:



Let's consider all possible ways to get from starting pt. through station $S_{1,j}$
We have 2 choices . . .

→ Through $S_{1,j-1}$, then directly to $S_{1,j}$

→ —11— $S_{2,j-1}$, —11— transfer over to $S_{1,j}$

- Suppose that the fastest way thro. $S_{1,j}$ is through $S_{1,j-1}$
  → We must have taken fastest way entry through $S_{1,j-1}$
  → If there was a faster way through $S_{1,j-1}$, we would have used it instead.
  → Same is the case with $S_{2,j-1}$

Generalization: An optimal soln. to the problem "find fastest way thro. $S_{1,j}$" contains within it an optimal soln. to sub-problems: "find fastest way through $S_{1,j-1}$ or $S_{2,j-1}$"

→ This is called optimal sub-structure prop.

2) Recursively define the value of an optimal soln.: Define the value of an optimal soln. in terms of optimal soln. to sub-problems

Def^n:

→ f^* : fastest time to get through the entire factory

→ $f_i[j]$: fastest possible time to get a chassis from starting point through station $S_{i,j}$

$$f_1[j] = \begin{cases} e_1 + a_{1,1} & j = 1 \\ \min(f_1[j-1] + a_{1,j},\ f_2[j-1] + t_{2,j-1} + a_{1,j}) & \cdots j \geq 2 \end{cases}$$

$$f_2[j] = \begin{cases} e_2 + a_{2,1} & \cdots j = 1 \\ \min(f_2[j-1] + a_{2,j},\ f_1[j-1] + t_{1,j-1} + a_{2,j}) & \cdots i \geq 2 \end{cases}$$

$$f^* = \min(f_1[n] + x_1,\ f_2[n] + x_2)$$

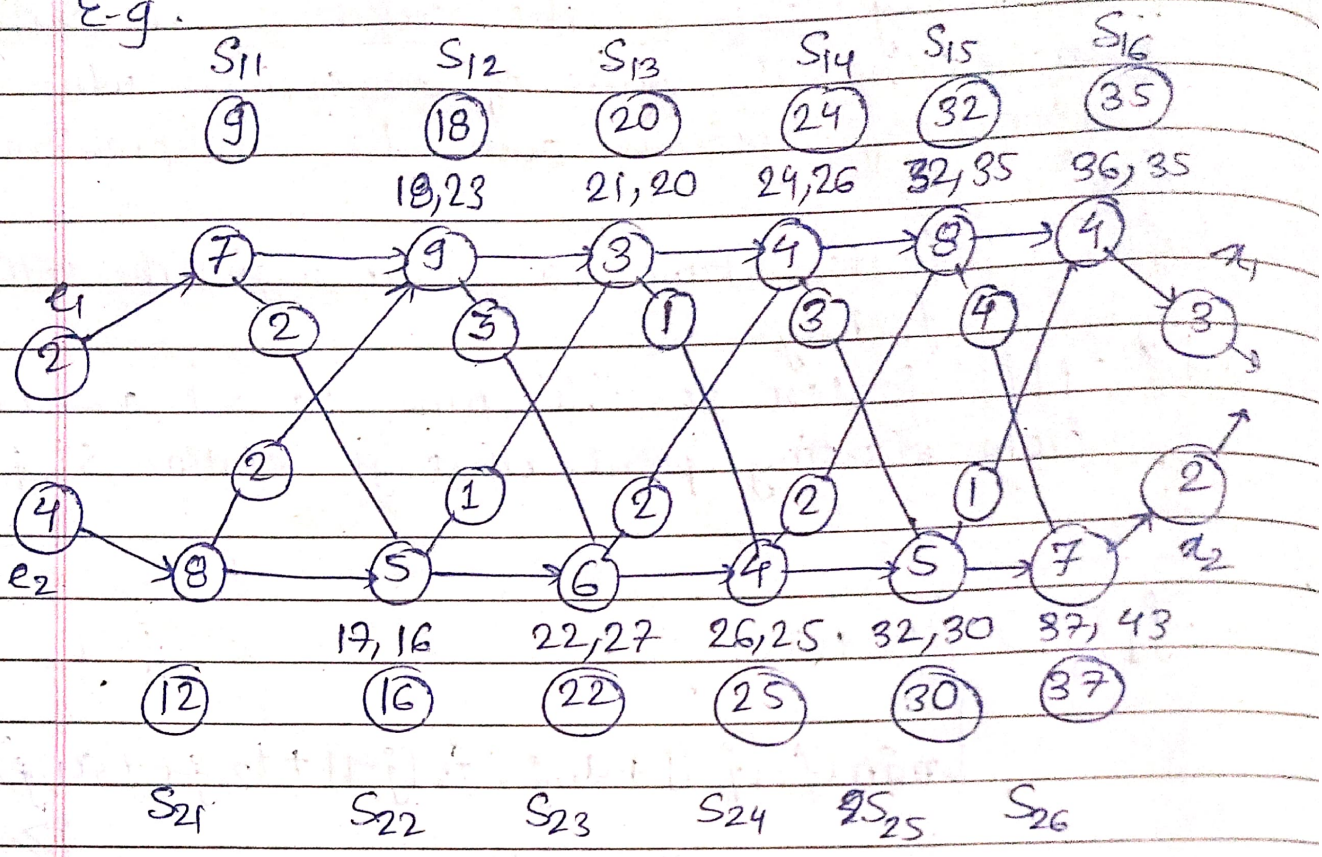
$l_i[j]$ = line no. (1 or 2) whose station $j-1$ is used in a fastest way through $S_{i,j}$.

