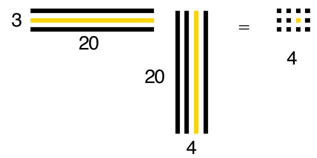
Matrix-chain Products

- Given: p_{i-1} x p_i matrices A_i , $1 \le i \le n$
- Problem: Compute $A_1 \bullet A_2 \bullet \ldots \bullet A_n$



Work = 3 * 20 * 4

In general:
p × q times q × r
costs p*q*r

Matrix-chain Products

- Given: p_{i-1} x p_i matrices A_i , $1 \le i \le n$
- Problem: Compute $A_1 \bullet A_2 \bullet \ldots \bullet A_n$

- In What Order?
- Example: A•B•C, where:
  - A is 1 x 10
  - B is 10 x 2
  - C is 2 x 10

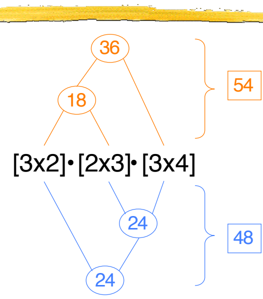
(A • B) • C	A • (B • C)
1*10*2 + 1*2*10	1*10*10 + 10*2*10
40	300

A Greedy Algorithm?

In above example, it was best to start with the cheapest adjacent pair.

Always true?

No.



Simple Algorithm

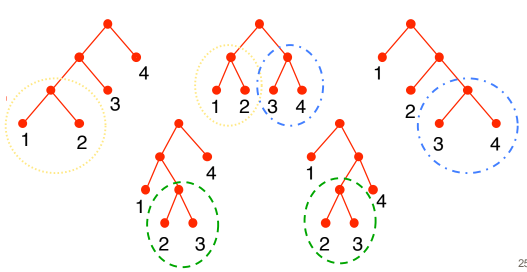
- Just try all possible parenthesizations
- How many are there?
  - $P(1) = 1$

$$P(n) = \sum_{k=1}^{n-1} P(k)P(n-k), n > 1$$

- $$P(n) = \frac{1}{n}\binom{2n-2}{n-1} = \Omega\left(\frac{4^n}{n^{3/2}}\right)$$

Repeated Subproblems

- All 5 Parenthesizations of $A_1 \bullet A_2 \bullet A_3 \bullet A_4$:



Optimal Substructure:

- **Theorem:** if the last multiply is $(A_1 \ldots A_i) \bullet (A_{i+1} \ldots A_n)$, then $A_1 \ldots A_i$ is optimally parenthesized, as is $A_{i+1} \ldots A_n$.

  Proof: Could improve if not.

- Useful? Two problems:
  - Don't know i.
  - $(A_1 \ldots A_i)$ is a prefix of input, but not $(A_{i+1} \ldots A_n)$

Optimal Substructure: Strengthened Induction Hyp.

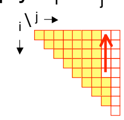
- **Theorem:** if the last mult in opt calculation of $A_i \ldots A_j$ is $(A_i \ldots A_k) \bullet (A_{k+1} \ldots A_j)$, then $A_i \ldots A_k$ is optimally parenthesized, as is $A_{k+1} \ldots A_j$.
  Proof: Could improve if not.
- Let M[i,j] = min ops to multiply $A_i \ldots A_j$

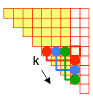
$$M[i,j] = \begin{cases} 0 & i = j \\ \min_{i \le k < j} (M[i,k] + M[k+1,j] + p_{i-1} p_k p_j) & i < j \end{cases}$$

An O(n³) Algorithm

// Goal: M[i,j] = min ops to multiply $A_i \ldots A_j$

for j := 1 to n do
  M[j,j] := 0;
  for i := (j - 1) downto 1 do
    M[i,j] := $\min_{i \le k < j}(p_{i-1} p_k p_j$ +
          M[i,k]+M[k+1,j]);





Example:

	1	2	3	4
1				
2				
3				
4				

$p_0 = 2$
$p_1 = 3$
$p_2 = 1$
$p_3 = 5$
$p_4 = 1$

A_1: 2x3
A_2: 3x1
A_3: 1x5
A_4: 5x1

Notes

- Diagonal M[i,i+2], e.g., gives best cost for multiplying adjacent triples $A_i A_{i+1} A_{i+2}$
  - Exercise: rewrite alg to compute successive diagonals instead of successive columns
  - Question: can it be rewritten to compute successive rows?
- $n^3 \rightarrow n \log n$ time is possible (but not easy)
- General structure of algorithm is useful for other problems on trees
  - E.g., go look up "CKY" alg for context-free parsing