Lecture 9
All -Pairs Shortest Paths Problem

1. Matrix multiplication
2. Floyd-Warshall algorithm
3. Transitive closure
4. Johnson's algorithm

All –pairs shortest paths problem

- Problem: Given a directed graph $G=(V, E)$, and a weight function $w: E \rightarrow R$, for each pair of vertices u, v, compute the shortest path weight $\delta(u, v)$, and a shortest path if exists.
- No negative cycles!
- Output: a $V \times V$ matrix $D=(d_{ij})$, where, d_{ij} contains the weight of a shortest path from vertex i to vertex j.

Methods

1) Application of single source shortest path algorithms
2) Direct methods to solve the problem:
    1) Matrix multiplication
    2) Floyd-Warshall algorithm
    3) Johnson's algorithm for sparse graphs
3) Transitive closure (Floyd-Warshall algorithm)

Motivation

- Computer network
- Aircraft network (e.g. flying time, fares)
- Railroad network
- Table of distances between all pairs of cities for a road atlas

Single source shortest path algorithms

If edges are non-negative:

Running the Dijkstra's algorithm n-times, once for each vertex as the source

running time: $O(V^2 \log V + VE)$ //using Fibonacci-heap

Single source shortest path algorithms

Negative-weight edges:

Bellman-Ford algorithm

Running time: $O(V^2 E)$

Dynamic Programming

- Characterize the structure of an optimal solution.
- Recursively define the value of an optimal solution.
- Compute the value of an optimal solution in a bottom-up fashion.

Data structure

- Adjacency matrix
- w: E $\rightarrow \Re$ as V x V matrix W

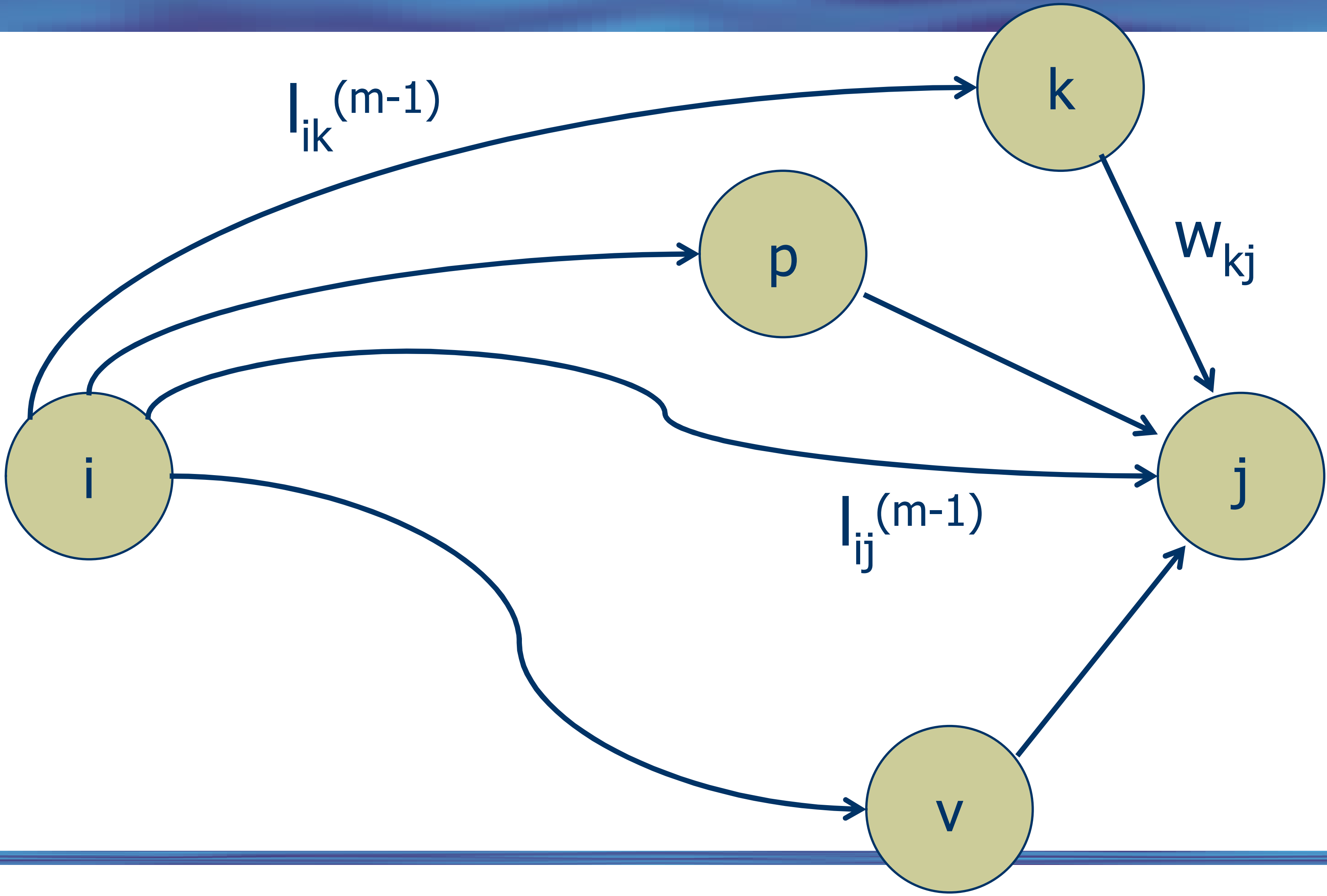
$$w_{ij} = \begin{cases} 0 & \text{if } i = j, \\ \text{weight of edge } (i,j) & \text{if } i \neq j \text{ and } (i,j) \in E, \\ \infty & \text{if } i \neq j \text{ and } (i,j) \notin E \end{cases}$$

Matrix multiplication (idea)

$l_{ij}^{(m)}$: minimum weight of any path from i to j

that contains at most m edges

Matrix multiplication (idea)



Matrix multiplication (idea)

$$l_{ij}^{(m)} = \min\left(l_{ij}^{(m-1)}, \min_{1 \leq k \leq n} \{l_{ik}^{(m-1)} + w_{kj}\}\right)$$

- Check all possible predecessors k of j and compare!

Matrix multiplication (structure)

$$l_{ij}^{(1)} = w_{ij}$$

$$l_{ij}^{(m)} = \min \left(l_{ij}^{(m-1)}, \min_{1 \le k \le n} \{l_{ik}^{(m-1)} + w_{kj}\}\right)$$
$$= \min_{1 \le k \le n} \{l_{ik}^{(m-1)} + w_{kj}\}$$

Matrix multiplication (structure)

