Matrix Chain Multiplication

we saw better, when $n=m=k$, but let us use naive one. doesn't matter this lecture

$A \in \mathbb{R}^{n \times m}$, $B \in \mathbb{R}^{m \times k}$.

Multiplying A and B: naively nmk time.

Then, given $A_1, \dots, A_k$ s.t. $A_i \in \mathbb{R}^{n_i \times m_i}$ with $m_i = n_{i+1}$ $\forall i \in [n-1]$,
can we compute $\underline{A_1 \cdot A_2 \cdot \cdots \cdot A_k}$? (let $n_{k+1} := m_k$)

First Q. Is it even "well-defined"?

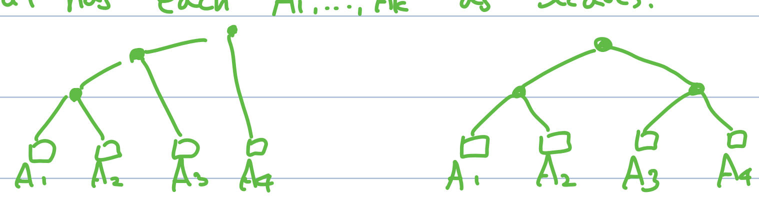
We only defined multiplication of two matrices $A \cdot B$, so when we write $A_1 \cdots A_k$, we need to specify <u>the order in which we multiply</u>

two interpretations (same)

① put parantheses s.t. if we multiply from inner ones, each parantheses contains always two matrices.

(e.g., $(((A_1 A_2) A_3) A_4)$ vs $((A_1 A_2)(A_3 A_4))$ vs ...)

② Construct a full binary tree (i.e., each non-leaf node has exactly two children) that has each $A_1, \dots, A_k$ as leaves.



Second Q. Okay, then is it okay to write $A_1 \cdots A_k$?

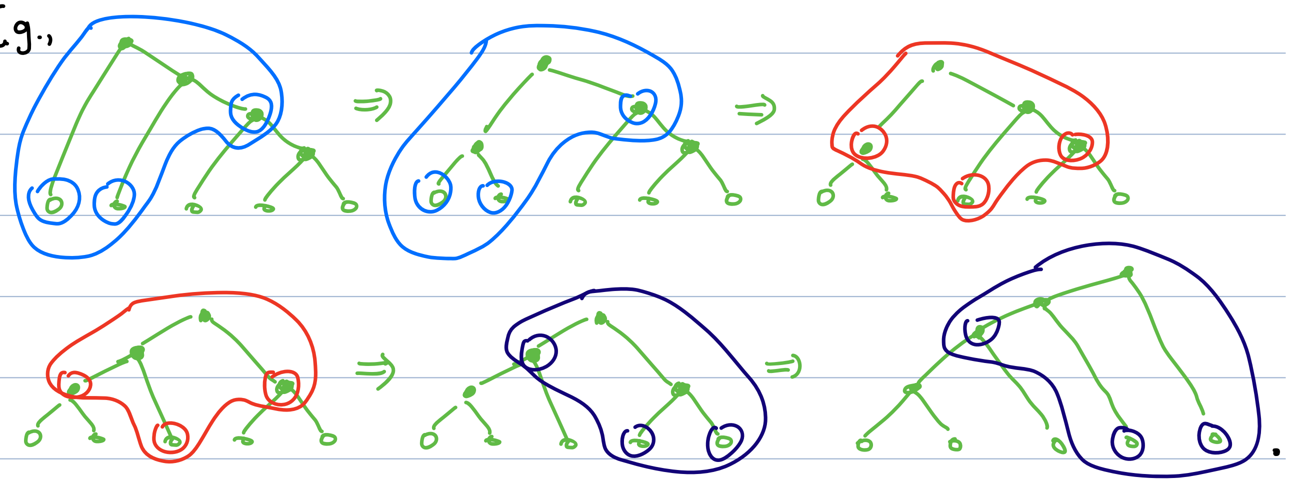
A: Yes, if you only care about the answer $B = A_1 \cdots A_k \in \mathbb{R}^{n_1 \times m_k}$.

Lemma. $\forall A_1, A_2, A_3$, $(A_1 A_2) A_3 = A_1 (A_2 A_3)$ (associativity)

(Note: $A_1 A_2 \neq A_2 A_1$ (NON-commutativity)).

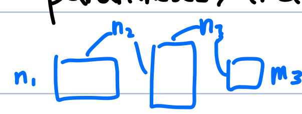
Once you have this, then any parantheses/tree give same answer

E.g.,



Okay, the answer doesn't depend on parantheses/tree,

but "computation time" does!!



$((A_1 A_2) A_3)$: 2 multiplications. Time : $n_1 n_2 n_3 + n_1 n_3 m_3$

$(A_1 (A_2 A_3))$: " Time : $n_2 n_3 m_3 + n_1 n_2 m_3$

if $n_1 = 1$, $n_2 = n_3 = m_3 = 10$, $((A_1 A_2) A_3)$ wins.

if $n_1 = n_2 = n_3 = 10$, $m_3 = 1$ $(A_1 (A_2 A_3))$ wins.

