Alin Tomescu | Week 14, Wednesday, May 7th, 2014 | Recitation 21
6.006 Intro to Algorithms | Prof. Srinivas Devadas | Prof. Nancy Lynch | Prof. Vinod Vaikuntanathan

Matrix chain multiplication

Input: A chain of matrices $A_1, A_2, \ldots, A_n$ where A_i has dimensions $m_{i-1} \times m_i$ (rows by columns). Note that consecutive matrices are *compatible* and can be multiplied.

Output: Give a parenthesization for the product $A_1 \times A_2 \times \ldots \times A_n$ that achieves the minimum number of element by element multiplications. Multiplying two matrices $A_{p,q} \times B_{q,r}$ requires pqr multiplications.

Thoughts: We have n matrices and $n-1$ ways to split up the product into two parenthesized products:

$$A_1 \times A_2 \times \ldots \times A_n = (A_1 \times A_2 \times \ldots \times A_k) \times (A_{k+1} \times A_{k+2} \times \ldots \times A_n)$$

If we can figure out how to pick this k for the size n problem, we've split our problem into two subproblems that we can (further recursively) solve using the same splitting mechanism.

The cost of computing our product of size n is split into *three costs*:

- Computing $A_{1..k} = A_1 \times A_2 \times \ldots \times A_k$
  - We further need to minimize the cost of this product w/ optimal parenthesization
- Computing $A_{k+1..n} = A_{k+1} \times A_{k+2} \times \ldots \times A_n$
  - We further need to minimize the cost of this product w/ optimal parenthesization
- Computing $A_{1..k} \times A_{k+1..n}$

Our problem formulation seems to be:

$$cost[1,n] = \min_{1<k<n} \{cost[1,k] + cost[k+1,n] + cost\ of\ multiplying\ the\ two\ expressions\}$$

What is the cost of multiplying the two expressions? Let's first see the dimensions of the two resulting matrices:

$$A_1 \times A_2 \times \ldots \times A_k = M_{rows(A_1),cols(A_k)} = M_{m_0,m_k}$$

$$A_{k+1} \times A_{k+2} \times \ldots \times A_n = M_{rows(A_{k+1}),cols(A_n)} = M_{m_k,m_n}$$

Then multiplying the two expressions should give us a matrix of size:

$$(A_1 \times A_2 \times \ldots \times A_k) \times (A_{k+1} \times A_{k+2} \times \ldots \times A_n) = M_{m_0,m_k} \times M_{m_k,m_n} = M_{m_0,m_n}$$

The cost for multiplying these two matrices is:

$$m_0 \times m_k \times m_n$$

Thus, we have:

$$cost[1,n] = \min_{1<k<n} \{cost[1,k] + cost[k+1,n] + m_0 \times m_k \times m_n\}$$

We can now notice the optimal substructure and define the recurrence.

Recurrence:

$$cost[i,j] = \text{minimum cost of multiplying matrices } A_i, A_{i+1}, \ldots, A_{j-1}, A_j$$

$$cost[i,j] = \min_{i<k<j} \{cost[i,k] + cost[k+1,j] + m_{i-1} \times m_k \times m_j\}$$

Base cases: A single matrix is easy to multiply with nothing.

$$cost[i,i] = 0$$

Multiplying two matrices $A_i \times A_{i+1}$ costs $m_{i-1} \times m_i \times m_{i+1}$ (see how matrix dimensions are defined above).

$$cost[i,i+1] = m_{i-1} \times m_i \times m_{i+1}$$

For anything bigger than 2 matrices we can use the recurrence to compute the minimum (optimal cost) solution.

Answer to the problem: $cost[1,n]$ will give us the minimum cost of the parenthesization. If we want to recover the actual parenthesization we have to store some extra information (like the k that was chosen) each time we compute $cost[i,j]$. We will need another matrix for this.

How to compute this bottom up: First, notice that bigger problems depend on smaller problems. So we should first compute all problems of size 3 products using the base case answers for size 2 products.

Then we can compute size 4, 5, 6, …, n.

Or, we can be lazy and use *memoization*.