ALG 11

Dynamic programming

Longest increasing subsequence (LIS)

Matrix chain multiplication

Longest increasing subsequence (LIS)

The longest increasing subsequence may not be contiguous.

5 4 9 11 5 3 2 10 0 8 6 1 7

Solution: 4 5 6 7

Possible problem modifications

Subsequence properties:
decreasing, non-decreasing, non-increasing, arithmetic,
with bounded growth rate, with weighted elements, ... etc., ...

not explicitly analysed here

Standard DP approach

Transform to known problem, define appropriate DAG according to the subsequence properties, find longest path in DAG.

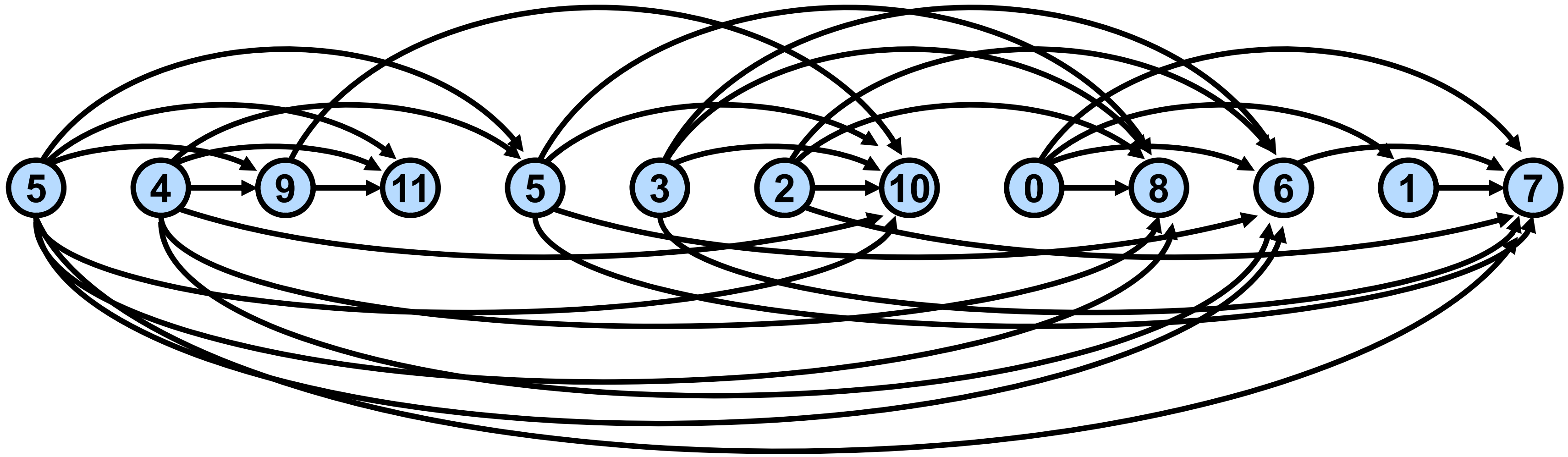
Longest increasing subsequence (LIS)

Standard DP approach

Transformation to the earlier problem

The sequence elements are DAG nodes. DAG is topologically sorted, position in sequence = position in top. ordering.
Edge x --> y exists if and only if
order of x is lower than order of y and also x<y.

Find longest path in this DAG.



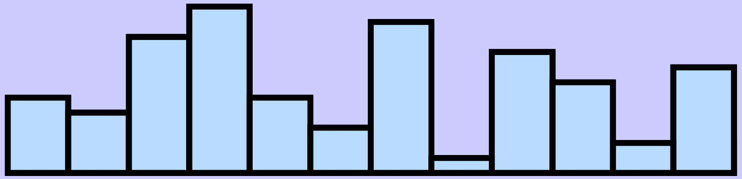
Algorithm is known, its complexity is $\Theta(N+M) \subseteq O(N^2)$.
If the sequence is increasing then the complexity is $\Theta(N^2)$.

Longest increasing subsequence (LIS)

Original faster DP approach

Regster optimal subsequences of all possible lengths and in each step update one of them.

k	1	2	3	4	5	6	7	8	9	10	11	12



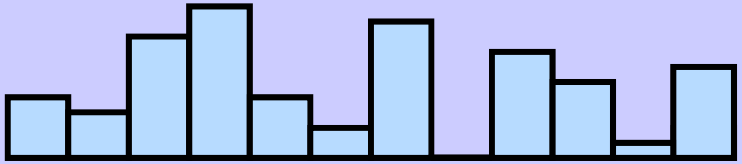
V	5	4	9	11	5	3	10	1	8	6	2	7
p	--											
iL	--											

DP table:
k .. index of element
V .. value of element
p .. predecessor
iL .. index of the last element in an increasing optimal subsequence with length d = 1, 2, ..., N.

For each index k:
Let d be the index of the biggest element, which satisfies
V[iL[d]] < V[k].
Set iL[d+1] := k, p[k] := iL[d], if such d exists.
Else iL[1] := k, p[k] := null.

Longest increasing subsequence (LIS)

k	1	2	3	4	5	6	7	8	9	10	11	12



V	4	3	8	10	4	2	9	0	7	5	1	6

k = 11

p	--	--	2	3	2	--	5	--	5	5	8	
iL	8	11	10									
V[iL]	0	1	5									

k = 12

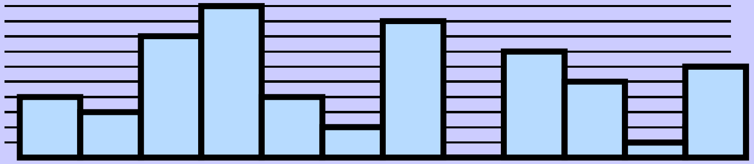
p	--	--	2	3	2	--	5	--	5	5	8	10
iL	8	11	10	12								
V[iL]	0	1	5	6								

k .. index of element
V .. value of element
p .. predecessor
iL .. index of the last element in an increasing optimal subsequence with length d = 1, 2, ..., N.