- Compute a series of matrices

$$L^{(1)},\ L^{(2)},\ \ldots,\ L^{(n-1)}$$

$$L^{(m)} = L^{(m-1)} \cdot W$$

- **Final matrix** $L^{(\mathbf{n-1})}$ contains the final shortest-path weights

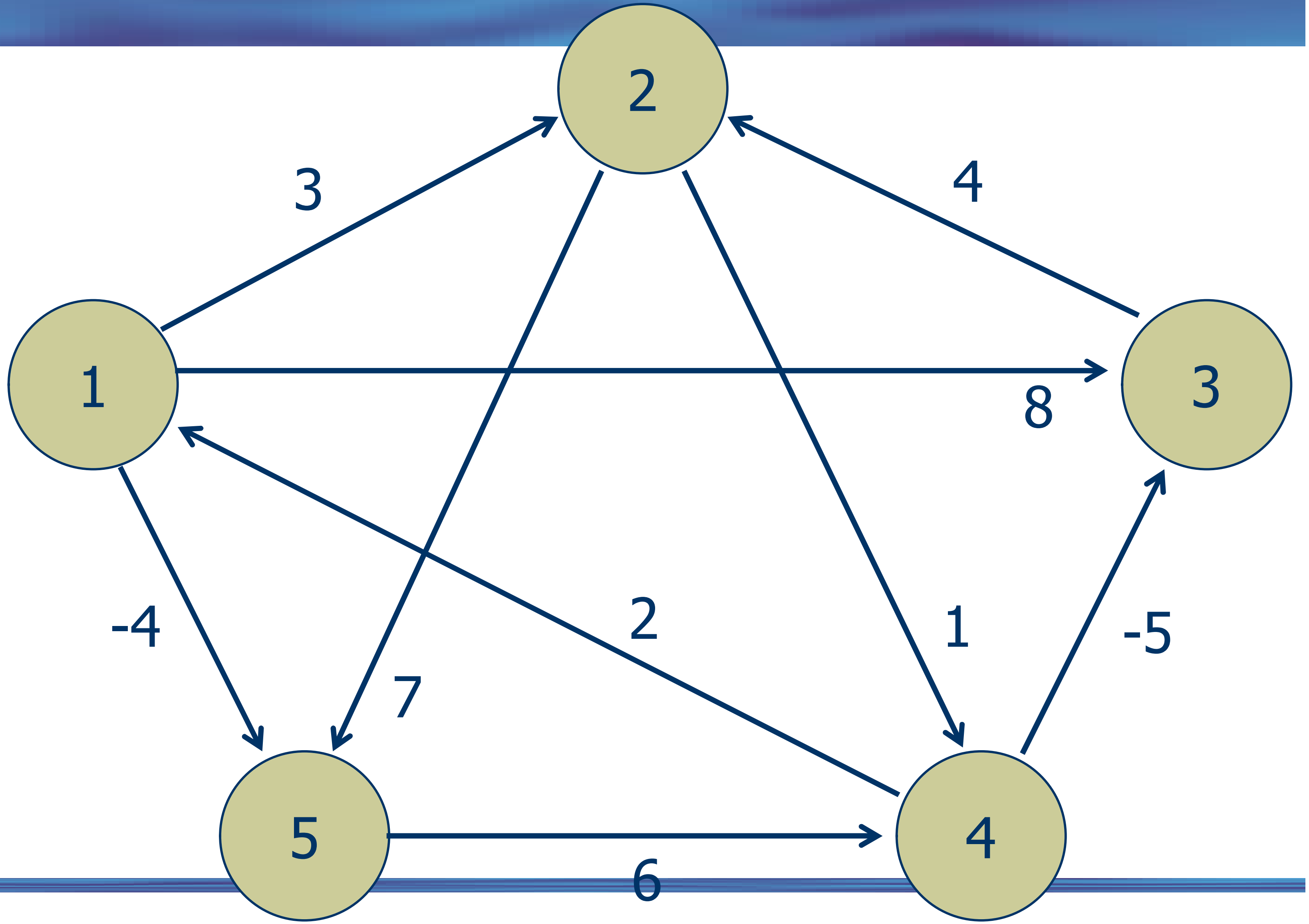
Matrix multiplication (pseudocode)

EXTEND-SHORTEST-PATHS(L, W)

$n \leftarrow rows[L]$
let $L' = (l'_{ij})$ be an $n \times n$ matrix
for $i \leftarrow 1$ **to** n
    do for $j \leftarrow 1$ **to** n
        do $l'_{ij} \leftarrow \infty$
            for $k \leftarrow 1$ **to** n
                do $l'_{ij} \leftarrow \min(l'_{ij}, l_{ik} + w_{kj})$
return L'

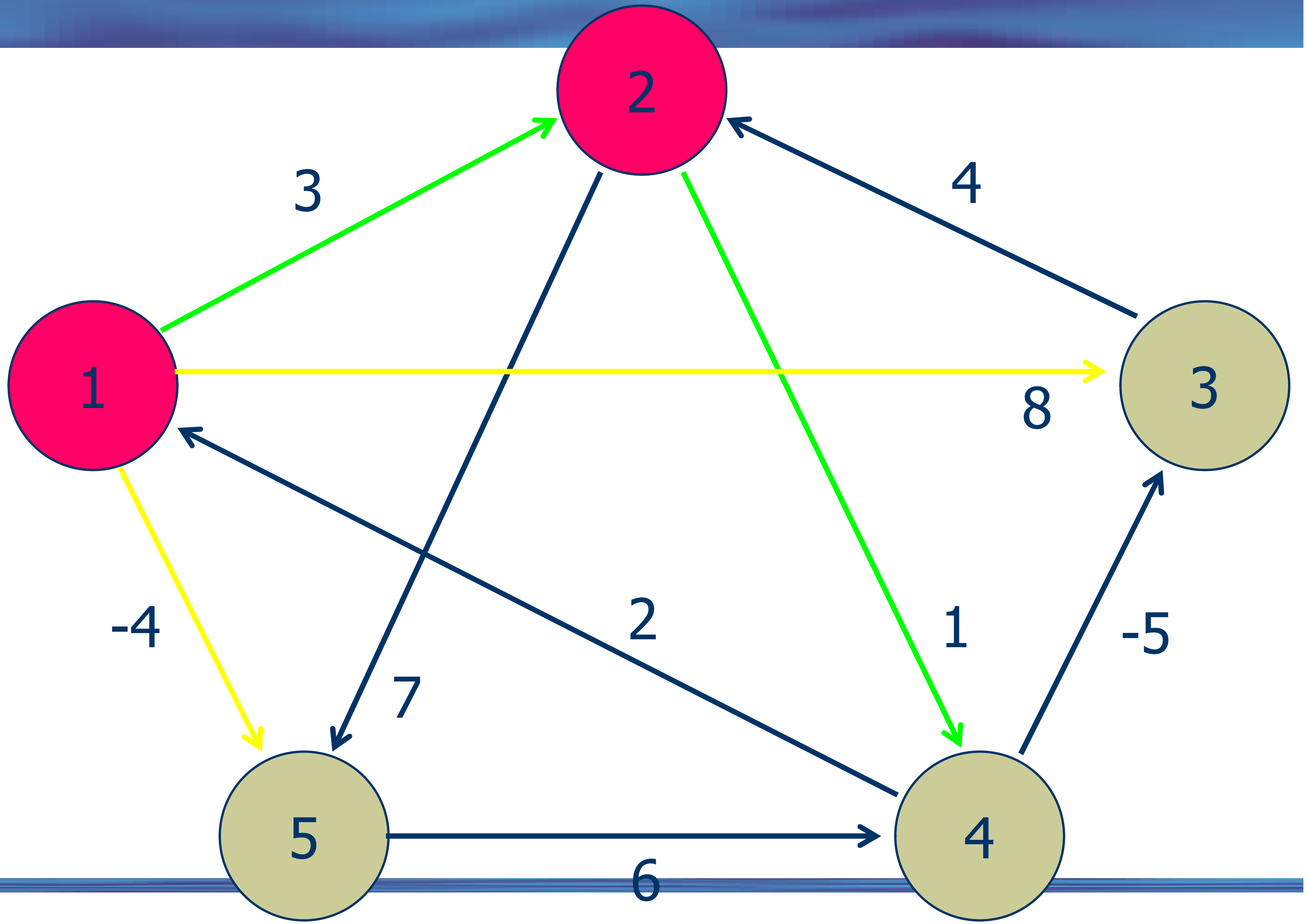
Matrix multiplication (example)

