CSE 431/531: Algorithm Analysis and Design (Fall 2024)
Dynamic Programming

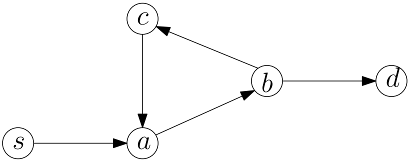
Lecturer: Kelin Luo

Department of Computer Science and Engineering
University at Buffalo

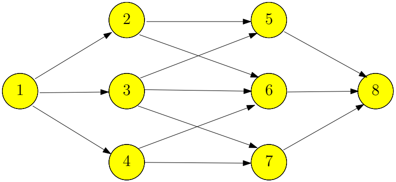
Outline

Directed Acyclic Graphs

Def. A directed acyclic graph (DAG) is a directed graph without (directed) cycles.



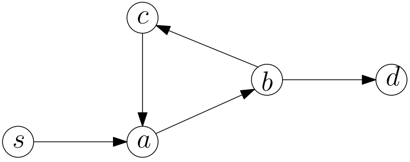
not a DAG



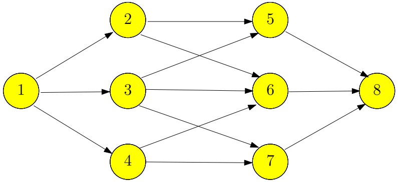
a DAG

Directed Acyclic Graphs

Def. A directed acyclic graph (DAG) is a directed graph without (directed) cycles.



not a DAG



a DAG

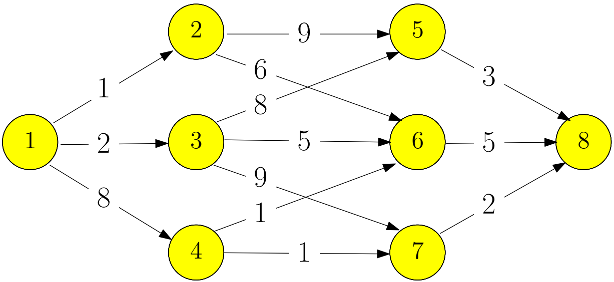
Lemma A directed graph is a DAG if and only its vertices can be topologically sorted.

Shortest Paths in DAG

Input: directed acyclic graph $G = (V, E)$ and $w : E \to \mathbb{R}$.

Assume $V = \{1, 2, 3 \cdots, n\}$ is topologically sorted: if $(i, j) \in E$, then $i < j$

Output: the shortest path from 1 to i, for every $i \in V$

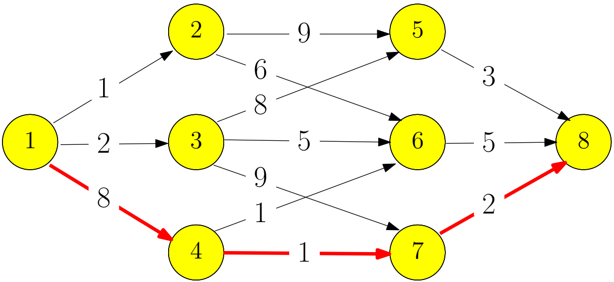


Shortest Paths in DAG

Input: directed acyclic graph $G = (V, E)$ and $w : E \to \mathbb{R}$.

Assume $V = \{1, 2, 3 \cdots, n\}$ is topologically sorted: if $(i, j) \in E$, then $i < j$

Output: the shortest path from 1 to i, for every $i \in V$



Shortest Paths in DAG

- $f[i]$: length of the shortest path from 1 to i

$$f[i] = \begin{cases} & i = 1 \\ & i = 2, 3, \cdots, n \end{cases}$$

Shortest Paths in DAG

- $f[i]$: length of the shortest path from 1 to i

$$f[i] = \begin{cases} 0 & i = 1 \\ & i = 2, 3, \cdots, n \end{cases}$$

Shortest Paths in DAG

- $f[i]$: length of the shortest path from 1 to i

$$f[i] = \begin{cases} 0 & i = 1 \\ \min_{j:(j,i)\in E} \{f(j) + w(j,i)\} & i = 2, 3, \cdots, n \end{cases}$$