For each index k:
Let d be an index of max. elem. which satisfies V[iL[d]] < V[k].
Then iL[d+1] := k, p[k] := iL[d], if such d exists.
Else iL[1] := k, p[k] := null.

Longest increasing subsequence (LIS)

k	1	2	3	4	5	6	7	8	9	10	11	12



V	4	3	8	10	4	2	9	0	7	5	1	6

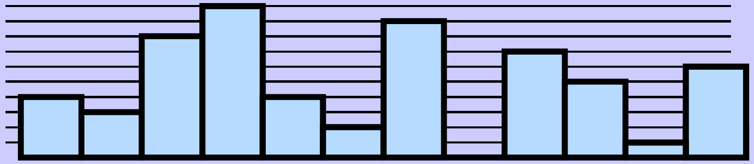
p	--											
iL	1											
V[iL]	4											

p	--	--										
iL	2											
V[iL]	3											

p	--	--	2									
iL	2	3										
V[iL]	3	8										

p	--	--	2	3								
iL	2	3	4									
V[iL]	3	8	10									

k	1	2	3	4	5	6	7	8	9	10	11	12



V	4	3	8	10	4	2	9	0	7	5	1	6

p	--	--	2	3	2							
iL	2	5	4									
V[iL]	3	4	10									

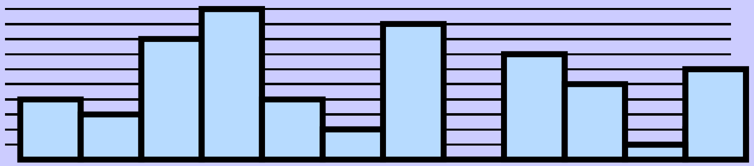
p	--	--	2	3	2	--						
iL	6	5	4									
V[iL]	2	4	10									

p	--	--	2	3	2	--	5					
iL	6	5	7									
V[iL]	2	4	9									

p	--	--	2	3	2	--	5	--				
iL	8	5	7									
V[iL]	0	4	9									

Longest increasing subsequence (LIS)

k	1	2	3	4	5	6	7	8	9	10	11	12

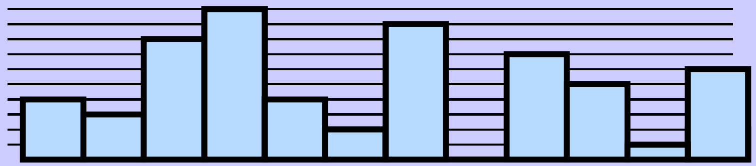


V	4	3	8	10	4	2	9	0	7	5	1	6

p	--	--	2	3	2	--	5	--				
iL	8	5	7									
V[iL]	0	4	9									

p	--	--	2	3	2	--	5	--	5			
VL	8	5	9									
V[iL]	0	4	7									

k	1	2	3	4	5	6	7	8	9	10	11	12



V	4	3	8	10	4	2	9	0	7	5	1	6

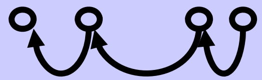
p	--	--	2	3	2	--	5	--	5	5		
iL	8	5	10									
V[iL]	0	4	5									

p	--	--	2	3	2	--	5	--	5	5	8	
iL	8	11	10									
V[iL]	0	1	5									

p	--	--	2	3	2	--	5	--	5	5	8	10
iL	8	11	10	12								
V[iL]	0	1	5	6								

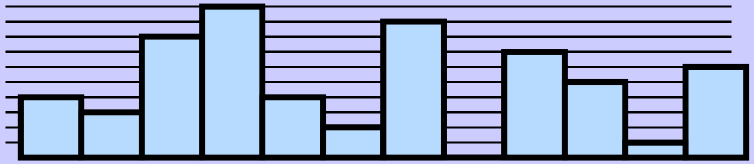
Optimal path reconstruction

The last defined element in iL is the index of the last element of one of the optimal subsequences of the whole sequence. The references in array p represent this subsequence.



Longest increasing subsequence (LIS)

Asymptotic complexity



k	1	2	3	4	5	6	7	8	9	10	11	12
V	4	3	8	10	4	2	9	0	7	5	1	6
p	--	--	2	3	2	--	5	--	5	5	8	10
iL	8	11	10	12								
V[iL]	0	1	5	6								

For each index k:
Let d be the index of the biggest element, which satisfies V[iL[d]] < V[k].

The values V[iL[d]], d = 1,2, ... form a non-decreasing sequence.

In each step k the value V[k] is fixed.
The biggest element V[iL[d]] which satisfies V[iL[d]] < V[k] can be found in time O(log N) by binary search.
There are N steps, the resulting asymptotic complexity is O(N · log N), it can be shown to be exactly Θ(N · log N).

Matrix chain multiplication

Example instance of the problem

Compute in most effective way the matrix product

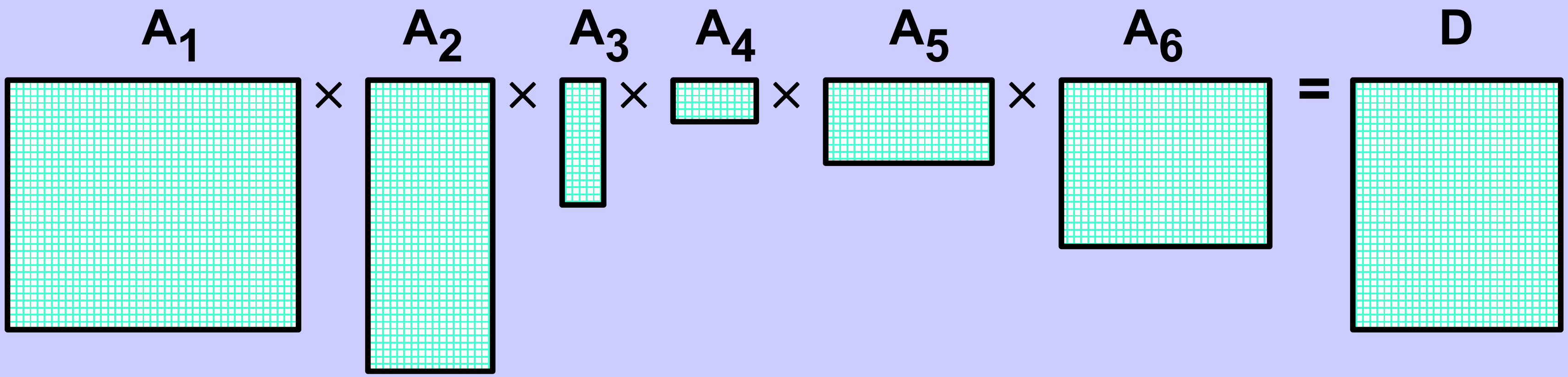
$A_1 \times A_2 \times A_3 \times A_4 \times A_5 \times A_6$,

where the dimensions of the matrices are (in the given order)