$L_{14}^{(2)}$



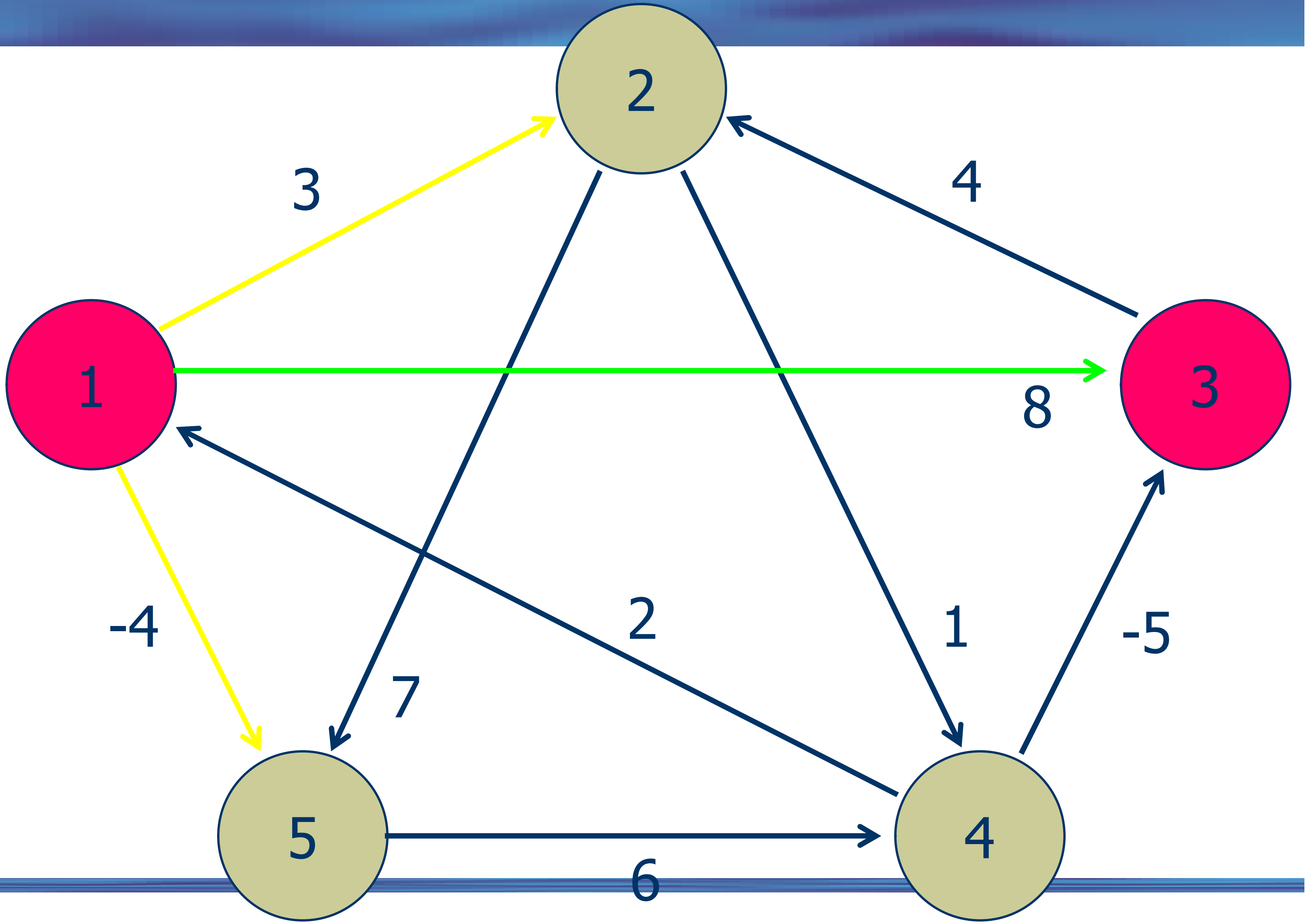
Matrix multiplication (example)

$L_{14}^{(2)}$



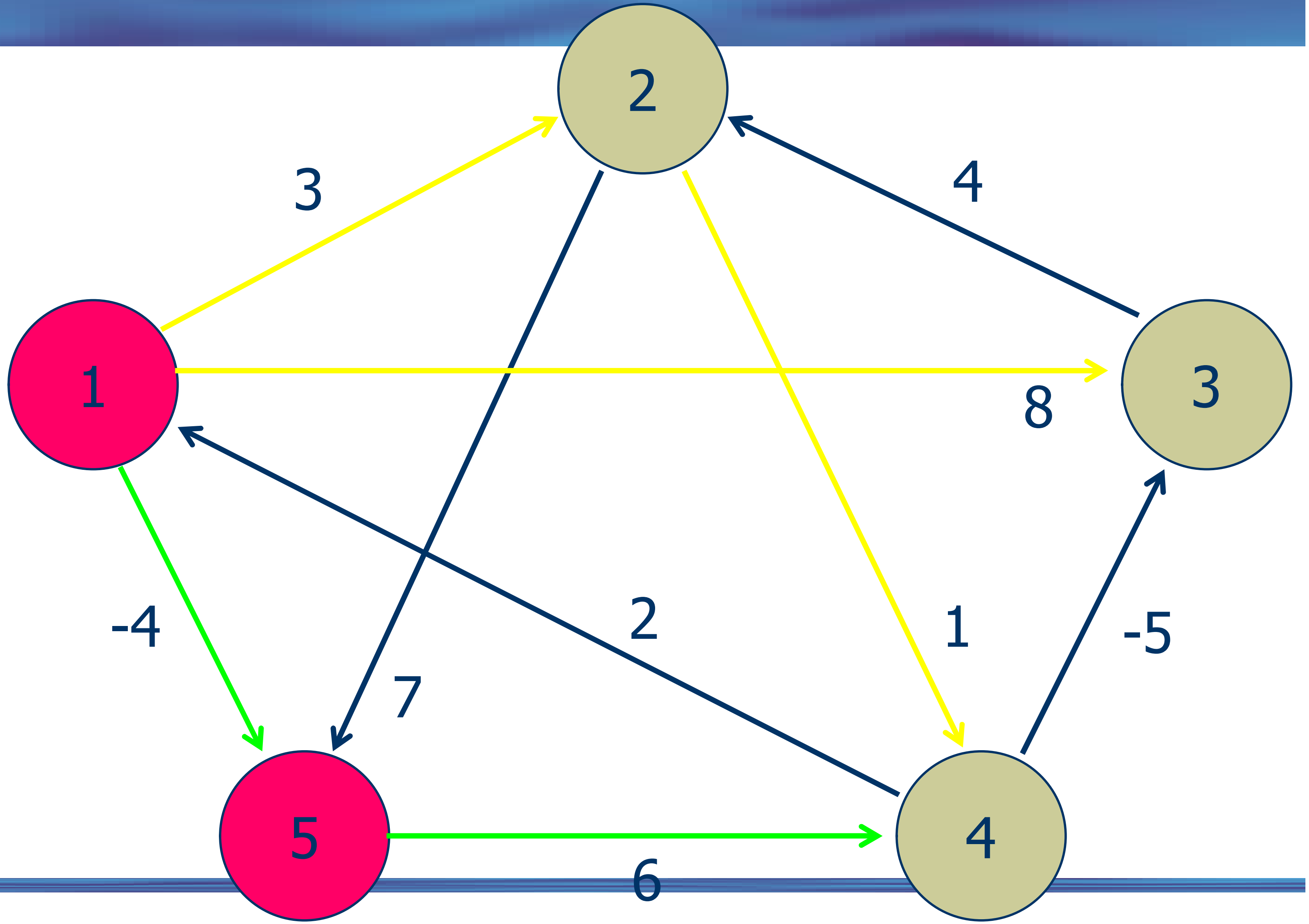
matrix multiplication (example)

