11-02-25

* Matrix multiplicatⁿ

Input : $A = [a_{ij}]$, $B = [b_{ij}]$
Output : $C = [c_{ij}]$

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

Running time $= \Theta(n^3)$

Divide & Conquer - Idea :

$n \times n$ matrix $= 2 \times 2$ matrix of $(n/2) \times (n/2)$ submatrices

$$\begin{bmatrix} r & s \\ t & u \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \cdot \begin{bmatrix} e & f \\ g & h \end{bmatrix}$$

$$C \quad = \quad A \cdot B$$

$$\left.\begin{array}{l} r = ae + bg, \quad s = af + bh \\ t = ce + dg, \quad u = cf + dh \end{array}\right\} \begin{array}{l} 4 \text{ additions.} \\ 8 \text{ multiplications.} \end{array}$$

$$T(n) = \underset{\text{submatrices}}{8} \cdot T\underbrace{(n/2)}_{\text{size of submatrix}} + \underbrace{\Theta(n^2)}_{\text{work adding submatrices}}$$

$$\underbrace{T(n) = \Theta(n^3)}_{\text{found using Master Th}^m} \implies \text{no improvement}$$

- Strassen's Idea

Multiply 2X2 matrices with only 7 recursive mults.

$P_1 = a\cdot(f-h)$

$P_2 = (a+b)\cdot h$

$P_3 = (c+d)\cdot e$

$P_4 = d\cdot(g-e)$

$P_5 = (a+d)\cdot(e+h)$

$P_6 = (b-d)\cdot(g+h)$

$P_7 = (a-c)\cdot(e+f)$

$r = P_5 + P_4 - P_2 + P_6$

$s = P_1 + P_2$

$t = P_3 + P_4$

$u = P_5 + P_1 - P_3 - P_7$

7 mults., 18 add/subs.

Note: No reliance on commutativity ~~##~~ of mult.

On solving for r we get the same value as found in divide & conquer approach.

1.) Divide: divide 2X2 matrices in $(n/2)\times(n/2)$ submatrices.

2.) Conquer: perform 7 mults recursively on the submatrices.

3.) Combine: Form matrix C using add/subs.

Analysis: $T(n) = 7\,T(n/2) + \Theta(n^2)$

$$n^{\log_b a} = n^{\log_2 7} \approx n^{2.81} \Longrightarrow \text{Case 1}: T(n) = \Theta(n^{\lg 7})$$

2.81 isn't much smaller than 3 but bcoz the difference is in the exp., its impact on running time is significant. This algo. beats the ordinary algo. on today's machines for $n \geqslant 32$ or so.

06-03-25

* Matrix-Chain Multiplicat^n

- paranthesize in a way that minimizes the no. of scalar multiplicat^{n}s.
- For a chain of matrices $\langle A_1, A_2, A_3, A_4 \rangle$, we can paranthesize the product in 5 distinct ways:

$(A_1(A_2(A_3A_4)))$
$(A_1((A_2A_3)A_4))$
$((A_1A_2)(A_3A_4))$
$((A_1(A_2A_3))A_4)$
$(((A_1A_2)A_3)A_4)$

- Matrix-Multiply:

```
if A.cols ≠ B.rows
    error "incompatible dimensions"
else let C be a new A.rows X B.cols
    for i=1 to A.rows
```

- Given a chain of n matrices $\langle A_1, A_2, A_3, \dots A_n \rangle$
- Counting no. of ~~paranthes~~ paranthesizat^{n}s of a seq. of n matrices by $P(n)$.
- The split b/w two subproducts may occur b/w k-th and $(k+1)$-st matrices for any k.

$$P(n) = \begin{cases} 1 & ; \text{if } n=1, \\ \sum_{k=1}^{n-1} P(k)P(n-k) & ; \text{if } n \geq 2. \end{cases}$$

sol^n to recurrence is $\Omega(4^n/n^{3/2})$.

catalan no.

- $P(n) = C(n) = \frac{(2n)!}{n!\,(n+1)!}$

no. of operators

- Applying DP
  - → characterize struct. of optimal solⁿ.
  - → recursively define value of optimal solⁿ.
  - → compute value of an optimal solⁿ in a bottom-up fashion.
  - → construct optimal solⁿ from computed info.

