IE 509 : WEEK 10

Submission Exercise: Chain Matrix Multiplication

Given real matrices $A_1, .., A_k$ of dimensions $m_i \times n_i, i = 1, .., k-1$, our goal is to compute the product $B = A_1 \cdot A_2 \cdots A_k$. Computing B involves $k-1$ matrix multiplications: e.g., we can first obtain $A_{1,2} = A_1 \cdot A_2$, and then $A_{1,3} = A_{1,2} \cdot A_3$, and so on until $B = A_{1,k-1} \cdot A_k$. The order of these multiplications will determine the effort in computing B. For example, let $k = 4$, and let the dimensions of the matrices be 50×20, 20×1, 1×10 and 10×100, respectively. Multiplying a $m \times n$ matrix with a $n \times p$ matrix takes mnp operations, and the following table shows three different ways of multiplying these matrices.

Parenthesization	Cost Computation	Cost
$A_1 \cdot ((A_2 \cdot A_3) \cdot A_4)$	$20 \cdot 1 \cdot 10 + 20 \cdot 10 \cdot 100 + 50 \cdot 20 \cdot 100$	120,200
$(A_1 \cdot (A_2 \cdot A_3)) \cdot A_4$	$20 \cdot 1 \cdot 10 + 50 \cdot 20 \cdot 10 + 50 \cdot 10 \cdot 100$	60,200
$(A_1 \cdot A_2) \cdot (A_3 \cdot A_4)$	$50 \cdot 20 \cdot 1 + 1 \cdot 10 \cdot 100 + 50 \cdot 1 \cdot 100$	7,000

Let $c(i, j)$ denote the minimum cost of computing $A_{i,j} = A_i \cdot A_{i+1} ... A_j$. Then, our goal is to compute $c(1, n)$. Consider the following algorithm for this problem:

for $i = 1$ **to** k **do**
  $c(i, i) \leftarrow 0$;
end
for $s = 1$ **to** $k - 1$ **do**
  for $i = 1$ **to** $k - s$ **do**
    $j \leftarrow i + s$;
    $c(i, j) \leftarrow \min\{c(i, k) + c(k + 1, j) + m_{i-1} \cdot m_k \cdot m_j \ : \ i \leq k < j\}$;
  end
end
return $c(1, n)$;

1. Write a python function that will take the dimensions of the matrices $A_1, A_2, ..., A_k$ as inputs and compute $c(1, n)$.

2. Modify your implementation so that the output gives the optimal order(parenthesization) in which the matrices are supposed to be multiplied.