Design and Analysis of Algorithms
Supplemental

Si Wu

School of CSE, SCUT

cswusi@scut.edu.cn

TA: 1684350406@qq.com

Outline

- **LP Duality**
- **Longest Common Substring**
- **All-pairs Shortest Paths**
- **Chain Matrix Multiplication**

LP Duality

Primal problem.

$$\begin{array}{rrcrcr} \text{(P)} \ \max & 13A & + & 23B & & \\ \text{s. t.} & 5A & + & 15B & \leq & 480 \\ & 4A & + & 4B & \leq & 160 \\ & 35A & + & 20B & \leq & 1190 \\ & A & , & B & \geq & 0 \end{array}$$

Goal. Find a lower bound on optimal value.

Easy. Any feasible solution provides one.

Ex 1. $(A, B) = (34, 0) \Rightarrow z^* \geq 442$

Ex 2. $(A, B) = (0, 32) \Rightarrow z^* \geq 736$

Ex 3. $(A, B) = (7.5, 29.5) \Rightarrow z^* \geq 776$

Ex 4. $(A, B) = (12, 28) \Rightarrow z^* \geq 800$

LP Duality

Primal problem.

$$
\begin{aligned}
\text{(P)}\quad \max\ & 13A + 23B \\
\text{s. t.}\quad & 5A + 15B \le 480 \\
& 4A + 4B \le 160 \\
& 35A + 20B \le 1190 \\
& A\ ,\quad B \ge 0
\end{aligned}
$$

Goal. Find an upper bound on optimal value.

Ex 1. Multiply 2nd inequality by 6: $24A + 24B \le 960$.

$$\Rightarrow \quad z^* = \underbrace{13A + 23B}_{\text{objective function}} \le 24A + 24B \le 960.$$

LP Duality

Primal problem.

$$
\begin{aligned}
\text{(P)}\quad \max\ & 13A + 23B \\
\text{s. t.}\quad & 5A + 15B \leq 480 \\
& 4A + 4B \leq 160 \\
& 35A + 20B \leq 1190 \\
& A\ ,\quad B \geq 0
\end{aligned}
$$

Goal. Find an upper bound on optimal value.

Ex 2. Add 2 times 1st inequality to 2nd inequality:

$$\Rightarrow \quad z^* = 13A + 23B \leq 14A + 34B \leq 1120.$$

LP Duality

Primal problem.

$$\begin{array}{rrcrcr} \text{(P)} \ \max & 13A & + & 23B & & \\ \text{s. t.} & 5A & + & 15B & \leq & 480 \\ & 4A & + & 4B & \leq & 160 \\ & 35A & + & 20B & \leq & 1190 \\ & A & , & B & \geq & 0 \end{array}$$

Goal. Find an upper bound on optimal value.

Ex 2. Add 1 times 1st inequality to 2 times 2nd inequality:

$$\Rightarrow \qquad z^* = 13A + 23B \ \leq \ 13A + 23B \ \leq \ 800.$$

Recall lower bound. $(A, B) = (34, 0) \Rightarrow z^* \geq 442$

Combine upper and lower bounds: z* = 800.

LP Duality

Primal problem.

$$
\begin{aligned}
\text{(P)}\quad \max\ & 13A + 23B \\
\text{s. t.}\ & 5A + 15B \le 480 \\
& 4A + 4B \le 160 \\
& 35A + 20B \le 1190 \\
& A\ ,\ B \ge 0
\end{aligned}
$$

Idea. Add nonnegative combination (C, H, M) of the constraints s.t.

$$
\begin{aligned}
13A + 23B &\le (5C + 4H + 35M)\,A + (15C + 4H + 20M)\,B \\
&\le 480C + 160H + 1190M
\end{aligned}
$$

Dual problem. Find best such upper bound.

$$
\begin{aligned}
\text{(D)}\ \min\ & 480C + 160H + 1190M \\
\text{s. t.}\ & 5C + 4H + 35M \ge 13 \\
& 15C + 4H + 20M \ge 23 \\
& C, H, M \ge 0
\end{aligned}
$$

LP Duality: Economic Interpretation

Brewer: find optimal mix of beer and ale to maximize profits.

$$\begin{array}{rrcrcr}
\text{(P)} \ \max & 13A & + & 23B & & \\
\text{s. t.} & 5A & + & 15B & \le & 480 \\
 & 4A & + & 4B & \le & 160 \\
 & 35A & + & 20B & \le & 1190 \\
 & A & , & B & \ge & 0
\end{array}$$

Entrepreneur: buy individual resources from brewer at min cost.

- *C, H, M* = unit price for corn, hops, malt.
- Brewer won't agree to sell resources if $5C + 4H + 35M < 13$.

$$\begin{array}{rrcrcrcr}
\text{(D)} \ \min & 480C & + & 160H & + & 1190M & & \\
\text{s. t.} & 5C & + & 4H & + & 35M & \ge & 13 \\
 & 15C & + & 4H & + & 20M & \ge & 23 \\
 & C & , & H & , & M & \ge & 0
\end{array}$$

LP Duals

Canonical form.

$$\text{(P)} \quad \max c^T x \quad \text{s. t.} \; Ax \le b, \quad x \ge 0$$

$$\text{(D)} \quad \min y^T b \quad \text{s. t.} \; A^T y \ge c, \quad y \ge 0$$

Double Dual

Canonical form.

$$
\begin{array}{llll}
\text{(P)} & \max c^T x & & \\
& \text{s. t.} \ Ax & \leq & b \\
& x & \geq & 0
\end{array}
\qquad
\begin{array}{llll}
\text{(D)} & \min y^T b & & \\
& \text{s. t.} \ A^T y & \geq & c \\
& y & \geq & 0
\end{array}
$$

Property. The dual of the dual is the primal.

Pf. Rewrite (D) as a maximization problem in canonical form; take dual.

$$
\begin{array}{llll}
\text{(D')} & \max & -y^T b & \\
& \text{s. t.} & -A^T y \leq & c \\
& & y \geq & 0
\end{array}
\qquad
\begin{array}{llll}
\text{(DD)} & \min & -c^T z & \\
& \text{s. t.} & -(A^T)^T z \geq & -b \\
& & z \geq & 0
\end{array}
$$

Taking Duals

LP dual recipe.

Primal (P)	maximize	minimize	Dual(D)
constraints	$a\,x = b_i$ $a\,x \le b_i$ $a\,x \ge b_i$	y_i unrestricted $y_i \ge 0$ $y_i \le 0$	variables
variables	$x_j \ge 0$ $x_j \le 0$ unrestricted	$a^{\mathrm{T}}y \ge c_j$ $a^{\mathrm{T}}y \le c_j$ $a^{\mathrm{T}}y = c_j$	constraints

Pf. Rewrite LP in standard form and take dual.

LP Strong Duality

Theorem. [Gale–Kuhn-Tucker 1951, Dantzig–von Neumann 1947]

For $A \in \Re^{m\times n}$, $b \in \Re^{m}$, $c \in \Re^{n}$, if (P) and (D) are nonempty, then max = min.

$$
\begin{array}{llrcl}
\text{(P)} & \max c^T x & & & \\
& \text{s. t.} & Ax & \le & b \\
& & x & \ge & 0
\end{array}
\qquad\qquad
\begin{array}{llrcl}
\text{(D)} & \min y^T b & & & \\
& \text{s. t.} & A^T y & \ge & c \\
& & y & \ge & 0
\end{array}
$$

Generalizes:

- Dilworth’s theorem.
- König–Egervary theorem.
- Max-flow min-cut theorem.
- von Neumann’s minimax theorem.
- …

LP Weak Duality

Theorem. For $A \in \Re^{m\times n}$, $b \in \Re^m$, $c \in \Re^n$, if (P) and (D) are nonempty, then max $\leq$ min.

$$\begin{array}{llrcl} \text{(P)} & \max c^T x & & & \\ & \text{s. t.} & Ax & \leq & b \\ & & x & \geq & 0 \end{array} \qquad \begin{array}{llrcl} \text{(D)} & \min y^T b & & & \\ & \text{s. t.} & A^T y & \geq & c \\ & & y & \geq & 0 \end{array}$$

Pf. Suppose $x \in \Re^m$ is feasible for (P) and $y \in \Re^n$ is feasible for (D).

