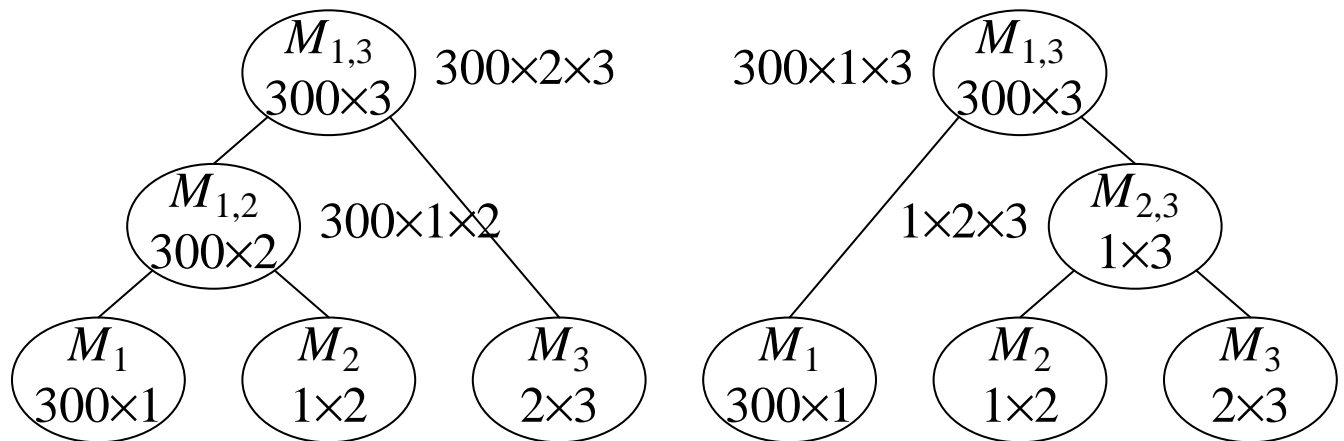


AN OPTIMAL SCHEME FOR EVALUATING A MATRIX PRODUCT $M_1 \times M_2 \times \dots \times M_N$

Problem: Given the matrices M_1, M_2, \dots, M_N , where M_i is of the order $n_i \times n_{i+1}$, find an optimal scheme for evaluating the matrix product $M_1 \times M_2 \times \dots \times M_N$.

Example. If A is an $m \times n$ matrix and B an $n \times p$ matrix, then we need to compute $m \times n \times p$ multiplications of the form $a_{ij} b_{jk}$ for the product $A \times B$. The first scheme below for $M_1 \times M_2 \times M_3$ needs $600 + 1800 = 2400$ multiplications compared to only $6 + 900 = 906$ in the second.



Shown next to each intermediate node is the number of multiplications of the form $a_{ij} b_{jk}$ used for that node.

- #(schemes for multiplying N matrices)
 = #(full binary trees with N terminal nodes)
 = #(balanced be -strings of length $2(N - 1)$),
 which is exponential in N .

Question: Show all schemes for multiplying 4 matrices.

ALGORITHM FOR DETERMINING AN OPTIMAL SCHEME FOR $M_1 \times M_2 \times \dots \times M_N$

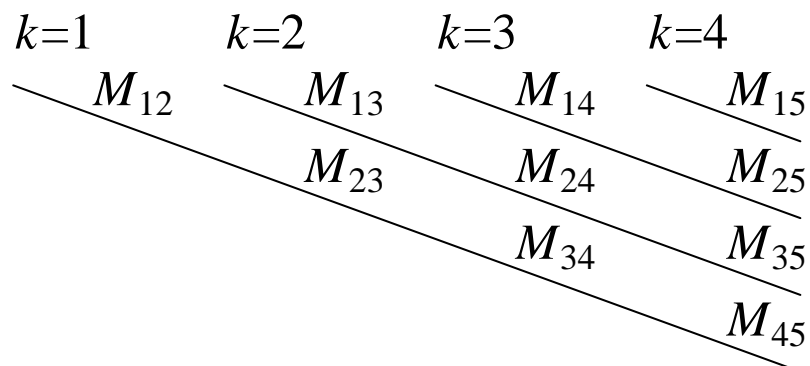
Notation:

