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 50×20

B is 20×1

C is 1×10

D is 10×100.

Matrix multiplication is associative so we can compute:

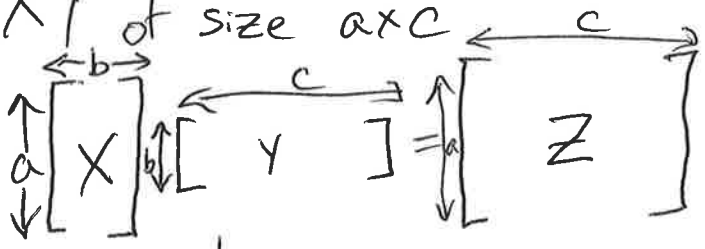
$((A \times B) \times C) \times D$ or $(A \times B) \times (C \times D)$ or $(A \times (B \times C)) \times D$ etc....

What's the best way?

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

(# of columns of X = # of rows of Y)

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:

Cost of multiplying XY is abc

Earlier example:

	Cost
$((A\times B)\times C)\times D$	$50\times 20\times 1 + 50\times 1\times 10 + 50\times 10\times 100 = 51{,}500$
$(A\times B)\times(C\times D)$	$50\times 20\times 1 + 1\times 10\times 100 + 50\times 1\times 100 = 7{,}000$
$(A\times(B\times C))\times D$	$20\times 1\times 10 + 50\times 20\times 10 + 50\times 10\times 100 = 60{,}200$

Which ordering has min cost?

General problem:

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

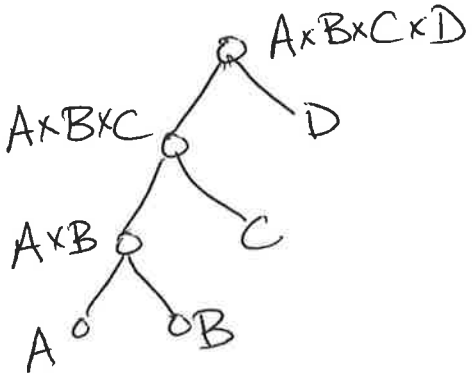
What's min cost for multiplying $A_1\times A_2\times\cdots\times A_n$?

Input: $m_0, m_1, \ldots, m_n$

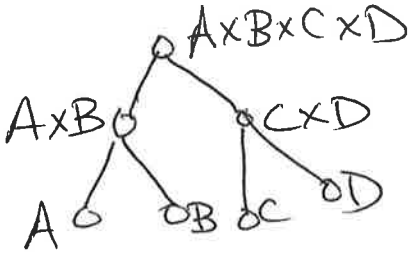
Goal: min cost for computing $A_1\times\cdots\times A_n$

Graphical view: View as a binary tree

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



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



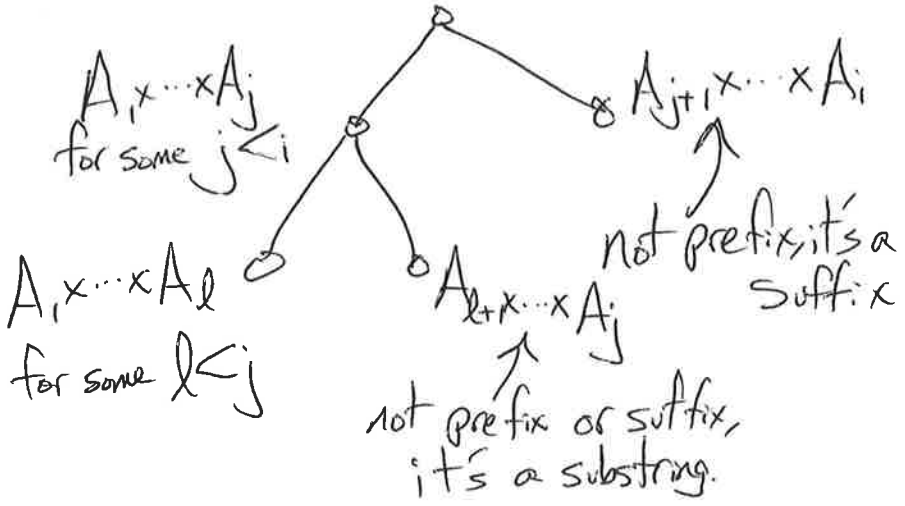
Leaves are $A_1, \ldots, A_n$.
Root is $A_1 \times \cdots \times A_n$ & internal nodes are intermediate computations