- $y \geq 0, A x \leq b \;\Rightarrow\; y^T A x \leq y^T b$
- $x \geq 0, A^T y \geq c \;\Rightarrow\; y^T A x \geq c^T x$
- Combine: $c^T x \leq y^T A x \leq y^T b$

Review: Simplex Tableaux

subtract $c_B^T A_B^{-1}$ times constraints

$$
\begin{aligned}
c_B^T x_B + c_N^T x_N &= Z \\
A_B x_B + A_N x_N &= \mathrm{b} \\
x_B \quad , \quad x_N &\geq 0
\end{aligned}
$$

initial tableaux

$$
\begin{aligned}
(c_N^T - c_B^T A_B^{-1} A_N) x_N &= Z - c_B^T A_B^{-1} \mathrm{b} \\
I x_B + A_B^{-1} A_N x_N &= A_B^{-1} \mathrm{b} \\
x_B \quad , \quad x_N &\geq 0
\end{aligned}
$$

tableaux corresponding to basis B

multiply by A_B^{-1}

Primal solution. $x_B = A_B^{-1}\mathrm{b} \geq 0, x_N = 0$

Optimal basis. $c_N^T - c_B^T A_B^{-1} A_N \leq 0$

Simplex Tableaux: Dual Solution

subtract $c_B^T A_B^{-1}$ times constraints

$$
\begin{aligned}
c_B^T x_B + c_N^T x_N &= Z \\
A_B x_B + A_N x_N &= \mathrm{b} \\
x_B \;,\quad x_N &\geq 0
\end{aligned}
\qquad\qquad
\begin{aligned}
(c_N^T - c_B^T A_B^{-1} A_N) x_N &= Z - c_B^T A_B^{-1}\mathrm{b} \\
I x_B + A_B^{-1} A_N x_N &= A_B^{-1}\mathrm{b} \\
x_B \;,\quad x_N &\geq 0
\end{aligned}
$$

initial tableaux

tableaux corresponding to basis B

multiply by A_B^{-1}

Primal solution. $x_B = A_B^{-1}\mathrm{b} \geq 0, x_N = 0$

Optimal basis. $c_N^T - c_B^T A_B^{-1} A_N \leq 0$

Dual solution. $\boxed{y^T = c_B^T A_B^{-1}}$

$$
\begin{aligned}
y^T b &= c_B^T A_B^{-1} b \\
&= c_B^T x_B + c_N^T x_N \\
&= c^T x
\end{aligned}
$$

min ≤ max

$$
\begin{aligned}
y^T A &= \begin{bmatrix} y^T A_B & y^T A_N \end{bmatrix} \\
&= \begin{bmatrix} c_B^T A_B^{-1} A_B & c_B^T A_B^{-1} A_N \end{bmatrix} \\
&= \begin{bmatrix} c_B^T & c_B^T A_B^{-1} A_N \end{bmatrix} \\
&\geq \begin{bmatrix} c_B^T & c_N^T \end{bmatrix} \\
&= c^T
\end{aligned}
$$

dual feasible

LP Duality: Economic Interpretation

Brewer: find optimal mix of beer and ale to maximize profits.

$$
\begin{array}{rrcrcr}
\text{(P)} & \max \ 13A & + & 23B & & \\
\text{s. t.} & 5A & + & 15B & \le & 480 \\
 & 4A & + & 4B & \le & 160 \\
 & 35A & + & 20B & \le & 1190 \\
 & A & , & B & \ge & 0
\end{array}
$$

$A^* = 12$
$B^* = 28$
$OPT = 800$

Entrepreneur: buy individual resources from brewer at min cost.

$$
\begin{array}{rrcrcrcr}
\text{(D)} & \min \ 480C & + & 160H & + & 1190M & & \\
\text{s. t.} & 5C & + & 4H & + & 35M & \ge & 13 \\
 & 15C & + & 4H & + & 20M & \ge & 23 \\
 & C & , & H & , & M & \ge & 0
\end{array}
$$

$C^* = 1$
$H^* = 2$
$M^* = 0$
$OPT = 800$

LP duality. Market clears.

LP Duality: Sensitivity Analysis

Q. How much should brewer be willing to pay (marginal price) for additional supplies of scarce resources?

A. corn \$1, hops \$2, malt \$0.

Q. Suppose a new product "light beer" is proposed. It requires 2 corn, 5 hops, 24 malt. How much profit must be obtained from light beer to justify diverting resources from production of beer and ale?

A. At least 2 (\$1) + 5 (\$2) + 24 (\$0) = \$12 / barrel.

LP Duality Example

- Fine the dual of the following LP:

$$\text{Maximize} \qquad Z = 2x_1 + x_2$$

under constraints

$$\begin{array}{rcrcr} x_1 & + & x_2 & \geq & 4 \\ -x_1 & + & 2x_2 & \leq & 1 \\ -3x_1 & + & x_2 & = & -1 \end{array}$$

and $x_1 \geq 0$, $x_2 \in \mathsf{R}$.

LP Duality Example

- Fine the dual of the following LP:

$$\text{Maximize} \qquad Z = 2x_1 + x_2$$

under constraints

$$\begin{array}{rcrcr} x_1 & + & x_2 & \geq & 4 \\ -x_1 & + & 2x_2 & \leq & 1 \\ -3x_1 & + & x_2 & = & -1 \end{array}$$

and $x_1 \geq 0,\ x_2 \in \mathrm{R}.$

- The dual can be found as follows:

	Primal	Dual
Objective function	Max $Z = 2x_1 + x_2$	Min $W = -4y_1 + y_2 - y_3.$
Row (1)	$-x_1 - x_2 \leq -4$	$y_1 \geq 0$
Row (2)	$-x_1 + 2x_2 \leq 1$	$y_2 \geq 0$
Row (3)	$-3x_1 + x_2 = -1$	no sign constraint on y_3
Variable (1)	$x_1 \geq 0$	$-y_1 - y_2 - 3y_3 \geq 2$
Variable (2)	x_2 has no sign constraint	$-y_1 + 2y_2 + y_3 = 1$

Longest Common Substring

A slightly different problem (longest common subsequence) with a similar solution

Given two strings $X = x_1x_2...x_m$ and $Y = y_1y_2...y_n$, find their longest common substring Z , i.e., a largest k for which there are indices i and j with $x_ix_{i+1}...x_{i+k-1} = y_jy_{j+1}...y_{j+k-1}$.

For example:
X : DEADBEEF
Y : EATBEEF
Z : BEEF //pick the longest contiguous substring

Show how to do this by dynamic programming.

LCS Solution

Step 1: Space of Subproblems

For $1 \le i \le m$, and $1 \le j \le n$,

- Define $d_{i,j}$ to be the length of the longest common substring ending at x_i and y_j . (Does this work?)
- Let D be the $m \times n$ matrix $[d_{i,j}]$.
  - How does D provide answer?

LCS Solution

Step 2: Recursive Formulation

Case 1: If $x_i = y_j$, then $z_k = x_i = y_j$ and

z_{k-1} is a LCS of X and Y ending at x_{i-1} and y_{j-1}

Case 2: If $x_i \neq y_j$, then there cannot be a common substring ending at x_i and y_j!

$$d_{i,j} = \begin{cases} d_{i-1,j-1} + 1 & if\ x_i = y_j \\ 0 & if\ x_i \neq y_j \end{cases}$$

Finally, we can find length of longest common substring by finding maximum $d_{i,j}$ among all possible ending position i and j.

$$LCSSubString(X, Y) = \max\{d_{i,j}\}$$

LCS Solution

Step 3: Bottom-up Computation

Similar to *Longest Common Subsequence* we set the first row and column of the matrix $d[0,j]$ and $d[i,0]$ to be 0.

