1. Matrix Chain Multiplication

Suppose we wish to compute the product of three matrices: \( A_1A_2A_3 \). The dimensions of \( A_1 \) are \( d_0 \times d_1 \), the dimensions of \( A_2 \) are \( d_1 \times d_2 \), and the dimensions of \( A_3 \) are \( d_2 \times d_3 \).

Since matrix multiplication is associative, there are two different ways computing this product: either \(((A_1A_2)A_3)\) or \((A_1(A_2A_3))\). Both will yield the same result, but one way may be faster than the other. In general, multiplying a \( p \times q \) matrix by a \( q \times r \) matrix takes \( pqr \) scalar multiplications (and the product is a \( p \times r \) matrix).

(a) Suppose \( d_0 = 10, d_1 = 100, d_2 = 5, \) and \( d_3 = 50 \). Which way of computing \( A_1A_2A_3 \) is faster?

Suppose we now have \( n \) matrices in our multiplication chain: \( A_1A_2...A_n \). Given \( d_0, d_1, ...d_n \), we wish to design a dynamic programming algorithm to find the optimal way of computing the product. Notice that the number of possibilities increases sharply as \( n \) increases. For example, when \( n = 4 \) there are 5 possibilities: \((A_1(A_2(A_3A_4)))\), \((A_1((A_2A_3)A_4))\), \(((A_1A_2)(A_3A_4))\), \(((A_1(A_2A_3))A_4)\), or \(((A_1A_2)A_3)A_4)\).

Let’s define \( \text{BestCost}[i,j] \) as the optimal number of scalar multiplications needed to compute the sub-product \( A_iA_{i+1}...A_j \), for all \( i \geq j \).

(b) What base case(s) should we use? If you’re not sure yet, feel free to come back to this question later.

(c) Suppose the best way of computing \( A_iA_{i+1}...A_j \) is as \(((A_iA_{i+1}...A_k)(A_{k+1}A_{k+2}...A_j))\). What is the value of \( \text{BestCost}[i,j] \)? Hint: You’ll need 2 recursive references to other values in \( \text{BestCost} \).

(d) How many possible choices of \( k \) are there? Explain how to recursively compute \( \text{BestCost}[i,j] \) in general. (You may write a recurrence relation to do so, if you’d like.)

(e) In what order should we fill the array \( \text{BestCost} \)?

(f) Once \( \text{BestCost} \) is filled, how can we extract the optimal number of scalar multiplications needed to compute the whole chain?

(g) What is the runtime complexity of this dynamic programming algorithm?