Step 1: Optimal Structure

Find optimal substruct & then use it to construct an optimal solⁿ to the problem from optimal solⁿs to subproblems.

$A_{i\ldots j}$: the matrix resulting from evaluating the product $A_i A_{i+1} \ldots A_{i+j}$

We must split the product b/w A_k & A_{k+1} for some int. k in the range $i \leq k \leq j$.

i.e., for some val of k, we first compute the matrices $A_{i\ldots k}$ & $A_{k+1\ldots j}$ & then multiply them together to produce final product $A_{i\ldots j}$.

Step 2: Recursive Solⁿ

Can define $m[i,j]$ recursively as follows

→ If $i=j$, $m[i,j]=0$ for $1 \leq i = j \leq n$

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

The eq^n assumes that we know value of k, which we don't.

There are only j-i possible values for k, however, namely k=i, i+1, ..., j-1.

Optimal paranthesizat^n must use one of these values for k, so need to check them all to find the best.

∴ Recursive def^n for min. cost of paranthesizing product $A_i A_{i+1} \dots A_j$ becomes

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

The eq^n assumes that we know value of k, which we don't.

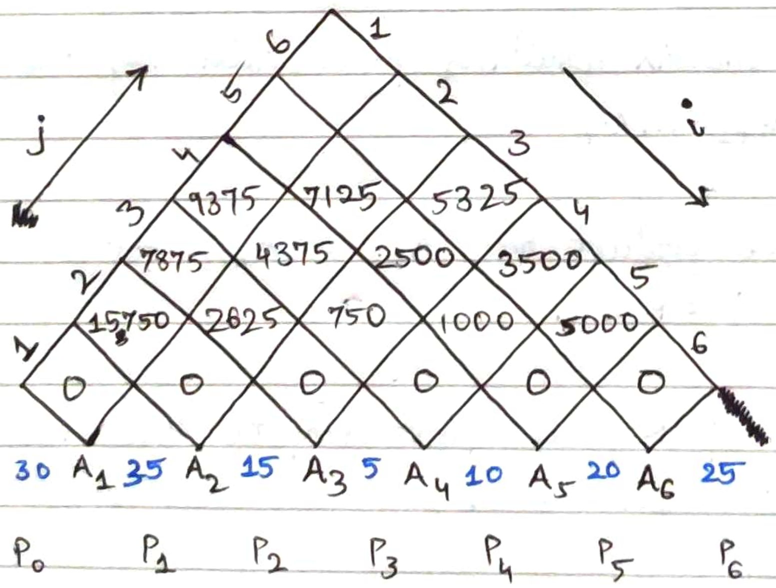
There are only $j-i$ possible values for k, however, namely $k=i$, $i+1, \ldots, j-1$.

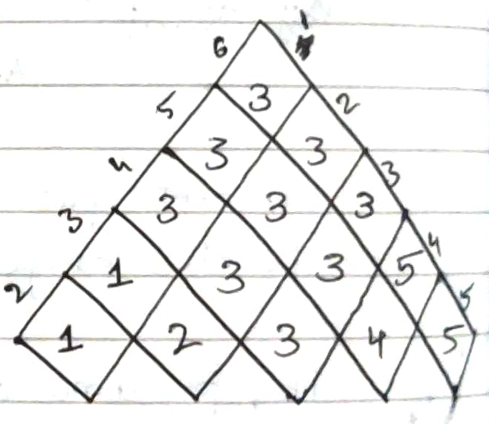
Optimal paranthesizat^n must use one of these values for k, so need to check them all to find the best.

$\therefore$ Recursive def^n for min. cost of paranthesizing product $A_i A_{i+1} \ldots A_j$ becomes

$$m[i,j] = \begin{cases} 0 & ; \text{ if } i=j \\ \min\limits_{i \le k < j} \{ m[i,k] + m[k+1,j] + P_{i-1} P_k P_j & ; \text{ if } i<j \end{cases}$$

eg.





