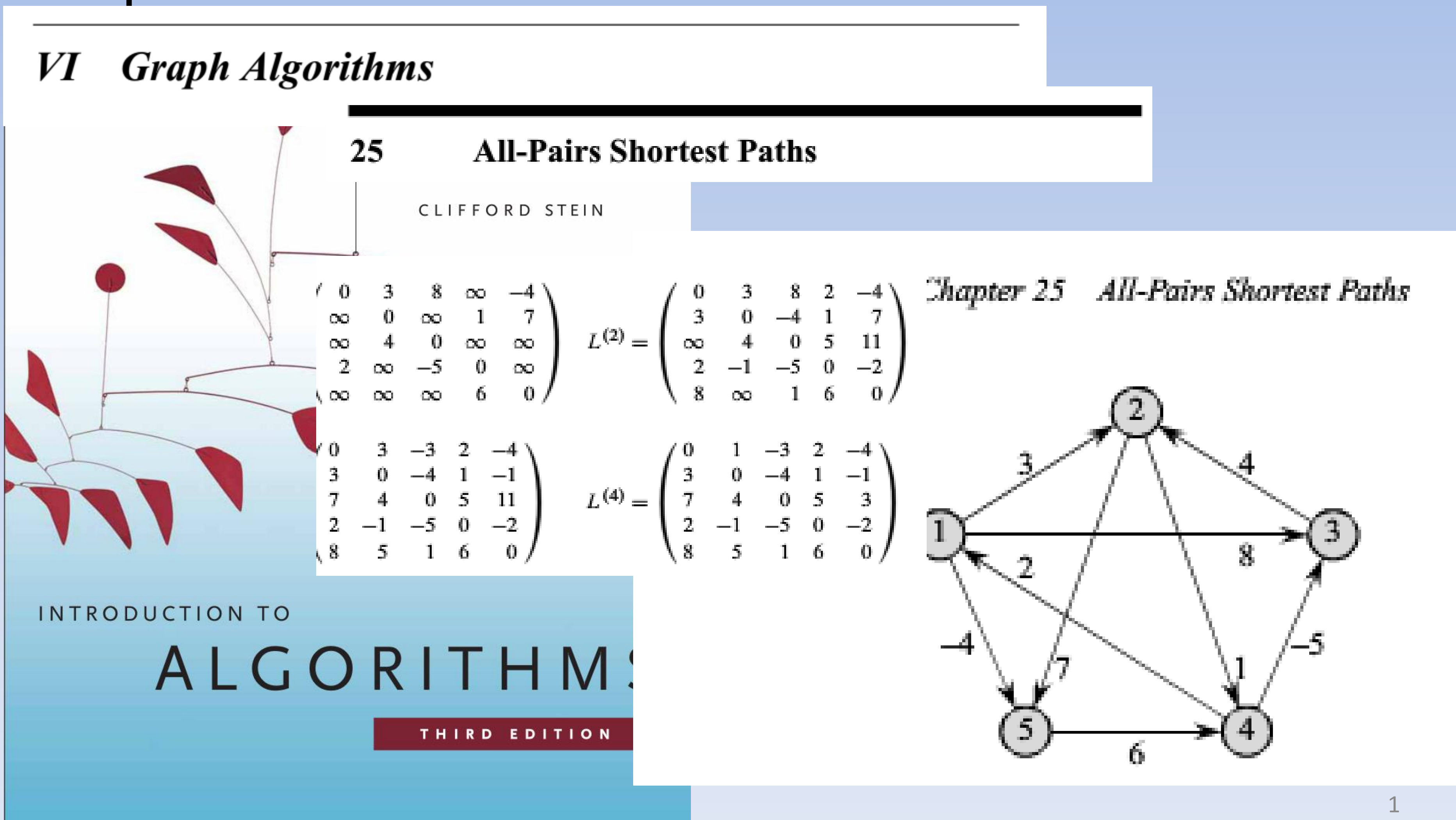
Design and Analysis of Algorithms

Section VI : Graph Algorithms

Chapter 25: All-Pairs Shortest Paths



All-Pairs Shortest Paths Problem

- Given:
  - Weighted, Directed Graph G=(V, E)
  - Weight Function w: E -> $\mathbb{R}$
    - Edges -> Real-Valued Weights
- Weight of path $P=<v_0, v_1, \ldots, v_k>$
  - $w(p) = \sum w(v_{i-1}, v_i)$
- Shortest-Path Weight $\delta(u,v)$ is the minimum weight path w(p) that goes from u to v, otherwise ∞
- The shortest path from u to v is any path p with a weight of $\delta(u,v)$
- **ALL-PAIRS SHORTEST PATHS**
  - For all pairs of vertices $u,v \in V$, $\delta(u,v)$

All-Pairs Shortest Paths

- Adjacency-List Representation
- Assume vertices are numbered 1, 2, … , |V|
- Input: n x n weight matrix W of an n-vertex directed graph G=(V,E)
- $W = (w_{ij})$, w_{ij}=
  - 0, if $i=j$
  - the weight of directed edge (i, j), if $i \neq j$ & $(i,j) \in E$
  - ∞ if $i \neq j$ & $(i,j) \notin E$

All-Pairs Shortest Paths Output

- n x n matrix D = (d_{ij})
  - d_{ij}= weight of shortest path from vertex i to vertex j.
  - $\delta(i,j)$
- Predecessor Matrix Π = (π_{ij})
- π_{ij} =
  - NIL, if i=j or no path from i to j.
  - predecessor to j on some shortest path from i to j.

Predecessor Subgraph

- $G_{\pi,i}=(V_{\pi,i}, E_{\pi,i})$
- $V_{\pi,i} = \{j \in V : \pi_{ij} \neq nil\} \cup \{i\}$
- $E_{\pi,i} = \{(\pi_{ij,} j) : j \in V_{\pi,I} - \{i\}$

Print-All-Pairs-Shortest-Path

PRINT-ALL-PAIRS-SHORTEST-PATH(Π, i, j)

if $i == j$
    print i
elseif π_{ij} == NIL
    print "no path from" i "to" j "exists"
else PRINT-ALL-PAIRS-SHORTEST-PATH(Π, i, π_{ij})
    print j

Dynamic Programming Steps

- Characterize the structure of optimal solution
- Recursively define the value of an optimal solution
- Compute the value of an optimal solution bottom-up.
- Construct the optimal solution from computed information.

Simple Recursive Solution

- Define $l_{ij}^{(m)}$ as the minimum weight of any path from vertex i to j that contains at most m edges.
- $l_{ij}^{(0)} =$
  - 0 if i=j
  - ∞ if i≠j

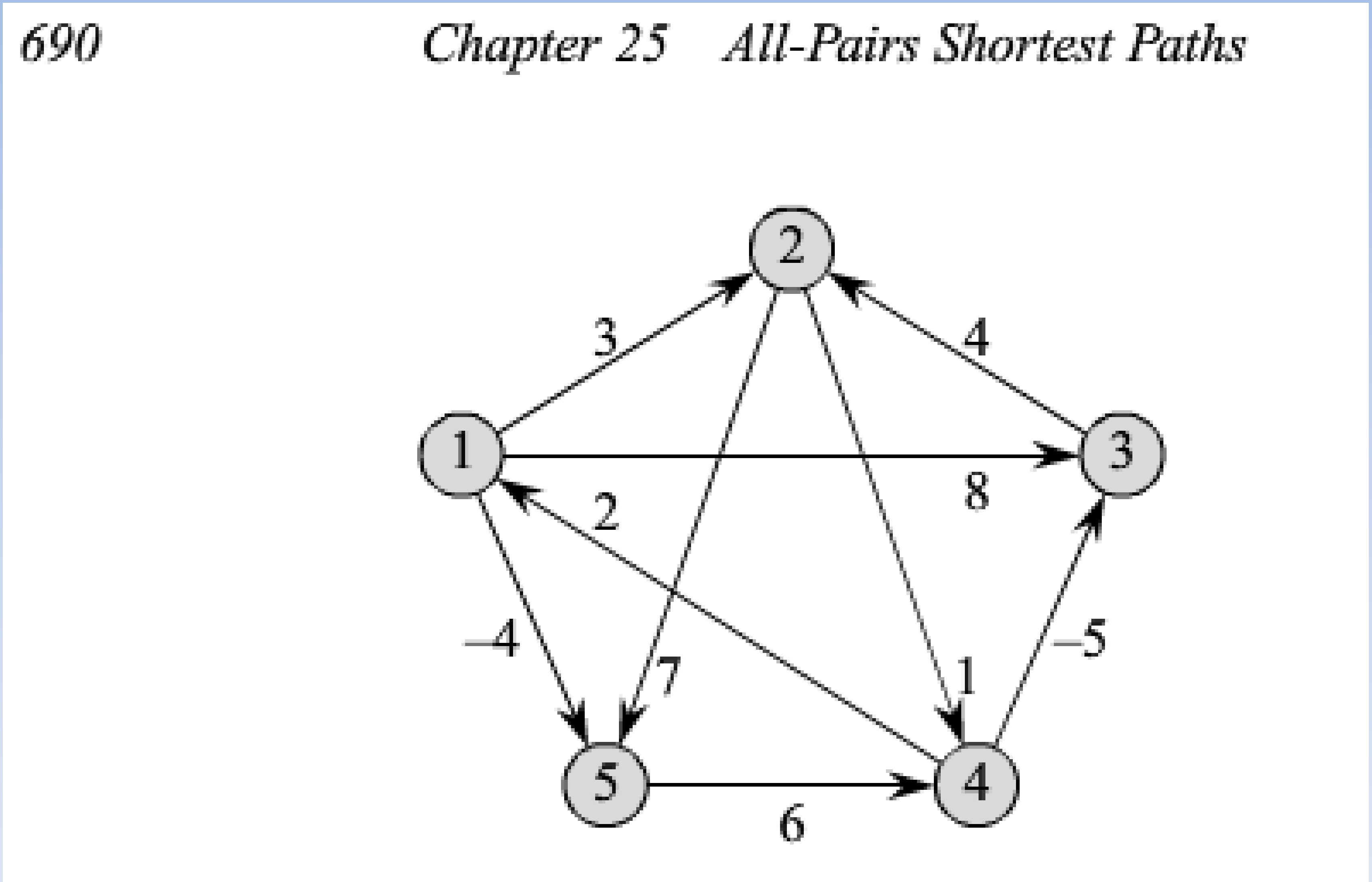
Single Source Relaxation Process

- Relax Edge (u, v) By:
  - Testing possible shortest path improvement to **v** by using current path to **u**
  - When improvements are possible update:
    - v.d: estimated shortest-path weight
    - v.π: v's parent

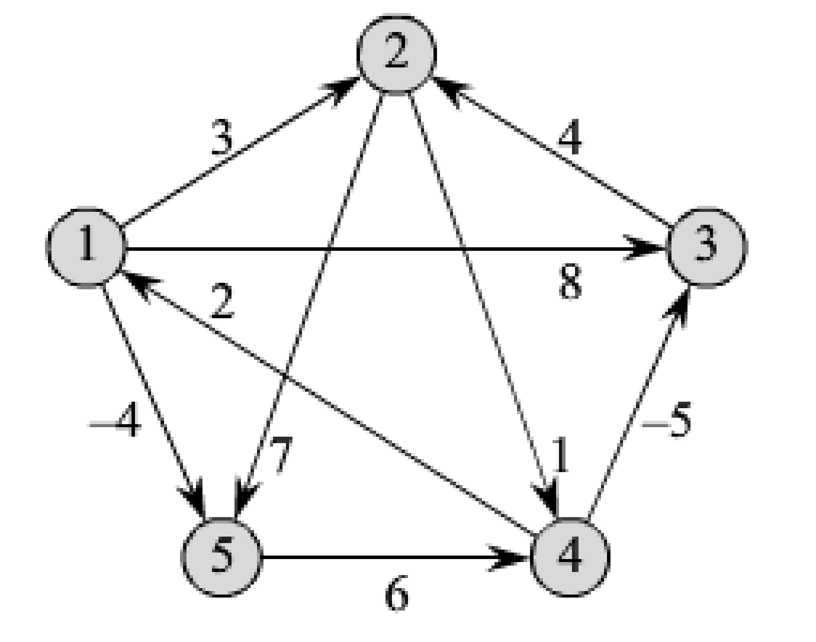
Recursive Solution

- Define $l_{ij}^{(m)}$ as the minimum weight of any path from vertex *i* to *j* that contains at most *m* edges.
- $l_{ij}^{(0)}$ =
  - 0 if i=j
  - ∞ if i≠j
- **Compute $l_{ij}^{(m)}$ as the minimum**
  - **$l_{ij}^{(m-1)}$ the minimum weight path from i to j with m-1 edges**
  - **$l_{ik}^{(m)} + w_{kj}$: for all possible k's.**

