WHEN SHOULD YOU USE DYNAMIC PROGRAMMING?
LOOK FOR THE FOLLOWING:

- OPTIMAL SUBSTRUCTURE (YOU CAN USE DYNAMIC PROGRAMMING.)
- OVERLAPPING SUBPROBLEMS (YOU SHOULD USE DYNAMIC PROGRAMMING.)

MATRIX CHAIN MULTIPLICATION.

CONSIDER THE PROBLEM OF MULTIPLYING TWO MATRICES A $(p \times q)$ AND B $(q \times r)$. THE PRODUCT AB HAS DIMENSIONS $(p \times r)$. ITS ij ENTRY IS

$$\sum_{k=1}^{q} a_{ik} b_{kj} \qquad (1 \le i \le p,\ 1 \le j \le r)$$

WHICH INVOLVES q SCALAR MULTIPLICATIONS. THUS THE TOTAL NUMBER OF SCALAR MULTIPLICATIONS PERFORMED IN COMPUTING AB IS pqr.

NOW CONSIDER THE PRODUCT OF THREE MATRICES ABC OF DIMENSIONS $(p \times q)$, $(q \times r)$, AND $(r \times s)$ RESPECTIVELY.

THERE ARE TWO POSSIBLE PARENTHESIZATIONS OF THIS PRODUCT:

(1) $A(BC)$ # SCALAR MULTIPLICATIONS $= pqs + qrs$

(2) $(AB)C$ # SCALAR MULTIPLICATIONS $= pqr + prs$

e.g. $p = 10$, $q = 100$, $r = 10$, $s = 100$. THEN (1) YIELDS 200,000 MULTIPLICATIONS AND (2) YIELDS 20,000.

THUS ONE PARENTHESIZATION MAY BE MORE EFFICIENT THAN ANOTHER.

FOR FOUR MATRICES THERE ARE FIVE PARENTHESIZATIONS.

LET $P(n)$ DENOTE THE NUMBER OF DISTINCT PARENTHESIZATIONS OF A PRODUCT OF n MATRICES.

EXCERCISE: SHOW $P(n)$ SATISFIES THE RECURRENCE

$$P(n) = \sum_{k=1}^{n} P(k)P(n-k) \qquad (n \geq 1).$$

THE SEQUENCE $P(n)$ IS CALLED THE <u>CATALAN NUMBERS</u>. ONE CAN SHOW THAT

$$P(n) = \frac{1}{n}\binom{2n-2}{n-1}$$

<u>EXERCISE</u>

USE STIRLING'S FORMULA TO SHOW

$$P(n) = \Theta\left(\frac{4^n}{n^{3/2}}\right)$$

<u>PROBLEM</u>

GIVEN A PRODUCT OF n MATRICES $A_1 A_2 \cdots A_n$ WHERE A_i HAS DIMENSIONS $P_{i-1} \times P_i$, DETERMINE A PARENTHESIZATION WHICH MINIMIZES THE TOTAL NUMBER OF SCALAR MULTIPLICATIONS PERFORMED.

AS USUAL THERE ARE REALLY TWO PROBLEMS

- FIND THE MINIMUM NUMBER OF MULTIPLICATIONS
- FIND THE PARENTHESIZATION WHICH ACHIEVES IT.

WE CAN SEE THAT THE BRUTE FORCE APPROACH IS INEFFICIENT SINCE THE NUMBER OF PARENTHESIZATIONS IS EXPONENTIAL.

To develop a dynamic programming solution, we consider the general subproblem of finding an optimal parenthesization of $A_i \cdots A_j$ where $1 \le i \le j \le n$.

Observe that an optimal parenthesization splits $A_i \cdots A_j$ into

$$(A_i \cdots A_k) \cdot (A_{k+1} \cdots A_j)$$

for some k $(i \le k < j)$.

Note also that if $A_i \cdots A_j$ is optimally parenthesized, then so are both $(A_i \cdots A_k)$ and $(A_{k+1} \cdots A_j)$

Proof: If the parenthesization of $(A_i \cdots A_k)$ is not optimal, then we can replace it with an optimal one yielding a parenthesization of $A_i \cdots A_k$ with fewer scalar multiplications. $=1$ This contradicts that our original parenthesization of $A_i \cdots A_k$ was optimal. $< j$ ///

Thus the principle of optimality is satisfied in this problem.

$$\overset{\text{OPTIMAL}}{\underset{(p_{i-1} \times p_k)}{(A_i \cdots A_k)}} \cdot \overset{\text{OPTIMAL}}{\underset{(p_k \times p_j)}{(A_{k+1} \cdots A_j)}} \quad \text{OPTIMAL}$$

LET $m[i,j]$ DENOTE THE MINIMUM NUMBER OF SCALAR MULTIPLICATIONS NECESSARY TO COMPUTE $A_i \cdots A_j$. IF $i=j$ THEN $m[i,j]=0$ SINCE THE PRODUCT CONSISTS OF JUST ONE MATRIX.