$j = 2$ to n

$i = 1$ or 2

l^* = Line no. whose station n is used in fastest way through entire factory.

E.g.



j →	1	2	3	4	5	6	Final
$f_1[j]$	9	18	20	24	32	35	38
$f_2[j]$	12	16	22	25	30	37	39

	2	3	4	5	6
$l_1[j]$	1	2	1	1	2
$l_2[j]$	1	2	1	2	2

Time complexity = O(NUM_STATIONS)
Space complexity = O(1)

Longest Common Subsequence

Def^n: Given a sequence $X = \langle x_1, x_2, x_3 \cdots x_m \rangle$ & another sequence $Z = \langle z_1, z_2, z_3 \cdots z_k \rangle$ is a sub-sequence of X if there exists a strictly increasing sequence $\langle i_1, i_2, i_3 \cdots i_k \rangle$ of indices of X such that $\forall\ j = 1, 2 \cdots k$ we have $x_i = z_j$

E.g. $Z = \langle B, C, D, B \rangle$ is a sub-sequence of $X = \langle A, B, C, B, D, A, B \rangle$ with corresponding index sequences $\langle 2, 3, 5, 7 \rangle$
All inc. order of positions

Def^n: Given 2 sequences X & Y, we say that a sequence Z is a <u>common sub-sequence</u> of X & Y, if Z is a <u>sub-sequence of both X & Y</u>.

If $X = \langle A, B, C, B, D, A, B \rangle$
$Y = \langle B, D, C, A, B, A \rangle$

Common sub-sequences are $\langle B, C, A \rangle$, $\langle B, C, B, A \rangle$, $\langle B, D, A, B \rangle$

Def^n: Given 2 sequences $X = \langle x_1, x_2, x_3 \cdots x_m \rangle$ & $Y = \langle y_1, y_2, y_3 \cdots y_n \rangle$ & with to find <u>longest maximum length common sub-sequence of X & Y</u>.

Step 1 Characterizing LCS

Def^n: ith prefix of sequence X - If is ith prefix sequence of X denoted by X_i

If is "i" number prefix symbol of X.

E.g. If $X = \langle A, B, C, B, D, A, B \rangle$

$X_4 = \langle A, B, C, B \rangle$

~~$X_0 = \langle \phi \rangle$~~

$X_0 = \phi$

Optimal Sub-structure of LCS:

Let $X = \langle x_1, x_2 \cdots x_m \rangle$ & $Y = \langle y_1, y_2 \cdots y_n \rangle$ be sequences & let $Z = \langle z_1, z_2, \cdots z_k \rangle$ be any LCS of X & Y.