Example Graph

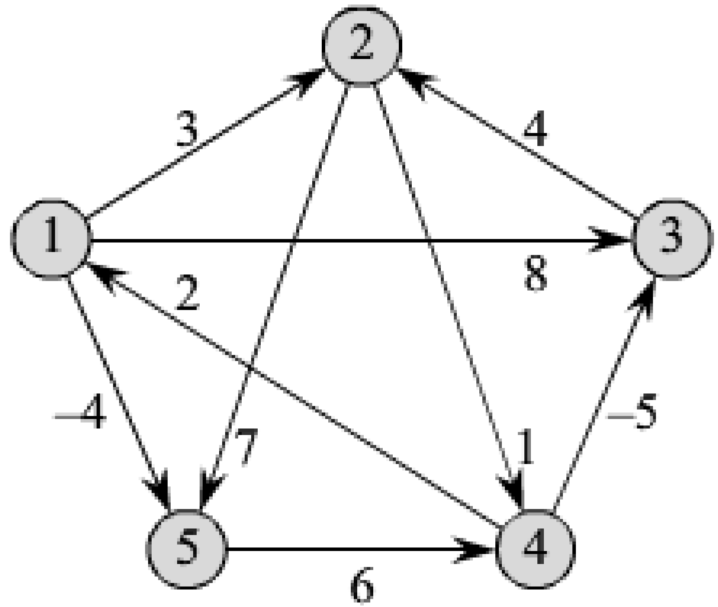


$$L^{(1)} \quad \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$



$L^{(1)}$

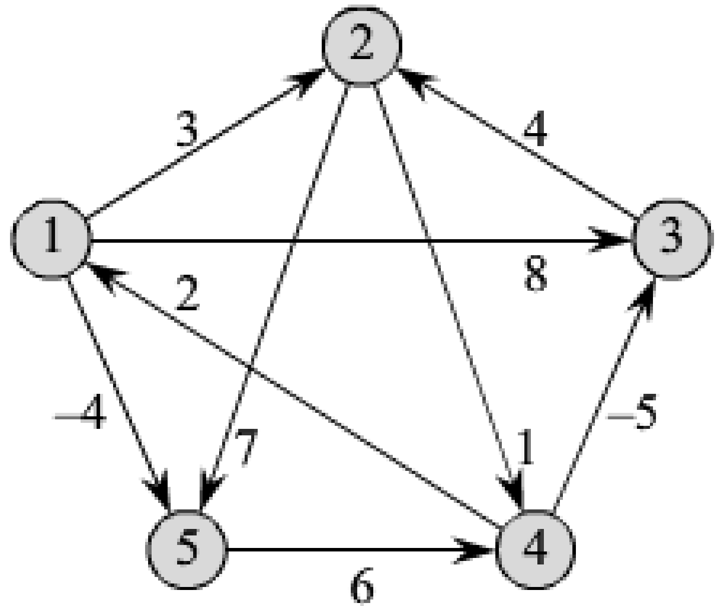
$$\left(\begin{array}{cc} 0 & 3 \\ \infty & 0 \\ \infty & 4 \\ 2 & \infty \\ \infty & \infty \end{array} \right.$$



- Shortest Path of length 1 weights =
  - (1, 2) w/ w(1,2) = 3

$L^{(1)}$

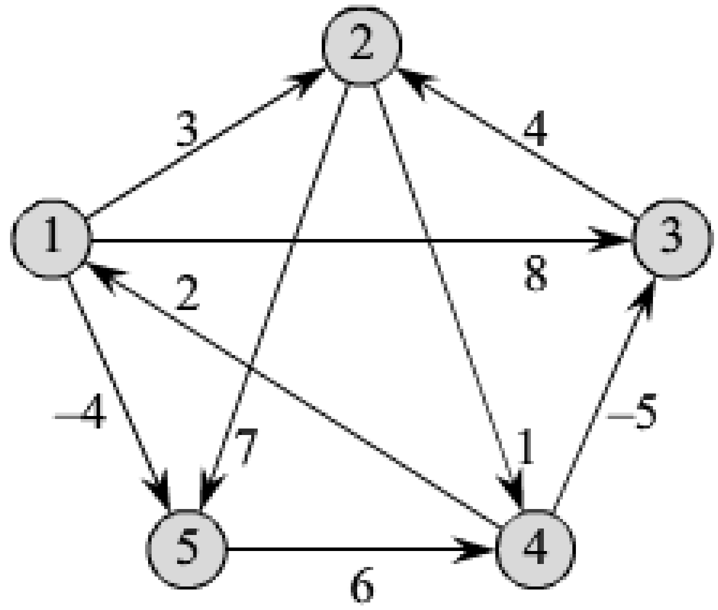
$$\left(\begin{matrix} 0 & 3 \\ \infty & 0 \\ \infty & 4 \\ 2 & \infty \\ \infty & \infty \end{matrix}\right.$$



- Shortest Path of length 1 weights =
  - (1, 2) w/ w(1,2) = 3
  - (4, 1) w/ w(4,1) = 2

$L^{(1)}$

$$\left(\begin{array}{cc} 0 & 3 \\ \infty & 0 \\ \infty & 4 \\ 2 & \infty \\ \infty & \infty \end{array}\right.$$

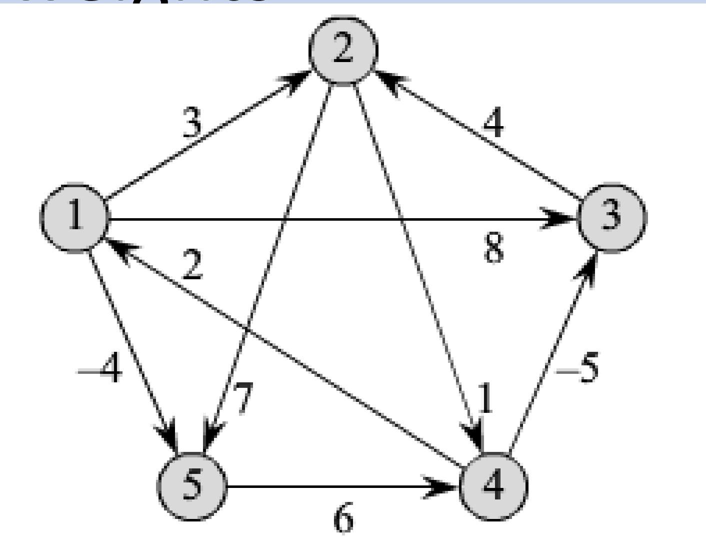


- Shortest Path of length 1 weights =
  - (1, 2) w/ w(1,2) = 3
  - (4, 1) w/ w(4,1) = 2
  - (3, 2) w/ w(3,2) = 4

$L^{(1)}$

$$\begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

- Shortest Path of length 1 weights =
  - (1, 2) w/ w(1,2) = 3
  - (4, 1) w/ w(4,1) = 2
  - (3, 2) w/ w(3,2) = 4



Floyd-Warshall

- Assume no negative-weight cycles
- Use a different dynamic-programming formulation

Shortest Path Structure: Intermediate Vertex

- $P = \langle v_1, v_2, \ldots, v_i \rangle$
  - Intermediate Vertex is any vertex $\{v_2, \ldots, v_{i-1}\}$
  - Does not include start/end v_1, v_i
- Given that we are using vertices {1, 2, …, n}
  - There is a linear ordering of our vertices
- Consider a subset of these vertices less than some value k.
- Explore paths using only intermediate vertices $v<k$
  - Iteratively Expand k
  - Construct paths using $v<k$ from those using $v<k-1$

Shortest Path Structure: Intermediate Vertex

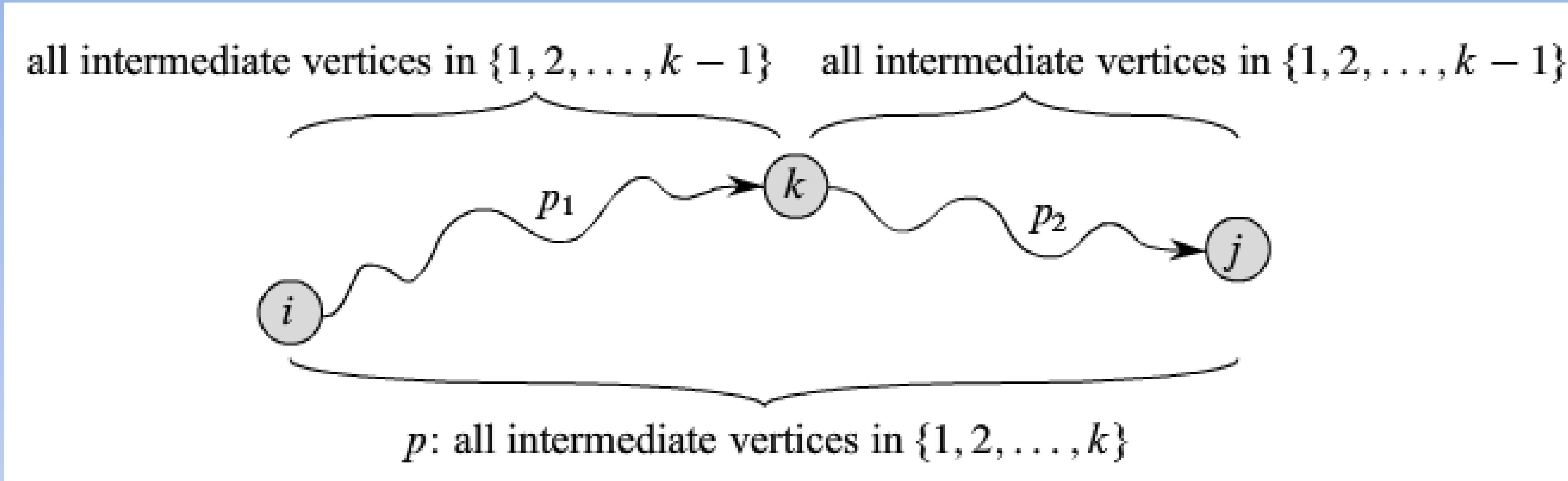
- Then consider a subset of vertices < k
  - {1, 2, 3, …, k}
- Then consider pairs of vertices i, j
  - consider all paths between i,j using only intermediate v in {1, 2, … , k}
  - Let p be minimum-weight path from previous set.
- Floyd-Warshall uses relationship between:
  - minimum-weight path using v<k-1
  - and that using v<k.

Shortest Path Structure
First Case

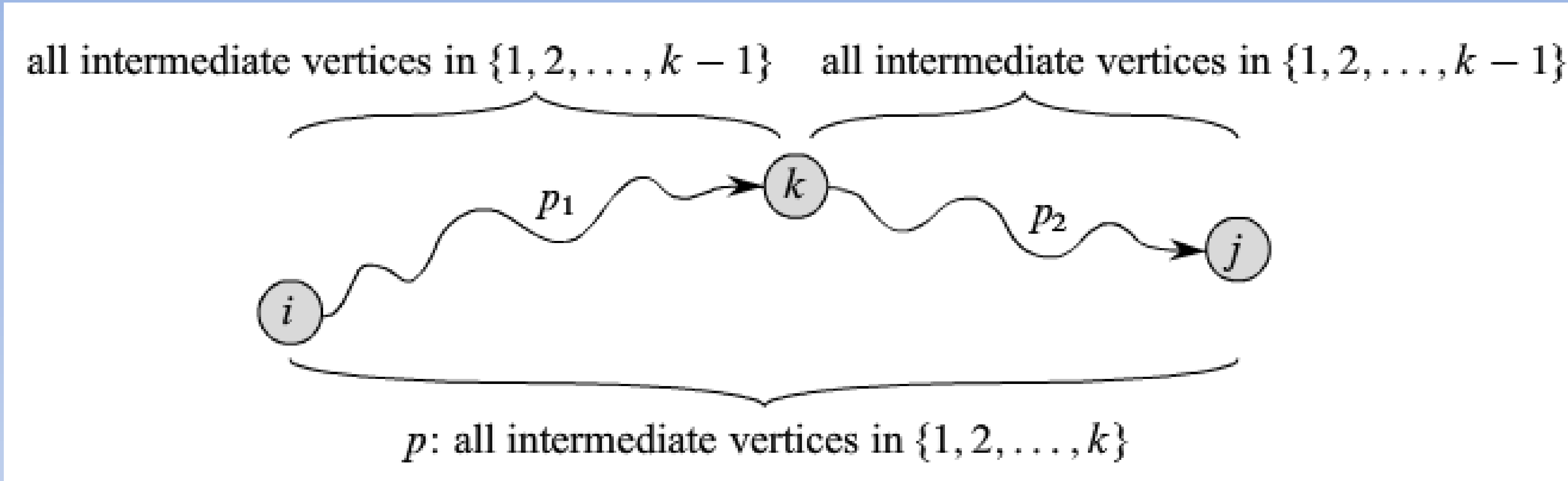
- Then consider:
  - subset of vertices v < k-1 = {1, 2, 3, …, k-1}
  - Let p be minimum-weight path between i,j with intermediate v only from {1, 2, 3, …, k-1}
- Then q the minimum path with intermediate v only in {1, 2, 3, …, k} has property:
  - k **is not** intermediate of q
    - Then the shortest path from k-1 case is same as case with k
    - Then all intermediate vertices must be only in {1, 2, …, k-1}

Shortest Path Structure Second Case

- k **is** intermediate of q (our new shortest path) then we can break it into two pieces
  - k is an intermediate node, and there are no cycles so it must break q into two parts: i –p_1-> k –p_2->j
  - Both p_1 and p_2 must have all of their vertices v in {1, 2, …, k-1}
  - p_1 and p_2 must both be shortest paths from there start and finish vertices using only v < k-1.