Shortest Paths in DAG

- Use an adjacency list for incoming edges of each vertex i

Shortest Paths in DAG

$f[1] \leftarrow 0$
for $i \leftarrow 2$ to n **do**
    $f[i] \leftarrow \infty$
    for each incoming edge (j, i) of i **do**
        if $f[j] + w(j, i) < f[i]$ **then**
            $f[i] \leftarrow f[j] + w(j, i)$

Shortest Paths in DAG

- Use an adjacency list for incoming edges of each vertex i

Shortest Paths in DAG

```
f[1] ← 0
for i ← 2 to n do
    f[i] ← ∞
    for each incoming edge (j, i) of i do
        if f[j] + w(j, i) < f[i] then
            f[i] ← f[j] + w(j, i)
            π(i) ← j
```

Shortest Paths in DAG

- Use an adjacency list for incoming edges of each vertex i

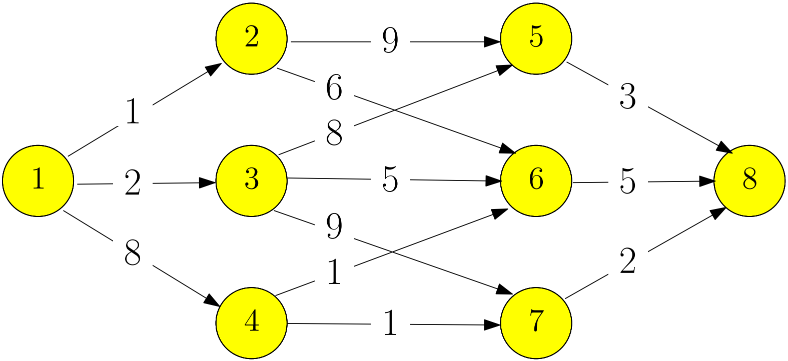
Shortest Paths in DAG

$f[1] \leftarrow 0$
for $i \leftarrow 2$ to n **do**
    $f[i] \leftarrow \infty$
    for each incoming edge (j, i) of i **do**
        if $f[j] + w(j, i) < f[i]$ **then**
            $f[i] \leftarrow f[j] + w(j, i)$
            $\pi(i) \leftarrow j$

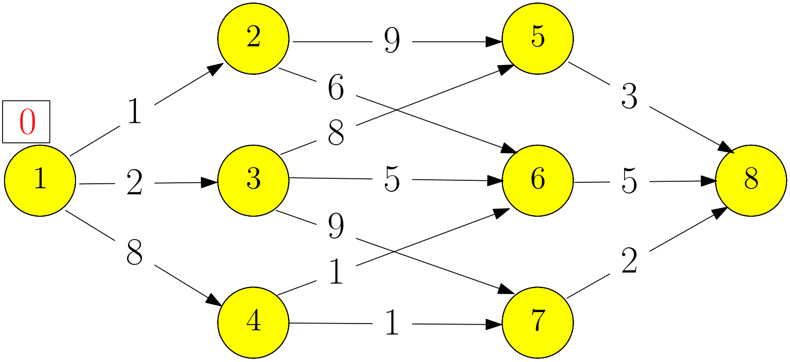
print-path(t)

if $t = 1$ **then**
    print(1)
    return
print-path($\pi(t)$)
print(“,”, t)

