Matrix Multiplication

Note Title 05/12/2012

Given matrices $A_0, A_1, \ldots, A_{N-1}$
of dimensions $n_0 \times n_1,\ n_1 \times n_2,\ \ldots,\ n_{N-1} \times n_N$,
determine the parenthesisation that optimises
the evaluation of $A_0 \cdot A_1 \cdot \ldots \cdot A_{N-1}$.

Formally, constructively evaluate

$$\langle \Downarrow p : p \text{ is a parenthesisation of } N \text{ symbols} : \text{cost}_A\, p \rangle$$

Dynamic Programming

A dynamic programming solution of an optimisation problem

$$\langle \Downarrow S : \text{soln}.S : \text{cost}.S \rangle$$

has 3 phases:

- an inductive characterisation of the solution space
- use of distributivity properties ("the principle of optimality") to obtain a corresponding characterisation of the objective function
- solution of the equations so obtained in topological order.

1. Parenthesisation

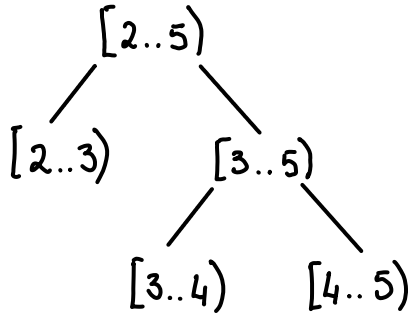
Suppose $0 \leqslant i < j \leqslant N$. A *parenthesisation* of the interval $[i..j)$ is a binary tree such that

(a) if $i+1 = j$, the tree is a leaf labelled $[i..j)$.

(b) if $i+1 < j$, the tree has root labelled $[i..j)$ and two subtrees;
the left subtree is a parenthesisation of $[i..k)$ and the right subtree is a parenthesisation of $[k..j)$ for some k, $i < k < j$.

Example

$(A_2 \cdot (A_3 \cdot A_4))$



2. The cost function

We will not be specific about the cost of evaluating matrix products — it could be, for example, the number of scalar multiplications — except to say that it is additive and depends only on the dimensions of the matrices. That is, we assume that, for given functions c and d,

$$\text{cost.}\left(\begin{array}{c}[i..i+1)\\ \bullet\end{array}\right) \;=\; d(i)$$

$$\text{cost.}\left(\begin{array}{ccc} & [i..j) & \\ [i..k) & & [k..j) \\ p & & q \end{array}\right) \;=\; c(i,k,j) + \text{cost.}p + \text{cost.}q$$

Suppose $i<j$.

Define

$$\text{mincost}(i,j) \;=\; \langle \Downarrow p:\ p \text{ parenthesises } [i..j) :\ \text{cost.}p \rangle .$$

Then

$$\begin{array}{ll} & \text{mincost}(i,i+1) \\ = & \quad \{ \text{ one-point rule } \} \\ & d(i) \ . \end{array}$$

Assume $i+1 < j$. Then

$mincost(i,j)$

$=$ { definition }

$\langle \Downarrow p : p \text{ parenthesises } [i..j) : cost.p \rangle$

$=$ { inductive characterisation of paren. }

$$\left\langle \Downarrow p : \left\langle \exists k : i<k<j : \begin{array}{c} p = [i..j) \\ [i..k) \quad [k..j) \\ q \quad r \end{array} \right\rangle : cost.p \right\rangle$$

$=$ { range disjunction, one-pt. rule, defn. of cost }

$\langle \Downarrow k : i<k<j :$

$\quad \langle \Downarrow q, r : q \text{ parenthesises } [i..k) \wedge r \text{ parenthesises } [k..j) : c(i,k,j) + cost.q + cost.r \rangle$

$\rangle$

"principle of optimality"

$=$ { distributivity of addition over $\downarrow$ }

$\langle \Downarrow k : i<k<j :$

$\quad c(i,k,j)$

$\quad + \langle \Downarrow q : q \text{ parenthesises } [i..k) : cost.q \rangle$

$\quad + \langle \Downarrow r : r \text{ parenthesises } [k..j) : cost.r \rangle$

$\rangle$

$=$ { definition of mincost }

$\langle \Downarrow k : i<k<j : c(i,k,j) + mincost(i,k) + mincost(k,j) \rangle$

3. Topological Ordering

An interval $[i..j)$ is a *subinterval* of $[k..l)$ if $k \leqslant i < j \leqslant l$.

The subinterval relation is a well-founded partial ordering on intervals.

Conclusion

Calculate $mincost(0,N)$ where

$$mincost(i, i+1) = d(i)$$

$$mincost(i,j) = \langle \Downarrow k : i < k < j : c(i,k,j) + mincost(i,k) + mincost(k,j) \rangle \quad \text{for } i+1 < j.$$

mincost can be evaluated in topological order of subintervals (eg. intervals of size 1, 2, 3, etc.).