- Now we want to exploit this and develop recursive solution.



- Now we want to exploit this and develop recursive solution.
- Develop matrix:
  - $d_{ij}^{(k)}$ = the weight of a shortest path from i to j using only intermediate v<k
  - $d_{ij}^{(0)}$ = No intermediate v means edge from i to j.
  - $d_{ij}^{(0)} = w_{ij}$

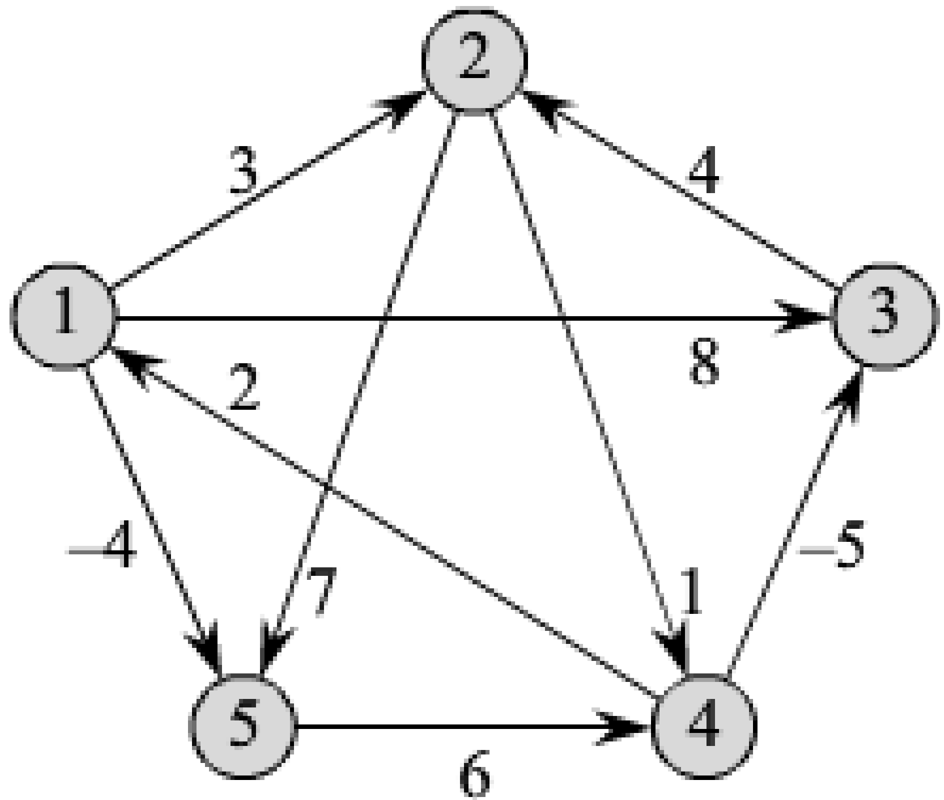
$d_{ij}^{(k)}$

$$d_{ij}^{(k)} = \begin{cases} w_{ij} & \text{if } k = 0 , \\ \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right) & \text{if } k \geq 1 . \end{cases} \tag{25.5}$$

$d_{ij}^{(k)}$

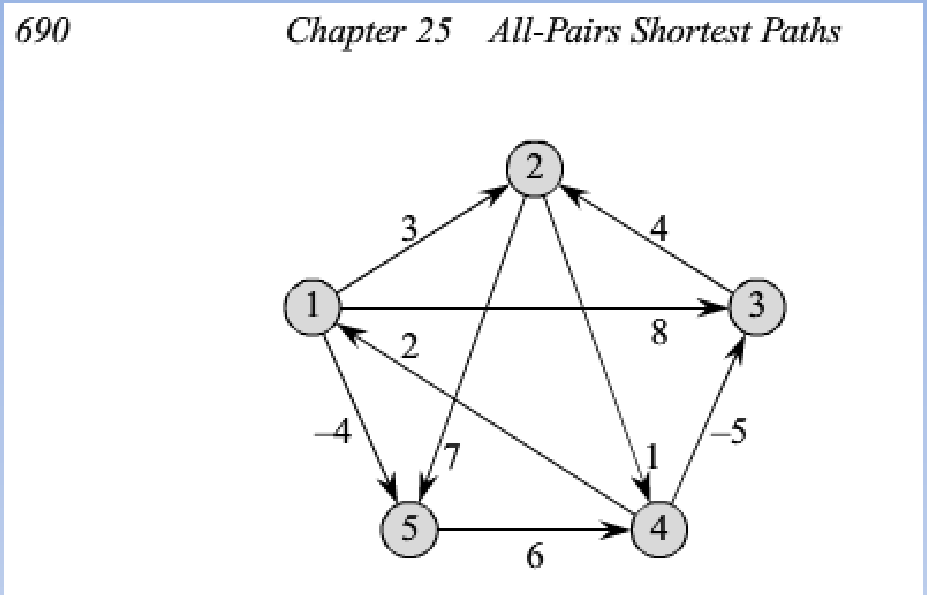
$$d_{ij}^{(k)} = \begin{cases} w_{ij} & \text{if } k = 0 , \\ \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right) & \text{if } k \geq 1 . \end{cases} \tag{25.5}$$

- $d_{ij}^{(n)}$ will have out shortest path info.



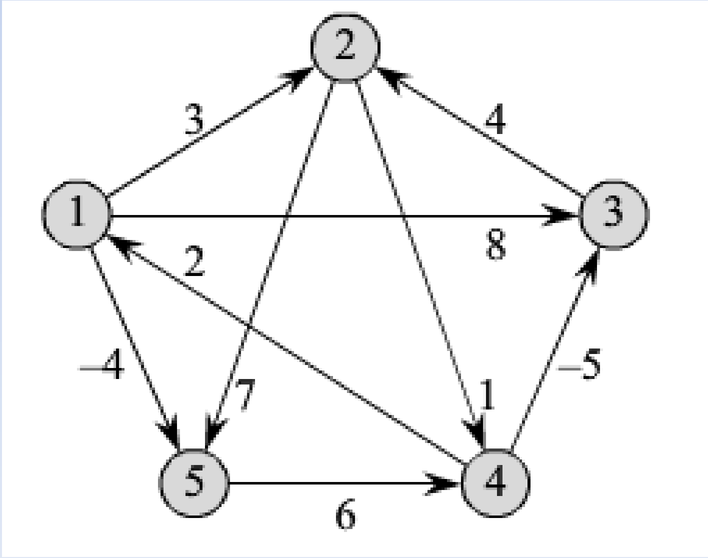
2
3
4
1
3
8
2
-4
7
1
-5
5
4
6

$D^{(0)}$



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

- Shortest Path with intermediates ≤ 0 =
  - (1, 2) w/ w(1,2) = 3
  - (4, 1) w/ w(4,1) = 2
  - (3, 2) w/ w(3,2) = 4

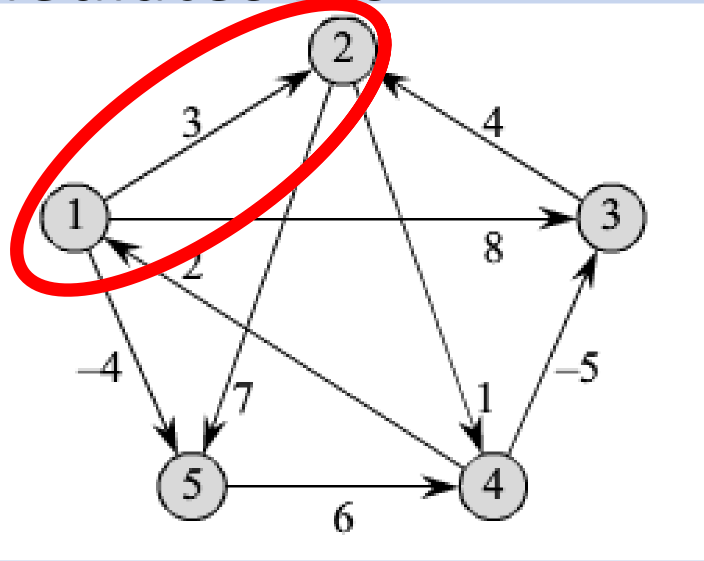


$\Pi^{(0)}$

$$\begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & \text{NIL} & 4 & \text{NIL} & \text{NIL} \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

- Shortest Path with intermediates ≤ 0 =

$$\begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

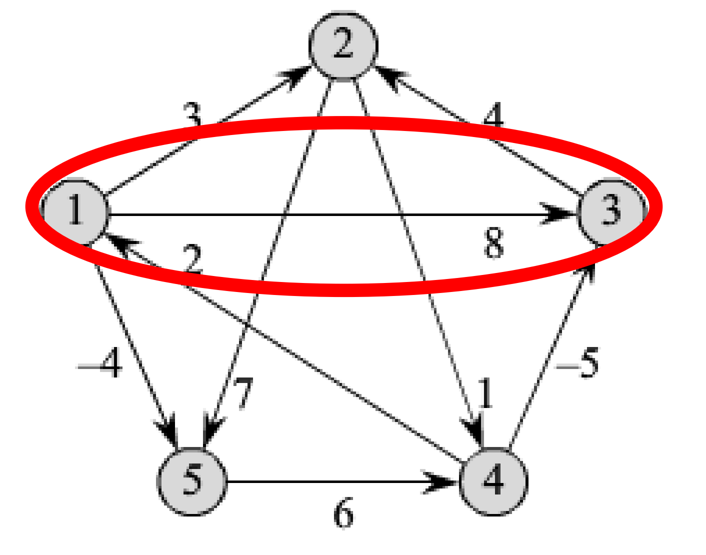


$\Pi^{(0)}$

$$\begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & \text{NIL} & 4 & \text{NIL} & \text{NIL} \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

- Shortest Path with intermediates ≤ 0 =

$$\begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

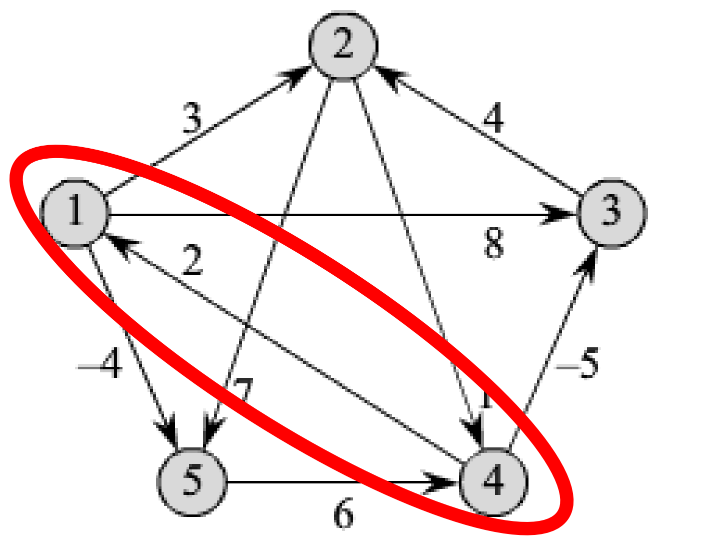


$\Pi^{(0)}$

$$\begin{pmatrix} \mathrm{NIL} & 1 & 1 & \mathrm{NIL} & 1 \\ \mathrm{NIL} & \mathrm{NIL} & \mathrm{NIL} & 2 & 2 \\ \mathrm{NIL} & 3 & \mathrm{NIL} & \mathrm{NIL} & \mathrm{NIL} \\ 4 & \mathrm{NIL} & 4 & \mathrm{NIL} & \mathrm{NIL} \\ \mathrm{NIL} & \mathrm{NIL} & \mathrm{NIL} & 5 & \mathrm{NIL} \end{pmatrix}$$