Calculate $d[1,j]$ for $j = 1, 2, \dots, \mathrm{n}$
Then, the $d[2,j]$ for $j = 1, 2, \dots, \mathrm{n}$
Then, the $d[3,j]$ for $j = 1, 2, \dots, \mathrm{n}$

etc., filling the matrix row by row and left to right.

For this problem we do not need to create another $m \times n$ matrix for storing arrows. Instead, we use l_{max} and p_{max} to store the largest length of common substring and its i position respectively. This suffices to reconstruct the solution.

LCS Solution

LONGEST-COMMON-SUBSTRING(X, Y)

$m \leftarrow length(X)$; $n \leftarrow length(Y)$;

$l_{max} \leftarrow 0$; $p_{max} \leftarrow 0$;

$\boldsymbol{for}$ i $\leftarrow 0$ $\boldsymbol{to}$ m // initialization

$\quad d[i, 0] \leftarrow 0$;

$\boldsymbol{for}$ $j \leftarrow 0$ $\boldsymbol{to}$ n

$\quad d[0, j] \leftarrow 0$;

$\boldsymbol{for}$ i $\leftarrow 1$ $\boldsymbol{to}$ m // dynamic programming

$\quad \boldsymbol{for}$ $j \leftarrow 1$ $\boldsymbol{to}$ n

$\quad\quad \boldsymbol{if}(x_i \neq y_j)$

$\quad\quad\quad d[i, j] \leftarrow 0$;

$\quad\quad$ ***else***

$\quad\quad\quad d[i, j] \leftarrow d[i-1, j-1] + 1$;

$\quad\quad\quad \boldsymbol{if}(d[i, j] > l_{max})$

$\quad\quad\quad\quad l_{max} \leftarrow d[i, j]$; $p_{max} \leftarrow i$;

$\quad\quad\quad \ldots\ldots$

$\boldsymbol{return}$ l_{max}, p_{max};

LCS Example

- Take the two strings: *X* = “EL GATO” and *Y* =“GATER”.
- We’ll fill in the following table *D*:

$$d_{i,j} = \begin{cases} d_{i-1,j-1} + 1 & if\ x_i = y_j \\ 0 & if\ x_i \neq y_j \end{cases}$$

LCS Example

- Take the two strings: *X* = “EL GATO” and *Y* =“GATER”.
- We’ll fill in the following table *D*:

$$d_{i,j} = \begin{cases} d_{i-1,j-1} + 1 & if\ x_i = y_j \\ 0 & if\ x_i \neq y_j \end{cases}$$

When filling *D*, we only look if the two letters in the strings are equal and if they are we add one to the element to the left and up.

	_	E	L	G	A	T	O
_	0	0	0	0	0	0	0
G	0	0	0	1	0	0	0
A	0	0	0	0	2	0	0
T	0	0	0	0	0	3	0
E	0	1	0	0	0	0	0
R	0	0	0	0	0	0	0

All-Pairs Shortest Paths

Input: weighted digraph $G = (V, E)$ with weight function $w : E \to \mathbb{R}$

Find: lengths of the shortest paths (i.e., distance) between all pairs of vertices in G.

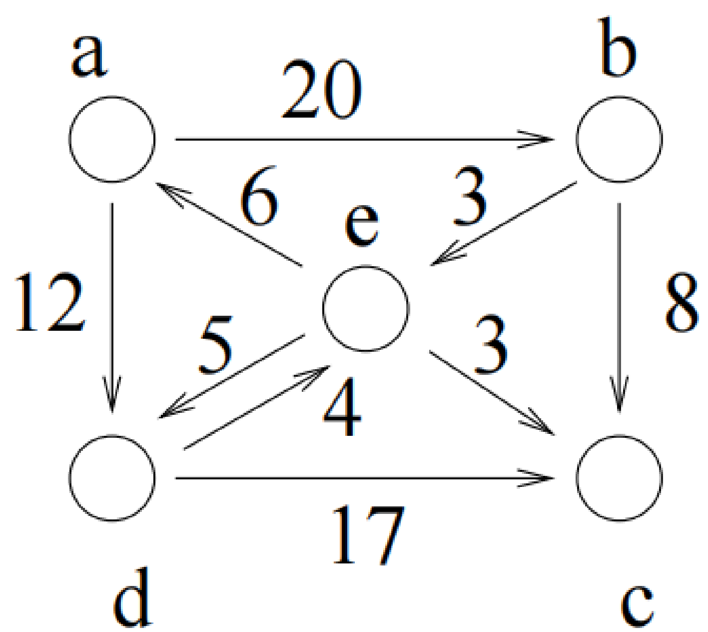
- we assume that there are no cycles with zero or negative cost.

All-Pairs Shortest Paths

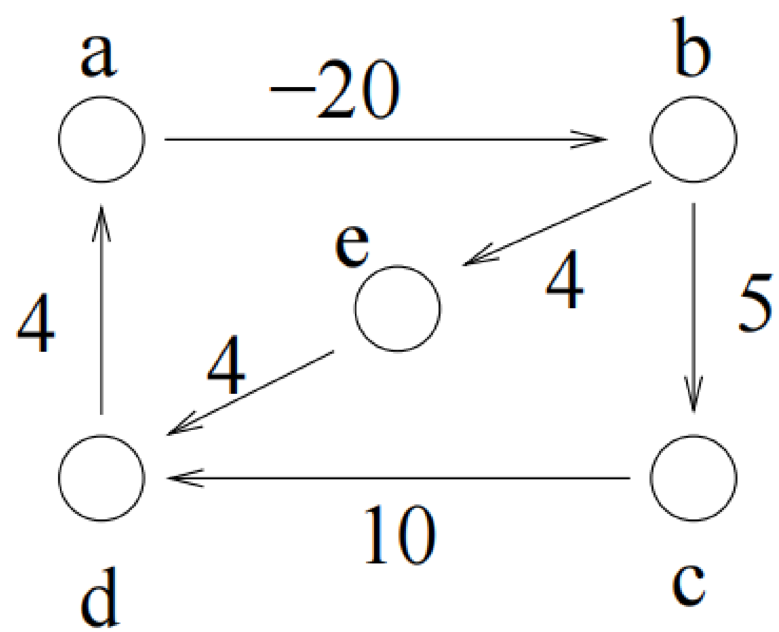
Input: weighted digraph $G = (V, E)$ with weight function $w : E \to \mathbb{R}$

Find: lengths of the shortest paths (i.e., distance) between all pairs of vertices in G.

- we assume that there are no cycles with zero or negative cost.



without negative cost cycle



with negative cost cycle

Input and Output Formats

Input Format:

- To simplify the notation, we assume that $V = \{1, 2, . . . , n\}$.
- Adjacency matrix: graph is represented by an n x n matrix containing edge weights

$$w_{ij} = \begin{cases} 0 & if \ \ i = j, \\ & \\ & \end{cases}$$

Input and Output Formats

Input Format:

- To simplify the notation, we assume that $V = \{1, 2, . . . , n\}$.
- Adjacency matrix: graph is represented by an n x n matrix containing edge weights

$$w_{ij} = \begin{cases} 0 & if\ \ i\ =\ j, \\ w(i,j) & if\ \ i \neq j\ and\ (i,j) \in E, \\ & \end{cases}$$

Input and Output Formats

Input Format:

- To simplify the notation, we assume that $V = \{1, 2, \ldots, n\}$.
- Adjacency matrix: graph is represented by an n x n matrix containing edge weights

$$w_{ij} = \begin{cases} 0 & if\ \ i = j, \\ w(i,j) & if\ \ i \neq j\ and\ (i,j) \in E, \\ \infty & if\ \ i \neq j\ and\ (i,j) \notin E. \end{cases}$$

Input and Output Formats

Input Format:

- To simplify the notation, we assume that $V = \{1, 2, \ldots, n\}$.
- Adjacency matrix: graph is represented by an n x n matrix containing edge weights

$$w_{ij} = \begin{cases} 0 & if \;\; i = j, \\ w(i,j) & if \;\; i \neq j \; and \; (i,j) \in E, \\ \infty & if \;\; i \neq j \; and \; (i,j) \notin E. \end{cases}$$

Output Format: an $n \times n$ matrix $D = [d_{ij}]$ in which d_{ij} is the length of the shortest path from vertex i to j.