- $M_{i,j} = M_i \times M_{i+1} \times \dots \times M_j$ for $1 \leq i \leq j \leq N$; $M_{i,i} = M_i$.
- $M_{1,N} = M_{1,j} \times M_{j+1,N}$ for each $1 \leq j < N$.
- $\text{optCost}(i, j) = \text{Min. \#(multiplications to compute } M_{i,j}\text{)}$.
 $\text{optCost}(i, i) = 0$ for each i .
- $\text{optTree}(i, j) = \text{A scheme to compute } M_{i,j} \text{ using only } \text{optCost}(i, j) \text{ many multiplications.}$

Example: $\text{OptCost}(1, 5)$ is the minimum of the following costs corresponding to the last matrix-product

- (1) $M_{11} \times M_{25}$: $\text{optCost}(M_{11}) + \text{optCost}(M_{25}) + n_1 n_2 n_6$
- (2) $M_{12} \times M_{35}$: $\text{optCost}(M_{12}) + \text{optCost}(M_{35}) + n_1 n_3 n_6$
- (3) $M_{13} \times M_{45}$: $\text{optCost}(M_{13}) + \text{optCost}(M_{45}) + n_1 n_4 n_6$
- (4) $M_{14} \times M_{55}$: $\text{optCost}(M_{14}) + \text{optCost}(M_{55}) + n_1 n_5 n_6$

Idea: Find an optimal scheme for each $M_{i,i+k}$ along the k th diagonal line for $k = 1, 2, 3,$ and 4 :



CONTD.

Algorithm OPT-SCHEME:

1. For ($i = 1$ to N) initialize $\text{optCost}(i, i) = 0$ and let $\text{optTree}(i, i)$ consists of a single node M_i .
2. For ($k=1$ to $N - 1$ and $i=1$ to $N - k$) do the following:
 - (a) Let $\text{optCost}(i, i + k) = \min_{0 \leq j \leq k-1} \{ \text{optCost}(i, i + j) + \text{optCost}(i + j + 1, i + k) + n_i n_{i+j+1} n_{i+k+1} \}$
 - (b) Let $\text{optTree}(i, i + k)$ consists of a root node, the left subtree $\text{optTree}(i, i + j)$, and the right subtree $\text{optTree}(i + j + 1, i + k)$ corresponding to the first j that gives the minimum in (a).

Terms $n_i n_{i+j+1} n_{i+k+1}$ Computed for $N = 4$:

$\text{optCost}(1, 2)$ $M_{12}: n_1 n_2 n_3$	$\text{optCost}(1, 3)$ $M_{11} \times M_{23}: n_1 n_2 n_4$ $M_{12} \times M_{33}: n_1 n_3 n_4$	$\text{optCost}(1, 4)$ $M_{11} \times M_{24}: n_1 n_2 n_5$ $M_{12} \times M_{34}: n_1 n_3 n_5$ $M_{13} \times M_{44}: n_1 n_4 n_5$
	$\text{optCost}(2, 3)$ $M_{23}: n_2 n_3 n_4$	$\text{optCost}(2, 4)$ $M_{22} \times M_{34}: n_2 n_3 n_5$ $M_{23} \times M_{44}: n_2 n_4 n_5$
		$\text{optCost}(3, 4)$ $M_{34}: n_3 n_4 n_5$

COMPLEXITY ANALYSIS FOR Step(2)

$k = 1$: Find an optimal scheme for each $M_{i,i+1}$, $1 \leq i \leq N - 1$.

- Each case considers 1 new scheme corresponding to $M_i \times M_{i+1}$.

$k = 2$: Find an optimal scheme for each $M_{i,i+2}$, $1 \leq i \leq N - 2$.

- Each case considers 2 new schemes corresponding to $M_{i,i} \times M_{i+1,i+2}$ and $M_{i,i+1} \times M_{i+2,i+2}$, using an optimal scheme for each $M_{i,j}$ involved.

$k = N - 1$: Find an optimal scheme for $M_{1,N}$.

- Consider $N - 1$ new schemes corresponding to $M_{1,1} \times M_{2,N}$, $M_{1,2} \times M_{3,N}$, \dots , $M_{1,N-1} \times M_{N,N}$, using an optimal scheme for each $M_{i,j}$ involved.