30×35, 35×15, 15×5, 5×10, 10×20, 20×25.

(The dimesion of the resulting matrix D is 30×20).

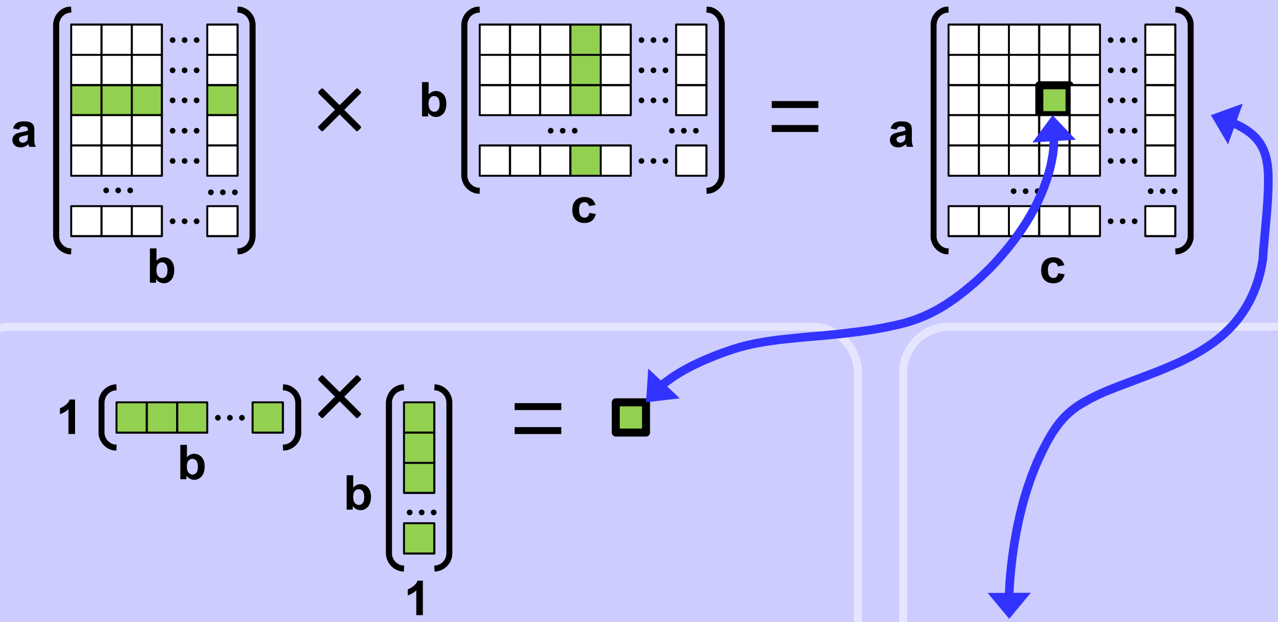
Matrices dimensions depicted to scale



Example follows [CLRS], chapter 15.

Matrix chain multiplication

Number of multiplications in two matrices product



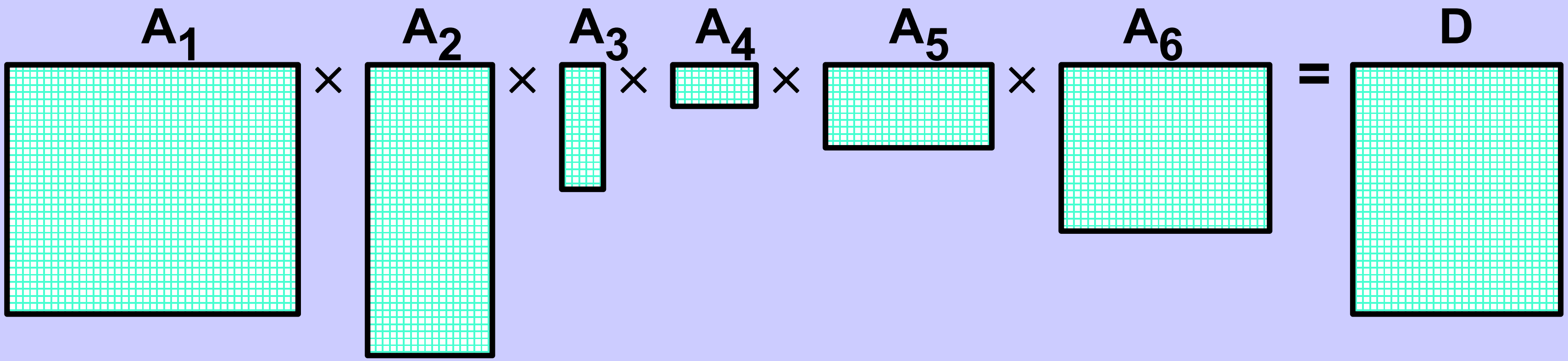
b multiplications yield one element of the result matrix.

a * c elements in the result matrix

Calculating product of two matrices of sizes a×b and b×c require a * b * c multiplications of numbers (floats, doubles, etc.).

We do not consider summation here, it can be analysed analogously.

