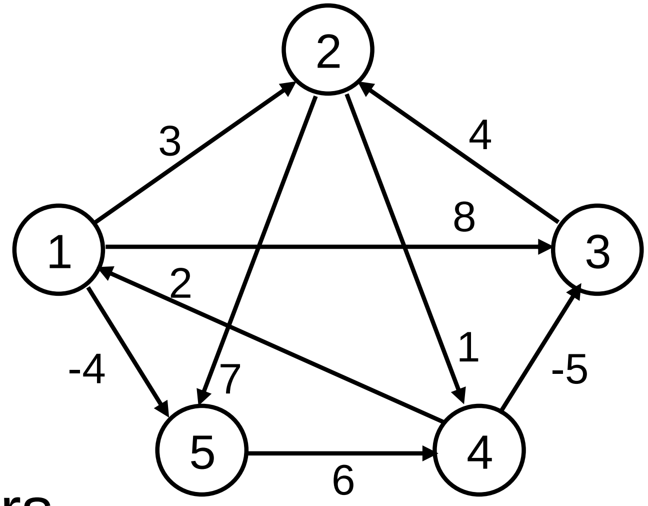
Chapter 25

All-Pairs Shortest Paths

All-Pairs Shortest Paths

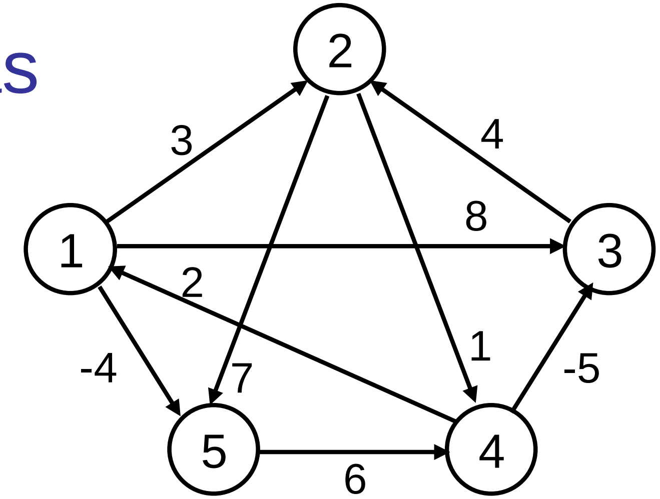
- **Given:**
  - Directed graph G = (V, E)
  - Weight function w : E → **R**
- **Compute:**
  - The shortest paths between all pairs of vertices in a graph
  - Representation of the result: an n × n matrix of shortest-path distances δ(u, v)



All-Pairs Shortest Paths - Solutions

- Run **BELLMAN-FORD** once from each vertex:
  - $O(V^2E)$, which is $O(V^4)$ if the graph is dense ($E = \Theta(V^2)$)
- If no negative-weight edges, could run **Dijkstra's** algorithm once from each vertex:
  - $O(VE\lg V)$ with binary heap, $O(V^3\lg V)$ if the graph is dense
- We can solve the problem in $O(V^3)$ in all cases, with no elaborate data structures

All-Pairs Shortest Paths



- Assume the graph (G) is given as adjacency matrix of weights
  - $W = (w_{ij})$, $n \times n$ matrix, $|V| = n$
  - Vertices numbered 1 to n

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

- Output the result in an $n \times n$ matrix

  $D = (d_{ij})$, where $d_{ij} = \delta(i, j)$
- Solve the problem using dynamic programming

Dynamic-programming method

- 1. Characterize the structure of an optimal solution.
- 2. Recursively define the value of an optimal solution.
- 3. Compute the value of an optimal solution in a bottom-up fashion.
- 4. constructing an optimal solution from computed information.

Shortest Paths and Matrix Multiplication

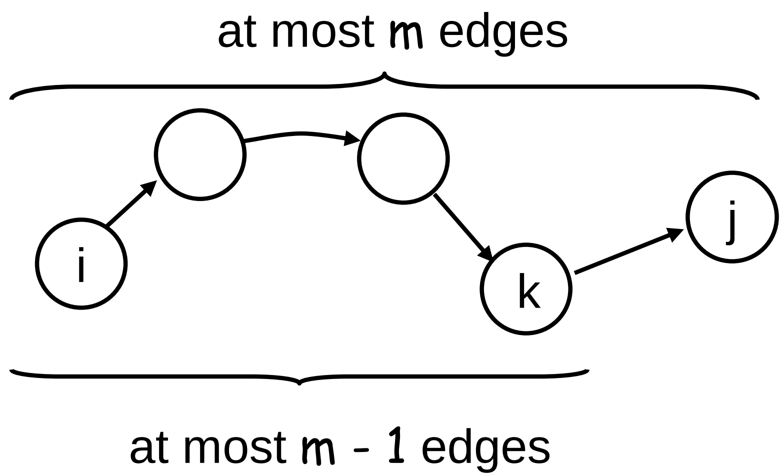
- Problem representation:

  find the shortest paths that have at most m edges

- Find shortest paths of length m = 1
- Continue "expanding" the paths' lengths
- This process is similar to matrix multiplication