$m[1,2] = m[1,1] + m[2,2] + P_0 P_1 P_2$

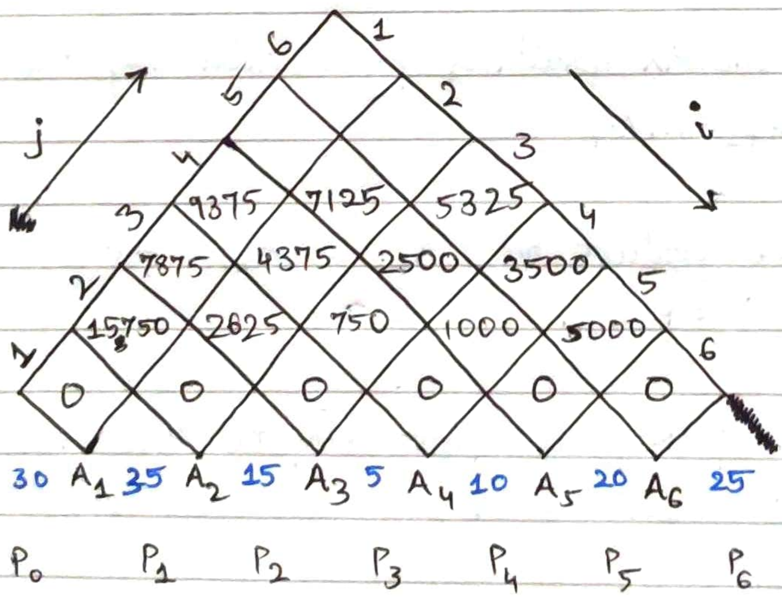
$k=1$ $\quad = 0 + 0 + 30 \times 35 \times 15$

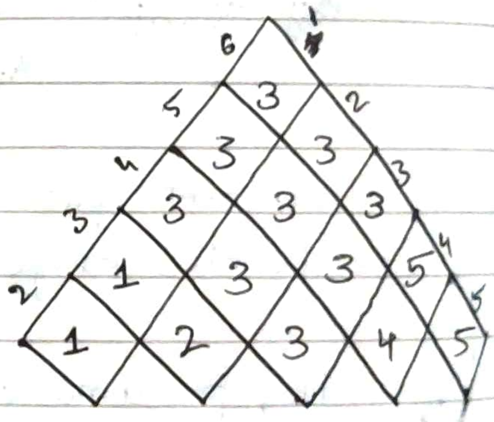
$= 15{,}750$

$m[3,4] = m[3,3] + m[4,4] + P_2 P_3 P_4$

$k=3$ $\quad = 0 + 0 + 15 \times 5 \times 10 = 750$

eg.





$m[1,2] = m[1,1] + m[2,2] + P_0 P_1 P_2$

$k=1$ $\quad = 0 + 0 + 30 \times 35 \times 15$

$= 15,750$

$m[3,4] = m[3,3] + m[4,4] + P_2 P_3 P_4$

$k=3$ $\quad = 0 + 0 + 15 \times 5 \times 10 = 750$

$\cancel{m[1,3] = m[1,2]}$

$$m[1,3] = \min \begin{cases} m[1,1] + m[2,3] + P_0 P_1 P_3 = 0 + 2625 + 5250 \\ \qquad = 7875 \\ m[1,2] + m[3,3] + P_0 P_2 P_3 = 15750 + 0 + 2250 \\ \qquad = 18000 \end{cases}$$

$= 7875$

$$m[2,4] = \min \begin{cases} m[2,2] + m[3,4] + P_1 P_2 P_4 = 0 + 750 + 5250 \\ \qquad = 6000 \\ m[2,3] + m[4,4] + P_1 P_3 P_4 = 2625 + 0 \; 1750 \\ \qquad = 4375 \end{cases}$$

$= 4375$

07-03-25

$$m[3,5] = \min \begin{cases} m[3,3] + m[4,5] + P_2 P_3 P_5 = 0 + 1000 + 15(5 \times 20) \\ \qquad = 2500 \\ m[3,4] + m[5,5] + P_2 P_4 P_5 = 750 + 0 + 15(10 \times 20) \\ \qquad = 3750 \end{cases}$$