Matrix chain multiplication

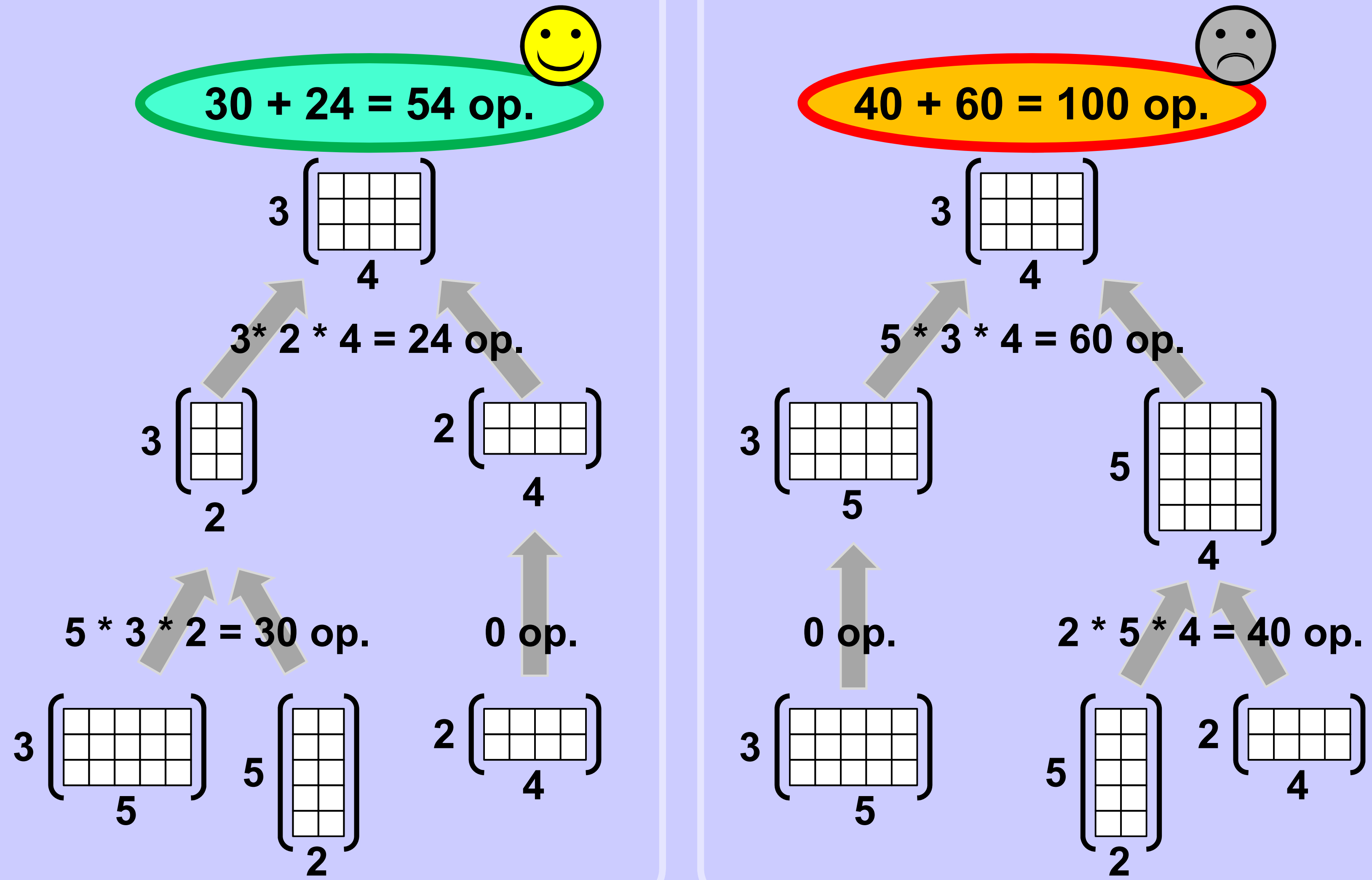


We consider different parenthesizations and thus different orders of calculations induced by those parenthesizations.

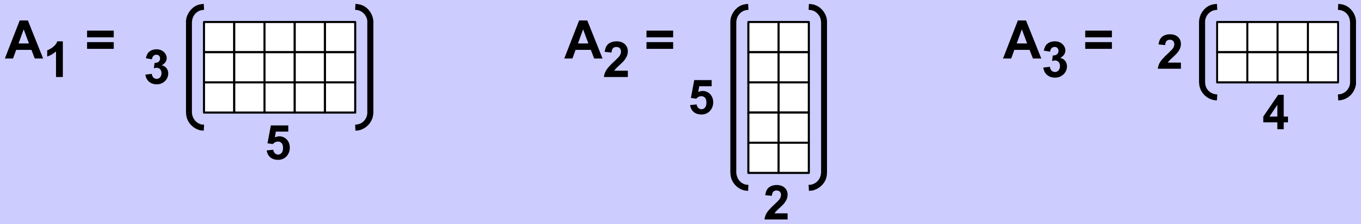
Evaluation order	Corresponding expression	# operations
left to right	$((((A_1 \times A_2) \times A_3) \times A_4) \times A_5) \times A_6$	43 500
right to left	$A_1 \times (A_2 \times (A_3 \times (A_4 \times (A_5 \times A_6))))$	47 500
worst	$A_1 \times ((A_2 \times ((A_3 \times A_4) \times A_5)) \times A_6)$	58 000
best	$(A_1 \times (A_2 \times A_3)) \times ((A_4 \times A_5) \times A_6)$	15 125

Matrix chain multiplication

Example: Comparison of multiplication of 3 matrices



Matrix chain multiplication



Product $(A_1 \times A_2) \times A_3$ requires 54 multiplications .

Product $A_1 \times (A_2 \times A_3)$ requires 100 multiplications.

Obviously, the order of parentheses is important .

Catalan numbers C_N

Product $A_1 \times A_2 \times A_3 \times ... \times A_N$ can be parenthesized in

C_N = Comb(2N, N) / (N+1) different ways.

$C_1, C_2, ..., C_7$ = 1, 1, 2, 5, 14, 42, 132. $C_N > 2^N$ pro N > 7.

