Chain Matrix Multiply

Example for 4 matrices A, B, C, D

Want to compute $A \times B \times C \times D$ most efficiently

Say $A$ is $50 \times 20$
$B$ is $20 \times 1$
$C$ is $1 \times 10$
$D$ is $10 \times 100$

Matrix multiplication is associative so we can compute:

$((A \times B) \times C) \times D$ or $(A \times B) \times (C \times D)$ or $(A \times (B \times C)) \times D$

etc....

What's the best way?

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

Let $Z = XY$ of size $a \times c$

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

$b$ multiplications & $b-1$ additions.
Z has ac entries & thus computing Z takes a total of abc multiplications & a(b-1)c additions. Since multiplies are more expensive say:

\[
\text{cost of multiplying } XY \text{ is } abc
\]

\[\begin{align*}
\text{Earlier example:} & \\
((A\times B) \times C) \times D & \quad \text{cost} \\
& 50 \times 20 \times 1 + 50 \times 1 \times 10 + 50 \times 10 \times 100 = 51,500 \\
(A\times B) \times (C \times D) & \quad 50 \times 20 \times 1 + 50 \times 10 \times 100 + 50 \times 1 \times 100 = 7,800 \\
(A \times (B \times C)) \times D & \quad 20 \times 1 \times 10 + 50 \times 20 \times 10 + 50 \times 10 \times 100 = 69,200 \\
\end{align*}\]

Which ordering has min cost?

\[\begin{align*}
\text{General problem:} & \\
\text{For } n \text{ matrices } A_1, A_2, \ldots, A_n & \\
\text{where } A_i \text{ is of size } m_{i-1} \times m_i & \\
\text{What's min cost for multiplying } A_1 \times A_2 \times \ldots \times A_n? & \\
\text{Input: } m_0, m_1, \ldots, m_n & \\
\text{Goal: min cost for computing } A_1 \times \ldots \times A_n &
\end{align*}\]
Graphical view: View as a binary tree

$$((A \times B) \times C) \times D$$

$$A \times B$$

$$A \times B \times C \times D$$

Leaves are $$A_1, \ldots, A_n$$. Root is $$A_1 \times \ldots \times A_n$$

& internal nodes are intermediate computations

DP algorithm:

Try prefixes for subproblems.

Let $$C(i) = \min \text{ cost for computing } A_1 \times A_2 \times \ldots \times A_i$$

Look at the tree: root is $$A_1 \times \ldots \times A_i$$

$$A_1 \times \ldots \times A_j$$ for some $$j < i$$

$$A_1 \times \ldots \times A_l$$ for some $$l < j$$

$$A_r \times \ldots \times A_j$$ not a prefix, it's a suffix

$$A_r \times \ldots \times A_j$$ not prefix or suffix, it's a substring.
Try substrings (instead of prefixes) for subproblems

For \(1 \leq i \leq j \leq n\),
let \(C(i,j) = \min \) cost for computing \(A_i \times \ldots \times A_j\)

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

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

Therefore,
\[
C(i,j) = \min_{l} \left\{ C(i,l) + C(l+1,j) + m_{i-1} m_l m_j \right\}, \quad i \leq l \leq j-1
\]
How to fill the table?

Base case is $C(i,i)$ for $i=1 \rightarrow N$ which is the diagonal.

$C = \begin{bmatrix}
C(i,i) & C(i,i+1) & \cdots & C(i,N) \\
C(i+1,i) & C(i+1,i+1) & \cdots & C(i+1,N) \\
\vdots & \vdots & \ddots & \vdots \\
C(N,i) & C(N,i+1) & \cdots & C(N,N)
\end{bmatrix}$

Next we do $C(i,i+1)$ which uses $C(i,i)$ & $C(i+1,i+1)$.

Then $C(i,i+2)$, etc.

Let $s = j-i$ = "width" of subproblem.

Base case: $s=0 \Rightarrow j=i$ so $C(i,j) = C(i,i)$.

Fill by $s=0 \rightarrow N-1$.

Goal: $C(l,n)$.
**Chained Multiply \((m_0, m_1, ..., m_n)\):**

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 \(C(i,j) > m_{i-1}m_l m_j + C(i,l) + C(l+j)\)

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

Return \((C(i,n))\)

**Running time:** 3 nested for loops of size \(O(n)\) so \(O(n^3)\) total time.
Independent sets:

For a graph $G=(V,E)$, a subset $S \subseteq V$ is an independent set if it does not cover any edge, i.e.,

for all $(x,y) \in E$, either $x \notin S$ and/or $y \notin S$. (so both endpoints not in $S$)

Max independent set problem:

Given a graph $G=(V,E)$, find the size $|S|$ of the an independent set.

Hard problem on general graphs:

NP-hard (even to approximate within $n^{0.99}$ factor)

Suppose $G$ is a tree $T$ (so no cycles)

Root $T$ at some vertex $r$ & think of $T$ as hanging from $r$.

So for a vertex $v$, let the children of $v$ be with respect to $T$ rooted at $r$. 
DP alg. for max-IS on trees:

For vertex \( v \),

let \( I(v) = \) size of largest independent set in subtree rooted at \( v \). (this is the tree hanging from \( v \) & below).

For \( I(v) \), either include \( v \) or not:

- if include \( v \), then no neighbor of \( v \) is in the IS. So we can remove \( v \) & its children. Those subtrees at grandchildren are disconnected so independent of each other. Hence in this case,

\[
I(v) = 1 + \sum_{\text{grandchildren } w \text{ of } v} I(w)
\]

- if we don't include \( v \), then we can remove \( v \) & then the subtrees at children of \( v \) are disconnected & thus independent of each other. Hence,

\[
I(v) = \sum_{\text{children } z \text{ of } v} I(z).
\]

Therefore,

\[
I(v) = \max \left\{ 1 + \sum_{\text{grandchildren } w \text{ of } v} I(w), \sum_{\text{children } z \text{ of } v} I(z) \right\}
\]
Run DFS starting from r & fill in the table as finish processing v.  
Postorder numbering takes $O(|V|+|E|)$ = $O(|V|)$ time.