Matrix-chain multiplication

Problem statement:

Input: *A sequence (chain) of* $(A_1, A_2, \ldots, A_n)$ *of matrices, where* A_i *is of order* $p_{i-1} \times p_i$.

Output: *full parenthesization (ordering) for the product* $A_1 \times A_2 \times \cdots \times A_n$ *that minimizes the number of (scalar) multiplications.*

Matrix-chain multiplication

- Counting the total number of orderings
  1. Define
     $P(n) =$ the number of orderings for a chain of n matrices
  2. Then for $n \geq 2$,
     $$P(n) = \sum_{k=1}^{n-1} P(k)P(n-k)$$
     and $P(1) = 1$
  3. It can be shown that $P(n) = \Omega(2^n)$

- A Brute-force solution by exhaustive search for determining the optimal ordering is infeasible!

Matrix-chain multiplication

DP – Step 1: *characterize the structure of an optimal ordering*

- An optimal ordering of the product $A_1 A_2 \cdots A_n$ splits the product between A_k and A_{k+1} for some k:

$$A_1 A_2 \cdots A_n = A_1 \cdots A_k \times A_{k+1} \cdots A_n$$

  and we first compute $A_1 \cdots A_k$ and $A_{k+1} \cdots A_n$, and then multiply them together.
- Key observation: the ordering of $A_1 \cdots A_k$ within this ("global") optimal ordering must be an optimal ordering of (sub-product) $A_1 \cdots A_k$. [1]
- Similar observation holds for $A_{k+1} \cdots A_n$
- Thus, an optimal ("global") solution contains within it the optimal ("local") solutions to subproblems. = **the optimal substructure property**

[1] Why? simply argue by contradiction: If there were a less costly way to order the product $A_1 \cdots A_k$, substituting that ordering within this (global) optimal ordering would produce another ordering of $A_1 A_2 \cdots A_n$, whose cost would be less than the optimum, a contradiction!

Matrix-chain multiplication

DP – Step 2: *recursively define the value of an optimal solution*

- Define

  $$m[i,j] = \text{min. number of multip. needed to compute } A_i \cdots A_j.$$

- By the definition,
  $m[1,n]$ = the cheapest way for the product $A_1 A_2 \cdots A_n$.

- $m[i,j]$ can be defined recursively:
  - if $i = j$, $m[i,i] = 0$.
  - if $i < j$, $m[i,j] = m[i,k] + m[k+1,j] + p_{i-1}p_kp_j$ for some k

Matrix-chain multiplication

DP – Step 2: *recursively define the value of an optimal solution*

- Thus, for $1 \leq i < j \leq n$,

$$m[i,j] = \begin{cases} 0 & \text{if } i = j \\ \min\limits_{i \leq k < j} \{m[i,k] + m[k+1,j] + p_{i-1}p_k p_j\} & \text{if } i < j \end{cases}$$

- In addition, to construct an optimal ordering, we keep track

$$s[i,j] = k_* = \text{the value s.t. } m[i,j] \text{ attains the minimum}$$

Matrix-chain multiplication

DP – Step 3: *compute the value of an optimal solution in a bottom-up approach*

- Compute $m[i, j]$ and $s[i, j]$ in a bottom-up approach. (pseudocode next page)

- Cost: $T(n) = \Theta(n^3)$ since
  1. compute roughly $n^2/2$ entries of m-table
  2. for each entry of m-table, it finds the minimum of fewer than n expressions.

Matrix-chain multiplication

```
matrix-chain-order(p)
create m[1...n,1...n] and s[1...n,1...n] and n = length(p)-1
for i = 1 to n
  m[i,i] = 0
endfor
for d = 2 to n
  for i = 1 to n-d+1
     j = i + d - 1
     //compute m[i,j]=min_k{...}
     m[i,j] = +infty
     for k = i to j-1
        q  = m[i,k] + m[k+1,j] + p[i-1]*p[k]*p[j]
        if q < m[i,j]
           m[i,j] = q
           s[i,j] = k  // track k such that min. is attained.
        endif
     endfor
  endfor
endfor
return m and s
```

Matrix-chain multiplication

DP – Step 4: *construct an optimal solution from computed information*

Example: Let `p = [30 35 15 5 10 20 25]`

`matrix-chain-order(p)` generates the m-array and s-array:

```
m = [ 0 15750 7875 9375 11875 15125 ]   s = [ 0 1 1 3 3 3 ]
    [ 0     0 2625 4375  7125 10500 ]       [ 0 0 2 3 3 3 ]
    [ 0     0    0  750  2500  5375 ]       [ 0 0 0 3 3 3 ]
    [ 0     0    0    0  1000  3500 ]       [ 0 0 0 0 4 5 ]
    [ 0     0    0    0     0  5000 ]       [ 0 0 0 0 0 5 ]
    [ 0     0    0    0     0     0 ]       [ 0 0 0 0 0 0 ]
```

By s-array, an optimal parenthesization/ordering is given by

$$(\, A_1 \,(\, A_2 \, A_3 \,)\,)\,(\,(\, A_4 \, A_5 \,)\, A_6 \,)$$