In general, checking each way of parenthesization separately leads to the exponencial complexity of the task.

Matrix chain multiplication

Illustration

All 14 different possibilities of product parenthesization of 5 factors

$$A_1 \times (A_2 \times (A_3 \times (A_4 \times A_5)))$$
$$A_1 \times (A_2 \times ((A_3 \times A_4) \times A_5))$$
$$A_1 \times ((A_2 \times A_3) \times (A_4 \times A_5))$$
$$A_1 \times ((A_2 \times (A_3 \times A_4)) \times A_5)$$
$$A_1 \times (((A_2 \times A_3) \times A_4) \times A_5)$$
$$(A_1 \times A_2) \times (A_3 \times (A_4 \times A_5))$$
$$(A_1 \times A_2) \times ((A_3 \times A_4) \times A_5)$$
$$(A_1 \times (A_2 \times A_3)) \times (A_4 \times A_5)$$
$$((A_1 \times A_2) \times A_3) \times (A_4 \times A_5)$$
$$(A_1 \times (A_2 \times (A_3 \times A_4))) \times A_5$$
$$(A_1 \times ((A_2 \times A_3) \times A_4)) \times A_5$$
$$((A_1 \times A_2) \times (A_3 \times A_4)) \times A_5$$
$$((A_1 \times (A_2 \times A_3)) \times A_4) \times A_5$$
$$(((A_1 \times A_2) \times A_3) \times A_4) \times A_5$$

Matrix chain multiplication

$A_1 \times (A_2 \times A_3 \times A_4 \ \dots \times A_{N-1} \times A_N)$

$(A_1 \times A_2) \times (A_3 \times A_4 \ \dots \times A_{N-1} \times A_N)$

$(A_1 \times A_2 \times A_3) \times (A_4 \ \dots \times A_{N-1} \times A_N)$

$(A_1 \times A_2 \times A_3 \times A_4) \times (\dots \times A_{N-1} \times A_N)$

$\dots$

$(A_1 \times A_2 \times A_3 \times A_4 \times \dots) \times (A_{N-1} \times A_N)$

$(A_1 \times A_2 \times A_3 \times A_4 \times \dots \times A_{N-1}) \times A_N$

N – 1 possible places where the expression is divided in two subexpressions, those are processed separately and finally multiplied together.

Let us suppose (as is usual in DP) that the optimum parenthesization is precomputed for all blue subexpressions.

Matrix chain multiplication

$$A_1 \times (A_2 \times A_3 \times A_4 \ \dots \ \times A_{N-1} \times A_N) = B[1,1] \times B[2,N]$$

$$(A_1 \times A_2) \times (A_3 \times A_4 \ \dots \ \times A_{N-1} \times A_N) = B[1,2] \times B[3,N]$$

$$(A_1 \times A_2 \times A_3) \times (A_4 \ \dots \ \times A_{N-1} \times A_N) = B[1,3] \times B[4,N]$$

$$(A_1 \times A_2 \times A_3 \times A_4) \times (\dots \times A_{N-1} \times A_N) = B[1,4] \times B[5,N]$$

$$\dots \qquad\qquad \dots$$

$$(A_1 \times A_2 \times A_3 \times A_4 \times \dots) \times (A_{N-1} \times A_N) = B[1,N-2] \times B[N-1,N]$$

$$(A_1 \times A_2 \times A_3 \times A_4 \times \dots \times A_{N-1}) \times A_N = B[1,N-1] \times B[N,N]$$

Matrix B[i, j] is the product of the corresponding subexpression.

Denote by r(X) resp. s(X) the number of rows resp. columns of matrix X. The matrix multiplication rules say:

$$r(B[i, j]) = r(A_i),\ s(B[i, j]) = s(A_j), \quad 1 \le i \le j \le N.$$

Matrix chain multiplication

Let MO[i, j] be minimum number of multiplications needed to compute B[i, j], i.e. minimum number of multiplications needed to compute the matrix $A_i \times A_{i+1} \times ... \times A_{j-1} \times A_j$.

B[1,1]	$\times$	B[2,N]	$MO[1,1] + r(A_1)*s(A_1)*s(A_N) + MO[2, N]$
B[1,2]	$\times$	B[3,N]	$MO[1,2] + r(A_1)*s(A_2)*s(A_N) + MO[3, N]$
B[1,3]	$\times$	B[4,N]	$MO[1,3] + r(A_1)*s(A_3)*s(A_N) + MO[4, N]$
. . .			
B[1,N-2]	$\times$	B[N-1,N]	$MO[1,N-2] + r(A_1)*s(A_{N-2})*s(A_N) + MO[N-1, N]$
B[1,N-1]	$\times$	B[N,N]	$\underbrace{MO[1,N-1]}_{\text{\# of multiplications in the left subexpression}} + \underbrace{r(A_1)*s(A_{N-1})*s(A_N)}_{\text{\# of multiplications in } B[1,.] \times B[.,N]} + \underbrace{MO[N, N]}_{\text{\# of multiplications in the right subexpression}}$

For MO[1,N], which is the solution of the whole problem, we get
$MO[1,N] = \min \{MO[1,k] + r(A_1)*s(A_k)*s(A_N) + MO[k+1, N] \mid k = 1..N-1\}$

Matrix chain multiplication

$MO[1,N] = \min \{MO[1,k] + r(A_1)*s(A_k)*s(A_N) + MO[k+1, N] \mid k = 1..N-1\}$

When values MO[i, j] for subexpressions shorter than [1, N] is known then the problem solution (= value MO[1, N]) , can be found in time $\Theta(N)$. (*)

Recursive and repeated exploitation of smaller subproblems solutions

The analysis which we performed with the whole expression
$A_1 \times A_2 \times A_3 \times \ldots \times A_N$,
can be analogously performed for each contiguous subexpression
$\ldots A_L \times A_{L+1} \times \ldots \times A_{R-1} \times A_R \ldots$, $1 \le L \le R \le N$.

