16.04.2020

18. Optimal Matrix chain Multiplication

A product of matrices is fully paranthesized if it is either

- a single matrix, or
- a product of 2 fully parenthesized matrices, surrounded by parentheses.

Each parenthesization defines a set of n-1 matrix multiplications.
We just need to pick the parenthesization that corresponds to best ordering.

There are $\frac{^{2n}C_n}{n+1}$ ways to parenthesize the matrix ∴ Trying out all possibilities is a bad idea, or $P(n) = \Omega(4^n/n^{3/2})$

Dynamic Programming steps:

1. Characterize the structure of an optimal solution.
2. Recursively define the value of an optimal solution.
3. Compute the value of an optimal solution bottom-up.
4. Construct an optimal solution from the computed info.

(i) <u>structure of optimal solution:</u>

If the outermost parenthesization is $((A_1 A_2 \dots A_i)(A_{i+1} \dots A_n))$, then the optimal solution consists of solving $A_{1,i}$ and $A_{i+1,n}$ optimally & then combining the solutions.

(ii) <u>Recursive solution</u>

In the enumeration of $P(n) = \Omega(4^n / n^{3/2})$ subproblems, a subproblem is of the form A_{ij} with $1 \le i, j \le n$, so there are $O(n^2)$ unique subproblems.

Let A_i be a $p_{i-1} \times p_i$ matrix.
Let $m[i,j]$ be the cost of computing A_{ij}.
If the final multiplication for A_{ij} is $A_{ij} = A_{ik} \cdot A_{k+1,j}$ then

$$m[i,j] = m[i,k] + m[k+1,j] + p_{i-1} p_k p_j.$$

We don't know 'k' a priori, so we take the minimum.

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

(iii) <u>Computing the optimal costs</u>

```
MATRIX-CHAIN-ORDER(p)
    n = p.length - 1
    let m[1..n, 1..n] and s[1..n-1, 2..n] be new tables.
    for i = 1 to n:
        m[i, i] = 0
    for l = 2 to n:        // l is the chain length.
        for i = 1 to n-l+1:
            j = i+l-1
            m[i,j] = ∞
            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
    return m and s
```

(iv) Constructing an optimal solution.

- Although MATRIX-CHAIN-ORDER determines the optimal no. of scalar multiplications needed to compute a matrix-chain product, it does not directly show how to multiply the matrices.
- the table $s[1..n-1, 2..n]$ gives us the info. we need to do so. Each entry $s[i,j]$ records a value of k such that an optimal parenthesization of $A_i A_{i+1} \ldots A_j$ splits the product between A_k and A_{k+1}.

```
PRINT-OPTIMAL-PARENS (s, i, j)
    if i==j:
        print "A";
    else print "("
        PRINT-OPTIMAL-PARENS (s, i, s[i,j])
        PRINT-OPTIMAL-PARENS (s, s[i,j]+1, j)
        print ")"
```