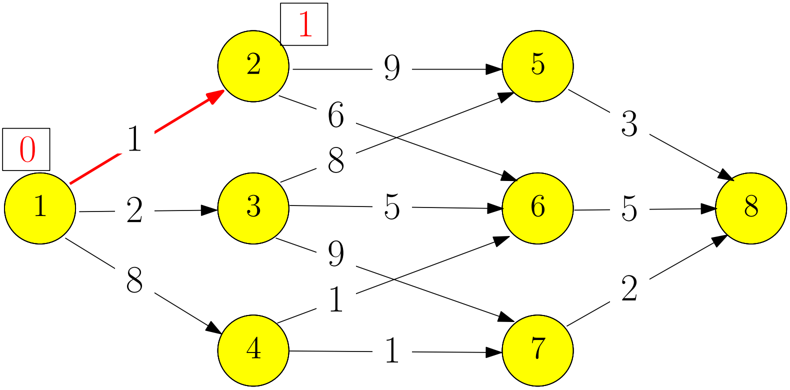
Example



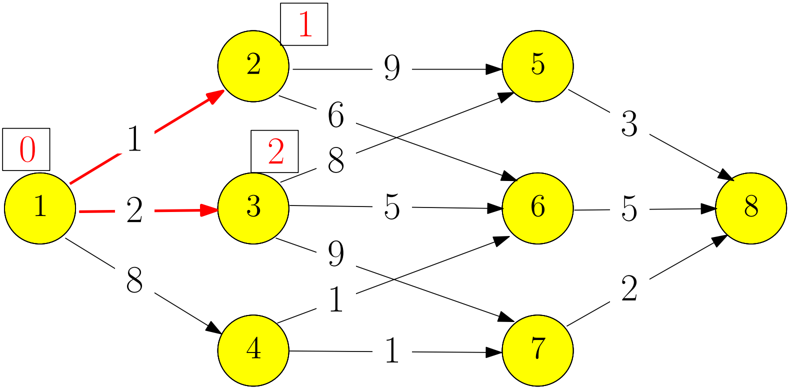
Example



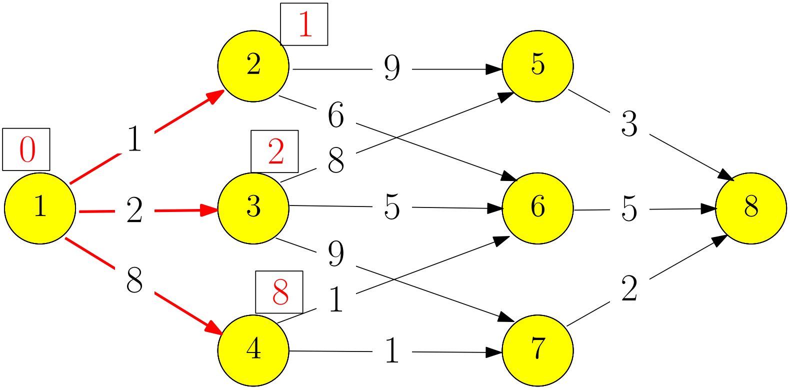
Example



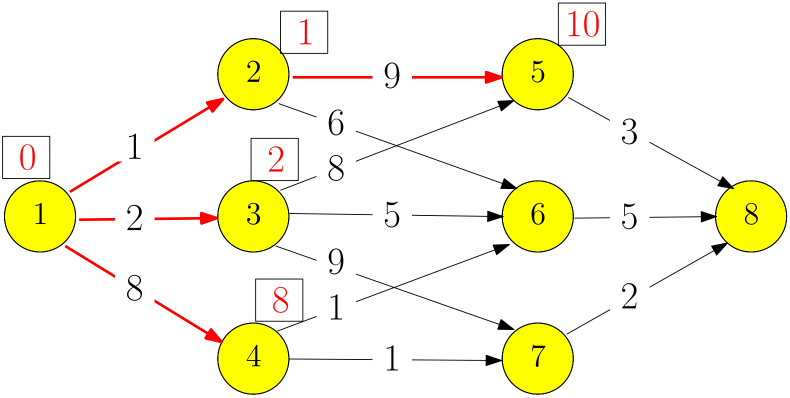
Example



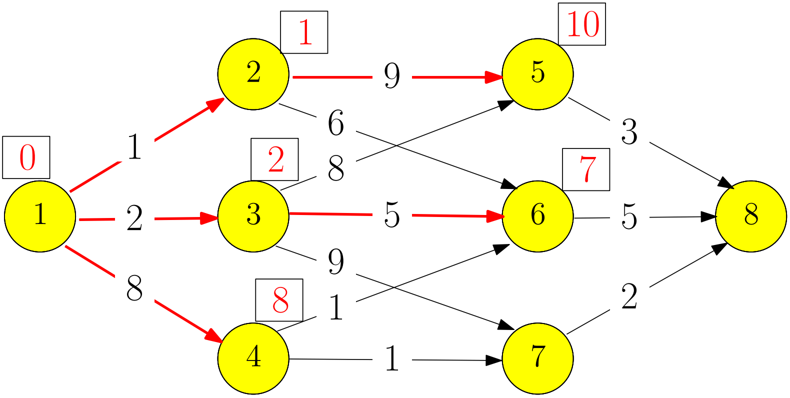
Example



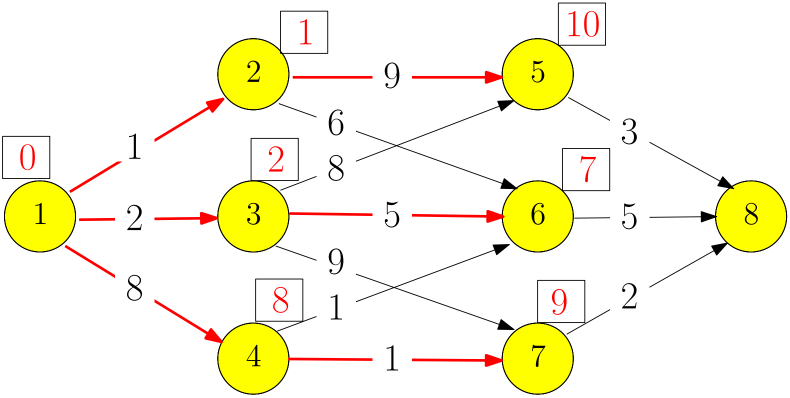
Example



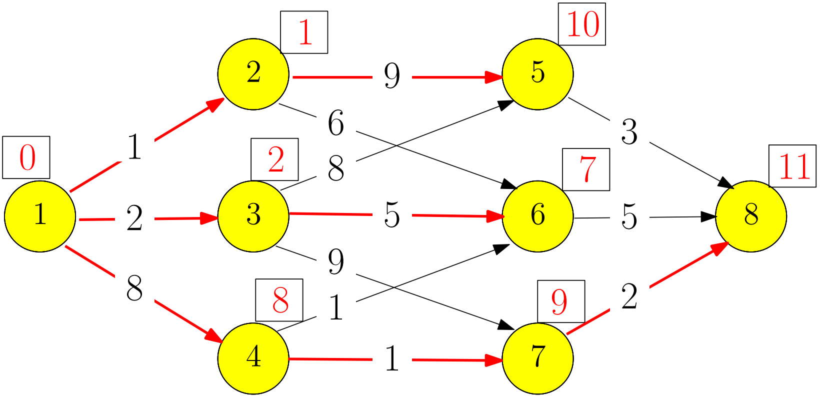
Example



Example



Example



Variant: Heavest Path in a Directed Acyclic Graph

Heaviest Path in a Directed Acyclic Graph

Input: directed acyclic graph $G = (V, E)$ and $w : E \to \mathbb{R}$.
Assume $V = \{1, 2, 3 \cdots, n\}$ is topologically sorted: if $(i, j) \in E$, then $i < j$

Output: the path with the largest weight (the heaviest path) from 1 to n.

- $f[i]$: weight of the heaviest path from 1 to i

$$f[i] = \begin{cases} & i = 1 \\ & i = 2, 3, \cdots, n \end{cases}$$

Variant: Heavest Path in a Directed Acyclic Graph

Heaviest Path in a Directed Acyclic Graph

Input: directed acyclic graph $G = (V, E)$ and $w : E \to \mathbb{R}$.
Assume $V = \{1, 2, 3 \cdots, n\}$ is topologically sorted: if $(i, j) \in E$, then $i < j$

Output: the path with the largest weight (the heaviest path) from 1 to n.