$L_{14}^{(2)}$



matrix multiplication (example)

$L_{14}^{(2)}$



matrix multiplication (example)

$$l_{14}^{(2)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \end{pmatrix} \bullet \begin{pmatrix} \infty \\ 1 \\ \infty \\ 0 \\ 6 \end{pmatrix}$$

$$= \min(\infty, 4, \infty, \infty, 2)$$

$$= 2$$

Relation to matrix multiplication

$C = A \cdot B$

$\Rightarrow \qquad c_{ij} = \Sigma_{k=1}^{n} \quad a_{ik} \quad \cdot \; b_{kj}$

compare:

$L^{(m)} = L^{(m-1)} \, . \, W$

$\Rightarrow \qquad l_{ij}^{(m)} = \min_{1 \leq k \leq n} \{l_{ik}^{(m-1)} + w_{kj}\}$

Relation to matrix multiplication

```
MATRIX-MULTIPLY(A, B)
n ← rows[A]
let C be an n × n matrix
for i ← 1 to n
    do for j ← 1 to n
        do c_ij ← 0
            for k ← 1 to n
                do c_ij ← c_ij + a_ik · b_kj
return C
```

Matrix multiplication (pseudo-code)

EXTEND-SHORTEST-PATHS(L, W)

$n \leftarrow rows[L]$
let $L' = (l'_{ij})$ be an $n \times n$ matrix
for $i \leftarrow 1$ **to** n
    do for $j \leftarrow 1$ **to** n
        do $l'_{ij} \leftarrow \infty$
            for $k \leftarrow 1$ **to** n
                do $l'_{ij} \leftarrow \min(l'_{ij}, l_{ik} + w_{kj})$
return L'

Matrix multiplication

Compute the sequence of n-1 matrices:

$$L^{(1)} = L^{(0)} \cdot W = W,$$

$$L^{(2)} = L^{(1)} \cdot W = W^2,$$

$$L^{(3)} = L^{(2)} \cdot W = W^3,$$

$$\ldots$$

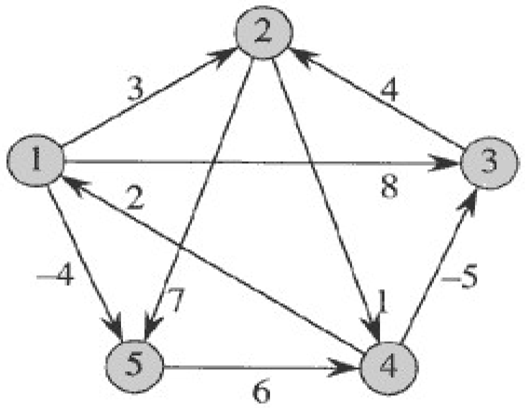
$$L^{(n-1)} = L^{(n-2)} \cdot W = W^{n-1}$$

Matrix multiplication (pseudo-code)

```
SLOW-ALL-PAIRS-SHORTEST-PATHS(W)
n ← rows[W]
L^(1) ← W
for m ← 2 to n − 1
        do L^(m) ← EXTEND-SHORTEST-PATHS(L^(m−1), W)
return L^(n−1)
```

How much space needed? Can you improve?

Matrix multiplication(example)



$$L^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad L^{(2)} = \begin{pmatrix} 0 & 3 & 8 & 2 & -4 \\ 3 & 0 & -4 & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & \infty & 1 & 6 & 0 \end{pmatrix}$$

$$L^{(3)} = \begin{pmatrix} 0 & 3 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \quad L^{(4)} = \begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

Figure 25.1 A directed graph and the sequence of matrices $L^{(m)}$ computed by SLOW-ALL-PAIRS-SHORTEST-PATHS. The reader may verify that $L^{(5)} = L^{(4)} \cdot W$ is equal to $L^{(4)}$, and thus $L^{(m)} = L^{(4)}$ for all $m \geq 4$.

Matrix multiplication (running time)

- $O(n^4)$

Improving the running time:

- compute not all $L^{(m)}$ matrices

  interested only in $L^{(n-1)}$, which is equal to $L^{(m)}$ for all integers $m \geq n-1$

Improving the running time

Compute the sequence

$L^{(1)} = W,$

$L^{(2)} = W^2 = W \cdot W,$

$L^{(4)} = W^4 = W^2 \cdot W^2,$

$L^{(8)} = W^8 = W^4 \cdot W^4$

...

$$L^{(2^{\lceil \log(n-1) \rceil})} = W^{(2^{\lceil \log(n-1) \rceil})} = W^{2^{\lceil \log(n-1) \rceil - 1}} \bullet W^{2^{\lceil \log(n-1) \rceil - 1}}$$

We need only $\lceil \log(n-1) \rceil$ matrix products

- $O(n^3 \log n)$

Improving running time

FASTER-ALL-PAIRS-SHORTEST-PATHS(W)