Step 1: Space of Subproblems

For m = 1, 2, 3, ...

Define $d_{ij}^{(m)}$ to be the length of the shortest path from i to j that contains at most m edges.

Let $D^{(m)}$ be the $n \times n$ matrix $[d_{ij}^{(m)}]$

We will see (next page) that solution D satisfies $D=D^{n-1}$.

Subproblems: (Iteratively)compute $D^{(m)}$for m=1,...,n-1.

Step 1: Space of Subproblems

Lemma

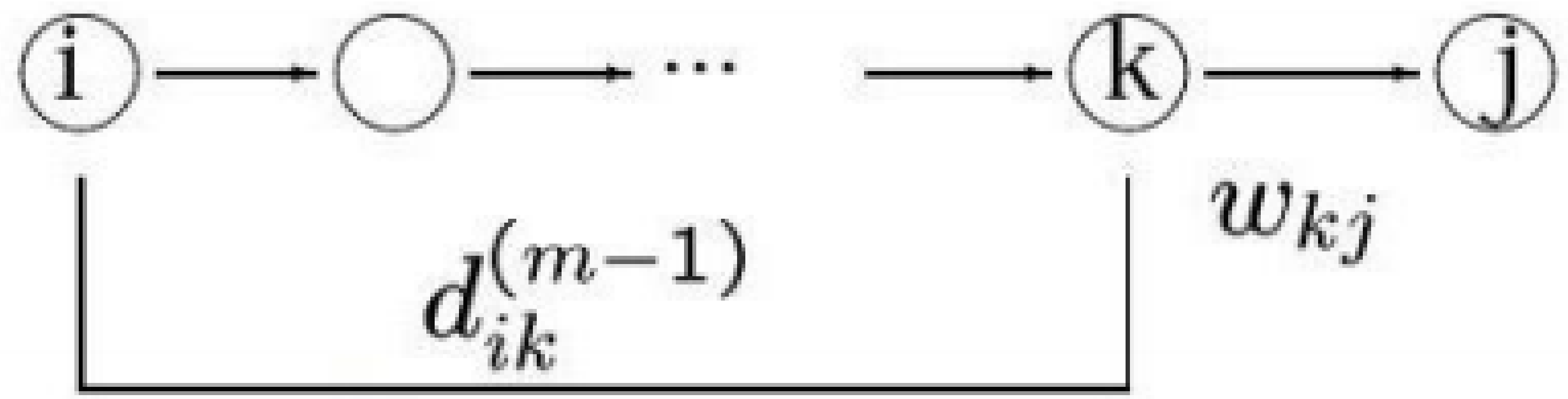
- $D^{(n-1)} = D$
- $d_{ij}^{(n-1)}$ = true distance from i to j

Proof

- We prove that any shortest path P from i to j contains at most n − 1 edges.
- First note that since all cycles have positive weight, a shortest path can have no cycles (if there were a cycle, we could remove it and lower the length of the path).
- A path without cycles can have length at most n − 1 (since a longer path must contain some vertex twice, that is, contain a cycle).

Step 2: Building $D^{(m)}$ from $D^{(m-1)}$

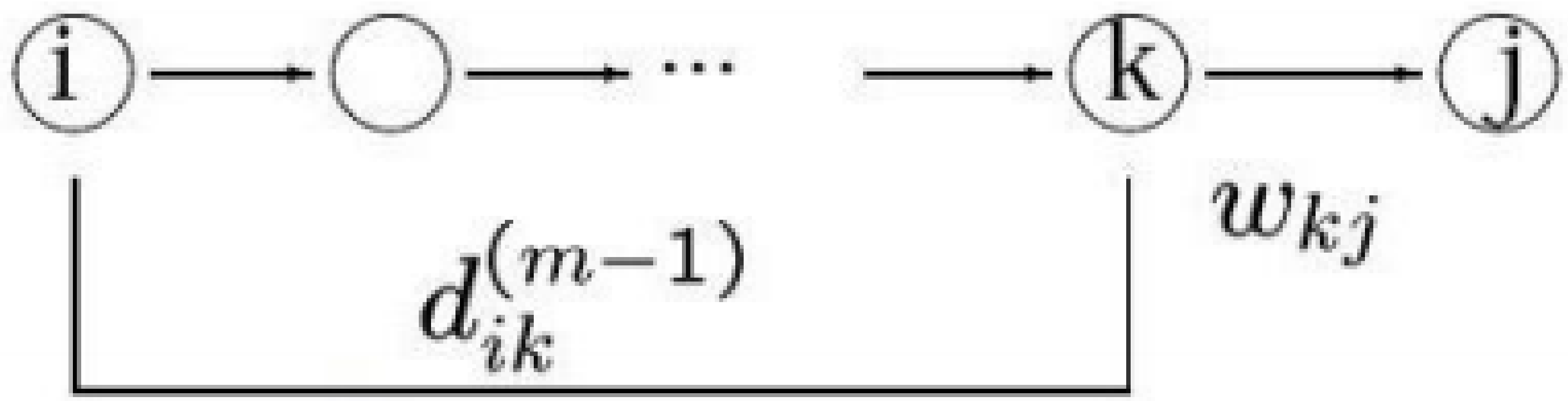
Consider a shortest path from i to j that contains at most m edges.



Let k be the vertex imriately before j on the shortest path.

Step 2: Building $D^{(m)}$ from $D^{(m-1)}$

Consider a shortest path from i to j that contains at most m edges.



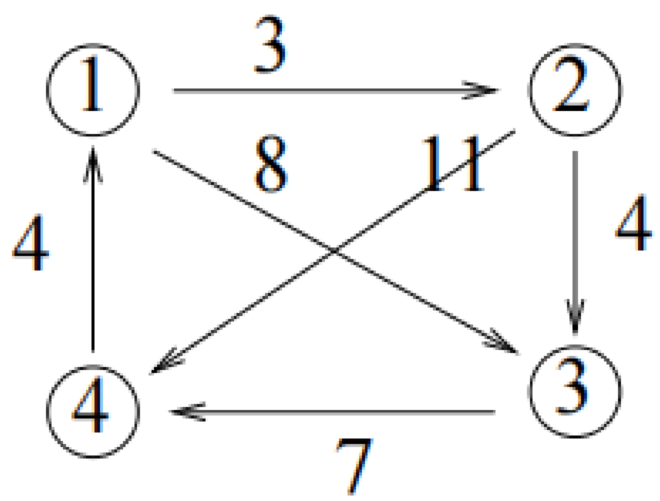
Let k be the vertex imriately before j on the shortest path.