1) If $x_m = y_n$ & $z_k = x_m = y_n$ implies,

"Z_{k-1} is an LCS of X_{m-1} & Y_{n-1}"

2) If $x_m \neq y_n$ & $z_k \neq x_m$ implies,

"Z is an LCS of X_{m-1} & Y"

3) If $x_m \neq y_n$ & $z_k \neq y_n$ implies

"Z is an LCS of X & Y_{n-1}"

Step 2 : Recursive Soln.

Let $c[i,j]$ = length of LCS of the sequences x_i & y_j.

$$c[i,j] = \begin{cases} 0 & \text{if } i=0 \text{ or } j=0 \\ c[i-1,j-1]+1 & i,j>0 \ \& \ x_i = y_j \\ \max(c[i,j-1], c[i-1,j]) & i,j>0 \ \& \ x_i \neq y_j \end{cases}$$

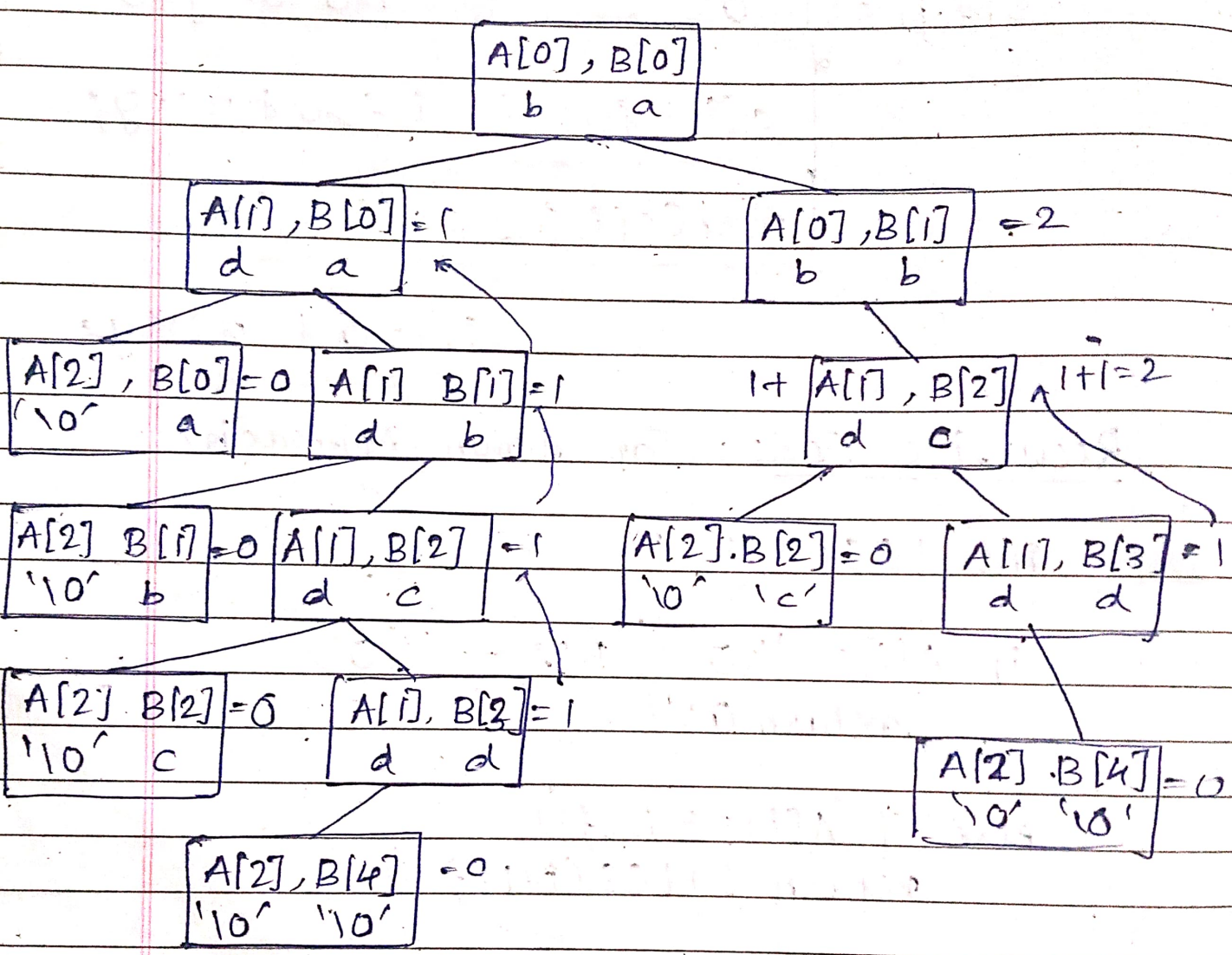
Recursive Code : (Top Down Approach)

```
int LCS(i, j) {
    if (A[i] == '\0' || B[j] == '\0')
        return 0;

    else if (A[i] == B[j])
        return 1 + LCS(i+1, j+1)

    else
        return max(LCS(i+1, j), LCS(i, j+1))
}
```