$n \leftarrow rows[W]$

$L^{(1)} \leftarrow W$

$m \leftarrow 1$

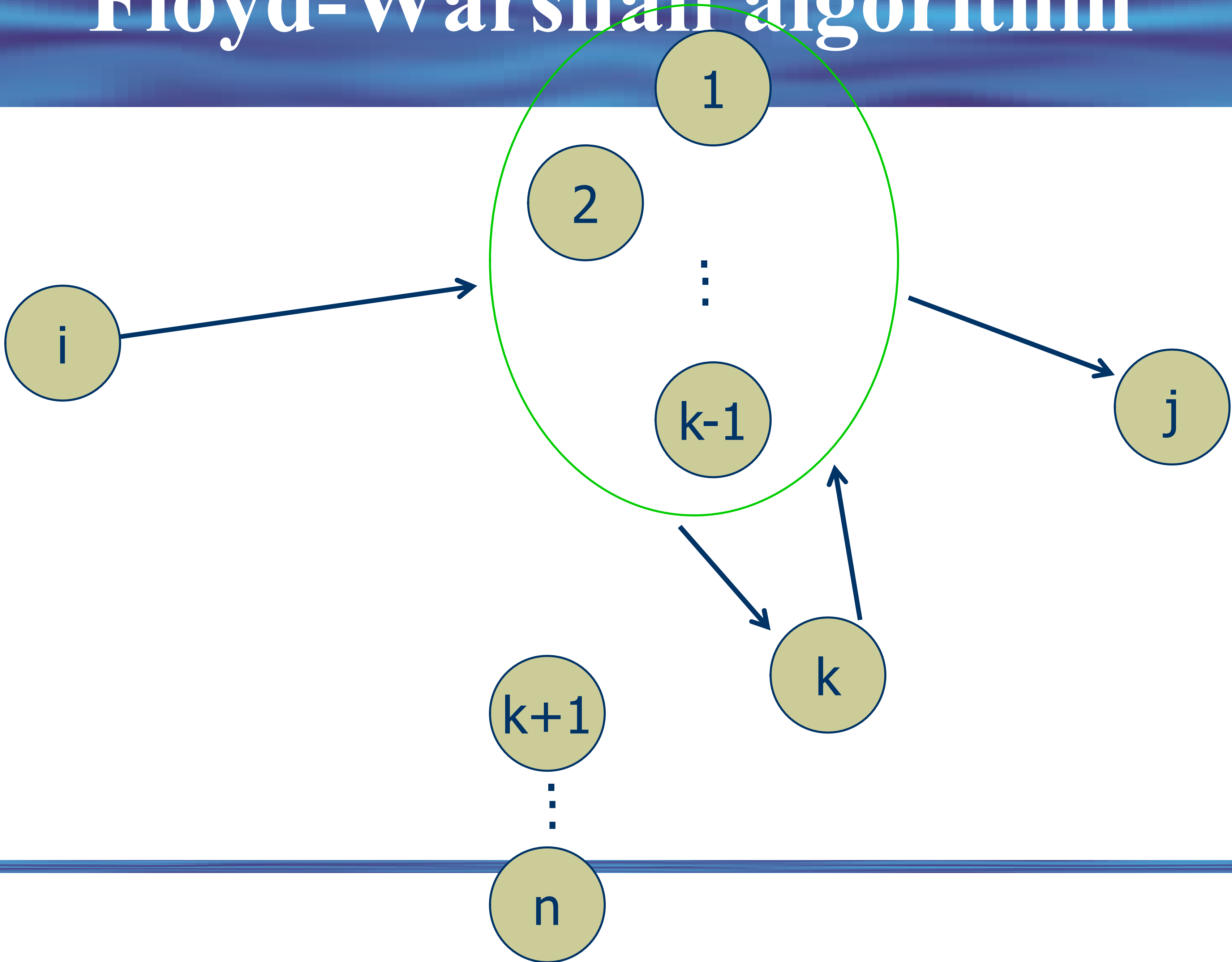
while $m < n - 1$

  do $L^{(2m)} \leftarrow$ EXTEND-SHORTEST-PATHS$(L^{(m)}, L^{(m)})$

    $m \leftarrow 2m$

return $L^{(m)}$

Floyd-Warshall algorithm



Floyd-Warshall algorithm

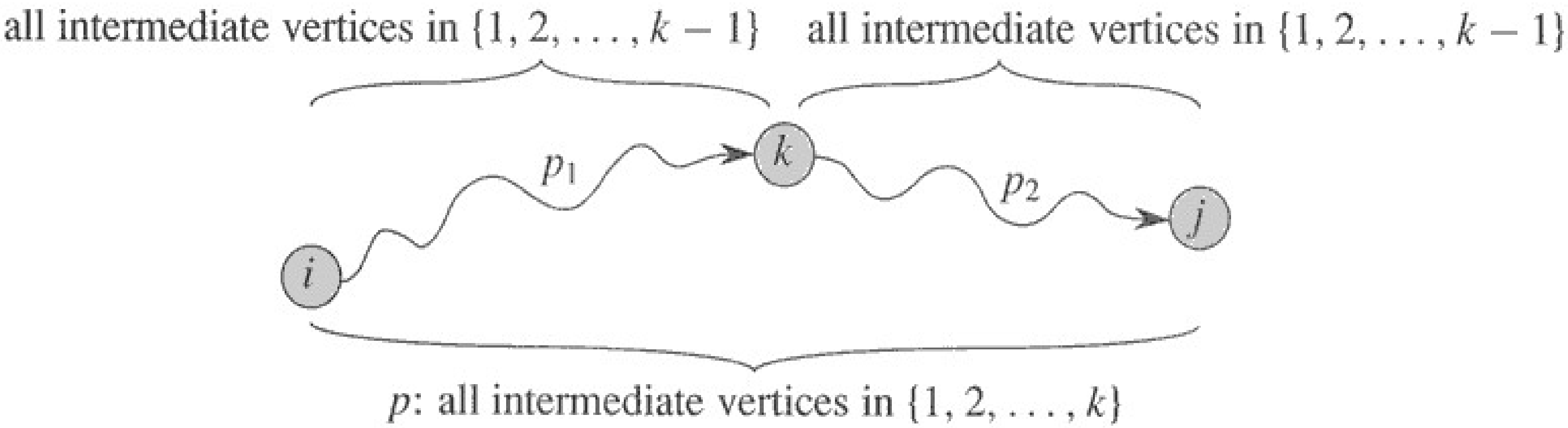


Figure 25.3 Path p is a shortest path from vertex i to vertex j, and k is the highest-numbered intermediate vertex of p. Path p_1, the portion of path p from vertex i to vertex k, has all intermediate vertices in the set $\{1, 2, \ldots, k-1\}$. The same holds for path p_2 from vertex k to vertex j.

Floyd-Warshall algorithm

$d_{ij}^{(k)}$:shortest path weight from i to j with intermediate vertices in the set {1,2,…,k}

$d_{ij}^{(0)} = w_{ij}$

(no intermediate vertices at all)

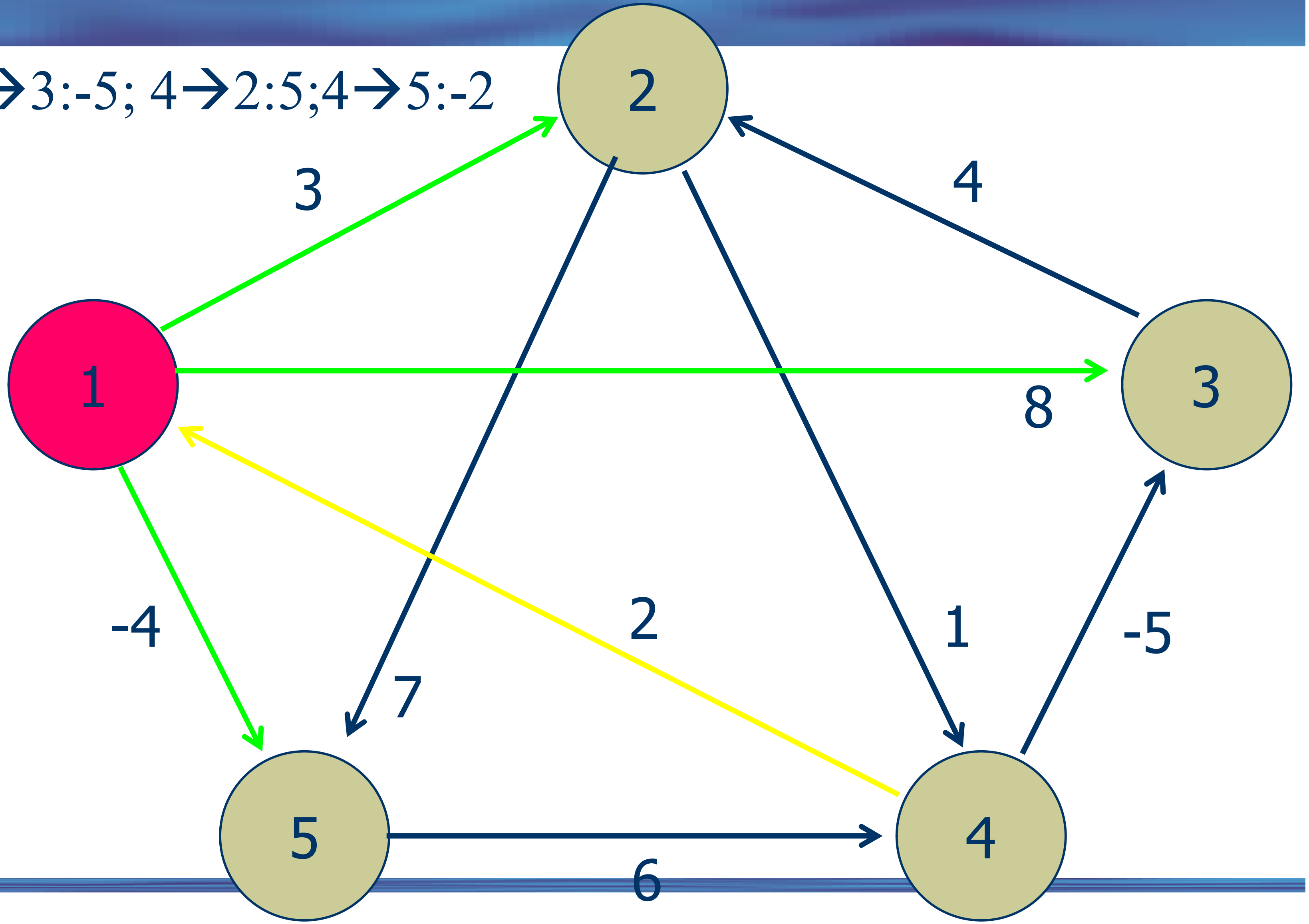
$d_{ij}^{(k)} = \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}) \quad \text{if } k \geq 1$