Total number of schemes considered:

$$\begin{aligned} & (N - 1). 1 + (N - 2). 2 + \dots + (N - (N - 1)). (N - 1) \\ & = N[1 + 2 + \dots + (N - 1)] - [1^2 + 2^2 + \dots + (N - 1)^2] \\ & = NN(N - 1)/2 - N(N - 1)(2N - 1)/6 = (N^2 - 1)N/6 = O(N^3) \end{aligned}$$

Evaluating a new scheme:

Two multiplications (for $n_i n_{i+j+1} n_{i+k+1}$) and 2 additions.

Complexity:

- Each new scheme is involved in one minimum computation.
- Total = $O(N^3)$.

Question: Give the number of terms $n_i n_j n_k$ evaluated in the brute-force method.

OPT-SCHEME: A D.P. ALGORITHM

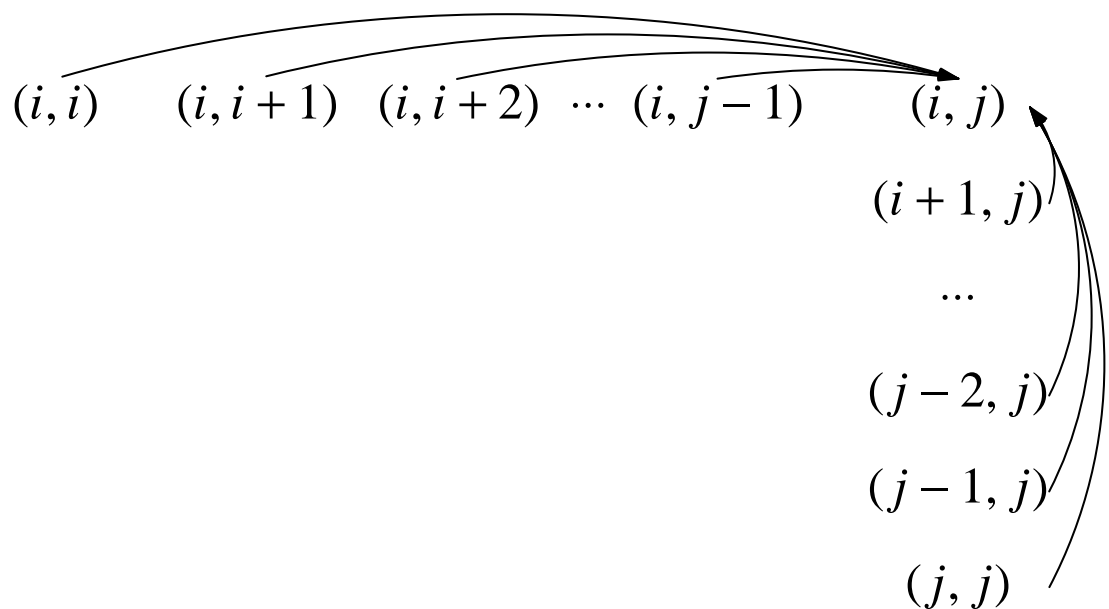
States: (i, j) , $1 \leq i \leq j \leq N$, corresponding to $M_{i,j}$.

Start-states: (i, i) , $1 \leq i \leq N$.

Values $V_{cost}(i, j)$ and $V_{scheme}(i, j)$:

- $V_{cost}(i, i) = 0$ and $V_{scheme}(i, i) = i$ for each $1 \leq i \leq N$.
- For $i < j$, $V_{cost}(i, j) = \text{optCost}(i, j)$ and $V_{scheme}(i, j) = p$, where $M_{i,p} \times M_{p+1,j} = M_{i,j}$ gives $\text{optCost}(i, j)$.