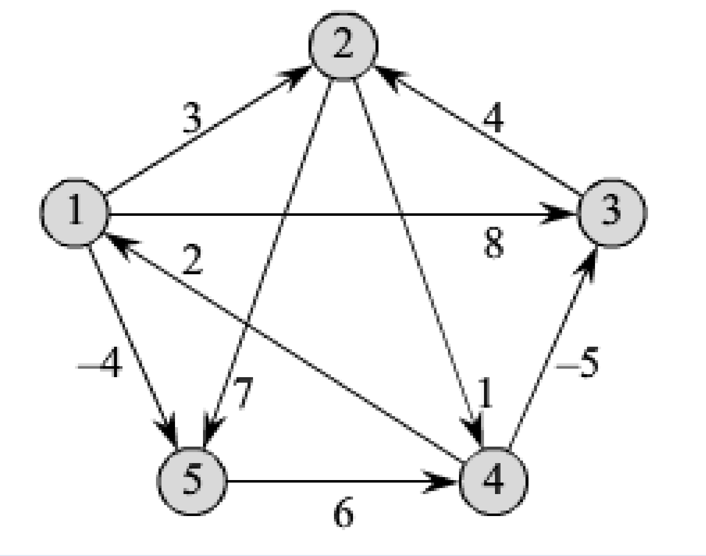
- Shortest Path with intermediates ≤ 0 =

$$\begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$



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

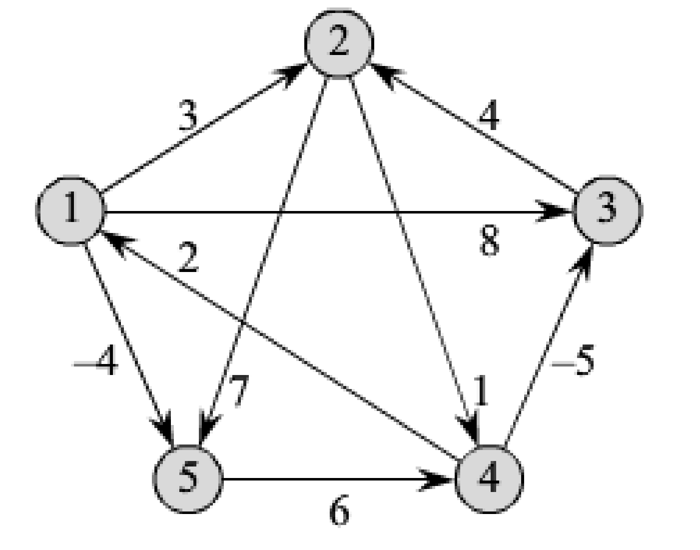
- Shortest Path with intermediates ≤ 1 =
  - p=(4, 1, 5)
  - w(p) = -2



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

- Shortest Path with intermediates ≤ 1 =
  - p=(4, 1, 5), w(p) = -2

$$\begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & 1 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$



FLOYD-WARSHALL(W)

```
n = W.rows
D^(0) = W
for k = 1 to n
    let D^(k) = (d_ij^(k)) be a new n × n matrix
    for i = 1 to n
        for j = 1 to n
            d_ij^(k) = min(d_ij^(k-1), d_ik^(k-1) + d_kj^(k-1))
return D^(n)
```

$$d_{ij}^{(k)} = \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right)$$

$D^{(1)}$

$$\begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

- Shortest Path with intermediates ≤ 1 =
  - p=(4, 1, 5), w(p) = -2
  - $d_{4,5}^{(1)}$ = min of
    - $d_{4,1}^{(0)} + d_{1,5}^{(0)}$
    - or: $d_{4,5}^{(0)}$

$$\begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

$D^{(2)}$

$$\begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & & & \\ 2 & 5 & & & \\ \infty & \infty & & & \end{pmatrix}$$

$$\begin{pmatrix} \text{NIL} & 1 & 1 & 2 & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & 1 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

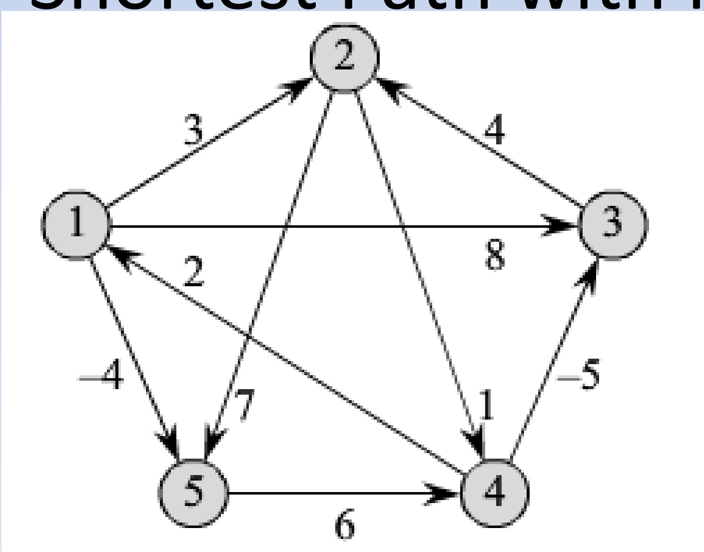
- Shortest Path with intermediates ≤ 2 =
  - p=(1, 2, 4), w(p) = 4
  - $d_{1,4}^{(2)}$ = min of
    - $d_{1,2}^{(1)} + d_{2,4}^{(1)}$
    - or: $d_{1,4}^{(1)}$

$$\begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

$D^{(3)}$

$$\begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

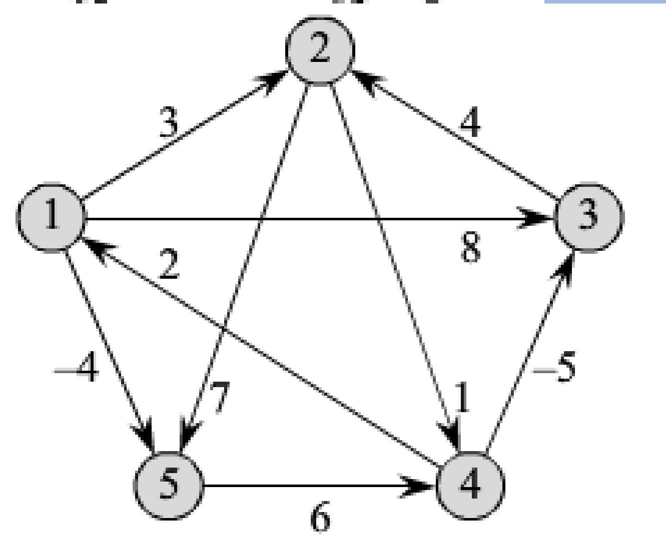
- Shortest Path with intermediates ≤ 3 =



$$\begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

$D^{(3)}$

$$\begin{pmatrix} 0 & 3 & 8 \\ \infty & 0 & \infty \\ \infty & 4 & 0 \\ 2 & -1 & -5 \\ \infty & \infty & \infty \end{pmatrix}$$



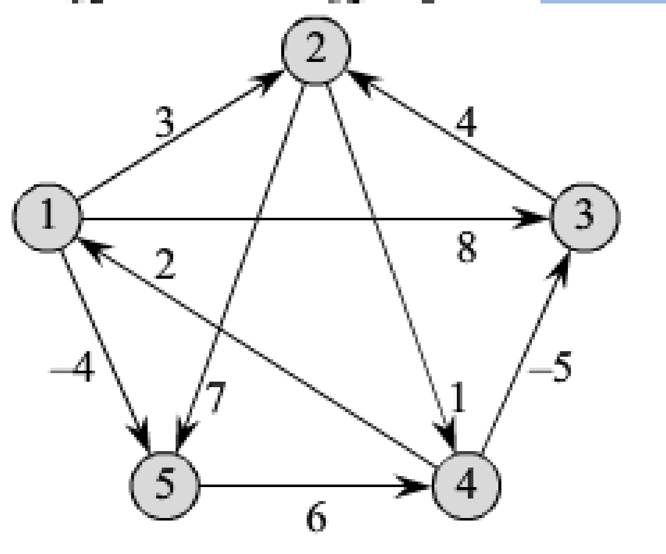
- Shortest Path with intermediates ≤ 3 =
  - p=(4, 3, 2), w(p) = -1
  - $D_{4,2}^{(3)}$ = min of
    - $d_{4,3}^{(2)} + d_{3,2}^{(2)}$
    - or: $d_{4,2}^{(2)}$

$$\begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

$D^{(3)}$

$$\begin{pmatrix} 0 & 3 & 8 \\ \infty & 0 & \infty \\ \infty & 4 & 0 \\ & -1 & -5 \\ & \infty & \infty \end{pmatrix}$$

$$\begin{pmatrix} \text{NIL} & 1 & 1 & 2 & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$



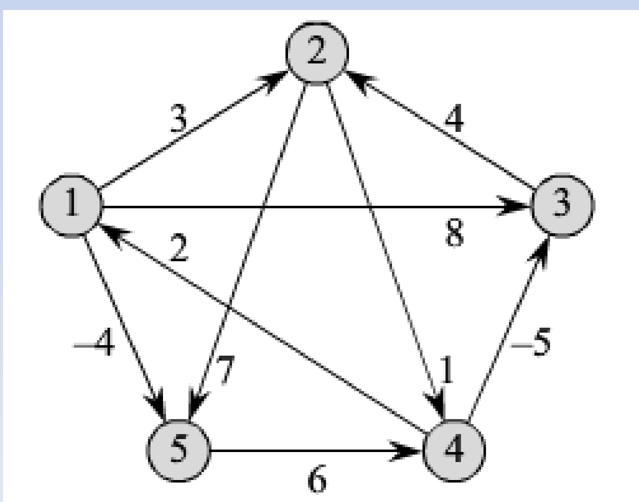
- Shortest Path with intermediates ≤ 3 =
  - $p=(4, 3, 2)$, $w(p) = -1$
  - $D_{4,2}^{(3)}$ = min of
    - $d_{4,3}^{(2)} + d_{3,2}^{(2)}$
    - or: $d_{4,2}^{(2)}$

$$\begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

$D^{(4)}$

$$\begin{pmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

- Shortest Path with intermediates ≤ 4 =



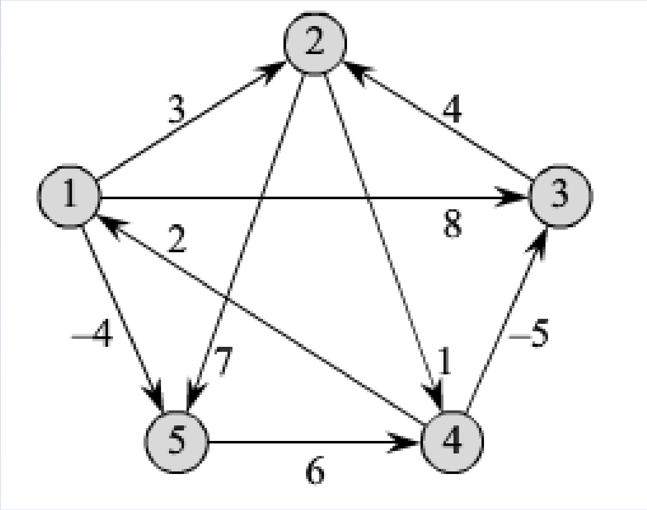
$$\begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

$D^{(4)}$ $\Pi^{(4)}$

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

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

- Shortest Path with intermediates ≤ 4 =



$$\begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

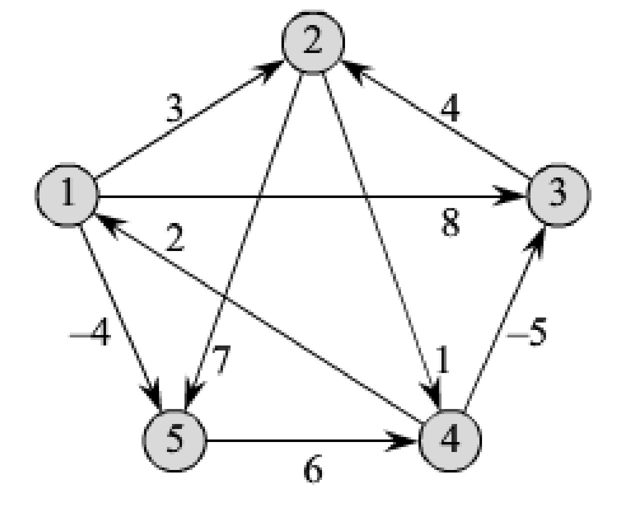
$D^{(5)}$

$$\begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

$\Pi^{(5)}$

$$\begin{pmatrix} \text{NIL} & 3 & 4 & 5 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$

- Shortest Path with intermediates ≤ 5 =



$$\begin{pmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

Computing Path

$$\pi_{ij}^{(0)} = \begin{cases} \text{NIL} & \text{if } i = j \text{ or } w_{ij} = \infty , \\ i & \text{if } i \neq j \text{ and } w_{ij} < \infty . \end{cases} \qquad (25.6)$$

$$\pi_{ij}^{(k)} = \begin{cases} \pi_{ij}^{(k-1)} & \text{if } d_{ij}^{(k-1)} \leq d_{ik}^{(k-1)} + d_{kj}^{(k-1)} , \\ \pi_{kj}^{(k-1)} & \text{if } d_{ij}^{(k-1)} > d_{ik}^{(k-1)} + d_{kj}^{(k-1)} . \end{cases} \qquad (25.7)$$

Floyd-Warshall Application: Transitive Closure

Transitive Closure of a DAG

- What are all the vertices reachable from any graph vertex?

- Define Transitive Closure Graph $G^*=(V, E^*)$
- $E^* = \{(i, j) : \text{if path from } i \text{ to } j \text{ in } G\}$

Transitive Closure of a DAG: Example Application

- Consider the graph G underlying any spreadsheet model,
  - The nodes are cells
  - And there is an arc from cell i to cell j if the result of cell j depends on cell i.
- When the value of a given cell is modified, the values of all reachable cells must also be updated.
- The identity of these cells is revealed by the transitive closure of G.
- Doing it fast is important

Transitive Closure of a DAG
Simple Solution

- What are all the vertices reachable from any graph vertex?