Result: $D^{(n)} = (d_{ij}^{(n)}) = d(i,j)$

(because all intermediate vertices are in the set {1, 2, …, n})

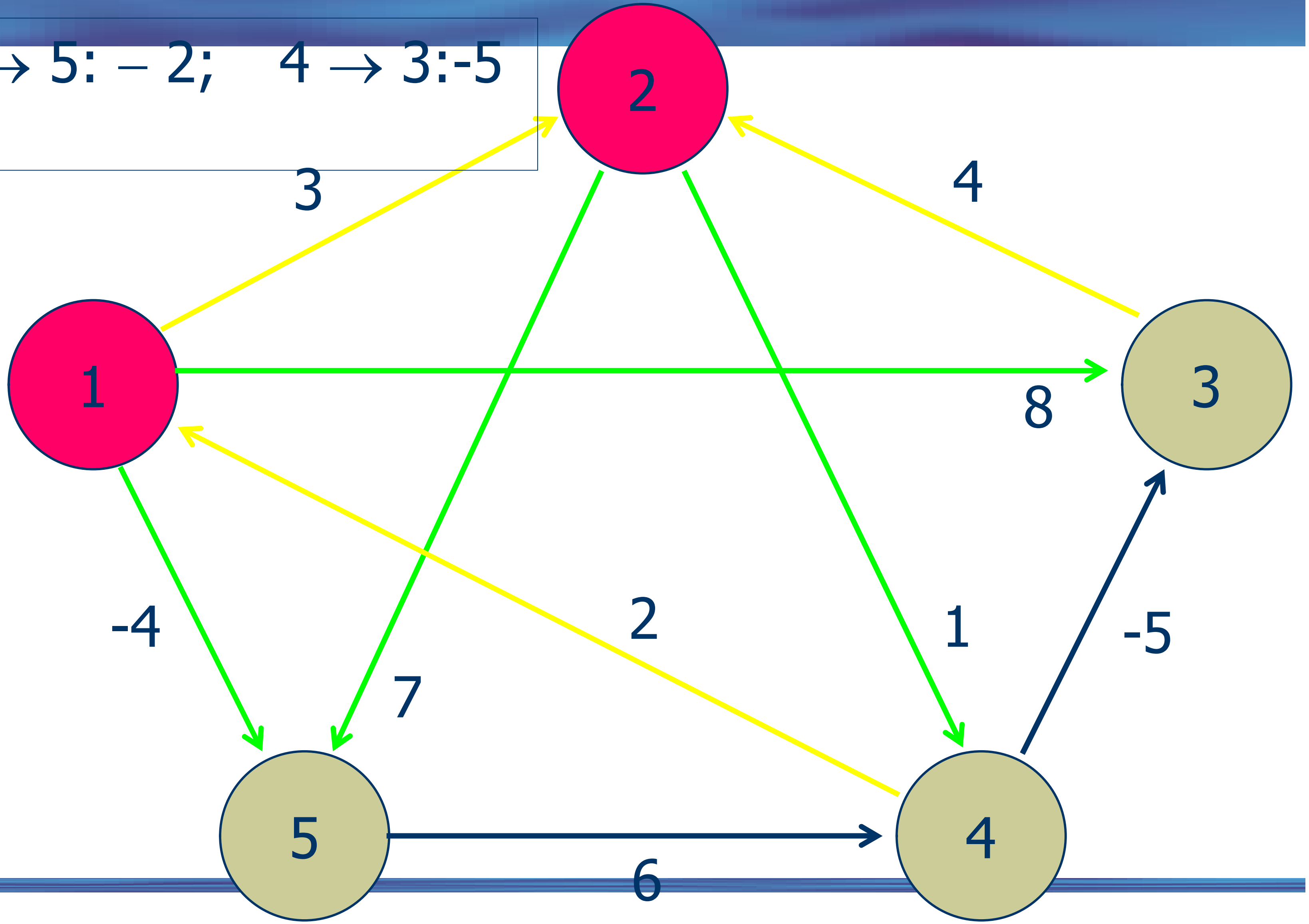
Floyd-Warshall algorithm

k=1: 4→3:-5; 4→2:5;4→5:-2



Floyd-Warshall algorithm

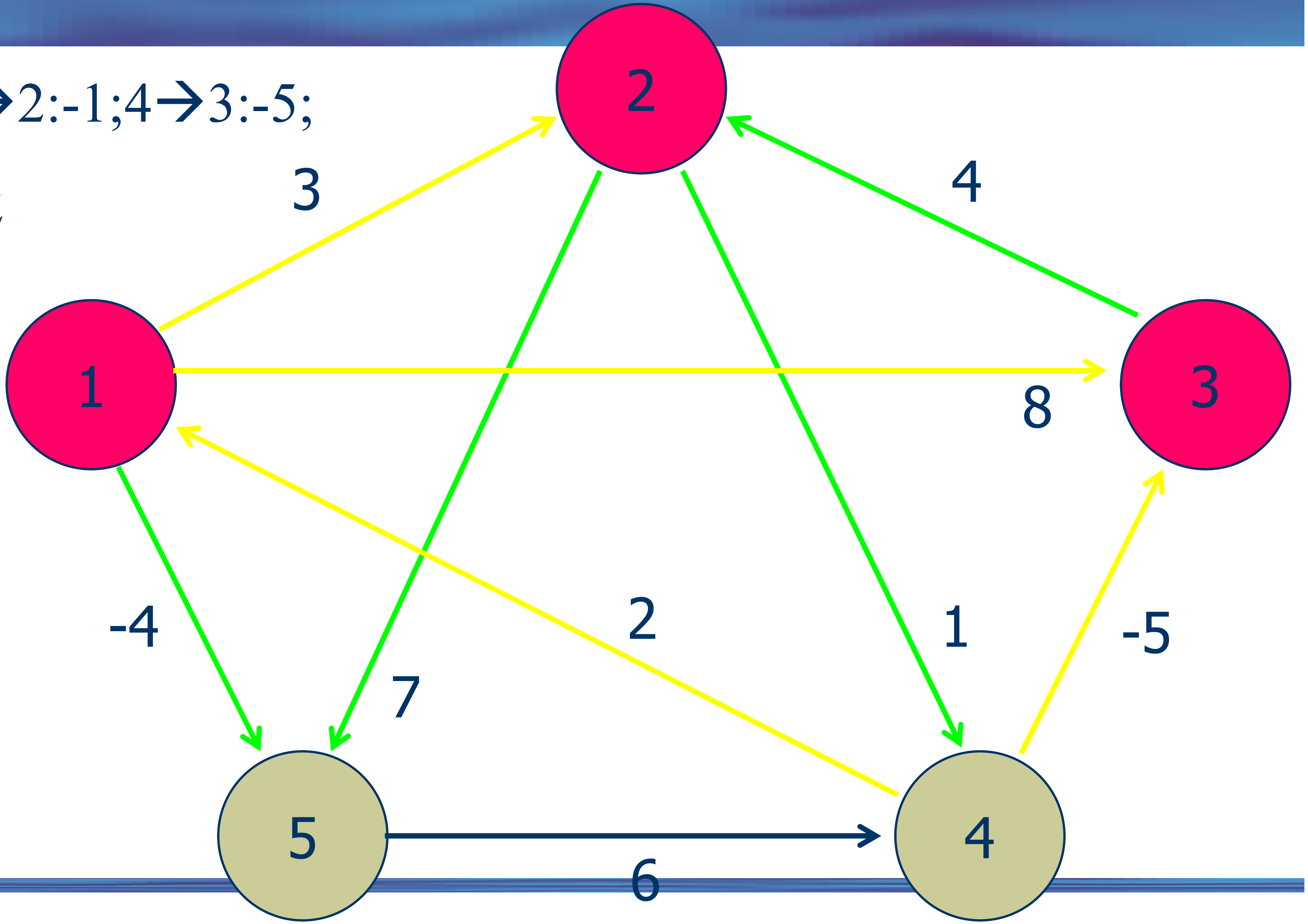
k=2: 4 → 5: – 2;   4 → 3:-5
4→2:5



Floyd-Warshall algorithm

k=3:4→2:-1;4→3:-5;

4→5:-2



Floyd-Warhsall algorithm (pseudo-code)

FLOYD-WARSHALL(W)

$n \leftarrow rows[W]$
$D^{(0)} \leftarrow W$
for $k \leftarrow 1$ **to** n
    do for $i \leftarrow 1$ **to** n
        do for $j \leftarrow 1$ **to** n
            do $d_{ij}^{(k)} \leftarrow \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right)$
return $D^{(n)}$

Notice that:

$$d_{ik}^{k} = \min\{d_{ik}^{k-1}, d_{ik}^{k-1} + d_{kk}^{k-1}\} = d_{ik}^{k-1}$$

$$d_{kj}^{k} = \min\{d_{kj}^{k-1}, d_{kk}^{k-1} + d_{kj}^{k-1}\} = d_{kj}^{k-1}$$

Floyd-Warhsall algorithm (less space)

FLOYD-WARSHALL$'(W)$

$n \leftarrow rows[W]$
$D \leftarrow W$
for $k \leftarrow 1$ **to** n
  do for $i \leftarrow 1$ **to** n
    do for $j \leftarrow 1$ **to** n
      do $d_{ij} \leftarrow \min(d_{ij}, d_{ik} + d_{kj})$
return D

Constructing a shortest path

- For k=0

