Chain Matrix Multiply:

Example: for 4 matrices A, B, C, D, we want to compute \( A \times B \times C \times D \) most efficiently.

Say A is 50x20
B is 20x1
C is 1x10
D is 10x100.

Since matrix multiplication is associative, we can compute:

\[
((A \times B) \times C) \times D
\]

or

\[
(A \times (B \times C)) \times D
\]

or

\[
A \times (B \times (C \times D))
\]
What's the best way?

Take $X$ of size $a \times b$ & $Y$ of size $b \times c$.

$Z = XY$ is of size $a \times c$.

$$
\begin{bmatrix}
  b \\
  c \\
\end{bmatrix}
\begin{bmatrix}
  x & y & Z \end{bmatrix}

Z_{ij} = \sum_{k=1}^{b} X_{ik}Y_{kj}

To compute $Z_{ij}$ takes $b$ multiplications (& $b-1$ additions)

ac entries of $Z$,

So $abc$ total multiplications (& similar # of additions)

Say cost of multiplying $X \times Y$ is $abc$. 
Therefore, for the earlier example, the cost for 

\[(\mathbf{A} \times \mathbf{B}) \times \mathbf{C} \times \mathbf{D}\]

is 

\[
(50)(20)(1) + (50)(1)(10) + (50)(10)(100)
\]

\[
= 1000 + 500 + 50,000 = 51,500
\]

\[(\mathbf{A} \times \mathbf{B}) \times (\mathbf{C} \times \mathbf{D})\]

\[
(50)(20)(1) + (1)(10)(100) + (50)(1)(100)
\]

\[
= 7,000
\]

\[(\mathbf{A} \times (\mathbf{B} \times \mathbf{C})) \times \mathbf{D}\]

\[
(20)(1)(10) + 50(20)(10) + (50)(10)(100)
\]

\[
= 200 + 10,000 + 50,000 = 60,200
\]

**General Problem:**

For \(n\) matrices \(A_1, A_2, \ldots, A_n\) where \(A_i\) is of size \(m_{i-1} \times m_i\),

what's min cost for multiplying \(A_1 \times A_2 \times \cdots \times A_n\)?

**Input:** \(M_0, M_1, M_2, \ldots, M_n\)

**Goal:** Determine the Parenthesization of min cost for computing \(A_1 \times \cdots \times A_n\).
Graphical view:

\((A \times B) \times (C \times D)\) as a binary tree:

\[ \text{Root: } A \times B \times C \times D \]

\[ \text{Internal nodes: } A \times B, C, D \]

\[ \text{Leaves: } A, B, C, D \]

The leaves correspond to \(A_1, A_2, \ldots, A_n\).

Internal nodes represent intermediate computations.

Root represents \(A_1 \times A_2 \times \ldots \times A_n\).
Try prefixes for subproblems:

\[ c(i) = \min \text{ cost for computing } A_1 \times A_2 \times \ldots \times A_i \]

But note that for the root, the left child is a prefix but the right child is a suffix

for some \( 1 \leq i \leq n-1 \)

So the computation at the root will have cost \( m_0 \times m_i \) (since \( A_1 \times \ldots \times A_i \) has size \( m_0 \times m_i \)), \( A_{i+1} \times \ldots \times A_n \) has size \( m_i \times m_n \)

Cost for \( A_1 \times \ldots \times A_i \) is \( c(i) \)

Cost for \( A_{i+1} \times \ldots \times A_n \) is ??

\[ \Rightarrow \text{ We need suffixes too!} \]
Then for next level:

\[ A_1 \times \ldots \times A_i \]
\[ A_i \times \ldots \times A_j \leftarrow A_{j+1} \times \ldots \times A_i \]
\[ 1 \leq j \leq i-1 \]

So we need substrings

**Attempt 2:**

For \( 1 \leq i \leq j \leq n \),

let \( C(i,j) = \min \) cost of computing \( A_i \times \ldots \times A_j \)

Base case: \( C(i,i) = 0 \)

For \( i < j \) try all \( l \) for the split where \( i \leq l \leq j-1 \)

\[ A_i \times \ldots \times A_j \]
\[ \leftarrow A_l \times \ldots \times A_{l+k} \]
\[ \leftarrow A_{l+k} \times \ldots \times A_j \]

Cost is \( m_{i-1} m_l m_j + C(i,l) + C(l+1,j) \)
Therefore,
\[ c(i,j) = \min_l \{ c(i,l) + c(l+1,j) + m_{i-l,m_{j-l}} \} \]
over \( l \) s.t. \( i \leq l \leq j-1 \)

How to fill the table?

\[ C = \begin{array}{c}
\text{fill in} \\
\end{array} \]

- only filling the upper-right portion

Since \( C(i,j) \) with \( i \leq j \)

Base case: \( C(i,i) \) \( i=1 \rightarrow n \) is the diagonal.

Then do \( C(i,i+1) \) since that uses \( C(i,i) \) \& \( C(i+1,i+1) \)

Then do \( C(i,i+2) \) which uses \( C(i,i), C(i,i+1), C(i+1,i+2), C(i+2,i+2) \)
Let $s = j - i$ be the "width" of the subproblem.

Base case $s = 0$: $C(i, i)$

Fill by $s = 0 \rightarrow n - 1$

Goal: compute $C(1, n)$ of width $s = n - 1$
Chan Matrix Multiply \( (m_0, m_1, \ldots, m_n) \)

For \( i = 1 \rightarrow n \), \( C(i, i) = 0 \)

For \( s = 1 \rightarrow n-1 \),

For \( i = 1 \rightarrow n-s \),

Let \( j = i+s \)

\( C(i, j) = \infty \)

For \( l = i \rightarrow j-1 \)

if \( C(i, j) > m_i \cdot m_k \cdot m_j + C(i, l) + C(l+1, j) \)

then \( C(i, j) = \)

Return \( C(1, n) \)

Running time: \( O(n^3) \)