- Run Floyd-Warshall with edge weights set to 1.
- If $d_{ij} < n$, there is a path,
- If $d_{ij} = \infty$, there is no path

Transitive Closure of a DAG Simple Solution

- What are all the vertices reachable from any graph vertex?

- Run Floyd-Warshall with edge weights set to 1.
- If $d_{ij} < n$, there is a path,
- If $d_{ij} = \infty$, there is no path
- Complexity = $\Theta(n^3)$

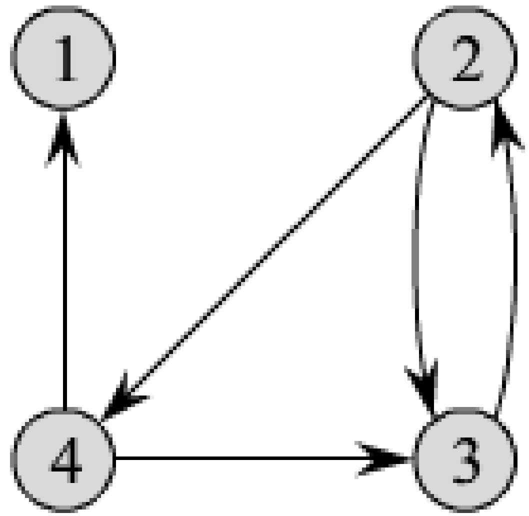
Transitive Closure of a DAG
Better Solution

- What are all the vertices reachable from any graph vertex?

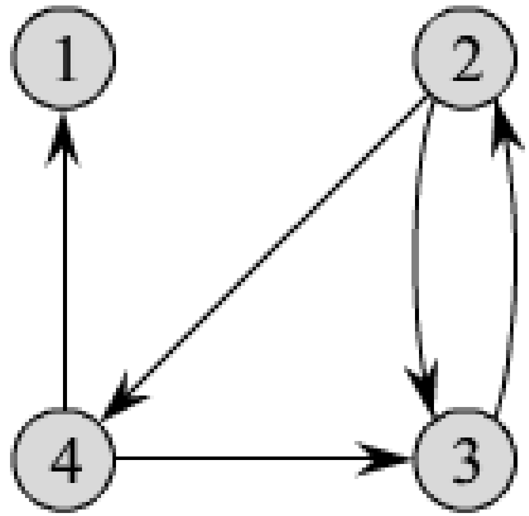
- Modify Floyd-Warshall by substituting logical AND and logical OR for MIN and ADD
- Complexity still $\Theta(n^3)$, but better in practice

Transitive Closure

- Define $T_{ij}^{(k)}$ as 1 if there exists a path from i to j with all itermediate vertices < k
- $E^* = \{(i,j) : T_{ij}^{(n)}$
- $T_{ij}^{(0)} =$
  - 0 if $i \neq j$ and $(i, j) \notin E$
  - 1 if $i = j$ or $(i, j) \in E$
- $k \geq 1$
  - $t_{ij}^{(k)} = t_{ij}^{(k-1)} \vee (t_{ik}^{(k-1)} \wedge t_{kj}^{(k-1)})$

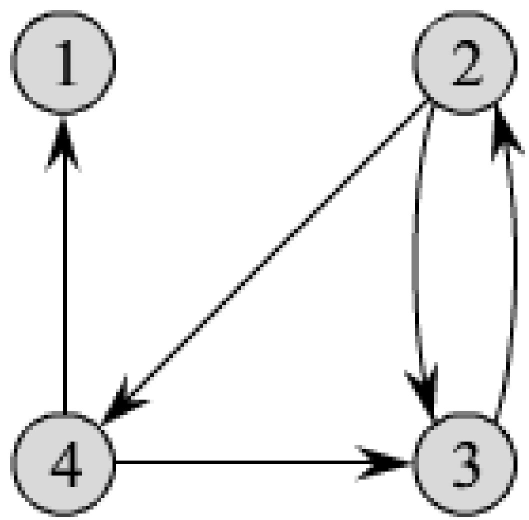


$$T^{(0)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

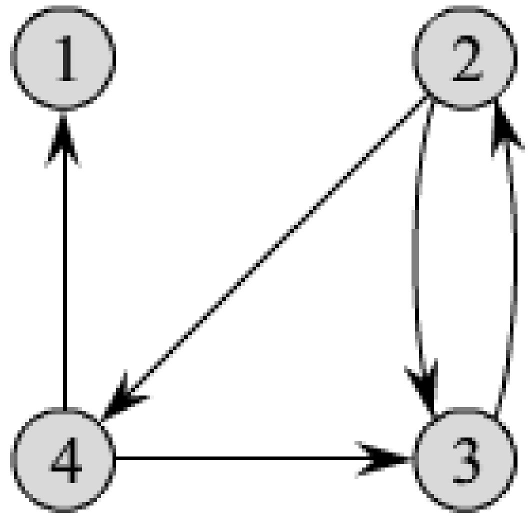


$$T^{(0)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

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



$$T^{(2)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix} \quad T^{(3)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$



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

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

TRANSITIVE-CLOSURE(G)

```
n = |G.V|
let T^(0) = (t_ij^(0)) be a new n × n matrix
for i = 1 to n
    for j = 1 to n
        if i == j or (i, j) ∈ G.E
            t_ij^(0) = 1
        else t_ij^(0) = 0
for k = 1 to n
    let T^(k) = (t_ij^(k)) be a new n × n matrix
    for i = 1 to n
        for j = 1 to n
            t_ij^(k) = t_ij^(k-1) ∨ (t_ik^(k-1) ∧ t_kj^(k-1))
return T^(n)
```

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

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

Another Approach to All-Pairs Shortest Paths

Johnson's Algorithm

- For Sparse Graphs – Better than Floyd-Warshall or Repeated Matrix Squaring
- $O(V^2 \lg V + VE)$

- Returns a matrix of shortest-path weights OR Reports Negative Weight Cycle

Johnson's Algorithm w/ Reweighting

- Given Graph G=(V,E) with all edge weights w>=0.
- Dijkstra's shortest path run on each vertex v in V w/ Fibonacci-Heap min-priority queue
  - $O(V^2 \lg V + VE)$
- If G has edge weights less than 0
  - Compute new set of non-negative weights

Dijkstra

- Solves the single-source shortest-paths problem on a weighted, directed graph G=(V,E)
- Requires all edge weights are nonnegative.
  - Assumes w(u,v) >= 0 for each edge (u,v) in E.
- With a good implementation, the running time of Dijkstra's algorithm is lower than that of the Bellman-Ford algorithm.

Dijkstra

- Dijkstra's algorithm maintains a set S of vertices whose final shortest-path weights from the source s have already been determined
- The algorithm:
  - repeatedly selects the vertex u from V-S with the minimum shortest-path estimate,
  - adds u to S,
  - and relaxes all edges leaving u.
- Book implementation uses min-priority queue Q of vertices, keyed by their d values.

Dijkstra

```
DIJKSTRA(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
S = ∅
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    S = S ∪ {u}
    for each vertex v ∈ G.Adj[u]
        RELAX(u, v, w)
```

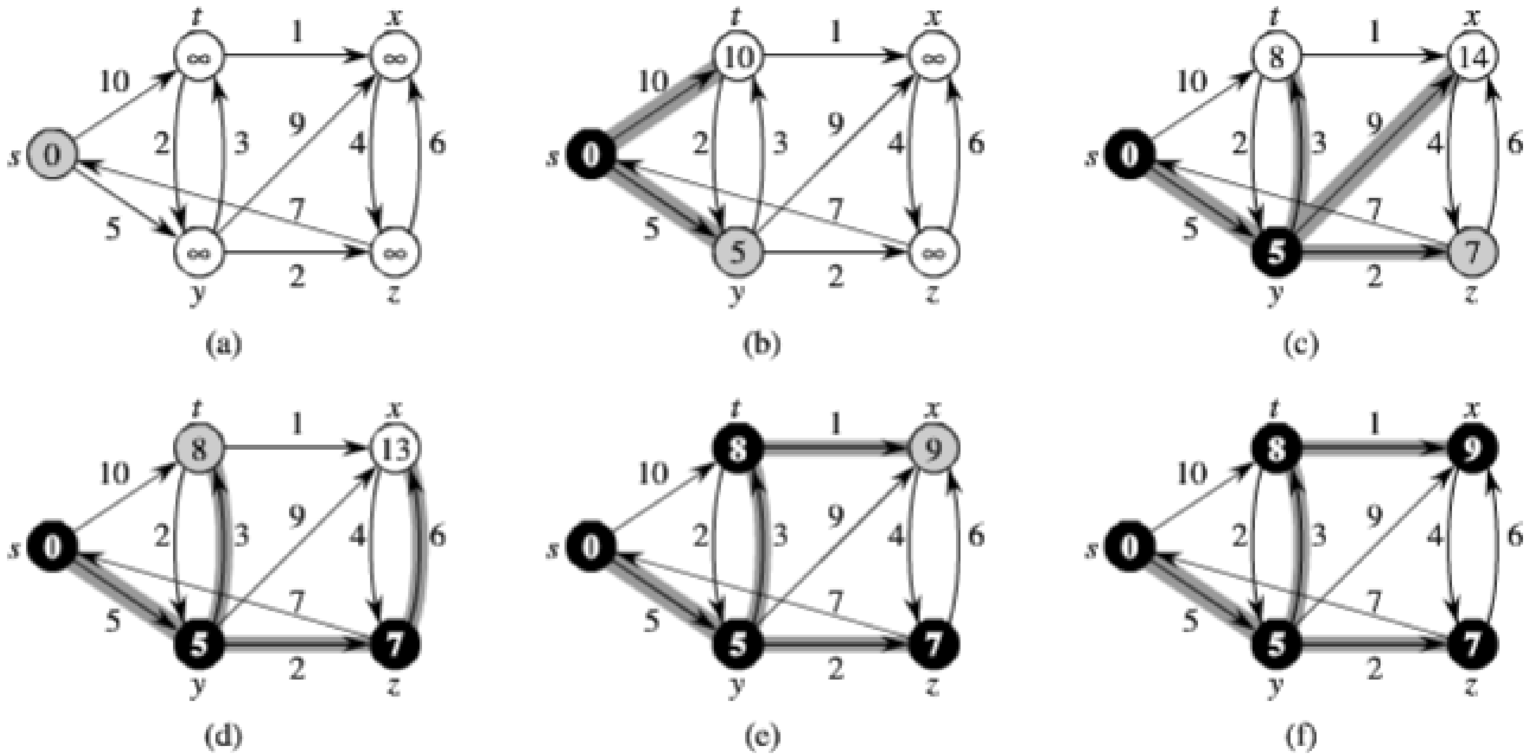


Figure 24.6 The execution of Dijkstra's algorithm. The source s is the leftmost vertex. The shortest-path estimates appear within the vertices, and shaded edges indicate predecessor values. Black vertices are in the set S, and white vertices are in the min-priority queue $Q = V - S$. **(a)** The situation just before the first iteration of the **while** loop of lines 4–8. The shaded vertex has the minimum d value and is chosen as vertex u in line 5. **(b)–(f)** The situation after each successive iteration of the **while** loop. The shaded vertex in each part is chosen as vertex u in line 5 of the next iteration. The d values and predecessors shown in part (f) are the final values.