$$\pi_{ij}^{(0)} = \begin{cases} NIL & if\ i = j \text{ or } w_{ij} = \infty \\ i & \text{if } i \neq j \text{ and } w_{ij} < \infty \end{cases}$$

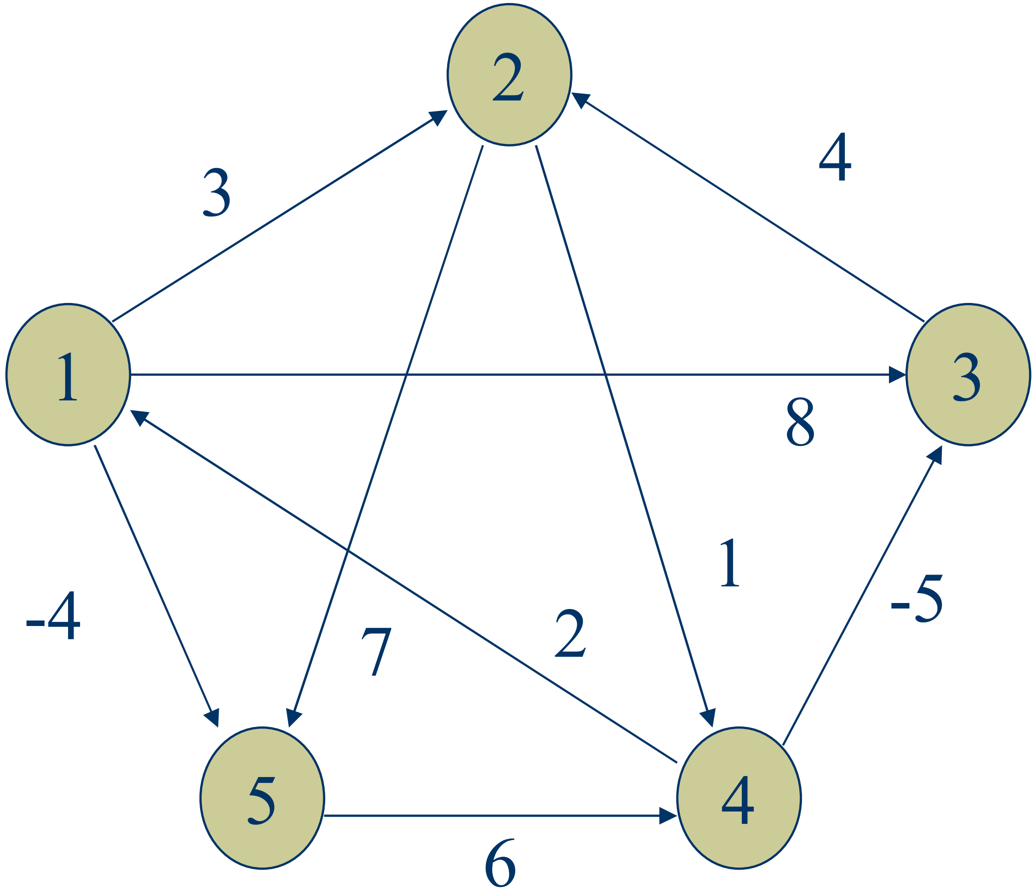
- For k>=1

$$\pi_{ij}^{(k)} = \begin{cases} \pi_{ij}^{(k-1)} & \text{if } d_{ij}^{(k-1)} \leq d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \\ \pi_{kj}^{(k-1)} & \text{if } d_{ij}^{(k-1)} > d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \end{cases},$$

Print all-pairs shortest paths

```
PRINT-ALL-PAIRS-SHORTEST-PATH(Π, i, j)
if i = j
    then print i
    else  if π_ij = NIL
              then print "no path from" i "to" j "exists"
              else  PRINT-ALL-PAIRS-SHORTEST-PATH(Π, i, π_ij)
                    print j
```

Example:



$$D^{(0)}=\begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

$$\prod^{(0)}=\begin{pmatrix} NIL & 1 & 1 & NIL & 1 \\ NIL & NIL & NIL & 2 & 2 \\ NIL & 3 & NIL & NIL & NIL \\ 4 & NIL & 4 & NIL & NIL \\ NIL & NIL & NIL & 5 & NIL \end{pmatrix}$$

$$D^{(1)}=\begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

$$\prod^{(1)}=\begin{pmatrix} NIL & 1 & 1 & NIL & 1 \\ NIL & NIL & NIL & 2 & 2 \\ NIL & 3 & NIL & NIL & NIL \\ 4 & 1 & 4 & NIL & 1 \\ NIL & NIL & NIL & 5 & NIL \end{pmatrix}$$