Optimal Substructure of a Shortest Path

- All subpaths of a shortest path are shortest paths
- Let p: a shortest path p from vertex i to j that contains at most m edges
- If $i = j$
  - $w(p) = 0$ and p has no edges

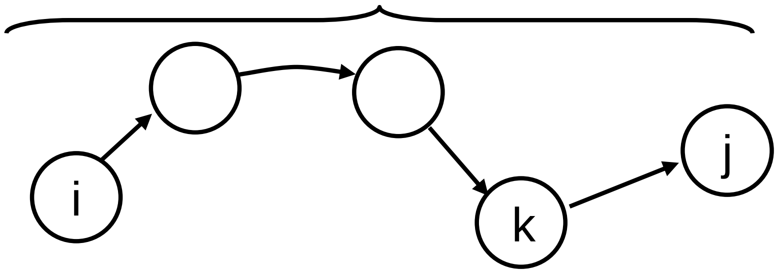


- If $i \neq j$: $p = i \overset{p'}{\rightsquigarrow} k \rightarrow j$
  - p' has at most m-1 edges
  - p' is a shortest path

$$\delta(i, j) = \delta(i, k) + w_{kj}$$

Recursive Solution

- $l_{ij}^{(m)}$ = weight of shortest path $i \rightsquigarrow j$ that contains at most m edges



- $m = 0$: $l_{ij}^{(0)} = \begin{cases} 0 & \text{if } i = j \\ \infty & \text{if } i \neq j \end{cases}$

- $m \geq 1$: $l_{ij}^{(m)} = \min \{ l_{ij}^{(m-1)} , \min\limits_{1 \leq k \leq n} \{l_{ik}^{(m-1)} + w_{kj}\} \}$

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

  – Shortest path from i to j with at most m – 1 edges

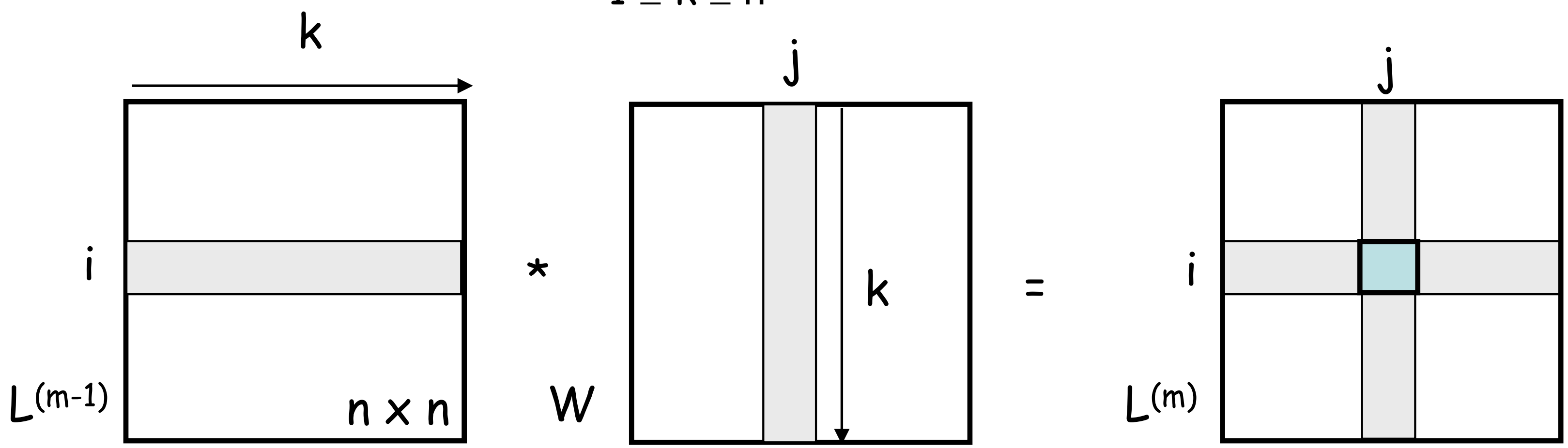
  – Shortest path from i to j containing at most m edges, considering all possible predecessors (k) of j

Computing the Shortest Paths

- $m = 1$: $l_{ij}^{(1)} = w_{ij}$ $\quad L^{(1)} = W$
  - The path between i and j is restricted to 1 edge
- Given $W = (w_{ij})$, compute: $L^{(1)}, L^{(2)}, \ldots, L^{(n-1)}$, where
- $L^{(n-1)}$ contains the actual shortest-path weights
- Given $L^{(m-1)}$ and $W \Rightarrow$ compute $L^{(m)}$
  - Extend the shortest paths computed so far by one more edge

Extending the Shortest Path

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



Replace: min $\rightarrow$ +
+ $\rightarrow$ •

Computing $L^{(m)}$ looks like matrix multiplication

EXTEND(*L*, *W*, *n*)