Johnson's Algorithm w/ Reweighting : $\widehat{w}$

1. $\forall$u, v $\in$ V, a shortest path between u and v with w must also be a shortest path with $\widehat{w}$
2. $\widehat{\mathrm{w}}$ is nonnegative for all edges

Computing New Weights

- Reweight $\hat{w}$ using function h(v) for v in V.
- $\hat{w}$(u, v) = w(u, v) + h(u) – h(v)

Computing New Weights

- Reweight $\hat{w}$ using function h(v) for v in V.
- $\hat{w}$(u, v) = w(u, v) + h(u) – h(v)

- Reweighting doesn't change Shortest Path

$$
\begin{aligned}
\hat{w}(p) &= \sum_{i=1}^{k} \hat{w}(v_{i-1}, v_i) \\
&= \sum_{i=1}^{k} (w(v_{i-1}, v_i) + h(v_{i-1}) - h(v_i)) \\
&= \sum_{i=1}^{k} w(v_{i-1}, v_i) + h(v_0) - h(v_k) \\
&= w(p) + h(v_0) - h(v_k) .
\end{aligned}
$$

Sum Telescopes

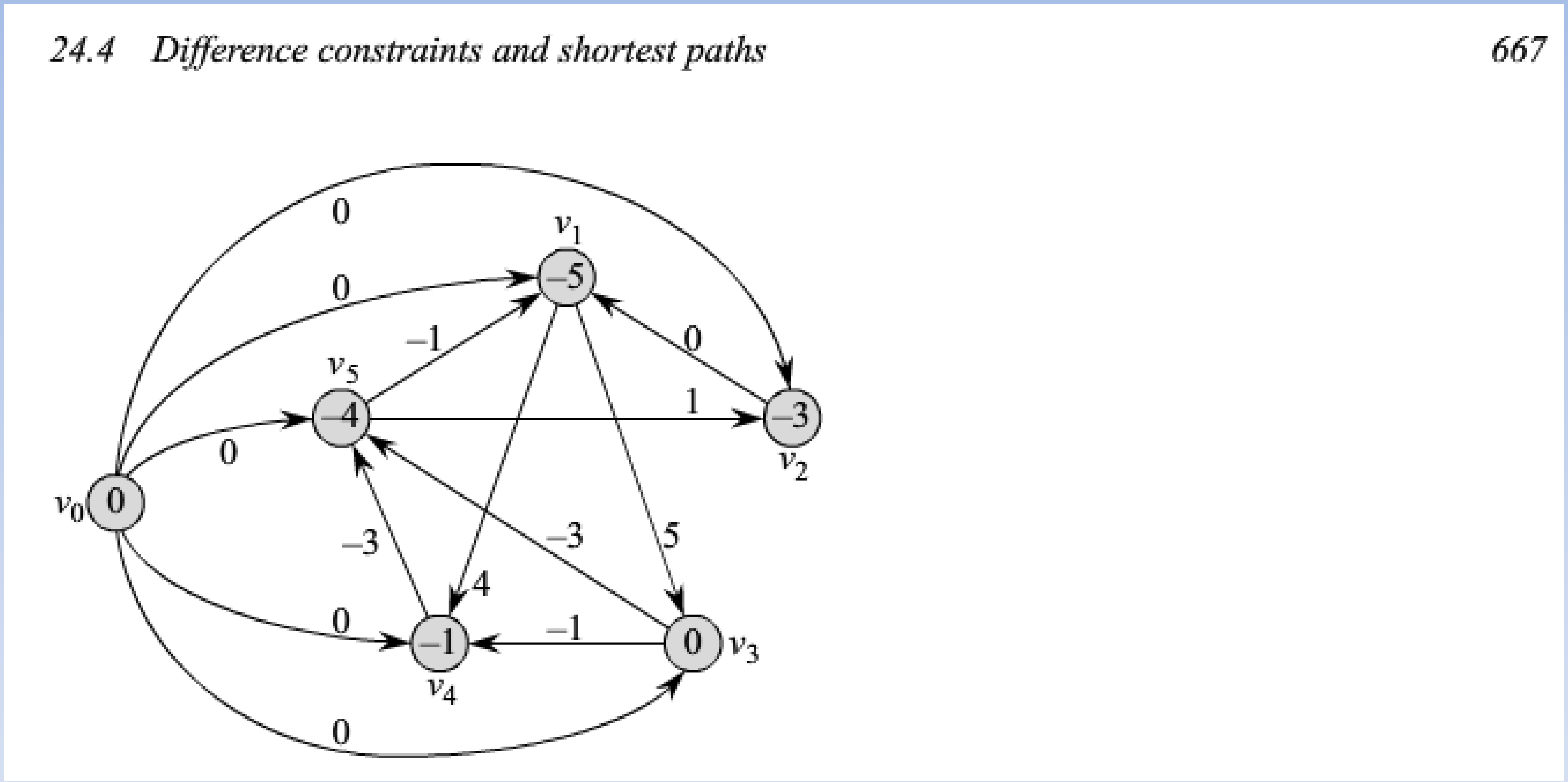
Computing New Weights

- Reweight $\hat{w}$ using function h(v) for v in V.
- $\hat{w}$(u, v) = w(u, v) + h(u) – h(v)

- Reweighting doesn't change Shortest Path
- Reweighting maintains negative weight cycles

$$\begin{aligned} \hat{w}(c) &= w(c) + h(v_0) - h(v_k) \\ &= w(c) , \end{aligned}$$

Example Constraint Graph



- Uses similar graph to produce reweighting function h(v)

Reweighting Function h(v)

- Given: G=(V,E) w/ w : E -> R
- Make G’ = (V’,E’) where
  - V’ = V + {s}
  - E’ = E + {(s, v) : v in V}
- Same as our Difference Constraints Graph

Reweighting Function h(v)

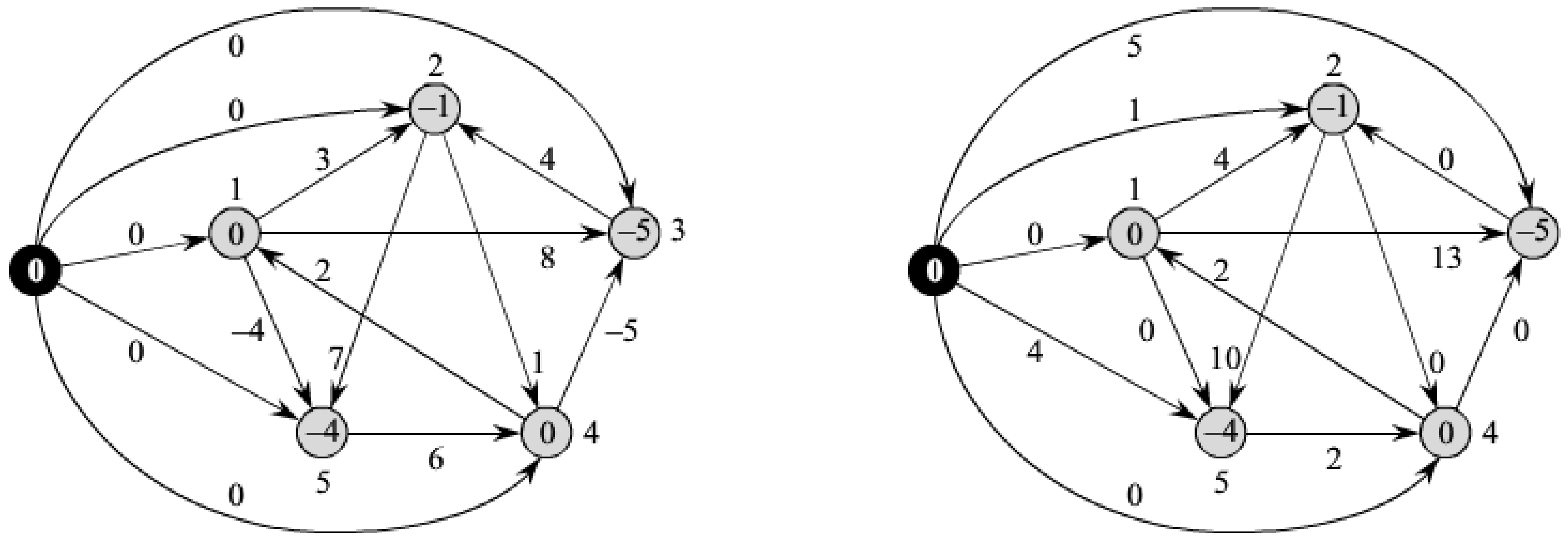
- Given: G=(V,E) w/ w : E -> R
- Make G' = (V',E') where
  - V' = V + {s}
  - E' = E + {(s, v) : v in V}
- Now G':
  - No shortest path include s unless they start with s.
  - No negative weight-cycle unless in G.
- Define h(v) = δ(s,v) for all v in V'

Reweighting Function h(v)

- Given: G=(V,E) w/ w : E -> R
- Make G' = (V',E') where
  - V' = V + {s}
  - E' = E + {(s, v) : v in V}

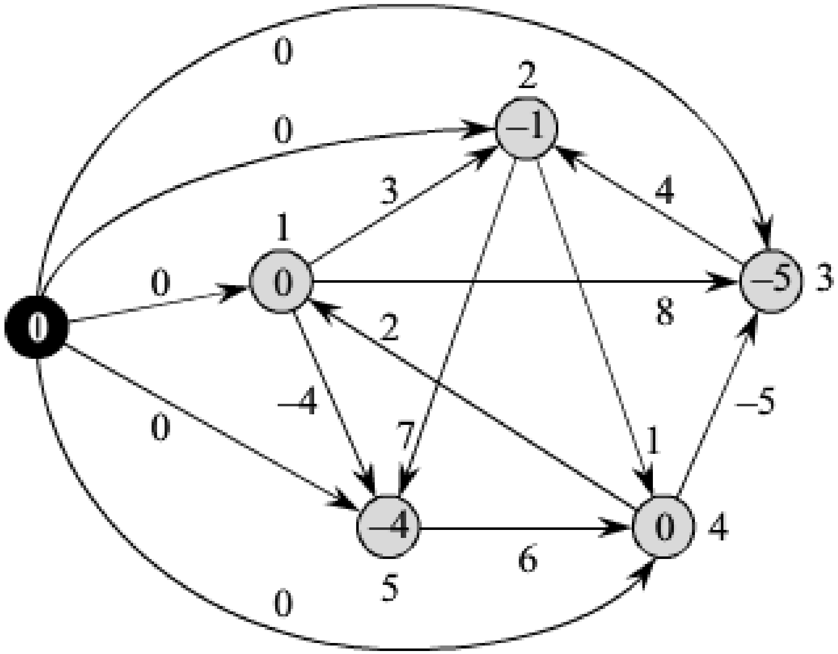
- Define h(v) = δ(s,v) for all v in V'
  - $h(v) \le h(u) + w(u,v)$
  - $0 \le h(u) - h(v) + w(u,v)$
  - $\widehat{w}(u, v) = w(u, v) + h(u) - h(v) \ge 0$
    - NONNEGATIVE

25.3 Johnson's algorithm for sparse graphs 703

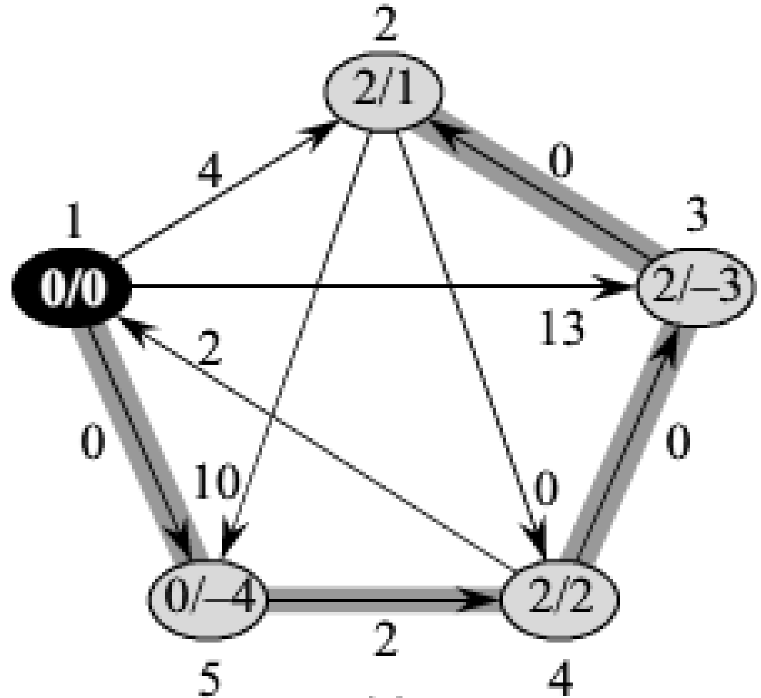


(a) (b)

- $\delta(s,3) = -5$
- $\delta(s,2) = -1$
- $\hat{w}(3, 2) = w(3,2) + \delta(s,3) - \delta(s,2) = 4 + (-5) - (-1)$

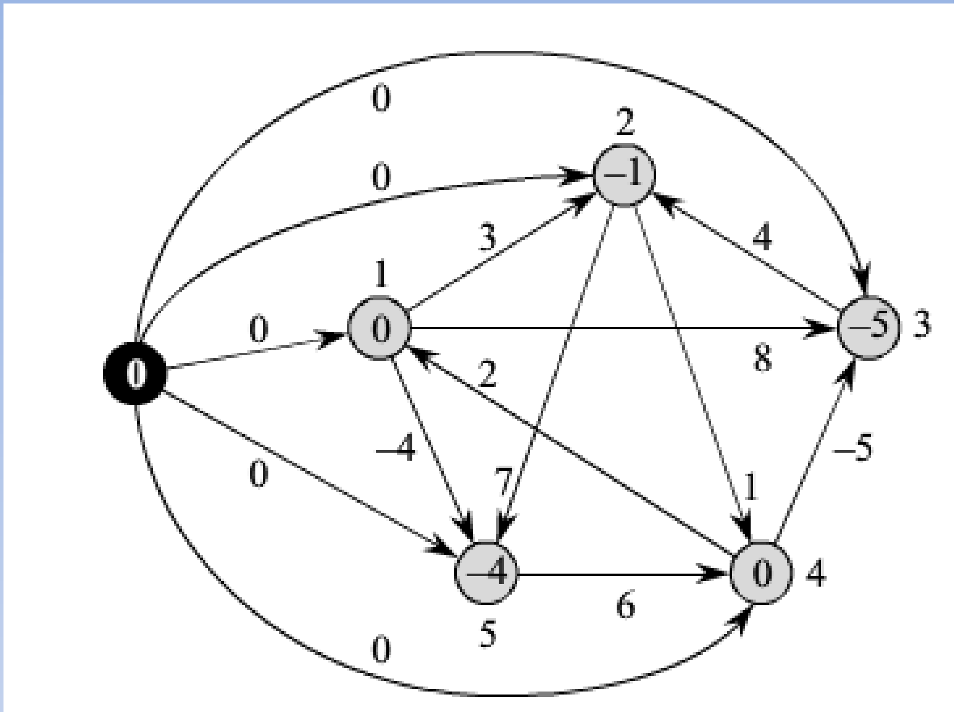


(a)