- $f[i]$: weight of the heaviest path from 1 to i

$$f[i] = \begin{cases} 0 & i = 1 \\ & i = 2, 3, \cdots, n \end{cases}$$

Variant: Heaviest Path in a Directed Acyclic Graph

Heaviest Path in a Directed Acyclic Graph

Input: directed acyclic graph $G = (V, E)$ and $w : E \to \mathbb{R}$.
Assume $V = \{1, 2, 3 \cdots, n\}$ is topologically sorted: if $(i, j) \in E$, then $i < j$

Output: the path with the largest weight (the heaviest path) from 1 to n.

- $f[i]$: weight of the heaviest path from 1 to i

$$f[i] = \begin{cases} 0 & i = 1 \\ \max_{j:(j,i)\in E} \{f(j) + w(j,i)\} & i = 2, 3, \cdots, n \end{cases}$$

Outline

Matrix Chain Multiplication

Matrix Chain Multiplication

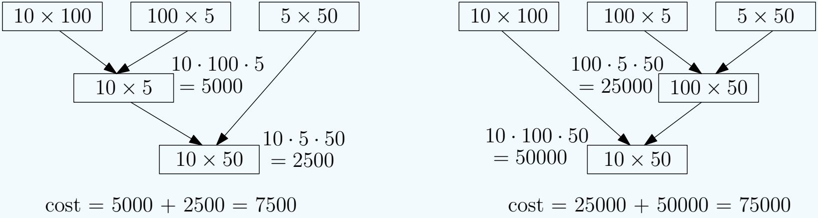
Input: n matrices $A_1, A_2, \cdots, A_n$ of sizes $r_1 \times c_1, r_2 \times c_2, \cdots, r_n \times c_n$, such that $c_i = r_{i+1}$ for every $i = 1, 2, \cdots, n-1$.

Output: the order of computing $A_1 A_2 \cdots A_n$ with the minimum number of multiplications

Fact Multiplying two matrices of size $r \times k$ and $k \times c$ takes $r \times k \times c$ multiplications.

Example:

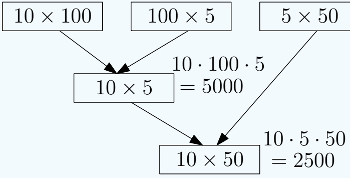
- $A_1 : 10 \times 100, \quad A_2 : 100 \times 5, \quad A_3 : 5 \times 50$



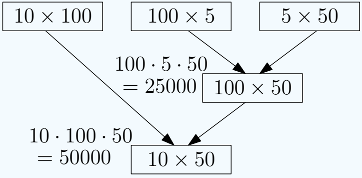
- $(A_1A_2)A_3$: $10 \times 100 \times 5 + 10 \times 5 \times 50 = 7500$
- $A_1(A_2A_3)$: $100 \times 5 \times 50 + 10 \times 100 \times 50 = 75000$

Example:

- $A_1 : 10 \times 100, \quad A_2 : 100 \times 5, \quad A_3 : 5 \times 50$



cost = 5000 + 2500 = 7500



cost = 25000 + 50000 = 75000

- $(A_1 A_2) A_3$: $10 \times 100 \times 5 + 10 \times 5 \times 50 = 7500$
- $A_1 (A_2 A_3)$: $100 \times 5 \times 50 + 10 \times 100 \times 50 = 75000$

Matrix Chain Multiplication: Design DP

Matrix Chain Multiplication: Design DP

- Assume the last step is $(A_1 A_2 \cdots A_i)(A_{i+1} A_{i+2} \cdots A_n)$

Matrix Chain Multiplication: Design DP

- Assume the last step is $(A_1 A_2 \cdots A_i)(A_{i+1} A_{i+2} \cdots A_n)$
- Cost of last step: $r_1 \times c_i \times c_n$

Matrix Chain Multiplication: Design DP

- Assume the last step is $(A_1 A_2 \cdots A_i)(A_{i+1} A_{i+2} \cdots A_n)$
- Cost of last step: $r_1 \times c_i \times c_n$
- Optimality for sub-instances: we need to compute $A_1 A_2 \cdots A_i$ and $A_{i+1} A_{i+2} \cdots A_n$ optimally