IF $i \le k < j$, AND k IS THE SPLIT POSITION OF AN OPTIMAL PARENTHESIZATION THEN

$$m[i,j] = m[i,k] + m[k+1,j] + p_{i-1}\, p_k\, p_j$$

SINCE WE DON'T KNOW k, WE DEFINE IN GENERAL

$$m[i,j] = \begin{cases} 0 & i = j \\ \min\limits_{i \le k < j} \left(m[i,k] + m[k+1,j] + p_{i-1}\, p_k\, p_j \right) & i < j \end{cases}$$

Ex. $n = 5$; $P_0 = 10$, $P_1 = 20$, $P_2 = 30$, $P_3 = 10$, $P_4 = 40$, $P_5 = 10$

TABLE m

	1	2	3	4	5
1	0	6000	8000	12000	13000
2	x	0	6000	14000	12000
3	x	x	0	12000	7000
4	x	x	x	0	4000
5	x	x	x	x	0

FOR INSTANCE, TO COMPUTE $m[2,5]$:

$$
m[2,5] = \min \begin{cases} m[2,2] + m[3,5] + P_1 P_2 P_5 = 13000 & (k=2) \\ m[2,3] + m[4,5] + P_1 P_3 P_5 = \boxed{12000} & (k=3) \checkmark \\ m[2,4] + m[5,5] + P_1 P_4 P_5 = 22000 & (k=4) \end{cases}
$$

OBSERVE THAT TO COMPUTE $m[2,5]$ ONE NEEDS ENTRIES $m[2, 2..4]$ AND $m[3..5, 5]$. THUS TO FILL THE TABLE, FIRST INITIALIZE THE MAIN DIAGONAL TO 0, THEN SUCCESSIVELY FILL EACH OFF DIAGONAL ABOVE THE MAIN.

i.e. $\quad m[i,i] = 0 \quad (1 \le i \le n)$

THEN $\quad m[i,i+1] \quad (1 \le i \le n-1)$

$\quad m[i,i+2] \quad (1 \le i \le n-2)$

AND IN GENERAL

$\quad m[i,i+\ell] \quad (1 \le i \le n-\ell)$

FOR $\ell = 1, 2, \ldots, n-1$

SEE P. 336 FOR PSEUDO-CODE.

FROM THIS TABLE WE CAN RE-CONSTRUCT THE VALUES K, WHICH GIVES THE SPLIT POINTS FOR EACH SUBPROBLEM.

A MORE EFFICIENT APPROACH IS TO STORE K VALUES IN A PARALLEL TABLE S[i, j] AS WE CONSTRUCT m[i, j]. (P. 336).

Ex. SAME AS ABOVE, WE SEE S[2,5] = 3, AND

TABLE S

	1	2	3	4	5
1	x	1	1	3	3
2	x	x	2	3	3
3	x	x	x	3	3
4	x	x	x	x	4
5	x	x	x	x	x

FROM THIS TABLE WE CAN CONSTRUCT THE OPTIMAL PARENTHESIZATION:

$$(A_1(A_2A_3))(A_4A_5).$$

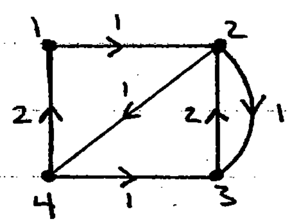
ALL PAIRS SHORTEST PATHS (APSP) (25.2)

CONSIDER A DIRECTED GRAPH IN WHICH A WEIGHT (COST) IS ASSIGNED TO EACH DIRECTED EDGE.

WE WRITE $G=(V,E)$ WHERE V IS THE VERTEX SET AND E IS THE SET OF DIRECTED EDGES. THE ADJACENCY MATRIX OF G IS DEFINED AS $W=(w_{ij})$ WHERE

$$w_{ij} = \begin{cases} 0 & \text{if } i=j \\ \text{WEIGHT OF DIR. EDGE } (i,j) & \text{if } i\neq j,\ (i,j)\in E \\ \infty & \text{if } i\neq j,\ (i,j)\notin E \end{cases}$$

EX.



$V=\{1,2,3,4\}$

$E=\{(1,2),(2,4),(4,1),(4,3),(3,2),(2,3)\}$

$$W = \begin{pmatrix} 0 & 1 & \infty & \infty \\ \infty & 0 & 1 & 1 \\ \infty & 2 & 0 & \infty \\ 2 & \infty & 1 & 0 \end{pmatrix}$$