E.g.

A	b	d	\0
	0	1	2

B	a	b	c	d	\0
	0	1	2	3	4



	a	b	c	d	'\0'
b	2	2			
d	1	1	1	1	
'\0'	0	0	0		0

O(m x n)

Time complexity = $O(mn)$
Space " = $O(mn)$

Dynamic Programming (Bottom-up approach)

$X = a\ b\ a\ a\ b\ a$

$Y = b\ a\ b\ b\ a\ b$

X ↓ / Y →	^	b	a	b	b	a	b
^	0	0	0	0	0	0	0
a	0	← 0	↖ 1	← 1	← 1	↖ 1	← 1
b	0	↖ 1	← 1	↖ 2	↖ 2	← 2	↖ 2
a	0	↑ 1	↖ 2	← 2	← 2	↖ 3	← 3
a	0	↑ 1	↖ 2	← 2	← 2	↖ 3	← 3
b	0	↖ 1	↑ 2	↖ 3	↖ 3	← 3	↖ 4
a	0	↑ 1	↖ 2	↑ 3	↑ 3	↖ 4	← ↑ 4 → Answer

→ If both characters aren't matching, take max. of top & left box & draw arrow to indicate that top/left value is taken & put the value in box

→ If characters are matching, fill ~~out~~ the box as 1 + top left diagonal value & draw the ~~arrow~~ arrow in the direction

For finding out the characters back-trace from ~~last~~ answer using arrows

~~+~~ If
→ If the arrow is left/top just move
→ else keep noting the characters from right to left.

Solution: baba & baab

Matrix Chain Multiplication:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \qquad B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix}$$

2×3

$m \times n$

3×2

$n \times k$

$$C = A \times B = \begin{bmatrix} a_{11} \times b_{11} + a_{12} \times b_{21} + a_{13} \times b_{31} & a_{11} \times b_{12} + a_{12} \times b_{22} + a_{13} \times b_{32} \\ a_{21} \times b_{11} + a_{22} \times b_{21} + a_{23} \times b_{31} & a_{21} \times b_{12} + a_{22} \times b_{22} + a_{23} \times b_{32} \end{bmatrix}$$

2×2

§ Multiplications req^d to multiply these 2 matrices $= 2 \times 3 \times 2$
$= 12$

$A_1 \times A_2 \times A_3$

$2 \times 3 \quad 3 \times 4 \quad 4 \times 2$

$d_0 \; d_1 \quad d_1 \; d_2 \quad d_2 \; d_3$

$(A_1 \times A_2) \times A_3$ $\qquad$ $A_1 \times (A_2 \times A_3)$

~~2 3 3 4~~

2 3 3 4 4 2

Single matrix

$(A_1 \times A_2) \times A_3$

2 3 3 4 4 2

$2 \times 3 \times 4 = 24$ 0

2 4 4 2

$= 2 \times 4 \times 2 = 16$

$= \boxed{\text{Total} = 24 + 16 = 40}$

$A_1 \times (A_2 \times A_3)$

2 3 3 4 4 2

0 $3 \times 4 \times 2 = 24$

2 3 3 2

$= 2 \times 3 \times 2 = 12$

$\boxed{\text{Total} = 24 + 12 = 36}$

Now we need do chock out for all icombinations & pick up the one with min. ng. of multiplications