Matrix Chain Multiplication: Design DP

- Assume the last step is $(A_1 A_2 \cdots A_i)(A_{i+1} A_{i+2} \cdots A_n)$
- Cost of last step: $r_1 \times c_i \times c_n$
- Optimality for sub-instances: we need to compute $A_1 A_2 \cdots A_i$ and $A_{i+1} A_{i+2} \cdots A_n$ optimally
- $opt[i, j]$: the minimum cost of computing $A_i A_{i+1} \cdots A_j$

Matrix Chain Multiplication: Design DP

- Assume the last step is $(A_1A_2\cdots A_i)(A_{i+1}A_{i+2}\cdots A_n)$
- Cost of last step: $r_1 \times c_i \times c_n$
- Optimality for sub-instances: we need to compute $A_1A_2\cdots A_i$ and $A_{i+1}A_{i+2}\cdots A_n$ optimally
- $opt[i,j]$: the minimum cost of computing $A_iA_{i+1}\cdots A_j$

$$opt[i,j] = \begin{cases} 0 & i = j \\ \min_{k: i\le k<j} \left(opt[i,k] + opt[k+1,j] + r_ic_kc_j\right) & i < j \end{cases}$$

Matrix Chain Multiplication: Design DP

matrix-chain-multiplication$(n, r[1..n], c[1..n])$

let $opt[i,i] \leftarrow 0$ for every $i = 1, 2, \cdots, n$
for $\ell \leftarrow 2$ to n **do**
  for $i \leftarrow 1$ to $n - \ell + 1$ **do**
    $j \leftarrow i + \ell - 1$
    $opt[i,j] \leftarrow \infty$
    for $k \leftarrow i$ to $j - 1$ **do**
      if $opt[i,k] + opt[k+1,j] + r_i c_k c_j < opt[i,j]$ **then**
        $opt[i,j] \leftarrow opt[i,k] + opt[k+1,j] + r_i c_k c_j$
return $opt[1,n]$

Recover the Optimum Way of Multiplication

matrix-chain-multiplication$(n, r[1..n], c[1..n])$

let $opt[i,i] \leftarrow 0$ for every $i = 1, 2, \cdots, n$
for $\ell \leftarrow 2$ to n **do**
  for $i \leftarrow 1$ to $n - \ell + 1$ **do**
    $j \leftarrow i + \ell - 1$
    $opt[i,j] \leftarrow \infty$
    for $k \leftarrow i$ to $j - 1$ **do**
      if $opt[i,k] + opt[k+1,j] + r_i c_k c_j < opt[i,j]$ **then**
        $opt[i,j] \leftarrow opt[i,k] + opt[k+1,j] + r_i c_k c_j$
        $\pi[i,j] \leftarrow k$
return $opt[1,n]$

Constructing Optimal Solution

Print-Optimal-Order(i, j)

if $i = j$ **then**
    print("A"$_i$)
else
    print("(")
    Print-Optimal-Order($i, \pi[i, j]$)
    Print-Optimal-Order($\pi[i, j] + 1, j$)
    print(")")