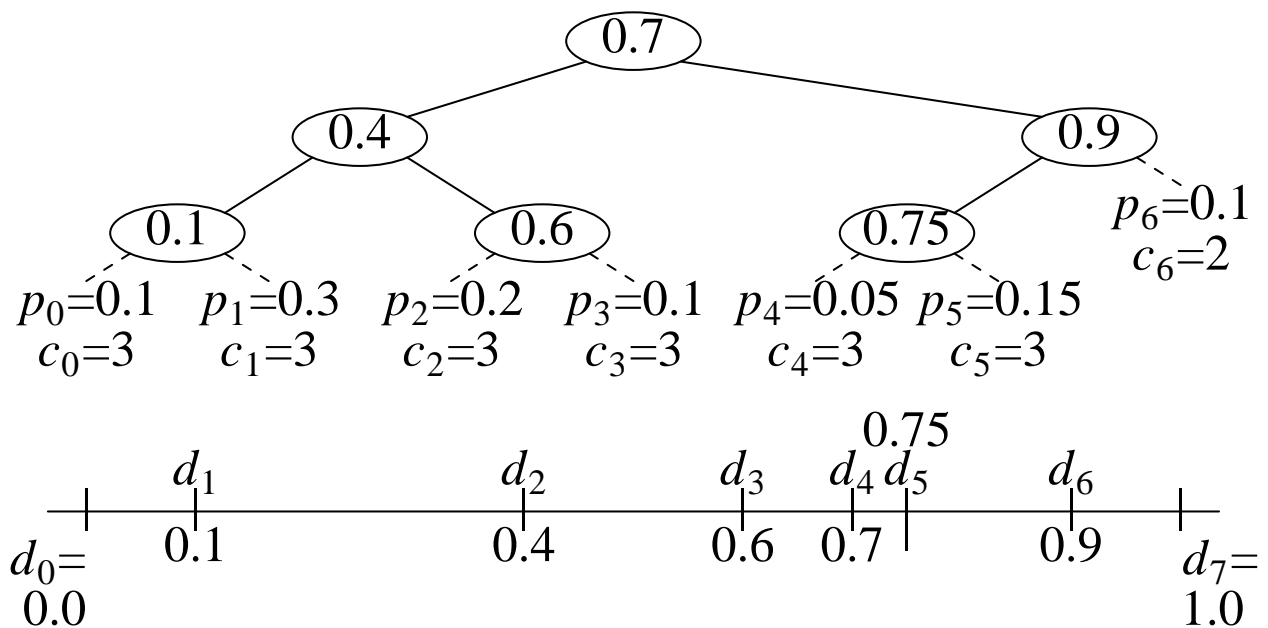
Relationship of (i, j) to Other States for $i < j$:



OPTIMAL BINARY SEARCH-TREE TO MINIMIZE NUMBER OF COMPARISONS

Problem: Given N data points $0 < d_1 < d_2 < \dots < d_N < 1$, find a binary search-tree which minimizes the average number of comparisons to test a random x in the interval $(0, 1)$ being one of the d_i 's. Note: $\text{prob}(x = d_i \text{ for some } i) = 0$.

Example: Shown below is a possible search-tree for the input $\langle 0.1, 0.4, 0.6, 0.7, 0.75, 0.9 \rangle$ for $N = 6$ and its cost.



- Each input x in the range $d_i < x < d_{i+1}$ follows the same search-path, and $\text{prob}(d_i < x < d_{i+1}) = p_i = d_{i+1} - d_i$, where $d_0 = 0$ and $d_{N+1} = 1$.
- The associated cost is $c_i = \#(\text{comparisons along the path to the corresponding missing child})$ as shown above.
- Total cost = $p_0c_0 + p_1c_1 + p_2c_2 + p_3c_3 + p_4c_4 + p_5c_5 + p_6c_6 = 2.9$.

ANALYSIS OF TOTAL COST

Contribution from Right Subtree at Root:

$$\begin{aligned}
 & 0.05 \times 3 + 0.15 \times 3 + 0.1 \times 2 \\
 &= 0.3 \times \left[\frac{0.05}{0.3} \times (2 + 1) + \frac{0.15}{0.3} \times (2 + 1) + \frac{0.1}{0.3} \times (1 + 1) \right] \\
 &= 0.3 \times \left[1 + \left(\frac{0.05}{0.3} \times 2 + \frac{0.15}{0.3} \times 2 + \frac{0.1}{0.3} \times 1 \right) \right] \\
 &= 0.3 \times \left[1 + \left(\text{cost of the subtree for data } \langle 0.75, 0.9 \rangle \right) \right. \\
 &\quad \left. \text{as a subset of } (0.7, 1.0) \right].
 \end{aligned}$$

