1 Matrix-Chain Multiplication

Let we have n matrices $A_1, A_2, \ldots, A_n$. The size of matrix A_i is $p_{i-1} \times p_i$. The size of the matrices is stored in an array $p[0, 1, \ldots, n]$ of size $n+1$. The size of matrix A_i is $p_{i-1} \times p_i$ which is basically $p_{i-1} \times p_i$ when accessing the size from the array.

1.1 Approach-1

Algorithm 1 is used to find the minimum cost to multiply matrices $A_i \ldots A_j$. Our aim is to find the minimum cost to multiply matrices $A_1 \ldots A_n$. So the initial call to Algorithm 1 is MATRIX-CHAIN$(p, 1, n)$.

Algorithm 1 MATRIX-CHAIN$(p[\ ], i, j)$

1: **if** $i = j$ **then** ▷ Single matrix
2: **return** 0
3: **end if**
4: `minCost` $\leftarrow \infty$ ▷ Initialize the minimum cost to multiply matrices $A_i \ldots A_j$
5: **for** $k \leftarrow i$ **to** $j-1$ **do** ▷ Check for all possible parenthesis between $A_i \ldots A_j$
6: `cost` $\leftarrow$ $\underbrace{\text{MATRIX-CHAIN}(p, i, k)}_{\text{Minimum cost to multiply } A_i \ldots A_k} + \underbrace{\text{MATRIX-CHAIN}(p, k+1, j)}_{\text{Minimum cost to multiply } A_{k+1} \ldots A_j} +$
$\underbrace{p_{i-1} p_k p_j}_{\text{Cost to multiply two matrices } \underbrace{[A_i \ldots A_k]}_{\text{size}=p_{i-1} \times p_k} \text{ and } \underbrace{[A_{k+1} \ldots A_j]}_{\text{size}=p_k \times p_j}}$
7: **if** `cost` $<$ `minCost` **then** ▷ Cost using current parenthesis is less than the minimum cost
8: `minCost` $\leftarrow$ `cost` ▷ Update minimum cost
9: **end if**
10: **end for**
11: **return** `minCost` ▷ Return the minimum cost to multiply matrices $A_i \ldots A_j$

$$
\begin{aligned}
T(n) &= \sum_{k=1}^{n-1} [T(k) + T(n-k) + \mathcal{O}(1)] + \mathcal{O}(1) \\
&= \left[\sum_{k=1}^{n-1} (T(k) + T(n-k)) \right] + n \\
&= \left[2 \sum_{k=1}^{n-1} T(k) \right] + n \qquad (1)
\end{aligned}
$$

$$
T(n-1) = \left[2 \sum_{k=1}^{n-2} T(k) \right] + (n-1) \qquad (2)
$$

$$
\begin{aligned}
T(n) - T(n-1) &= 2T(n-1) + 1 \\
T(n) &= 3T(n-1) + 1 \qquad (3)
\end{aligned}
$$

$$
\begin{aligned}
T(n) &= 3T(n-1)+1 \\
&= 3\left[3T(n-2)+1\right]+1 \\
&= 3^2T(n-2)+(1+3) \\
&= 3^2\left[3T(n-3)+1\right]+(1+3) \\
&= 3^3T(n-3)+(1+3+3^2) \\
&\vdots = \vdots \\
&= 3^{n-1}T(1)+(1+3+3^2+\cdots+3^{n-2}) \\
&= 0+\frac{\left(3^{n-1}-1\right)}{3-1} \qquad\qquad T(1)=0 \\
&= \frac{1}{2}\left(3^{n-1}-1\right) \qquad (4)
\end{aligned}
$$

1.2 Approach-2

Algorithm 2 is used to find the minimum cost to multiply matrices $A_i \dots A_j$. Our aim is to find the minimum cost to multiply matrices $A_1 \dots A_n$. So the initial call to Algorithm 2 is MATRIX-CHAIN-STORE$(p, 1, n)$. In this Algorithm, we are storing the cost in a 2D-array (say $\mathbb{T}$) of size $n \times n$ and initialize it with -1 (any negative value) because the cost to multiply matrices can be 0 or more.