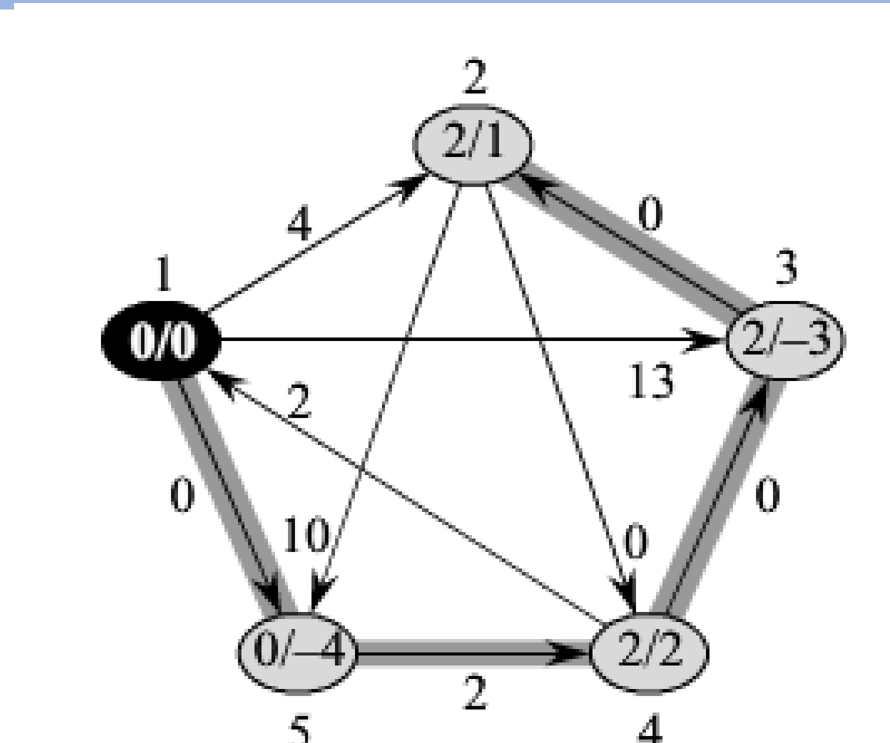


(c)

- δ(u,v) = $\hat{\delta}$(u,v) +h(v) – h(u)
  - $\widehat{w}$(u, v) = w(u, v) + h(u) – h(v)
  - $\widehat{w}$(u, v) + h(v) – h(u) = w(u, v)

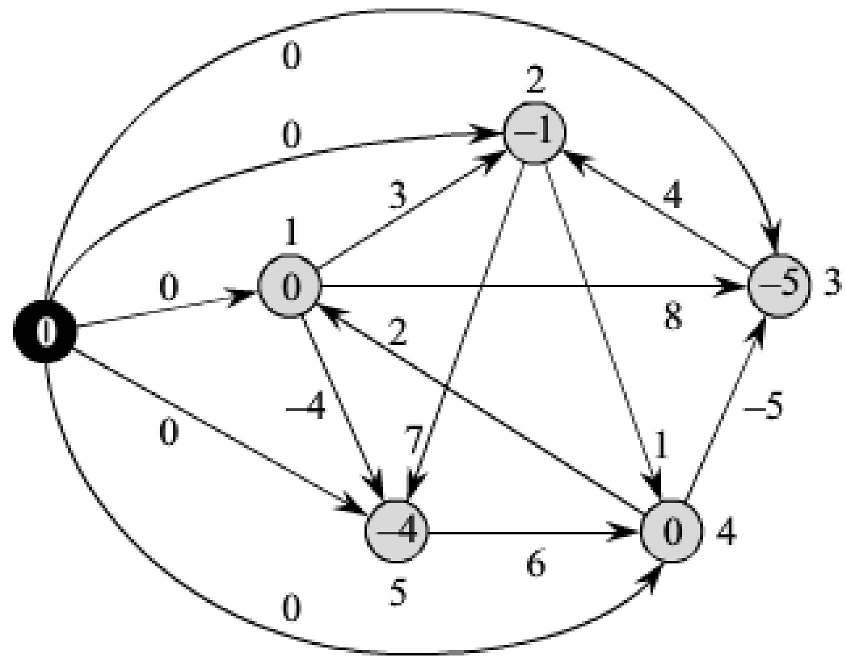


(a)

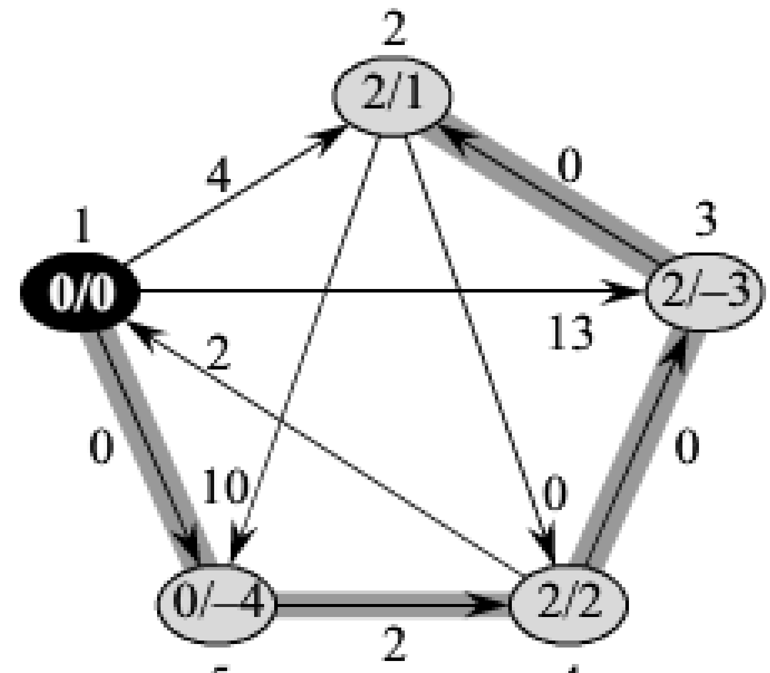


(c)

- $\delta(1,3) = w(1,5) + w(5,4) + w(4,3)$
- $\delta(1,4) = -4 + 6 + -5 = -3$

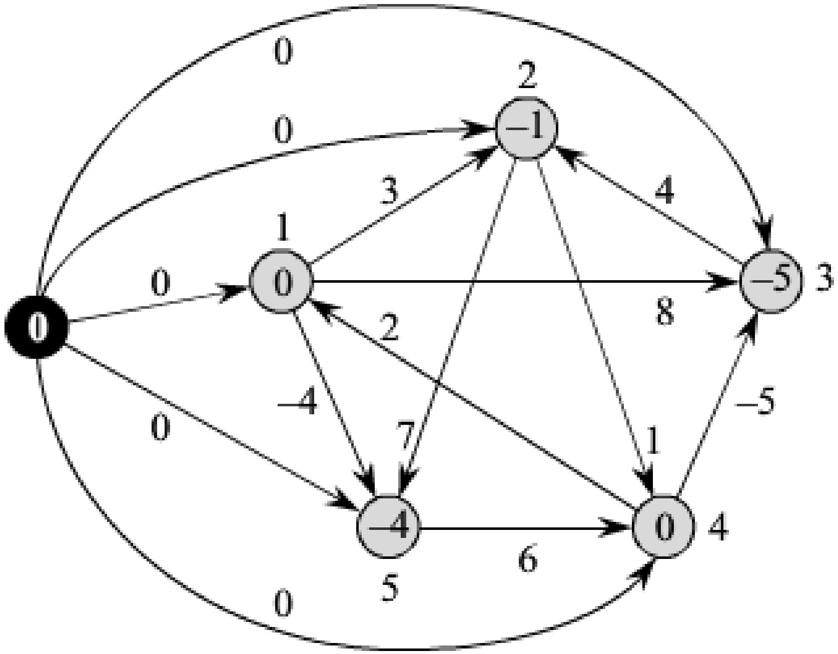


(a)

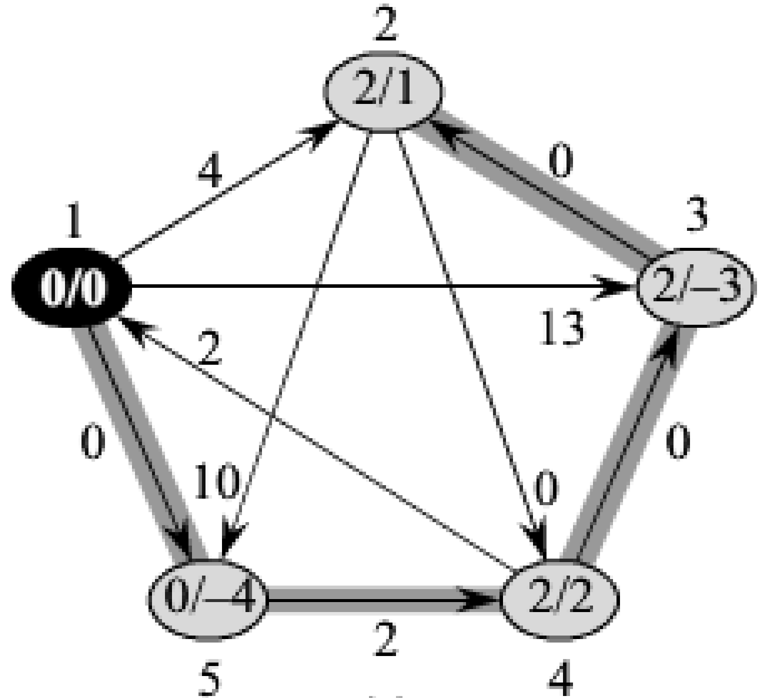


(c)

- $\hat{\delta}(1,4) = \hat{w}(1,5) + \hat{w}(5,4) + \hat{w}(4,3)$
- $\delta(1,4) = 0 + 2 + 0 = 2$



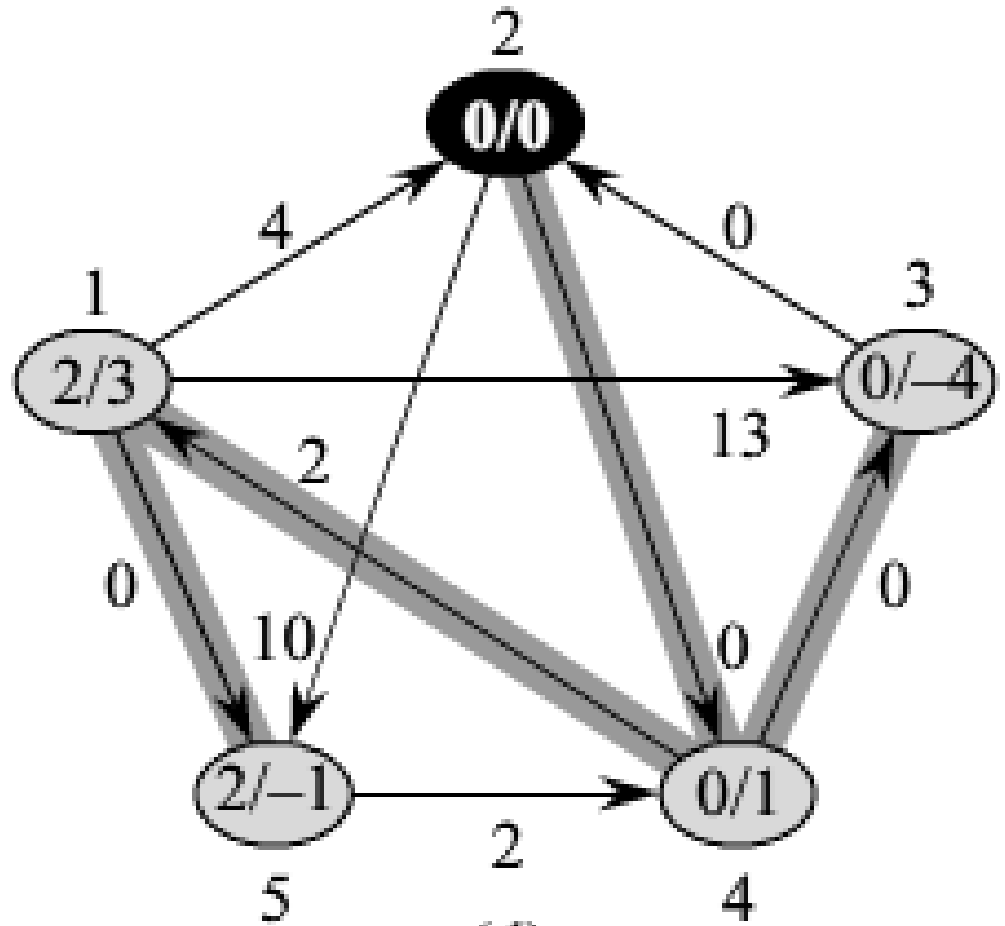
(a)