DP algorithm:

Try prefixes for subproblems.

Let $C(i)$ = min cost for computing $A_1 \times A_2 \times \cdots \times A_i$

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



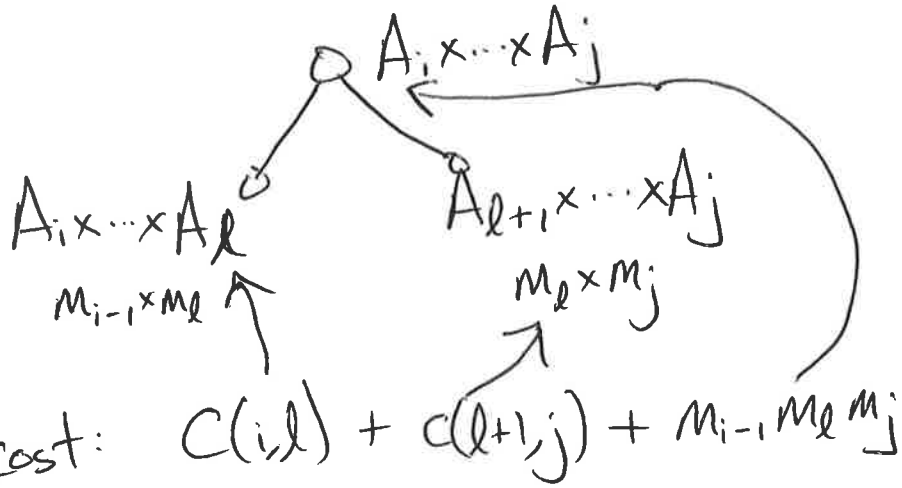
Try substrings (instead of prefixes) for subproblems

For $1 \le i \le j \le n$,

let $C(i,j)$ = min cost for computing $A_i \times \cdots \times A_j$

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

For $i < j$, try all ℓ for the split where $i \le \ell \le j-1$



Therefore,

$$C(i,j) = \min_{\ell} \{ C(i,\ell) + C(\ell+1,j) + m_{i-1} m_\ell m_j : i \le \ell \le j-1 \}$$

How to fill the table?

Base case is $C(i,i)$ for $i=1 \to n$ which is the diagonal

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 $s=0 \to n-1$.

Goal: $C(1,n)$.

ChainMultiply($m_0, m_1, \ldots, m_n$):

For $i=1 \to n$, $C(i,i)=0$

For $s=1 \to n-1$,

For $i=1 \to n-s$,

let $j=i+s$

$C(i,j)=\infty$

For $k=i \to j-1$

if $C(i,j) > m_{i-1} m_k m_j + C(i,k) + C(k+1,j)$

then $C(i,j)=$

Return $(C(1,n))$

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 $x \notin S$ &/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 a maximum an independent set.

Hard problem on general graphs:

NP-hard (even to approximate within $n^{.99}$ 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,

$$I(v) = \max\left\{ 1 + \sum_{\substack{\text{grandchildren } w \\ \text{of } v}} I(w) \;,\; \sum_{\substack{\text{children } z \\ \text{of } v}} I(z) \right\}.$$

(c)

Run DFS starting from r &
fill in the table as finish processing v.
$\Rightarrow$ Postorder numbering

takes $O(|V|+|E|) = O(|V|)$ time.