CS 380
Matrix Chain Multiplication Example from Lecture 22
April 19th, 2019

From lecture, we were considering optimizing the matrix chain $A_1A_2A_3A_4A_5A_6$ given the following dimensions.

matrix		A_1	A_2	A_3	A_4	A_5	A_6
Dimension		30x35	35x15	15x5	5x10	10x20	20x25

where matrix A_i has dimensions p_{i-1} x p_i.

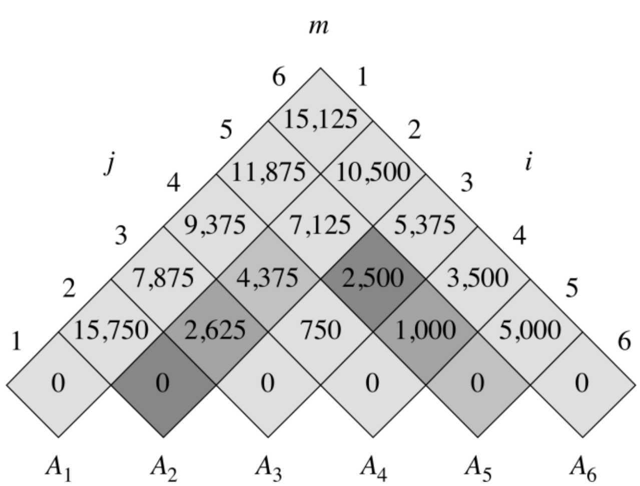
This necessitates computing two matrices, indicated by m and s, that store respectively the costs of multiplying matrix subchains (matrix m) in addition to the optimal dissection point k given a matrix subchain (matrix s).

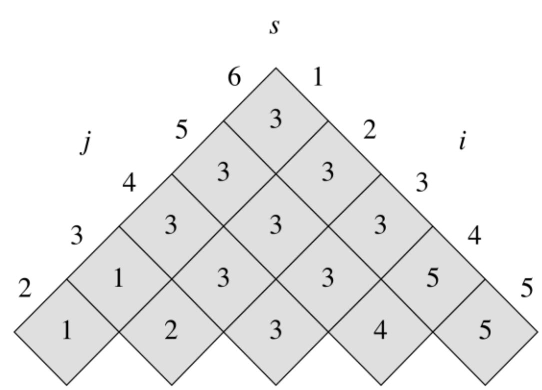
The matrix m is constructed in a row-wise manner using the computation below:

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

where m[i,j] is the minimum number of multiplications required to produce the subchain $A_i \ldots A_j$ given that it is dissected as $A_i \ldots A_k A_{(k+1)} \ldots A_j$. In particular, notice that we do NOT consider all possible ways to insert parenthesis into this matrix subchain, but rather the number of ways to dissect this chain into two subgroups $A_i \ldots A_k$ and $A_{(k+1)} \ldots A_j$ (i.e. the outer-most pairs of parenthesis). We need to look further down the tree to determine how best to dissect each of these sub (sub) matrix chains.

Regardless, it should be clear why the bottom row consists of 0's, and the second row consists of the costs of multiplying matrix A_i with matrix A_j.





Things become more interesting on the third row and above because there are more options to consider. In particular, consider the entry m[2,5], which indicates the minimum cost of multiplying the matrix chain $A_2A_3A_4A_5$ given some as-of-yet unknown dissection point k.

Notice that there are three possible ways to insert the outermost parenthesis to break this matrix chain into two subchains:

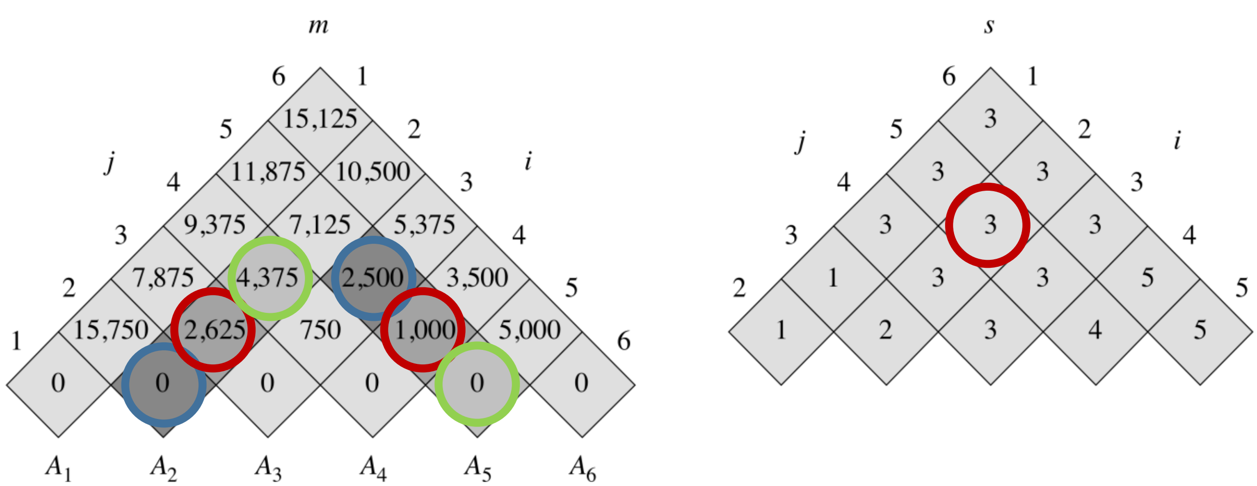
$(A_2)(A_3A_4A_5)$

$(A_2A_3)(A_4A_5)$

$(A_2A_3A_4)(A_5)$

$$m[2,5] = \min \begin{cases} m[2,2] + m[3,5] + p_1 p_2 p_5 = 0 + 2500 + 35 \cdot 15 \cdot 20 = 13{,}000 , \\ m[2,3] + m[4,5] + p_1 p_3 p_5 = 2625 + 1000 + 35 \cdot 5 \cdot 20 = 7125 , \\ m[2,4] + m[5,5] + p_1 p_4 p_5 = 4375 + 0 + 35 \cdot 10 \cdot 20 = 11{,}375 \end{cases}$$
$$= 7125 .$$

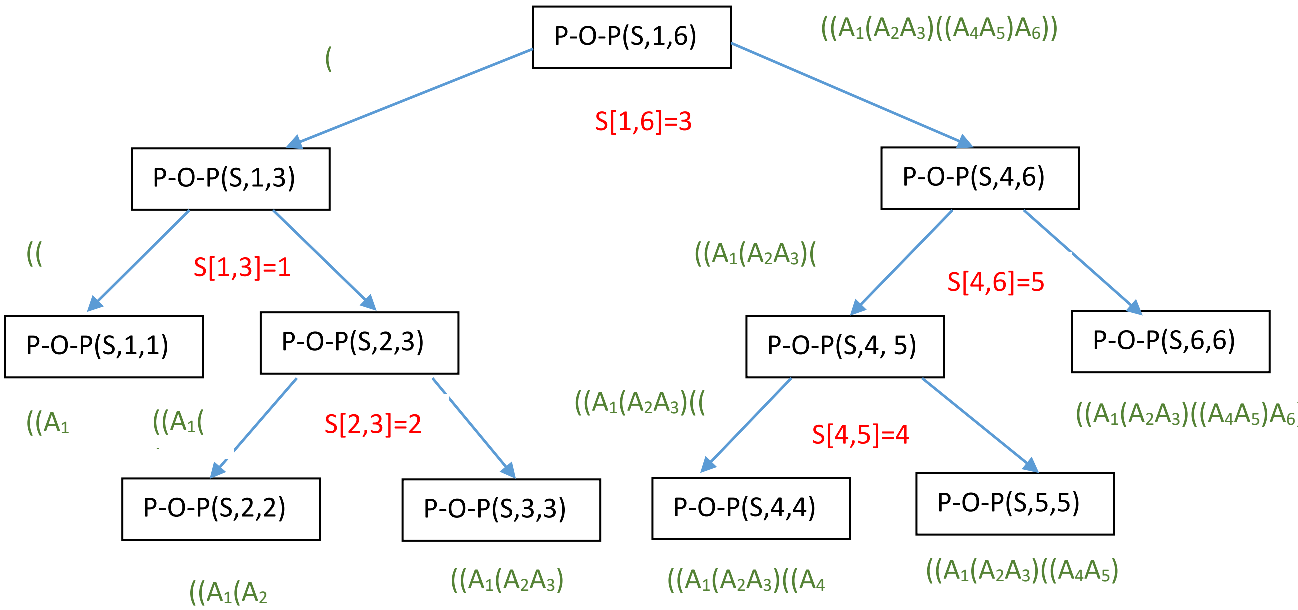
Note k = 3 corresponds to minimum, so enter this in s array



The optimal cost is entry m[1,6] = 15,125. Recovering the actual parenthesis structure requires the use of this recursive algorithm:

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

In our example, the sequence of function calls and output would be:



Exercise 15.2-1: Find an optimal parentheisization of a matrix-chain product whose sequence of dimensions is <5,10,3,12,5,50,6>.