The number of these subexpressions is the same as the number of index pairs (L, R), $1 \le L \le R \le N$. it is equal to Comb(N, 2) $\in \Theta(N^2)$.
A particular subproblem specified by (L, R) can be solved according to (*) in time O(N),
the whole solution time is therefore $O(N*N^2) = O(N^3)$.

Matrix chain multiplication

$$MO[L,R] \overset{*}{=} \min \{MO[L,k] + r(A_L)*s(A_k)*s(A_R) + MO[k+1,R] \mid k = L..R-1\}$$

Values MO[L,R] can be stored in 2D array at position [L][R].

Calcultion of MO[L,R] according to * depends on values MO[x,y] in which the difference y - x is less then the difference R - L.

The DP table is thus filled in the order of increasing difference R - L.

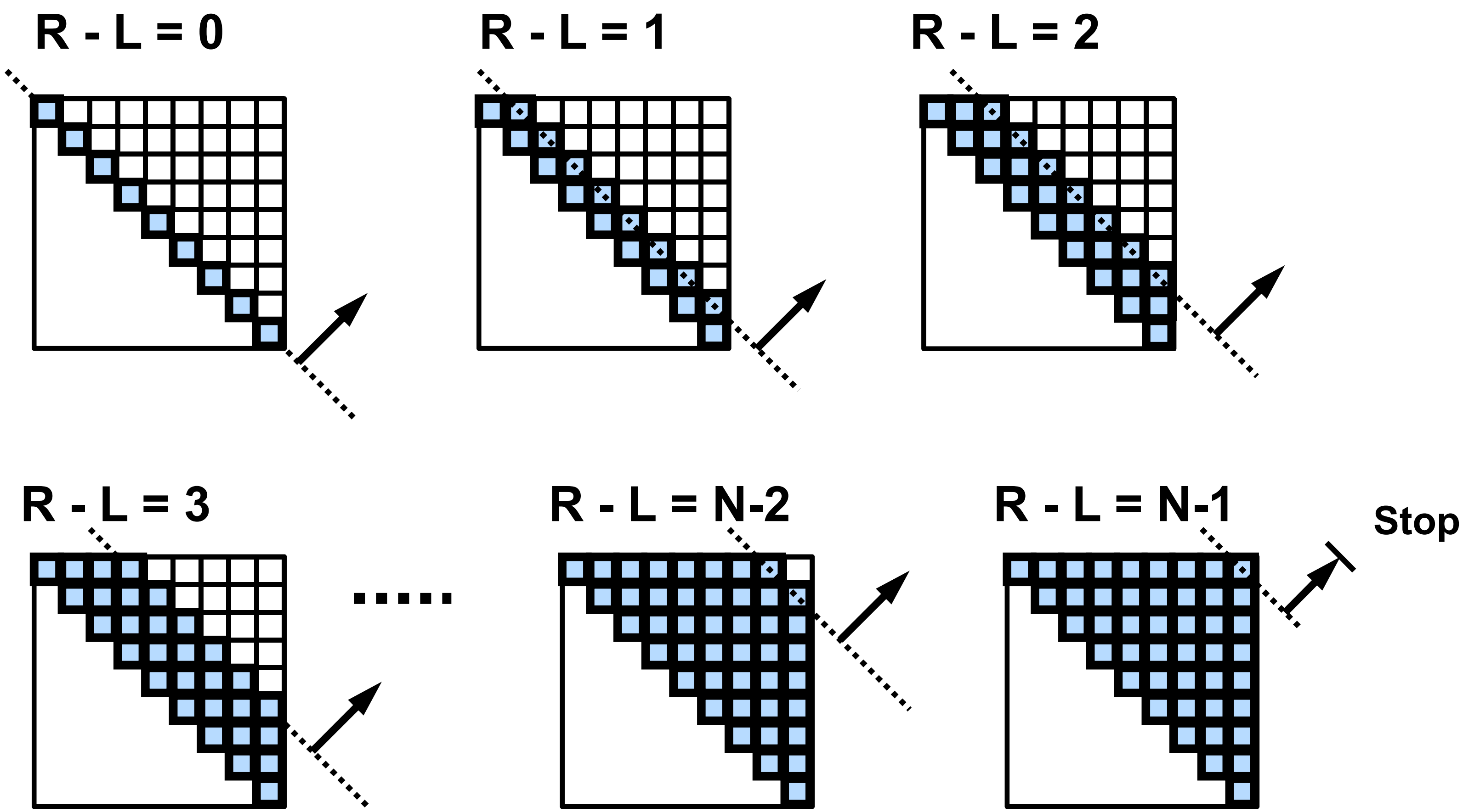
0. Calculate MO[L][R], where R-L = 0, it is the main diagonal.
1. Calculate MO[L][R], where R-L = 1, it is the diagonal just above the main diagonal.
2. Calculate MO[L][R], where R-L = 1, it is the diagonal just above the previous diagonal.

...

N-1. Calculate MO[L][R], where R-L = N-1, it is the upper right corner of the table.

Matrix chain multiplication

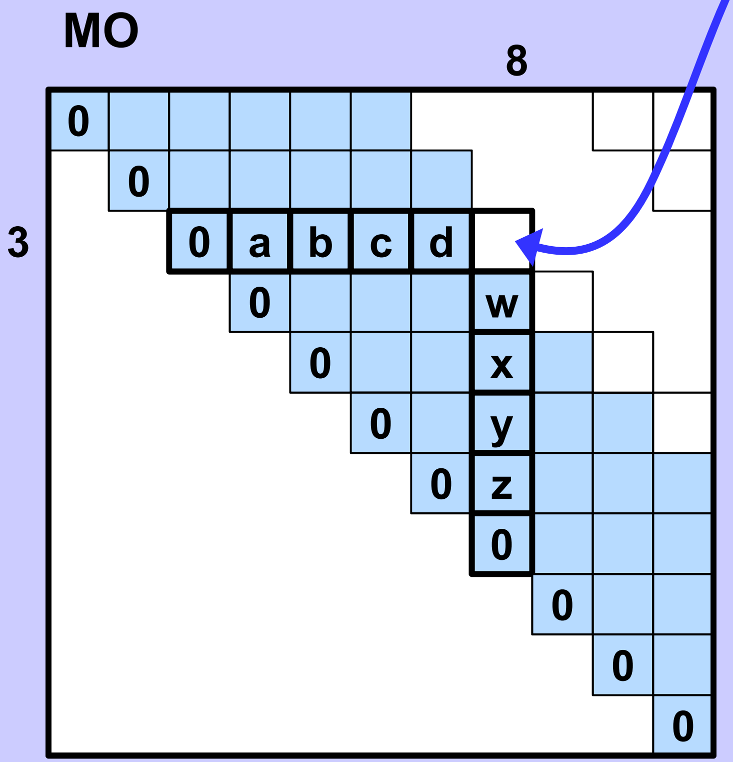
Calculation of the DP table -- progress scheme