The sub-path from i to k must be the shortest 1-k path with at most m-1 edges: $d_{ij}^{(m)} = d_{ik}^{(m-1)} + w_{kj}$

Since we don't know k,we try all possible choices: $d_{ij}^{(m)} = \min_{1 \le k \le n} \{d_{ik}^{(m-1)} + w_{kj}\}$

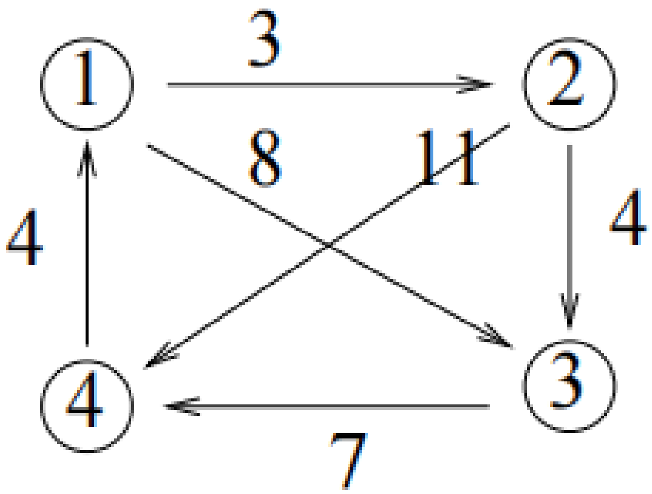
Example: Bottom-up Computation of $D^{(m-1)}$

$D^{(1)} = [w_{ij}]$ is just the weight matrix:



Example: Bottom-up Computation of $D^{(m-1)}$

$D^{(1)} = [w_{ij}]$ is just the weight matrix:

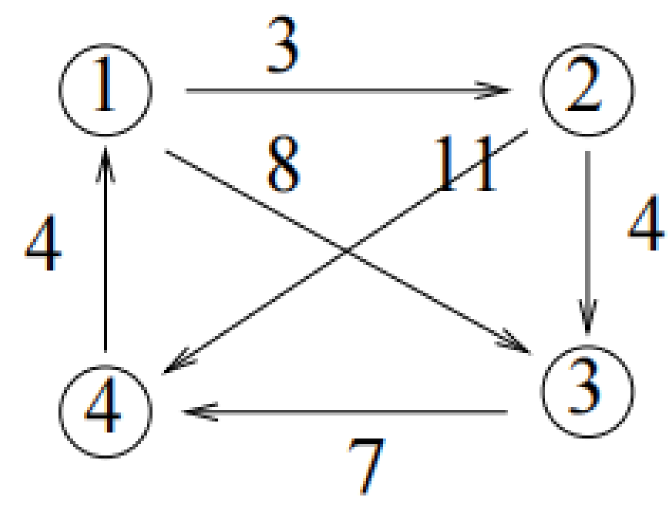


$$D^{(1)} = \begin{bmatrix} 0 & 3 & 8 & \infty \\ \infty & 0 & 4 & 11 \\ \infty & \infty & 0 & 7 \\ 4 & \infty & \infty & 0 \end{bmatrix}$$

$$d_{ij}^{(2)} = \min_{1 \le k \le 4} \{ d_{ik}^{(1)} + w_{kj} \}$$

Example: Bottom-up Computation of $D^{(m-1)}$

$D^{(1)} = [w_{ij}]$ is just the weight matrix:



$$D^{(1)} = \begin{bmatrix} 0 & 3 & 8 & \infty \\ \infty & 0 & 4 & 11 \\ \infty & \infty & 0 & 7 \\ 4 & \infty & \infty & 0 \end{bmatrix}$$

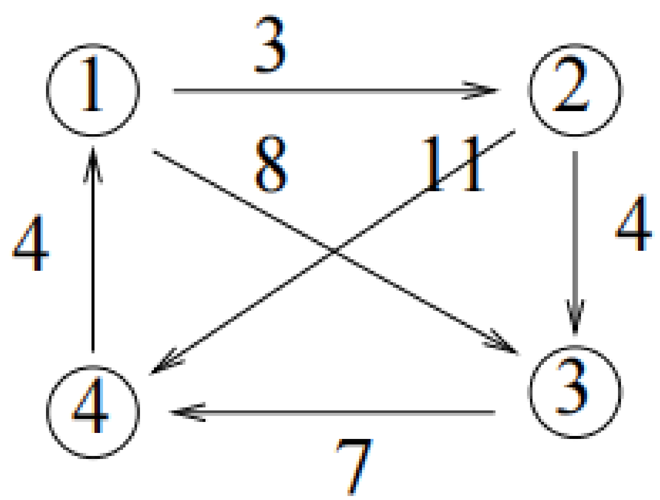
$$d_{ij}^{(2)} = \min_{1 \le k \le 4} \{d_{ik}^{(1)} + w_{kj}\}$$

$$D^{(2)} = \begin{bmatrix} 0 & 3 & 7 & 14 \\ 15 & 0 & 4 & 11 \\ 11 & \infty & 0 & 7 \\ 4 & 7 & 12 & 0 \end{bmatrix}$$

Example: Bottom-up Computation of $D^{(m-1)}$

$D^{(1)} = [w_{ij}]$ is just the weight matrix:

$$D^{(1)} = \begin{bmatrix} 0 & 3 & 8 & \infty \\ \infty & 0 & 4 & 11 \\ \infty & \infty & 0 & 7 \\ 4 & \infty & \infty & 0 \end{bmatrix}$$



$$\color{red}{d_{ij}^{(2)}} = \min_{1 \le k \le 4} \left\{ d_{ik}^{(1)} + w_{kj} \right\}$$

$$\color{red}{d_{ij}^{(3)}} = \min_{1 \le k \le 4} \left\{ d_{ik}^{(2)} + w_{kj} \right\}$$

$$D^{(2)} = \begin{bmatrix} 0 & 3 & 7 & 14 \\ 15 & 0 & 4 & 11 \\ 11 & \infty & 0 & 7 \\ 4 & 7 & 12 & 0 \end{bmatrix}$$

$$D^{(3)} = \begin{bmatrix} 0 & 3 & 7 & 14 \\ 15 & 0 & 4 & 11 \\ 11 & 14 & 0 & 7 \\ 4 & 7 & 11 & 0 \end{bmatrix}$$

$D^{(3)}$ gives the distances between any pair of vertices.

Review of Matrix Multiplication

- **Matrix**: An $n \times m$ matrix $A = [a[i, j]]$ is a two-dimensional array.

$$A = \begin{bmatrix} a[1,1] & a[1,2] & \cdots & a[1,m-1] & a[1,m] \\ a[2,1] & a[2,2] & \cdots & a[2,m-1] & a[2,m] \\ \vdots & \vdots & & \vdots & \vdots \\ a[n,1] & a[n,2] & \cdots & a[n,m-1] & a[n,m] \end{bmatrix},$$

which has n rows and m columns.

Review of Matrix Multiplication

- The product $C = AB$ of a $p \times q$ matrix A and a $q \times r$ matrix B is a $p \times r$ matrix C given by.

$$c[i,j] = \sum_{k=1}^{q} a[i,k]b[k,j], \quad \text{for } 1 \le i \le p \text{ and } 1 \le j \le r$$

- Complexity of Matrix multiplication: Note that C has pr entries and each entry takes $\Theta(q)$ time to compute so the total procedure takes $\Theta(pqr)$ time.

Remarks on Matrix Multiplication

- Matrix multiplication is associative, e.g.,

$$A_1A_2A_3 = (A_1A_2)A_3 = A_1(A_2A_3),$$

so parenthesization does not change result.

- Matrix multiplication is NOT commutative, e.g.,

$$A_1A_2 \neq A_2A_1$$

Matrix Multiplication of ABC

- Given $p \times q$ matrix A, $q \times r$ matrix B and $r \times s$ matrix C, ABC can be computed in two ways: $(AB)C$ and $A(BC)$.
- The number of multiplications needed are:

$$\begin{aligned} mult[(AB)C] &= pqr + prs, \\ mult[A(BC)] &= qrs + pqs. \end{aligned}$$

Implication: Multiplication "sequence" (parenthesization) is important!!

The Chain Matrix Multiplication Problem

- Definition (Chain matrix multiplication problem) :

  Given dimensions p_0,p_1, . . . ,p_n, corresponding to matrix sequence $A_1A_2 \ldots A_n$ in which Ai has dimension $p_{i-1} \times p_i$, determine the "multiplication sequence" that minimizes the number of scalar multiplications in computing $A_1A_2 \ldots A_n$.
- Question: Is there a better approach?