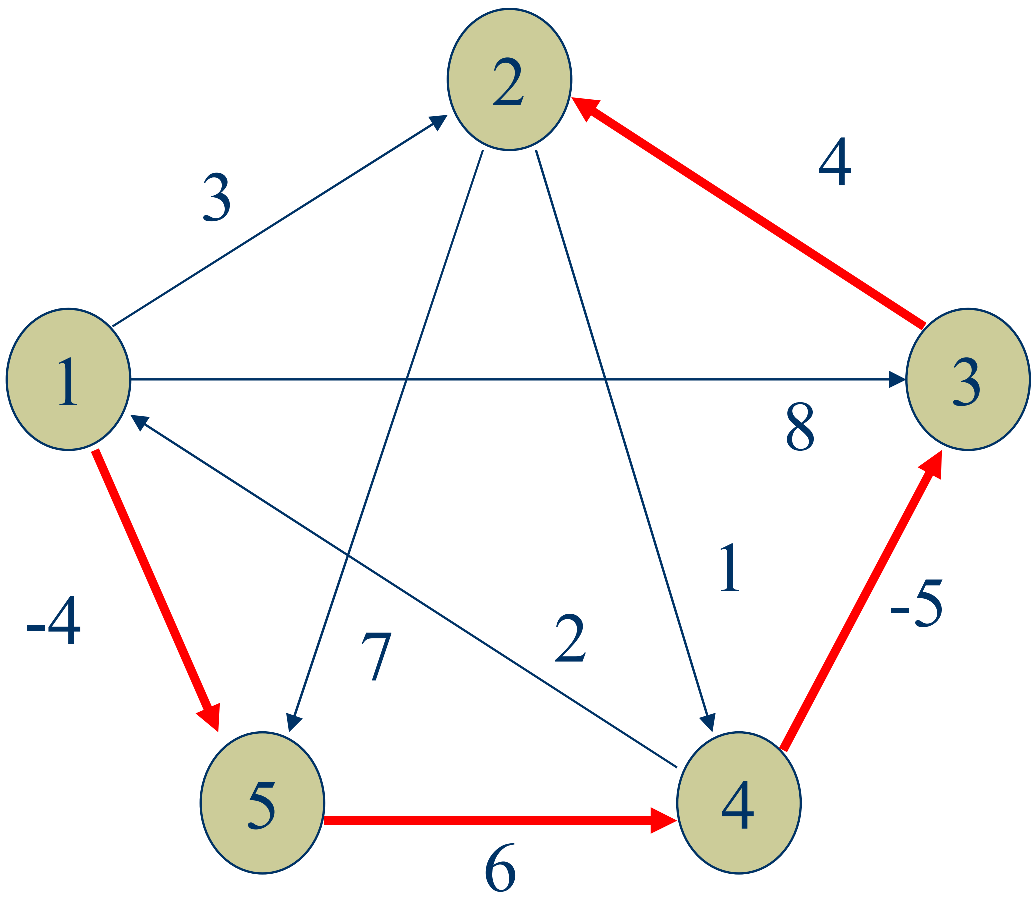
$$D^{(2)}=\begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \qquad \prod^{(2)}=\begin{pmatrix} NIL & 1 & 1 & 2 & 1 \\ NIL & NIL & NIL & 2 & 2 \\ NIL & 3 & NIL & 2 & 2 \\ 4 & 1 & 4 & NIL & 1 \\ NIL & NIL & NIL & 5 & NIL \end{pmatrix}$$

$$D^{(3)}=\begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \qquad \prod^{(3)}=\begin{pmatrix} NIL & 1 & 1 & 2 & 1 \\ NIL & NIL & NIL & 2 & 2 \\ NIL & 3 & NIL & 2 & 2 \\ 4 & 3 & 4 & NIL & 1 \\ NIL & NIL & NIL & 5 & NIL \end{pmatrix}$$

$$D^{(4)}=\begin{pmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \qquad \prod^{(4)}=\begin{pmatrix} NIL & 1 & 4 & 2 & 1 \\ 4 & NIL & 4 & 2 & 1 \\ 4 & 3 & NIL & 2 & 1 \\ 4 & 3 & 4 & NIL & 1 \\ 4 & 3 & 4 & 5 & NIL \end{pmatrix}$$

$$D^{(5)}=\begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \qquad \prod^{(5)}=\begin{pmatrix} NIL & 3 & 4 & 5 & 1 \\ 4 & NIL & 4 & 2 & 1 \\ 4 & 3 & NIL & 2 & 1 \\ 4 & 3 & 4 & NIL & 1 \\ 4 & 3 & 4 & 5 & NIL \end{pmatrix}$$

Shortest path from 1 to 2 in $\Pi^{(5)}$

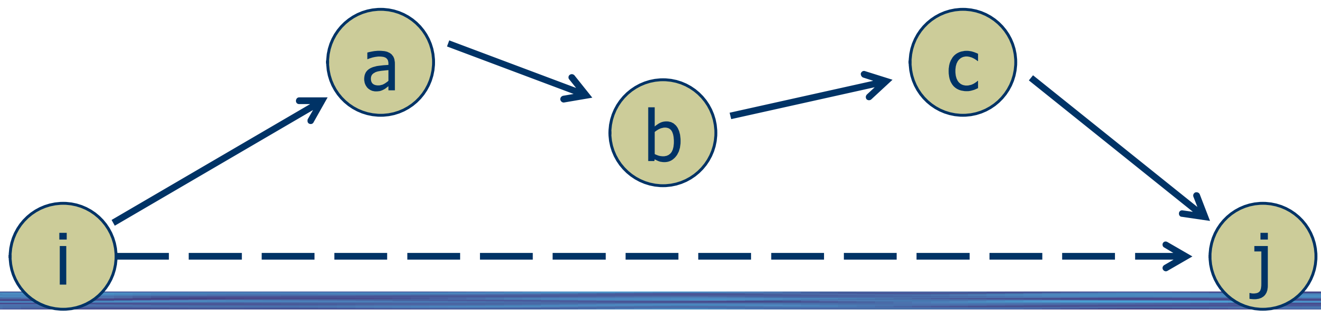


Transitive closure (the problem)

- Find out *whether* there is a path from i to j and compute

$$G^* = (V,E^*),$$

where $E^* = \{(i,j)$: there is a path from i to j in G$\}$



Transitive closure

- One way:

  set $w_{ij} = 1$ and

  run the Floyd-Warshall algorithm

- *running time* $O(n^3)$

transitive closure

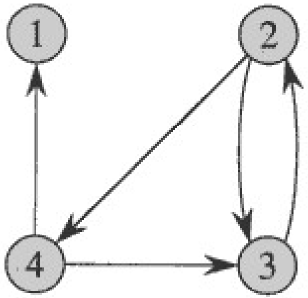
- Another way:
  substitute "+" and "min" by AND and OR
  in Floyd's algorithm

- *running time* $O(n^3)$

Transitive closure (pseudo-code)

```
TRANSITIVE-CLOSURE(G)
n ← |V[G]|
for i ← 1 to n
    do for j ← 1 to n
        do if i = j or (i, j) ∈ E[G]
            then t_ij^(0) ← 1
            else t_ij^(0) ← 0
for k ← 1 to n
    do for i ← 1 to n
        do for j ← 1 to n
            do t_ij^(k) ← t_ij^(k−1) ∨ (t_ik^(k−1) ∧ t_kj^(k−1))
return T^(n)
```

Transitive closure



$$T^{(0)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix} \quad T^{(1)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix} \quad T^{(2)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

$$T^{(3)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \quad T^{(4)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

Figure 25.5 A directed graph and the matrices $T^{(k)}$ computed by the transitive-closure algorithm.

Table of running times

algorithm	running time
Dijkstra‘s	$O(n^2 \log n + nm)$
Bellman-Ford	$O(n^2 m)$
matrix multiplication	$O(n^4)$
improved matrix mult.	$O(n^3 \log n)$
Floyd-Warshall	$O(n^3)$
Johnson‘s	$O(n^2 \log n + nm)$
Transitive closure	$O(n^3)$

Homework

- 25.2-1（not compulsory）
- 25.2-5
- 25.2-8