How to compute the optimal tree?

Dynamic Programming for Matrix Mult.

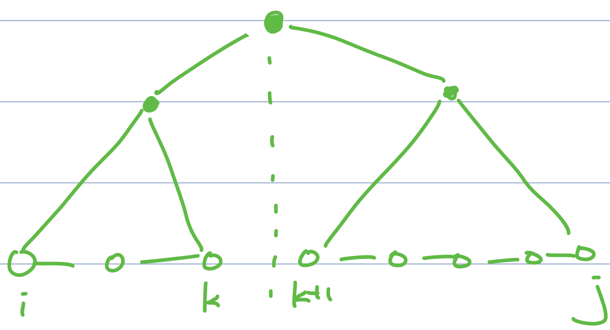
$\forall\, 1 \le i \le j \le k$, let

$T[i,j]$: Optimal running time to multiply $A_i \cdots A_j$.

Base case: $T[i,i] = 0 \quad \forall i \in [k]$.

Recurrence Relation.

$$T[i,j] = \min_{i \le k < j} \left(T[i,k] + T[k+1,j] + n_i n_{k+1} n_{j+1} \right)$$



$((A_1 \cdots A_k)(A_{k+1} \cdots A_j))$

outer paranthesis divides $A_1 \cdots A_k$ and $A_{k+1} \cdots A_j$

root divides $A_1 \cdots A_k$ and $A_{k+1} \cdots A_j$

Lemma. $T[i,j]$ is indeed optimal running time to multiply $A_i \cdots A_j$.

Pf. Induction on $j-i$. If $j=i$, easy.

Claim. $\forall\, i \le k < j$, $\left(T[i,k] + T[k+1,j] + n_i n_{k+1} n_{j+1} \right)$ is the optimal running time to multiply $A_i \cdots A_j$ given that the last multiplication is $(A_1 \cdots A_k)(A_{k+1} \cdots A_j)$!

Pf.

($n_i n_{k+1} n_{j+1}$ is the cost of last mult, and by induction hypothesis,

$T[i,k]$ = optimal time to multiply $A_1 \cdots A_k$. $(k-i < j-i)$

$T[k+1,j]$ = " " " " $A_{k+1} \cdots A_j$ $(j-k-1 < j-i)$). $\square$

So, (opt. time for $A_i \cdots A_j$) $= \min_{i \le k < j}$ (opt. time for $A_i \cdots A_j$ given last mult. is $(A_i \cdots A_k)(A_{k+1} \cdots A_j)$)

$$= \min_{i \le k < j} \left(T[i,k] + T[k+1,j] + n_i n_{k+1} n_{j+1} \right)$$

$$= T[i,j]$$

□.

Running time: $O(k^3)$

Optimal Binary Search Tree

Input: n keys $k_1 < \cdots < k_n$ and $(2n+1)$ probabilities describing a random search x.

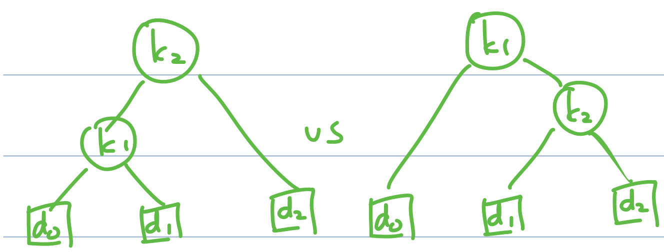
$p_i = \Pr[x = k_i]$. $\forall i \in [n]$ (let $d_i = (k_i, k_{i+1})$)

$q_i = \Pr[k_i < x < k_{i+1}]$ $\forall i \in [n+1]$ (assume $k_0 = -\infty$, $k_n = +\infty$)

Output: "Binary search tree": full binary tree whose

- non leaves correspond to $k_1 \dots k_n$. (left to right)
- leaves " to $d_0, \dots, d_n$

that minimizes "expected search time" = $\mathbb{E}_x$[depth of node containing x] +1



(left) (right)

If $x = d_2$, search time is 2 vs 3.

If $(p_1, p_2, q_0, q_1, q_2) = (0.1, 0.1, 0.2, 0.3, 0.3)$.

Expected search time = 2.4 vs 2.5

Same DP: Let $T[i,j]$ = optimal search time for $d_{i-1}, k_i, d_i, k_{i+1}, \dots, k_j, d_j$.

Base case: $T[i, i-1] = q_{i-1}$.

Recurrence Relation: $\forall i \le j$, $T[i,j] = \min_{i \le l \le j} \left(T[i, l-1] + T[l+1, j] + (q_{i-1} + p_i + q_i + \cdots + p_j + q_j)\right)$

Running time: $O(n^3)$ naively, but unlike matrix chain multiplication $O(n^2)$ is known!