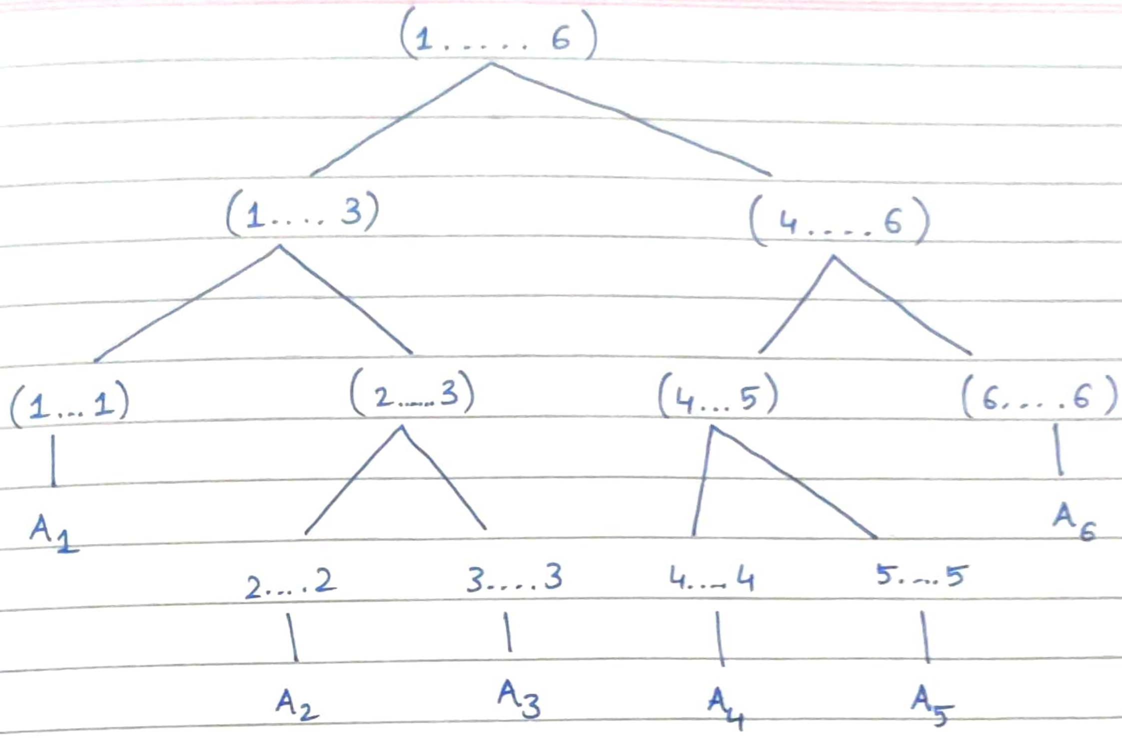
$= 2500$

$$m[4,6] = \min \begin{cases} m[4,4] + m[5,6] + P_3 P_4 P_6 = 0 + 5000 + 5 \times 10 \times 25 \\ \qquad = 6250 \\ m[4,5] + m[6,6] + P_3 P_5 P_6 = 1000 + 0 + 5 \times 20 \times 25 \\ \qquad = 3500 \end{cases}$$

$$m[1,4] = \min \begin{cases} m[1,1] + m[2,4] + P_0 P_1 P_4 \\ = 0 + 4375 + 30 \times 35 \times 10 = 14875 \\ \\ m[1,2] + m[3,4] + P_0 P_2 P_4 \\ = 15750 + 750 + 30 \times 15 \times 10 \\ = 21000 \\ \\ m[1,3] + m[4,4] + P_0 P_3 P_4 \\ = 7875 + 0 + 30 \times 5 \times 10 \\ = 9375 \end{cases}$$

$$m[2,5] = \min \begin{cases} m[2,2] + m[3,5] + P_1 P_2 P_5 \\ = 13000 \\ \\ m[2,3] + m[4,5] + P_1 P_3 P_5 \\ = 7125 \\ \\ m[2,4] + m[5,5] + P_1 P_4 P_5 \\ = 11325 \end{cases}$$

```
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], j)
        print ")"
```



O/P:

$$((A_1(A_2A_3))((A_4A_5)A_6))$$