Algorithm 2 MATRIX-CHAIN-STORE$(p[\ ], i, j)$

1: **if** $i = j$ **then** ▷ Single matrix
2: $\mathbb{T}_{\texttt{cost}}[i][j] \leftarrow 0$
3: **return** 0
4: **end if**
5: **if** $\mathbb{T}_{\texttt{cost}}[i][j] \neq -1$ **then** ▷ Minimum cost to multiply matrices $A_i \dots A_j$ is already computed
6: **return** $\mathbb{T}_{\texttt{cost}}[i][j]$ ▷ Retrieve minimum cost to multiply matrices $A_i \dots A_j$ from the table and return
7: **end if**
8: $\texttt{minCost} \leftarrow \infty$ ▷ Initialize the minimum cost to multiply matrices $A_i \dots A_j$
9: **for** $k \leftarrow i$ **to** $j - 1$ **do** ▷ Check for all possible parenthesis between $A_i \dots A_j$
10: **if** $\mathbb{T}_{\texttt{cost}}[i][k] \neq -1$ **then** ▷ Minimum cost to multiply matrices $A_i \dots A_k$ is already computed
11: $\texttt{cost}_1 \leftarrow \mathbb{T}_{\texttt{cost}}[i][k]$ ▷ Retrieve minimum cost to multiply matrices $A_i \dots A_k$ from the table
12: **else** ▷ Minimum cost to multiply matrices $A_i \dots A_k$ is not computed
13: $\texttt{cost}_1 \leftarrow \underbrace{\text{MATRIX-CHAIN-STORE}(p, i, k)}_{\text{Minimum cost to multiply } A_i \dots A_k}$ ▷ Obtain the minimum cost to multiply matrices $A_i \dots A_k$
14: $\mathbb{T}_{\texttt{cost}}[i][k] \leftarrow \texttt{cost}_1$ ▷ Store the minimum cost to multiply matrices $A_i \dots A_j$ in the table. This step can be ignored because while computing the cost, the algorithm stores it in the table before returning the cost (See line no. 27 of this algorithm)
15: **end if**
16: **if** $\mathbb{T}_{\texttt{cost}}[k+1][j] \neq -1$ **then** ▷ Minimum cost to multiply matrices $A_{k+1} \dots A_j$ is already computed
17: $\texttt{cost}_2 \leftarrow \mathbb{T}_{\texttt{cost}}[k+1][j]$ ▷ Retrieve minimum cost to multiply matrices $A_{k+1} \dots A_j$ from the table
18: **else** ▷ Minimum cost to multiply matrices $A_{k+1} \dots A_j$ is not computed
19: $\texttt{cost}_2 \leftarrow \underbrace{\text{MATRIX-CHAIN-STORE}(p, k+1, j)}_{\text{Minimum cost to multiply } A_{k+1} \dots A_j}$ ▷ Obtain the minimum cost to multiply matrices $A_{k+1} \dots A_j$
20: $\mathbb{T}_{\texttt{cost}}[k+1][j] \leftarrow \texttt{cost}_2$ ▷ Store the minimum cost to multiply matrices $A_{k+1} \dots A_j$ in the table. This step can be ignored because while computing the cost, the algorithm stores it in the table before returning the cost (See line no. 27 of this algorithm)
21: **end if**
22: $\texttt{cost} \leftarrow \texttt{cost}_1 + \texttt{cost}_2 + \underbrace{p_{i-1}p_kp_j}_{\text{Cost to multiply two matrices } \underbrace{[A_i \dots A_k]}_{\text{size}=p_{i-1} \times p_k} \text{ and } \underbrace{[A_{k+1} \dots A_j]}_{\text{size}=p_k \times p_j}}$ ▷ Cost to multiply $A_i \dots A_j$
23: **if** $\texttt{cost} < \texttt{minCost}$ **then** ▷ Cost using current parenthesis is less than the minimum cost
24: $\texttt{minCost} \leftarrow \texttt{cost}$ ▷ Update minimum cost
25: **end if**
26: **end for**
27: $\mathbb{T}_{\texttt{cost}}[i][j] \leftarrow \texttt{minCost}$ ▷ Store the minimum cost to multiply matrices $A_i \dots A_j$ in the table
28: **return** minCost ▷ Return the minimum cost to multiply matrices $A_i \dots A_j$