Matrix chain multiplication

$$MO[L,R] = \min \{MO[L,k] + r(A_L)*s(A_k)*s(A_R) + MO[k+1,R] \mid k = L..R-1\}$$

Example of one cell calculation



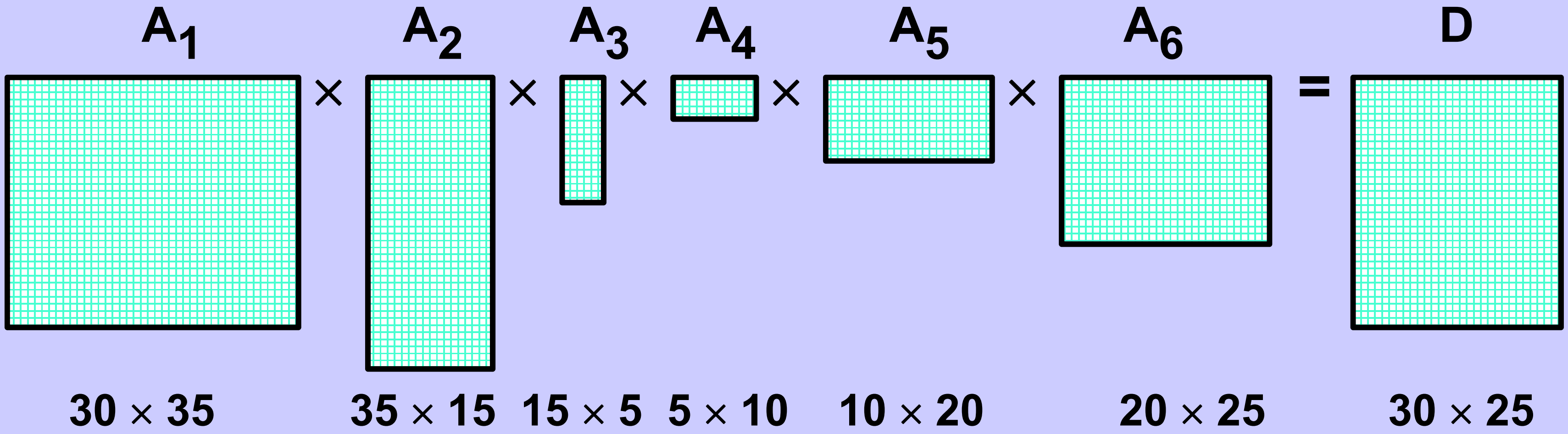
$MO[3,8] = \min \{$
$MO[3,3] + r(A_3)*s(A_3)*s(A_8) + MO[4,8],$
$MO[3,4] + r(A_3)*s(A_4)*s(A_8) + MO[5,8],$
$MO[3,5] + r(A_3)*s(A_5)*s(A_8) + MO[6,8],$
$MO[3,6] + r(A_3)*s(A_6)*s(A_8) + MO[7,8],$
$MO[3,7] + r(A_3)*s(A_7)*s(A_8) + MO[8,8]\}$

Denote $P[L, R] := r(A_L)*s(A_R)$. Then

$MO[3,8] = \min \{$
$0 + s(A_3)*P[3,8] + w,$
$a + s(A_4)*P[3,8] + x,$
$b + s(A_5)*P[3,8] + y,$
$c + s(A_6)*P[3,8] + z,$
$d + s(A_7)*P[3,8] + 0\}.$

Matrix chain multiplication

Matrices



MO	1	2	3	4	5	6
1	0	15750	7875	9375	11875	$\boxed{15125}$
2	0	0	2625	4375	7125	10500
3	0	0	0	750	2500	5375
4	0	0	0	0	1000	3500
5	0	0	0	0	0	5000
6	0	0	0	0	0	0

Optimum

Matrix chain multiplication

*

$MO[L,R] = \min \{MO[L,k] + r(A_L)*s(A_k)*s(A_R) + MO[k+1,R] \mid k = L..R-1\}$

When the value of MO[L,R] is established
we store in the 2D reconstruction table RT at the position [L][R]
the walue of k in which the minimum in * was attained.

The value k = RT[L][R] defines the division of the subexpression

$(A_L \times A_{L+1} \times \ldots \times A_R)$

into two smaller optimal subexpressions

$(A_L \times A_{L+1} \times \ldots \times A_k) \times (A_{k+1} \times A_{k+2} \times \ldots \times A_R)$.

