Dynamic Programming

- Divides the problem into overlapping subproblems of same type.
- In DAC the subproblems are independent.
- Subproblems share same subproblems.
- DP saves time by solving suproblem once and using it in problems which share this subproblem.
- It is applicable to problems which exhibit optimal substructure property.

Matrix Chain Product

- Consider a chain of matrix: $\langle A_1, A_2, \ldots, A_N \rangle$
- Objective: minimize the number of scalar multiplications for finding the product.
- Associativity holds: so

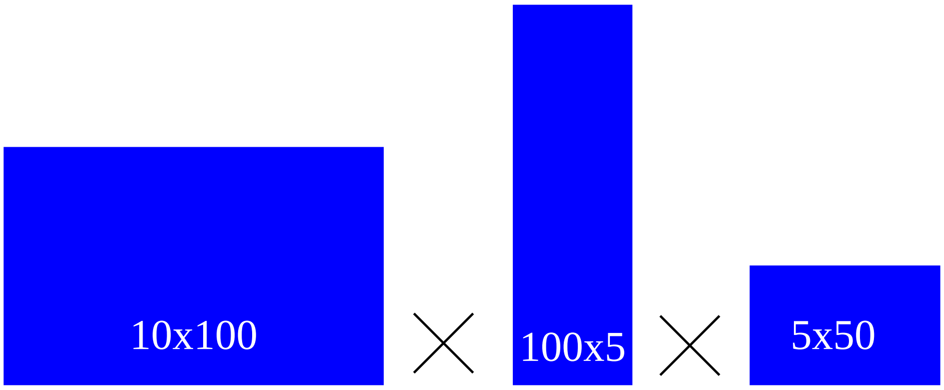
$$\begin{aligned} A_1 \times A_2 \times A_3 &= A_1 \times (A_2 \times A_3) \\ &= (A_1 \times A_2) \times A_3 \end{aligned}$$

- Define cost as follows:

$$\begin{aligned} \text{Cost}(A_1, A_2, A_3) &= \text{Cost}(A_2, A_3) \\ &\quad + \text{Cost}(A_1, (A_2 \times A_3)) \\ \text{Cost}(A_1, A_2, A_3) &= \text{Cost}(A_1, A_2) \\ &\quad + \text{Cost}((A_1 \times A_2), A_3), \text{ or} \end{aligned}$$

- Thus ordering the multiplication in several ways, costs can be different.
- The task: to find the order with least cost.

Matrix Chain Product



- Cost of $(A_1 \times A_2) \times A_3$ is

$$10 \times 100 \times 5 + 10 \times 5 \times 50 = 7500$$

- Cost of $A_1 \times (A_2 \times A_3)$ is

$$100 \times 5 \times 50 + 100 \times 10 \times 50 = 75000$$

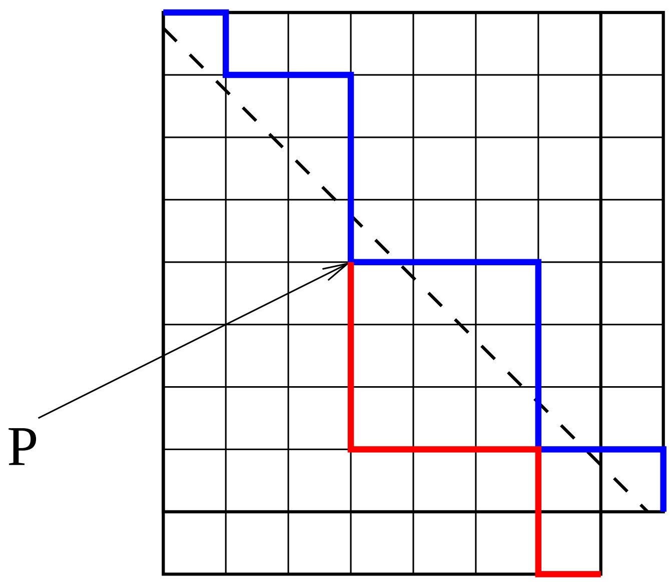
- Ten fold speedup can be achieved just changing the order!

Balanced Parentheses

- The ordering of matrix chain product is same as placing parentheses around n symbols, so that the expression is balanced.
- How many ways parentheses can be placed?
- Before the problem can be solved let us find its equivalent forms.

Number of Good Paths in a Grid

- In a grid of size $n \times n$ finding path from one corner point to other requires a sequence of right and down moves.
- Exactly $2n$ moves are required.
- But out of many sequence of moves we are interested in those that do not cross the main diagonal.



Good Paths in a Grid

- Consider the path shown in blue. It first crosses the main diagonal at P.
- We can reflect the part of path after P around main diagonal.
- Every right move becomes a down move and vice versa.
- The path becomes a path in $(n+1)\times n$ grid due to the reflection.
- This can be formally proved as follows.

Good Paths in a Grid

- When the path reaches at P it must have made one more down move than the number of right moves.
- Suppose there are k right moves then the number of down moves $= k + 1$.
- So remaining $n - k$ right moves and $n - k - 1$ down moves should be made to reach the other corner.
- But due to reflection $n - k - 1$ right moves are done and $n - k$ down moves are made.
- So total number of right and down moves are respectively

  $k+(n-k-1) = n-1$, and $k+1+(n-k) = n+1$.

Good Paths in a Grid

- Hence, every bad move is put in 1-1 correspondence with a path in $(n+1) \times (n-1)$ grid.
- Number of paths between two opposite corners in a $m \times k$ grid is $\binom{m+k}{m}$.
- Thus number of paths in $(n+1) \times (n-1)$ grid is $\binom{2n}{n+1}$.
- Implying number of good paths

$$C_n = \binom{2n}{n} - \binom{2n}{n+1} = \frac{1}{n+1}\binom{2n}{n}$$

- So exhaustive search for solution requires exponential time.

Optimal Parenthesization

- Let $A_{i...j}$ denote the matrix resulting from evaluationg product $A_i \times A_{i+1} \times \ldots \times A_j$
- Split the product $A_{1...n}$ between A_k and A_{k+1}, $1 \leq k < n$.
- That is compute $A_{1...k}$ and $A_{(k+1)...n}$ separately then multiply the two resulting matrices.
- Cost will be the cost of computing $A_{1...k}$ plus the cost of $A_{(k+1)...n}$.
- Note that the parenthesization of prefix subchain $A_1 \times A_2 \times \ldots \times A_k$ must be optimal also.
- An optmal solution has instances of optimal solutions of subproblems within it.
- So we define optimal solution of a problem instance as a combination of optimal solutions of smaller problem instances.