Developing a Dynamic Programming Algorithm

Step 1: Define Space of Subproblems

- Original Problem:

  Determine minimal cost multiplication sequence for $A_{1..n}$.

- Subproblems: For every pair $1 \le i \le j \le n$:

  Determine minimal cost multiplication sequence for $A_{i..j} = A_i A_{i+1} \dots A_j$.

  Note that $A_{i..j}$ is a $p_{i-1} \times p_j$ matrix.

- There are $\binom{n}{2} = \theta(n^2)$ such subproblems. (Why?)
- How can we solve larger problems using subproblem solutions?

Relationships among Subproblems

- At the last step of any optimal multiplication sequence (for a subbroblem), there is some k such that the two matrices $A_{i..k}$and $A_{k+1..j}$ are multipled together. That is,

$$A_{i..j} = (A_i \cdots A_k)(A_{k+1} \cdots A_j) = A_{i..k}A_{k+1..j}$$

- Question. How do we decide where to split the chain (what is k)?

  ANS: Can be any k. Need to check all possible values.

- Question. How do we parenthesize the two subchains $A_{i..k}$ and $A_{k+1..j}$?

- For some problems, **the subtrees will not overlap**.

  ANS: $A_{i..k}$ and $A_{k+1..j}$ must be computed optimally, so we can apply the same procedure recursively.

Relationships among Subproblems

Step 2: Constructing optimal solutions from optimal subproblem solution

- For *1 ≤ i ≤ j ≤ n*, let $m[i,j]$ denote the minimum number of multiplications needed to compute $A_{i..j}$. This optimum cost must satisify the following recursive definition.

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

$$A_{i..j} = A_{i..k}A_{k+1..j}$$

Developing a Dynamic Programming Algorithm

Step 3: Bottom-up computation of $m[i, j]$

- Recurrence:

Fill in the $m[i, j]$ table in an order, such that when it is time to calculate $m[i, j]$, the values of $m[i, k]$ and $m[k + 1, j]$ for all k are already available.

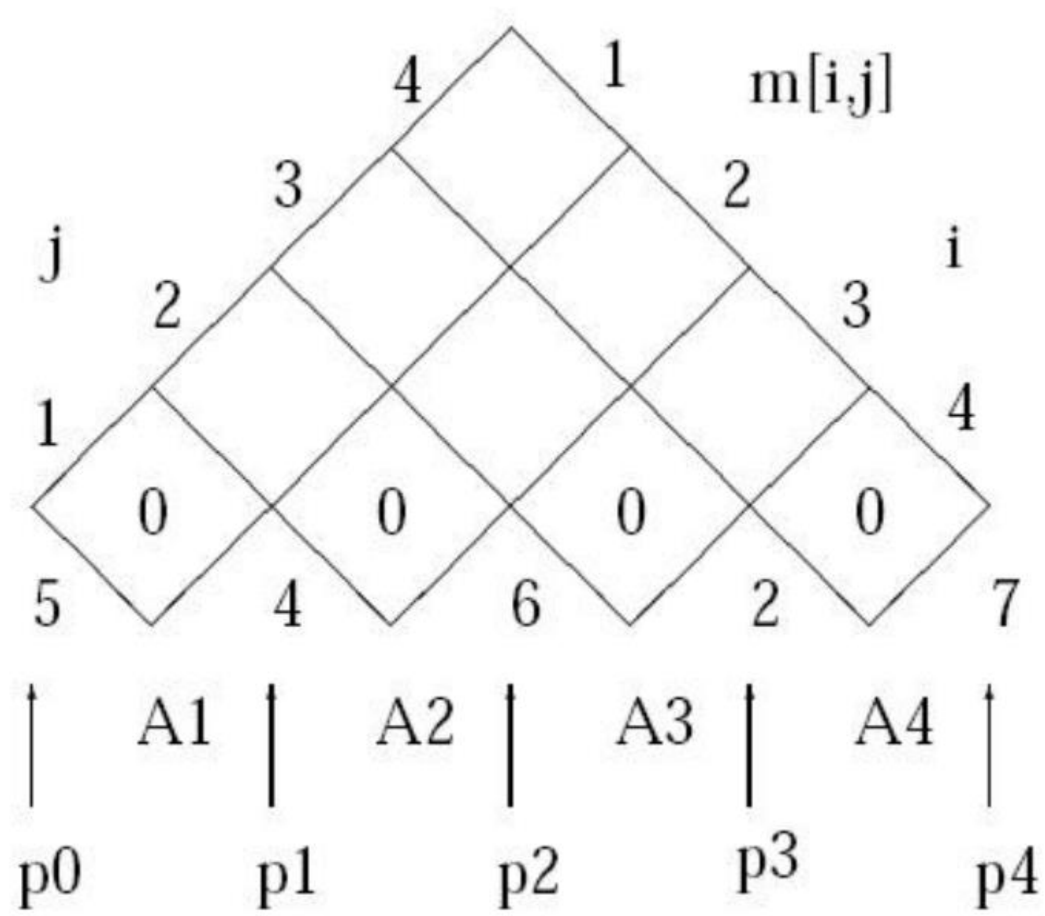
An easy way to ensure this is to compute them in increasing order of the size $(j - i)$ of the matrix-chain $A_{i..j}$:
$m[1, 2], m[2, 3], m[3, 4], \dots, m[n - 3, n - 2], m[n - 2, n - 1], m[n - 1, n]$
$m[1, 3], m[2, 4], m[3, 5], \dots, m[n - 3, n - 1], m[n - 2, n]$
$m[1, 4], m[2, 5], m[3, 6], \dots, m[n - 3, n]$
$\dots$
$m[1, n - 1], m[2, n]$
$m[1, n]$

Example for the Bottom-Up Computation

- Example.

  A chain of four matrices A_1, A_2, A_3 and A_4, with $p_0 = 5, p_1 = 4, p_2 = 6, p_3 = 2$ and $p_4 = 7$. Find $m[1,4]$.