The value RT[1, N] defines the division of the whole expression

$A_1 \times A_2 \times \ldots \times A_N$

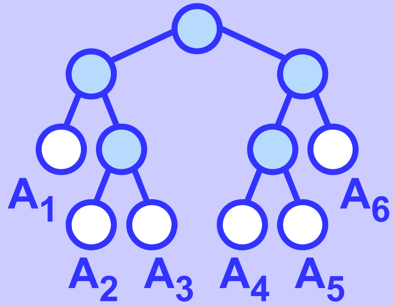
into the first two optimal subexpressions

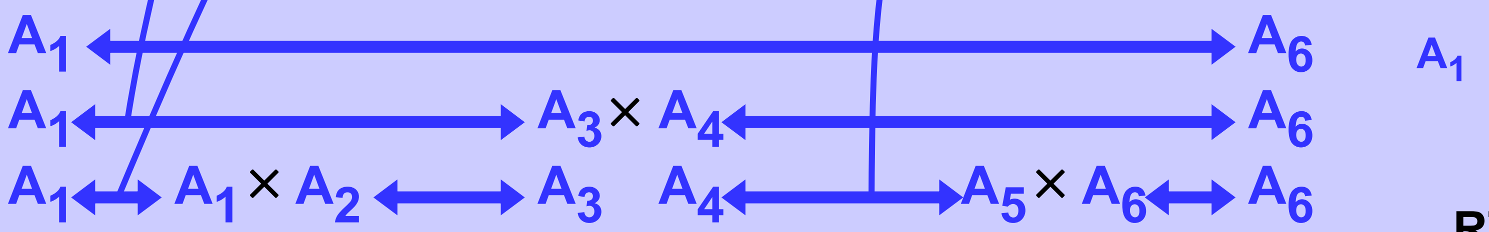
$(A_1 \times A_2 \times \ldots \times A_k) \times (A_{k+1} \times A_{k+2} \times \ldots \times A_N)$.

And then the reconstruction continues recursively analogously
for $(A_1 \times A_2 \times \ldots \times A_k)$ and for $(A_{k+1} \times A_{k+2} \times \ldots \times A_N)$ and so on.

Matrix chain multiplication

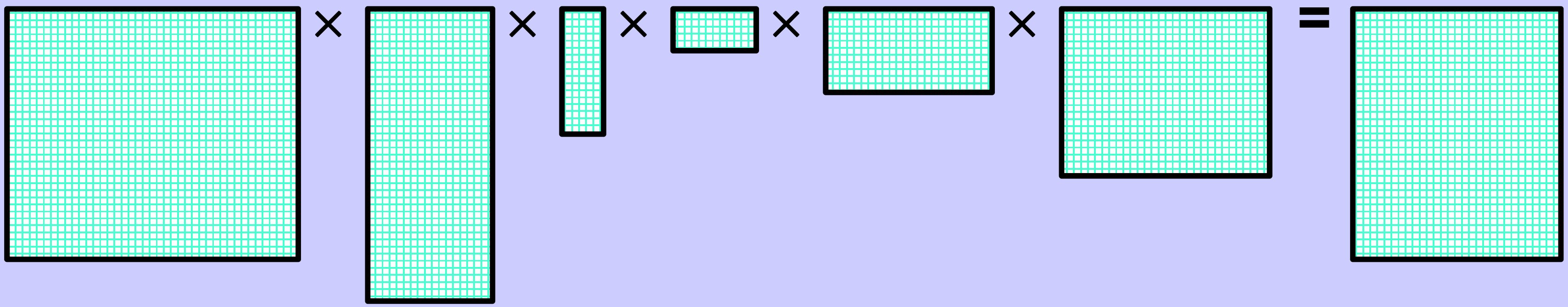
	1	2	3	4	5	6
1	0	1	$\boxed{1}$	3	3	$\boxed{3}$
2	0	0	2	3	3	3
3	0	0	0	3	3	3
4	0	0	0	0	4	$\boxed{5}$
5	0	0	0	0	0	5
6	0	0	0	0	0	0





$(A_1 \times (A_2 \times A_3)) \times ((A_4 \times A_5) \times A_6)$

RT
D

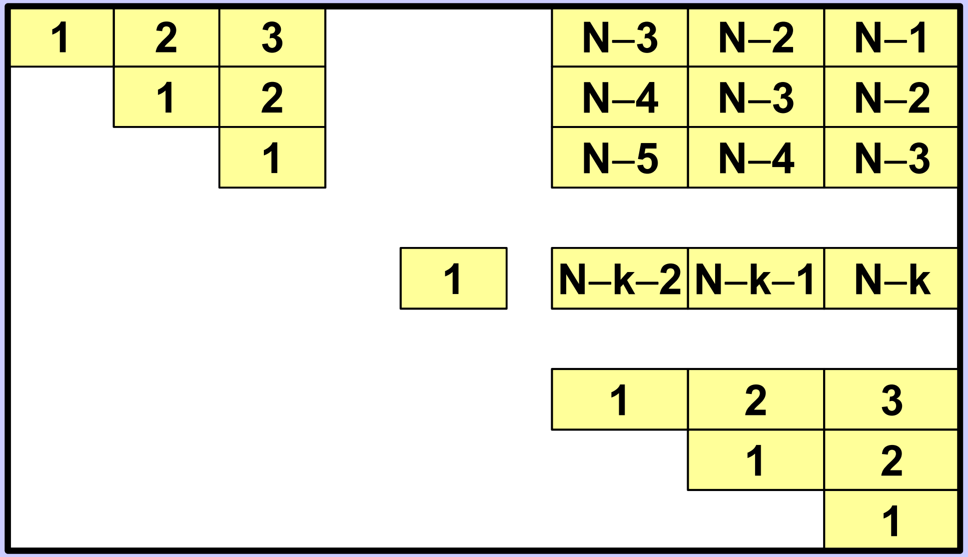


Matrix chain multiplication

Asymptotic complexity

The complexity of calculating one cell value is proportional to the number of other cells in the table used to perform this calculation.

Row index	Row sums
k = N–1	1/2 * (N–1) * N
k = N–2	1/2 * (N–2) * (N–1)
k = N–3	1/2 * (N–3) * (N–2)
k = k	1/2 * k * (k+1)
k = 3	1/2 * 3 * 4
k = 2	1/2 * 2 * 3
k = 1	1/2 * 1 * 2



Total

$$\frac{1}{2} * \sum_{k=1}^{N-1} k * (k+1) = \frac{1}{2} * \sum_{k=1}^{N-1} k^2 + \frac{1}{2} * \sum_{k=1}^{N-1} k$$

$$= 1/2 * (N-1) * N * (2N-1)/6 + 1/2 * (N-1) * N/2 \in \Theta(N^3)$$