$(A_1 \times A_2) \times A_3$

2 3 3 4 4 2

$c[1,2] = 24$ $c[3,3] = 0$

2 4 4 2

d_0 d_2 d_2 d_3

~~d_0~~ $+ d_0 \times d_2 \times d_3$

$c[1,2] + c[3,3] + d_0 \times d_2 \times d_3 = 40$

i j d_{i-1} d_k d_j

$A_1 \times (A_2 \times A_3)$

$c[1,1]$ 3 4 4 2

0 $c[2,3] = 24$

2 3 3 2

d_0 d_1 d_1 d_3

$d_0 \times d_1 \times d_3$

i k k+1 j

$c[1,1] + c[2,3] + d_0 \times d_1 \times d_3 = 36$

d_{i-1} d_k d_j

$\in \boxed{c[i,j] = c[i,k] + c[k+1,j] + d_{i-1} \times d_k \times d_j}$

$i \leq k < j$

$\boxed{c[i,j] = \min_{i \leq k < j} \{ c[i,k] + c[k+1,j] + d_{i-1} \times d_k \times d_j \}}$

$$c[i,j] = \min_{i \le k < j} \{ c[i,k] + c[k+1,j] + d_{i-1} \times d_k \times d_j \}$$

$$\underset{d_0\ d_1}{A_1} \times \underset{d_1\ d_2}{A_2} \times \underset{d_2\ d_3}{A_3} \times \underset{d_3\ d_4}{A_4}$$

No. of combinations = Take $N = n-1$

Here $n = 4$

$\therefore N = 4-1 = 3$

~~Formula =~~ 2^{nd}

Formula $= \frac{^{2N}C_N}{N+1} = \frac{^{2\times3}C_3}{3+1} = \frac{^6C_3}{4}$

$= \frac{6 \times 5 \times 4}{4 \times 3 \times 2 \times 1}$

$= 5$

1) $A_1 (A_2 (A_3 A_4))$ 2) $A_1 ((A_2 A_3) A_4)$

3) $(A_1 A_2)(A_3 A_4)$ 4) $(A_1 (A_2 A_3)) A_4$

5) $((A_1 A_2) A_3) A_4$

$c[1,4] = \min$

$k=1$ $c[1,1] + c[2,4] + d_0 \times d_1 \times d_4$

$k=2$ $c[1,2] + c[3,4] + d_0 \times d_2 \times d_4$

$k=3$ $c[1,3] + c[4,4] + d_0 \times d_3 \times d_4$

$c[4,4] = 0$

$A_1 (A_2 A_3 A_4)$

$(A_1 A_2)(A_3 A_4)$

$(A_1 A_2 A_3) A_4$

$$c[2,4] = \min_{2 \le k < 4} \begin{cases} k=2 & c[2,2] + c[3,4] + d_1 \times d_2 \times d_4 \\ k=3 & c[2,3] + c[4,4] + d_1 \times d_3 \times d_4 \end{cases}$$

c	j → 1	2	3	4
i ↓ 1	0	24	28	(58) → final ans
2		0	16	36
3			0	40
4				0

k	1	2	3	4
1		1	(1)	(3)
2			2	3
3				3
4				

This table gives the order of parentheses

For e.g.

$A_1 \times A_2 \times A_3 \times A_4$

3 2 2 4 4 2 2 5

d_0 d_1 d_2 d_3 d_4

$$c[1,2] = \min_{1 \le k < 2} \{ c[1,1] + c[2,2] + d_0 \times d_1 \times d_2 \}$$
$$0 + 0 + 3 \times 2 \times 4$$

$$c[2,3] = \min_{2 \le k < 3} \{ c[2,2] + c[3,3] + d_1 \times d_2 \times d_3 \}$$
$$0 + 0 + 2 \times 4 \times 2 = 16$$

$$c[3,4] = \min_{3 \le k < 4} \{ c[3,3] + c[4,4] + d_2 \times d_3 \times d_4 \}$$
$$0 + 0 + 4 \times 2 \times 5 = 40$$

$$c[1,3] = \min_{1 \le k < 3} \begin{cases} k=1 & c[1,1] + c[2,3] + d_0 \times d_1 \times d_3 \\ & 0 + 16 + 3 \times 2 \times 2 = 28 \\ k=2 & c[1,2] + c[3,3] + d_0 \times d_2 \times d_3 \\ & 24 + 0 + 3 \times 4 \times 2 = 48 \end{cases}$$