  S0: Initialization

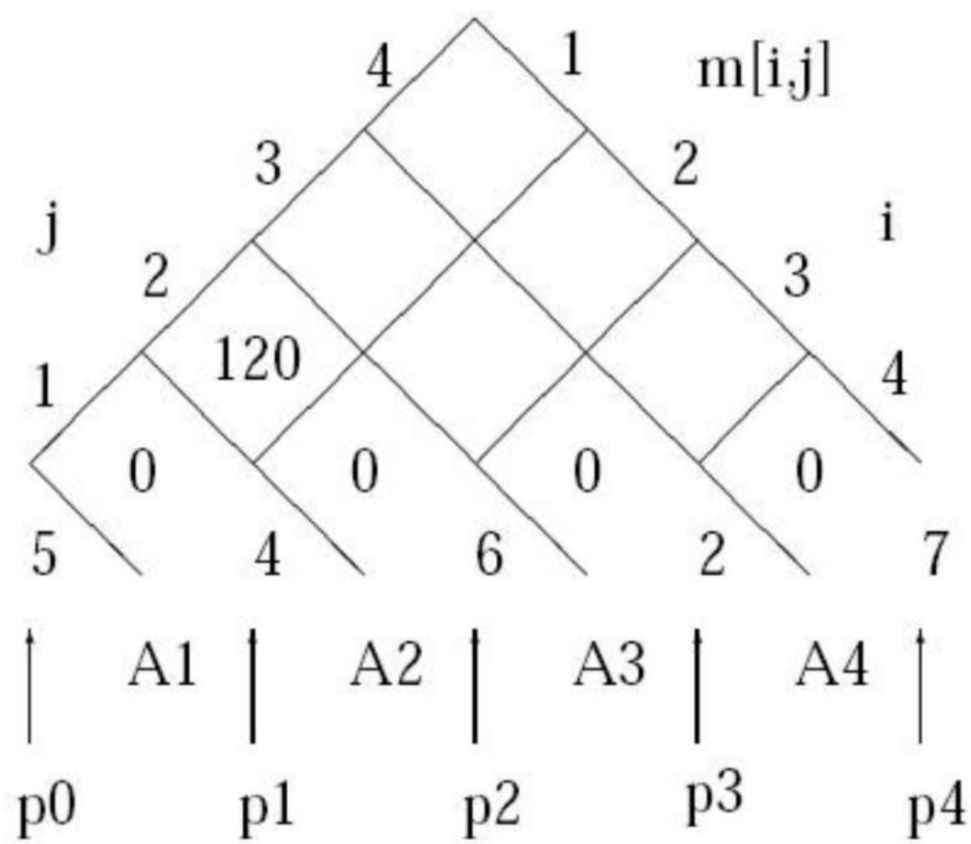


Example – Continued

- Step 1: Computing $m[1,2]$

By definition

$$m[1,2] = \min_{1\le k<2}(m[1,k] + m[k+1,2] + p_0 p_k p_2$$
$$= m[1,1] + m[2,2] + p_0 p_1 p_2 = 120$$

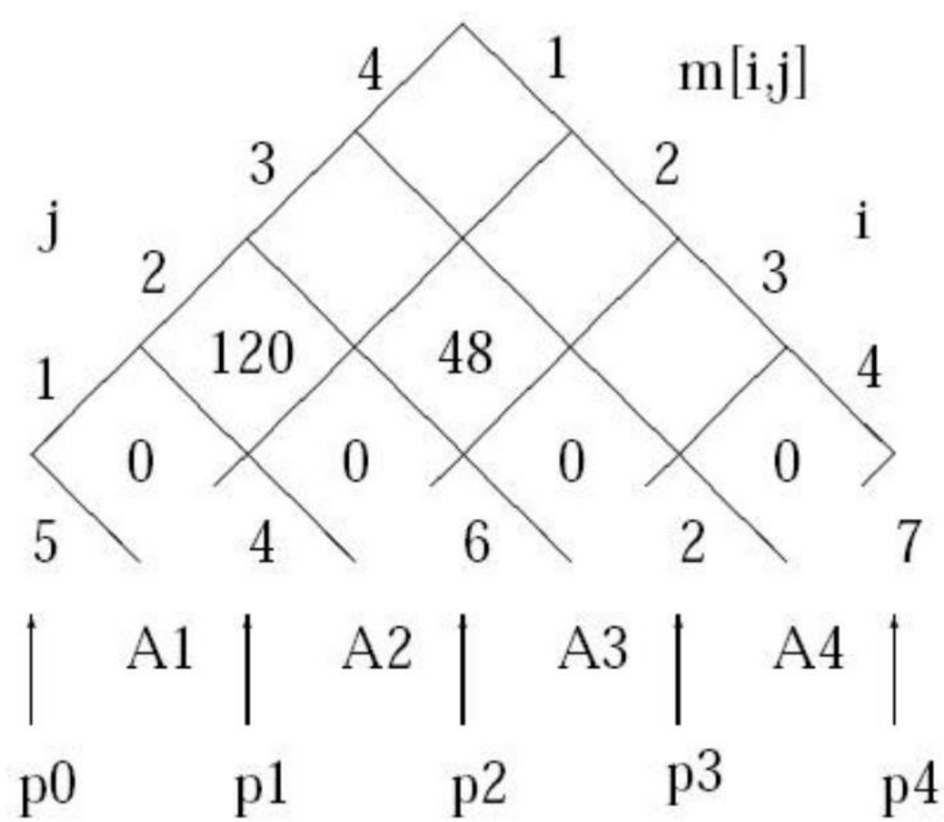


Example – Continued

- Step 2: Computing $m[2,3]$

By definition

$$
\begin{aligned}
m[2,3] &= \min_{2 \le k < 3} (m[2,k] + m[k+1,3] + p_1 p_k p_3 \\
&= m[2,2] + m[3,3] + p_1 p_2 p_3 = 48
\end{aligned}
$$

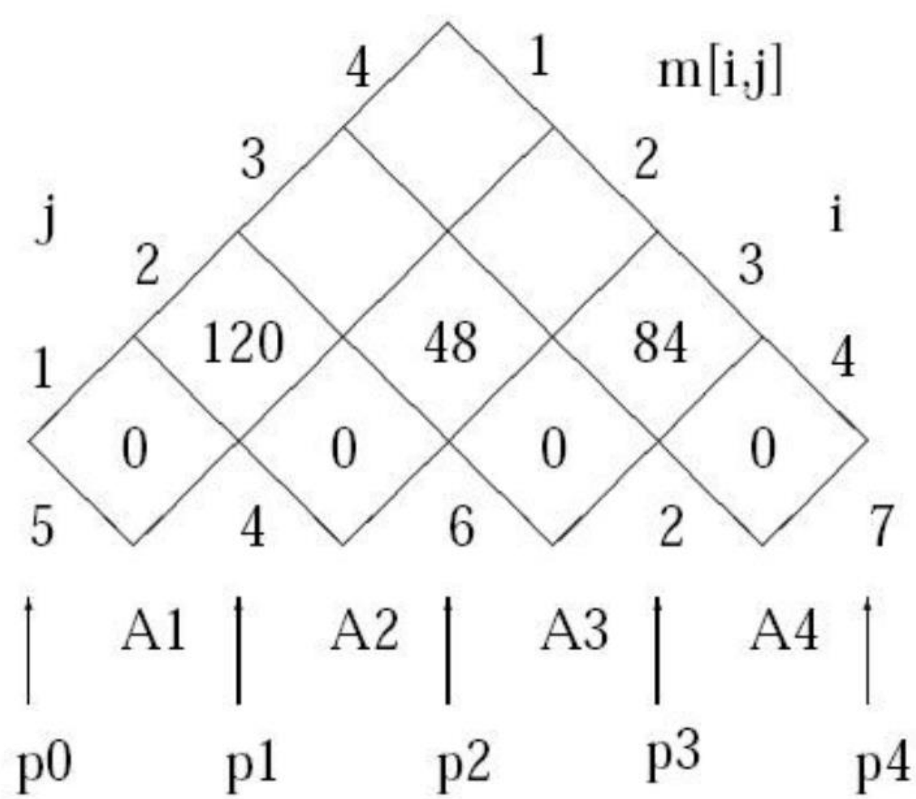


Example – Continued

- Step 3: Computing $m[3,4]$

By definition

$$
\begin{aligned}
m[3,4] \quad &= \quad \min_{3 \le k < 4} (m[3,k] + m[k+1,4] + p_2 p_k p_4 \\
&= \quad m[3,3] + m[4,4] + p_2 p_3 p_4 = 84
\end{aligned}
$$