(c)

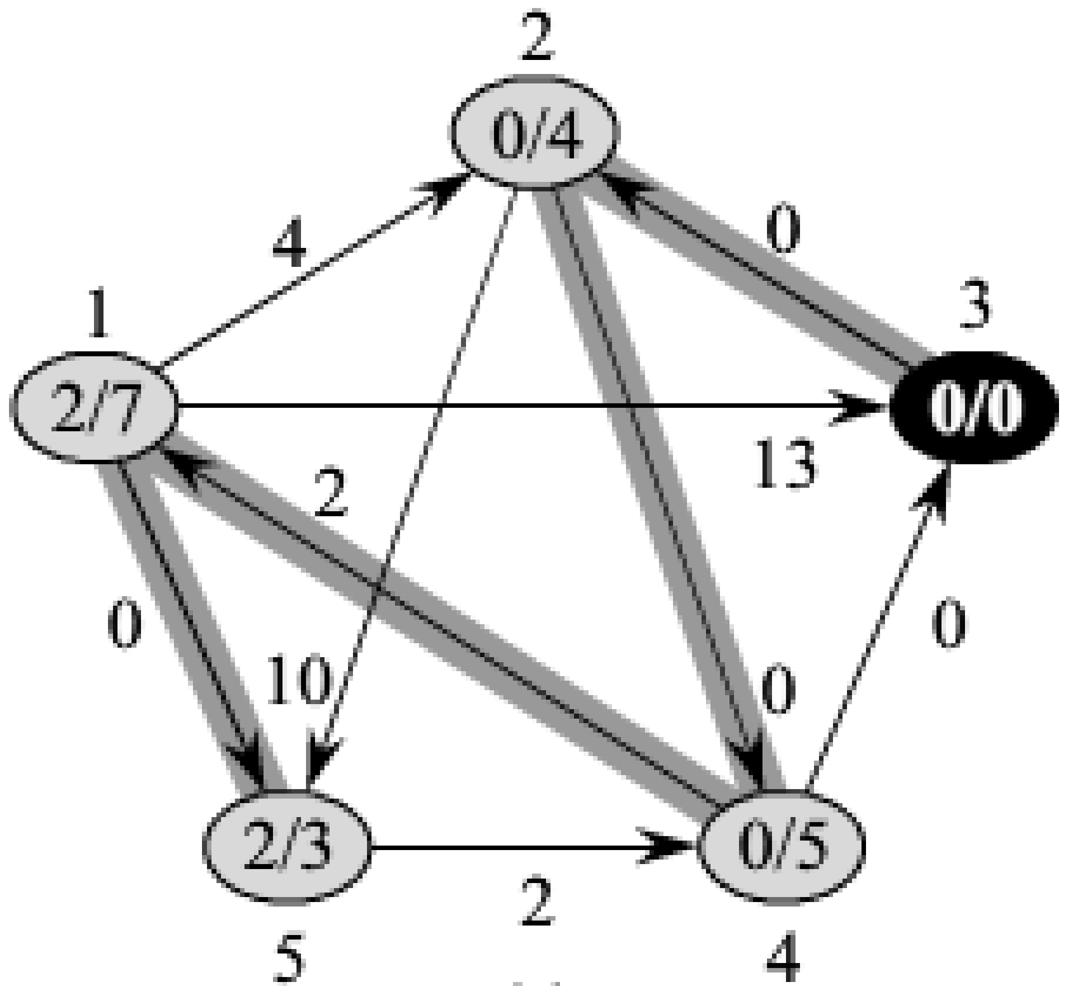
- $\delta(1,3) = \hat{\delta}(1,3) + h(3) - h(1)$
- $\delta(1,3) = 2 + -5 - 0 = -3$

Dijkstra w/ s=2



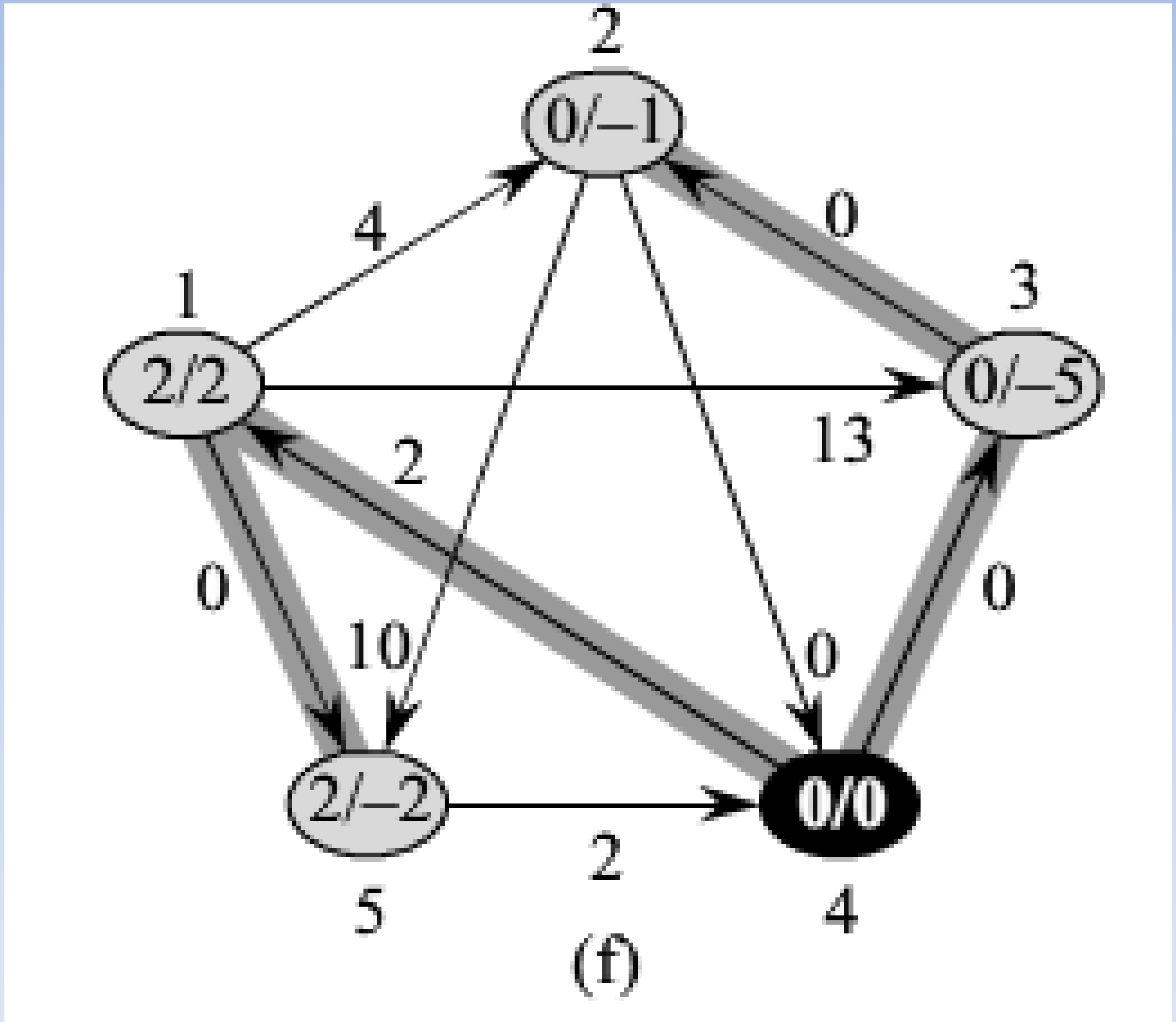
(d)

Dijkstra w/ s=3

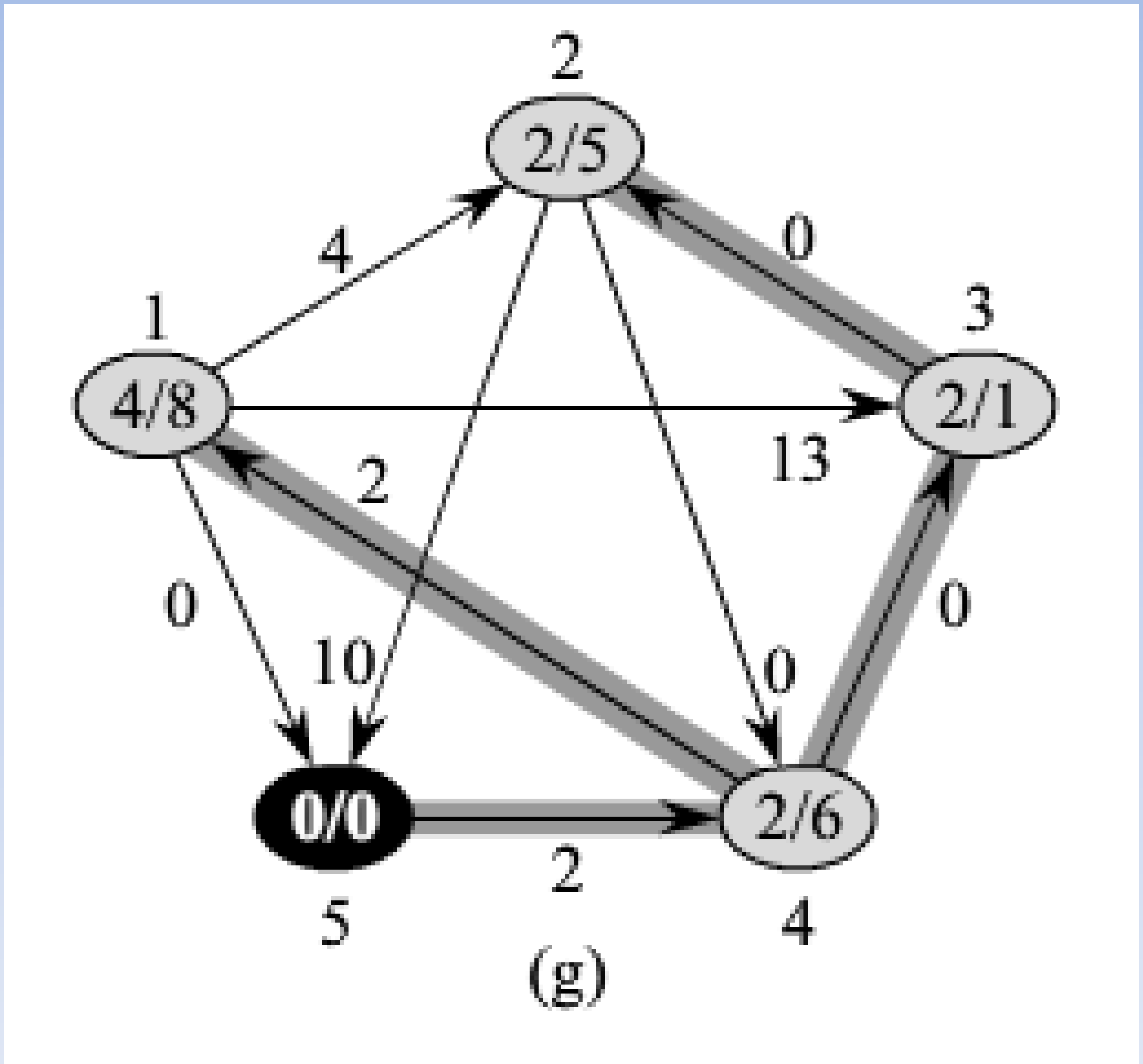


(e)

Dijkstra w/ s=4



Dijkstra w/ s=5



```
JOHNSON(G, w)
    compute G′, where G′.V = G.V ∪ {s},
            G′.E = G.E ∪ {(s, v) : v ∈ G.V}, and
            w(s, v) = 0 for all v ∈ G.V
    if BELLMAN-FORD(G′, w, s) == FALSE
            print "the input graph contains a negative-weight cycle"
    else for each vertex v ∈ G′.V
                set h(v) to the value of δ(s, v)
                        computed by the Bellman-Ford algorithm
            for each edge (u, v) ∈ G′.E
                ŵ(u, v) = w(u, v) + h(u) − h(v)
            let D = (d_uv) be a new n × n matrix
            for each vertex u ∈ G.V
                run DIJKSTRA(G, ŵ, u) to compute δ̂(u, v) for all v ∈ G.V
                for each vertex v ∈ G.V
                        d_uv = δ̂(u, v) + h(v) − h(u)
            return D
```

Johnson Analysis

- Fibonacci Heap yields $O(V^2 \lg V + VE)$
- Binary Min-Heap yields $O(VE \lg V)$