Contribution from Left Subtree at Root:

$$0.7 \times \left[1 + \left(\text{cost of the subtree for data } \langle 0.1, 0.4, 0.6 \rangle \right) \right. \\
 \left. \text{as a subset of } (0.0, 0.7) \right].$$

Cost of a Search-tree with d_i as Root:

$$\begin{aligned}
 & d_i [1 + (\text{cost of left subtree w.r.t the interval } (0, d_i))] \\
 & + (1 - d_i) [1 + (\text{cost of right subtree w.r.t the interval } (d_i, 1))] \\
 & = d_i \times \text{cost}(\text{left subtree}) + (1 - d_i) \times \text{cost}(\text{right subtree}) + 1
 \end{aligned}$$

Observation:

- Both the left-subtree and the right-subtree of an optimal search-tree are optimal for their respective data sets, when considered as a subset of the appropriate intervals.

A D.P. ALGORITHM FOR AN OPTIMAL SEARCH-TREE

Method:

- For each ($k = 1, 2, 3, \dots, N - 1$), compute successively the cost $c_{i,i+k}$ of an optimal $(k + 1)$ -node search-tree $T_{i,i+k}$ for the data $d_{i,i+k} = \langle d_i, d_{i+1}, \dots, d_{i+k} \rangle$, considered as a subset of the interval (d_{i-1}, d_{i+k+1}) , for $i = 1, 2, \dots, N - k$.
- Let $r_{i,i+k}$ be the root of such a $T_{i,i+k}$.

Algorithm:

Input: The data points ($0 <$) $d_1 < d_2 < \dots < d_N (< 1)$.

Output: An optimal binary search-tree.

1. [Initialize.] For each $1 \leq i \leq N$, let $c_{i,i} = 1$ and $r_{i,i} = d_i$ (one-node tree corresponding to $\langle d_i \rangle$).
2. For ($k = 1, 2, 3, \dots, N - 1$), compute an $T_{i,i+k}$:
 - (a) For (each $i = 1, 2, \dots, N - k$), let $c_{i,i+k} =$

$$1 + \min_{i \leq j \leq i+k} \left[\frac{d_j - d_{i-1}}{d_{i+k+1} - d_{i-1}} c_{i,j-1} + \frac{d_{i+k+1} - d_j}{d_{i+k+1} - d_{i-1}} c_{j+1,i+k} \right],$$
 where $c_{i,i-1} = c_{i+k+1,i+k} = 0$.
 - (b) Let $r_{i,i+k} = d_j$ corresponding to the first j which gives minimum in (a).
 - (c) If $j > i$, then the left subtree of the root of $T_{i,i+k}$ is $T_{i,j-1}$ and is empty if $j = i$.

Likewise, if $j < i + k$, then the right subtree of the root is $T_{j+1,i+k}$ and is empty if $j = i + k$.

CONTD.

Complexity:

- In step (2), the computation of $c_{i,i+k}$ for $T_{i,i+k}$ with $(k + 1)$ nodes requires $O(k)$ time, using the previously computed optimal tree costs with k or fewer nodes.
- Number of trees $T_{i,i+k}$ is $N - k$.
- Total complexity = $O((N - 1).1 + (N - 2).2 + \dots + 1.(N - 1)) = O(N^3)$.

Questions:

- ? Why do we consider the above algorithm efficient?
- ? Give the exact number of search-trees that are looked at (i.e., their costs are computed) in the above algorithm. Give the smallest value N_0 of N for which not all search-trees with N_0 nodes are looked at; show all search-trees with N_0 nodes that are not looked at. (Since these trees may depend on the input data, show a relevant input data for your answer.)
- ? Show the computation of an optimal search-tree for $d_1 = 0.1$, $d_2 = 0.3$, $d_3 = 0.9$.
- ? What would the optimal tree for $d_i = i/(N + 1)$ for $1 \leq i \leq N$ look like? How about for $d_i = [i/(N + 1)]^2$?
- ? For each of 5 possible search-trees with $N = 3$ nodes, find an input data so that it is the unique optimal search tree.
- ? Show that the greedy approach of merging two neighbors which have the smallest total probability does not work.