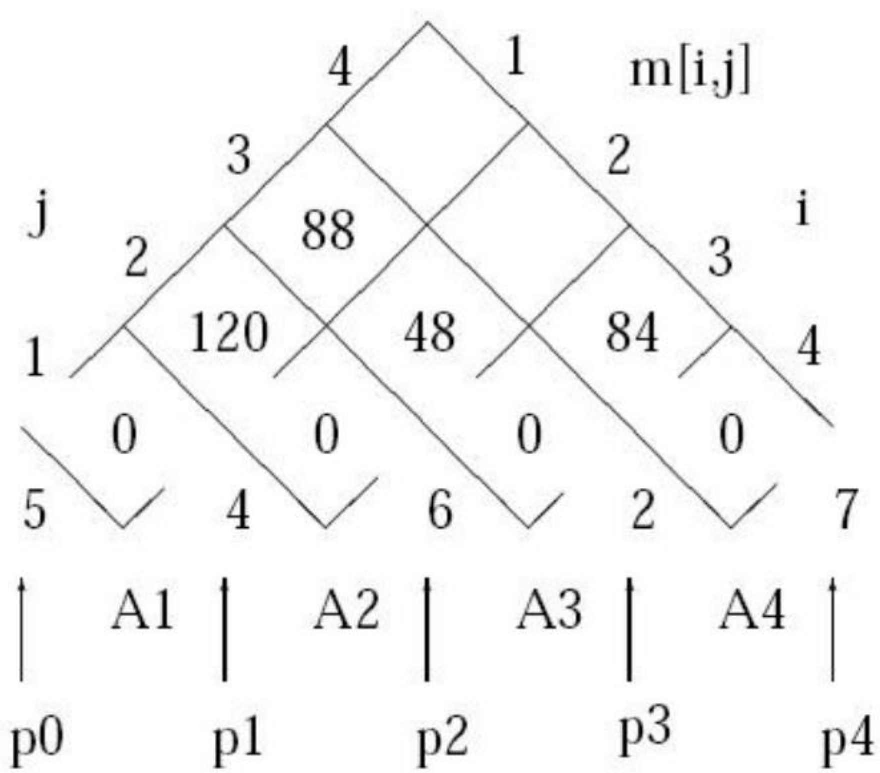


Example – Continued

- Step 4: Computing $m[1,3]$

By definition

$$\begin{aligned} m[1,3] \quad &= \quad \min_{1 \le k < 3} (m[1,k] + m[k+1,3] + p_0 p_k p_3 \\ &= \quad \min \begin{Bmatrix} m[1,1] + m[2,3] + p_0 p_1 p_3 \\ m[1,2] + m[3,3] + p_0 p_2 p_3 \end{Bmatrix} \\ &= \quad 88 \end{aligned}$$



Example – Continued

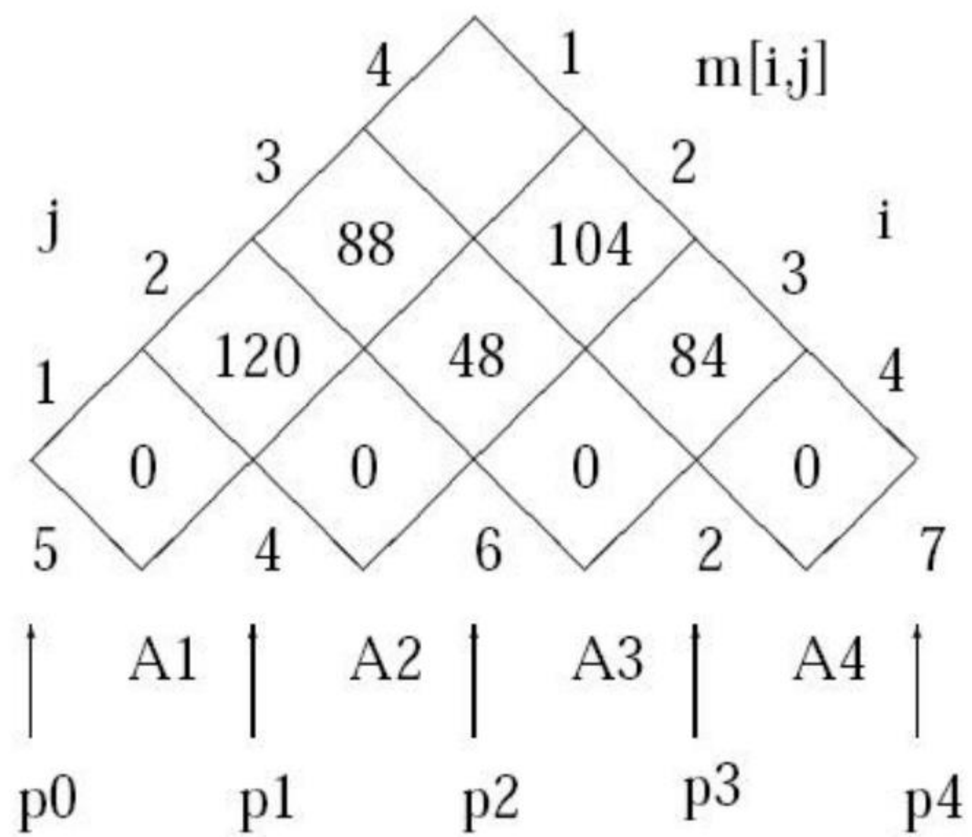
- Step 5: Computing $m[2,4]$

By definition

$$m[2,4] = \min_{2\le k<4} (m[2,k] + m[k+1,4] + p_1 p_k p_4$$

$$= \min \begin{cases} m[2,2] + m[3,4] + p_1 p_2 p_4 \\ m[2,3] + m[4,4] + p_1 p_3 p_4 \end{cases}$$

$$= 104$$

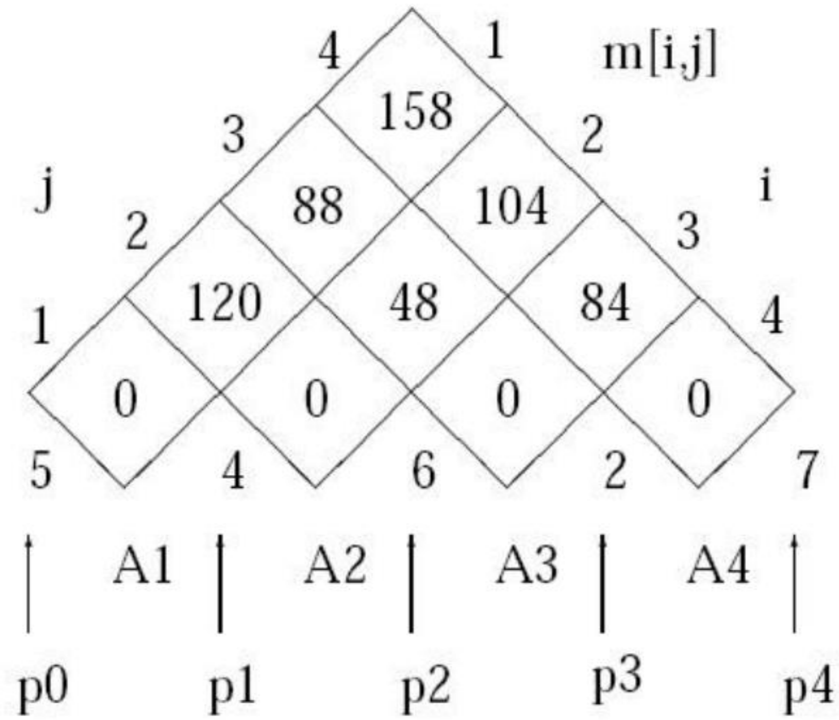


Example – Continued

- Step 6: Computing $m[1,4]$

By definition

$$
\begin{aligned}
m[1,4] &= \min_{1\le k<4}(m[1,k] + m[k+1,4] + p_0 p_k p_4 \\
&= \min\begin{Bmatrix} m[1,1] + m[2,4] + p_0 p_1 p_4 \\ m[1,2] + m[3,4] + p_0 p_2 p_4 \\ m[1,3] + m[4,4] + p_0 p_3 p_4 \end{Bmatrix} \\
&= 158
\end{aligned}
$$



The Dynamic Programming Algorithm

Matrix-Chain(p, n): // l is length of sub-chain

for $i = 1$ **to** n **do** $m[i, i] = 0$;

;

for $l = 2$ **to** n **do**

    for $i = 1$ **to** $n - l + 1$ **do**

        $j = i + l - 1$;

        $m[i, j] = \infty$;

        for $k = i$ **to** $j - 1$ **do**

            $q = m[i, k] + m[k + 1, j] + p[i - 1] * p[k] * p[j]$;

            if $q < m[i, j]$ **then**

                $m[i, j] = q$;

                $s[i, j] = k$;

            end

        end

    end

end

return m and s; (Optimum in $m[1, n]$)