matrix	A_1	A_2	A_3	A_4	A_5
size	3×5	5×2	2×6	6×9	9×4

$$\begin{aligned}
opt[1,2] &= opt[1,1] + opt[2,2] + 3 \times 5 \times 2 = 30, & \pi[1,2] &= 1 \\
opt[2,3] &= opt[2,2] + opt[3,3] + 5 \times 2 \times 6 = 60, & \pi[2,3] &= 2 \\
opt[3,4] &= opt[3,3] + opt[4,4] + 2 \times 6 \times 9 = 108, & \pi[3,4] &= 3 \\
opt[4,5] &= opt[4,4] + opt[5,5] + 6 \times 9 \times 4 = 216, & \pi[4,5] &= 4 \\
opt[1,3] &= \min\{opt[1,1] + opt[2,3] + 3 \times 5 \times 6, \\
&\qquad opt[1,2] + opt[3,3] + 3 \times 2 \times 6\} \\
&= \min\{0 + 60 + 90, 30 + 0 + 36\} = 66, & \pi[1,3] &= 2 \\
opt[2,4] &= \min\{opt[2,2] + opt[3,4] + 5 \times 2 \times 9, \\
&\qquad opt[2,3] + opt[4,4] + 5 \times 6 \times 9\} \\
&= \min\{0 + 108 + 90, 60 + 0 + 270\} = 198, & \pi[2,4] &= 2,
\end{aligned}$$

matrix	A_1	A_2	A_3	A_4	A_5
size	3×5	5×2	2×6	6×9	9×4

$$
\begin{aligned}
opt[3,5] &= \min\{opt[3,3] + opt[4,5] + 2 \times 6 \times 4, \\
&\qquad\quad opt[3,4] + opt[5,5] + 2 \times 9 \times 4\} \\
&= \min\{0 + 216 + 48, 108 + 0 + 72\} = 180, \\
\pi[3,5] &= 4, \\
opt[1,4] &= \min\{opt[1,1] + opt[2,4] + 3 \times 5 \times 9, \\
&\qquad\quad opt[1,2] + opt[3,4] + 3 \times 2 \times 9, \\
&\qquad\quad opt[1,3] + opt[4,4] + 3 \times 6 \times 9\} \\
&= \min\{0 + 198 + 135, 30 + 108 + 54, 66 + 0 + 162\} = 192, \\
\pi[1,4] &= 2,
\end{aligned}
$$

matrix	A_1	A_2	A_3	A_4	A_5
size	3×5	5×2	2×6	6×9	9×4

$$
\begin{aligned}
opt[2,5] &= \min\{opt[2,2] + opt[3,5] + 5 \times 2 \times 4,\\
&\qquad opt[2,3] + opt[4,5] + 5 \times 6 \times 4,\\
&\qquad opt[2,4] + opt[5,5] + 5 \times 9 \times 4\}\\
&= \min\{0 + 180 + 40, 60 + 216 + 120, 198 + 0 + 180\} = 220,\\
opt[1,5] &= \min\{opt[1,1] + opt[2,5] + 3 \times 5 \times 4,\\
&\qquad opt[1,2] + opt[3,5] + 3 \times 2 \times 4,\\
&\qquad opt[1,3] + opt[4,5] + 3 \times 6 \times 4,\\
&\qquad opt[1,4] + opt[5,5] + 3 \times 9 \times 4\}\\
&= \min\{0 + 220 + 60, 30 + 180 + 24,\\
&\qquad 66 + 216 + 72, 192 + 0 + 108\}\\
&= 234,\\
\pi[1,5] &= 2.
\end{aligned}
$$

matrix	A_1	A_2	A_3	A_4	A_5
size	3×5	5×2	2×6	6×9	9×4

opt, π	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$
$i = 1$	0, /	30, 1	66, 2	192, 2	234, 2
$i = 2$		0, /	60, 2	198, 2	220, 2
$i = 3$			0, /	108, 3	180, 4
$i = 4$				0, /	216, 4
$i = 5$					0, /

opt, π	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$
$i = 1$	0, /	30, 1	66, 2	192, 2	234, 2
$i = 2$		0, /	60, 2	198, 2	220, 2
$i = 3$			0, /	108, 3	180, 4
$i = 4$				0, /	216, 4
$i = 5$					0, /

Print-Optimal-Order(1,5)

  Print-Optimal-Order(1, 2)

    Print-Optimal-Order(1, 1)

    Print-Optimal-Order(2, 2)

  Print-Optimal-Order(3, 5)

    Print-Optimal-Order(3, 4)

      Print-Optimal-Order(3, 3)

      Print-Optimal-Order(4, 4)

    Print-Optimal-Order(5, 5)

Optimum way for multiplication: $((A_1A_2)((A_3A_4)A_5))$