$c[2,4] = \min_{2 \le k < 4} \begin{cases} k=2 & c[2,2] + c[3,4] + d_1 \times d_2 \times d_4 \\ & 0 + 40 + 2 \times 4 \times 5 = 80 \\ k=3 & c[2,3] + c[4,4] + d_1 \times d_3 \times d_4 \\ & 16 + 0 + 2 \times 2 \times 5 = 36 \end{cases}$

$c[1,4] = \min_{1 \le k < 4} \begin{cases} k=1 & c[1,1] + c[2,4] + d_0 \times d_1 \times d_4 \\ & 0 + 36 + 3 \times 2 \times 5 = 66 \\ k=2 & c[1,2] + c[3,4] + d_0 \times d_2 \times d_4 \\ & 24 + 40 + 3 \times 4 \times 5 = 124 \\ k=3 & c[1,3] + c[4,4] + d_0 \times d_3 \times d_4 \\ & 28 + 0 + 3 \times 2 \times 5 = 58 \end{cases}$

∴ Final ans. is 58

Check value in k table at c[1,4]

i.e. 3 ∴

Parenthesization:

~~A1 x~~

$(A_1 A_2 A_3) A_4$

Now see (1,3) in k table

∴ $((A_1)(A_2 A_3))(A_4)$

Time Complexity = $O(n^3)$

Space Complexity = $O(n^2)$

Recurrence relation:

MCM Solved Eg.

$M_1 \quad M_2 \quad M_3 \quad \ldots \quad M_4$

M_1	M_2	M_3	M_4
10×100	100×20	20×5	5×80
$P_0 \times P_1$	$P_1 \times P_2$	$P_2 \times P_3$	$P_3 \times P_4$

$$M[i,j] = \min \begin{cases} M[i,k] + M[k+1,j] + P_{i-1} \cdot P_k \cdot P_j \\ i \le k < j \end{cases}$$

$$M[1,2] = \overset{0}{M[1,1]} + \overset{0}{M[2,2]} + 10 \times 100 \times 20 = 20k$$

$$M[2,3] = M[2,2] + M[3,3] + 100 \times 20 \times 5 = 10k$$

$$M[3,4] = M[3,3] + M[4,4] + 20 \times 5 \times 80 = 8k$$

M	1	2	3	4
1	0	20k	15k	19k
2		0	10k	50k
3			0	8k
4				0

k	1	2	3	4
1		1	1	(3)
2			2	3
3				3
4				

Split at 3rd pos. first

$$M[1,3] = \min \begin{cases} \overset{0}{M[1,1]} + \overset{10k}{M[2,3]} + 10 \times 100 \times 5 = 15k \\ \overset{20k}{M[1,2]} + \overset{0}{M[3,3]} + \overset{1k}{10 \times 20 \times 5} = 21k \end{cases}$$

$$M[2,4] = \min \begin{cases} \underset{0}{M[2,2]} + \underset{8K}{M[3,4]} + 100 \times 20 \times 80 = \underset{160K}{168K} \\ \underset{10K}{M[2,3]} + \underset{0}{M[4,4]} + 100 \times 5 \times 80 = \underset{40K}{50K} \end{cases}$$

$$M[1,4] = \min \begin{cases} \underset{0}{M[1,1]} + \underset{50K}{M[2,4]} + 10 \times 100 \times 80 = \underset{80K}{130K} \\ \underset{20K}{M[1,2]} + \underset{8K}{M[3,4]} + 10 \times 20 \times 80 = \underset{16K}{44K} \\ \underset{15K}{M[1,3]} + \underset{0}{M[4,4]} + 10 \times 5 \times 80 = \underset{4K}{19K} \end{cases}$$

$(M_1\ M_2\ M_3)\ M_4$

↳ Now check in k table at (1,3)

∴ split at 1

$((M_1)\ (M_2\ M_3))\ M_4$