EXTEND-SHORTEST-PATHS(L, W)

```
n = L.rows
let L' = (l'_ij) be a new n × n matrix
for i = 1 to n
    for j = 1 to n
        l'_ij = ∞
        for k = 1 to n
            l'_ij = min(l'_ij, l_ik + w_kj)
return L'
```

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

Comparison with matrix multiplication

EXTEND-SHORTEST-PATHS(L, W)

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

SQUARE-MATRIX-MULTIPLY(A, B)

$n = A.rows$
let C be a new $n \times n$ matrix
for $i = 1$ **to** n
    for $j = 1$ **to** n
        $c_{ij} = 0$
        for $k = 1$ **to** n
            $c_{ij} = c_{ij} + a_{ik} \cdot b_{kj}$
return C

$$
\begin{aligned}
l^{(m-1)} &\rightarrow a\,, \\
w &\rightarrow b\,, \\
l^{(m)} &\rightarrow c\,, \\
\min &\rightarrow +\,, \\
+ &\rightarrow \cdot
\end{aligned}
$$

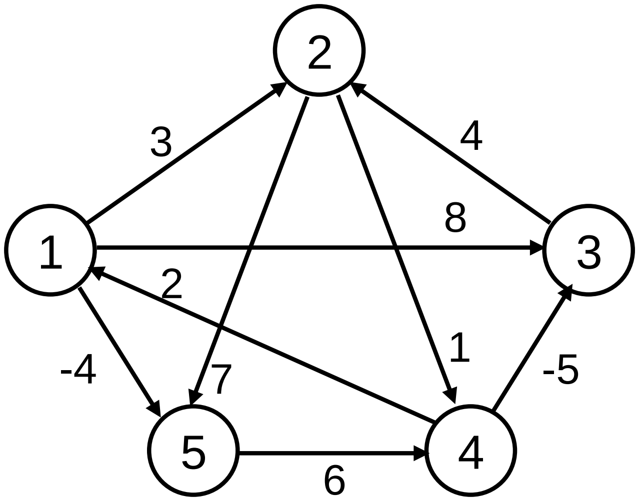
SLOW-ALL-PAIRS-SHORTEST-PATHS(W, n)

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

$n = W.rows$
$L^{(1)} = W$
for $m = 2$ **to** $n - 1$
    let $L^{(m)}$ be a new $n \times n$ matrix
    $L^{(m)} =$ EXTEND-SHORTEST-PATHS($L^{(m-1)}, W$)
return $L^{(n-1)}$

Running time: $\Theta(n^4)$

Example



$L^{(m-1)} = L^{(1)}$

0	3	8	∞	-4
∞	0	∞	1	7
∞	4	0	∞	∞
2	∞	-5	0	∞
∞	∞	∞	6	0

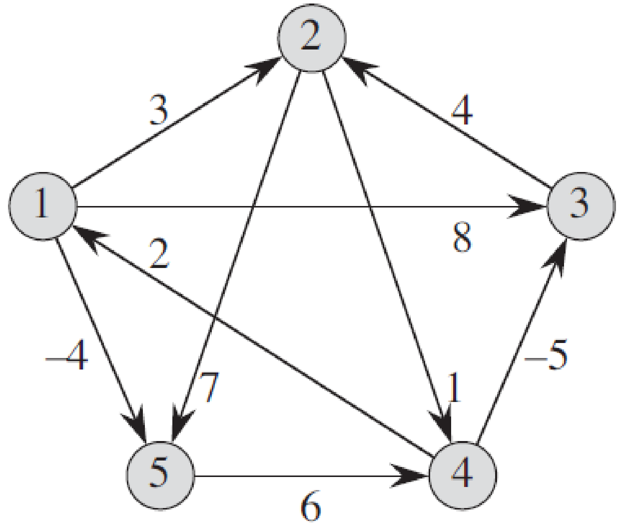
W

0	3	8	∞	-4
∞	0	∞	1	7
∞	4	0	∞	∞
2	∞	-5	0	∞
∞	∞	∞	6	0

$L^{(m)} = L^{(2)}$

0	3	8	2	-4
3	0	-4	1	7
∞	4	0	5	11
2	-1	-5	0	-2
8	∞	1	6	0

… and so on until $L^{(4)}$



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

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

Improving Running Time

- No need to compute all $L^{(m)}$ matrices
- If no negative-weight cycles exist:

  $L^{(m)} = L^{(n-1)}$ for all $m \geq n - 1$

- We can compute $L^{(n-1)}$ by computing the sequence:

$L^{(1)} = W$ $\qquad$ $L^{(2)} = W^2 = W \bullet W$

$L^{(4)} = W^4 = W^2 \bullet W^2$ $\qquad$ $L^{(8)} = W^8 = W^4 \bullet W^4 \ldots$

$$\Rightarrow 2^x = n - 1$$

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

FASTER-APSP(W, n)

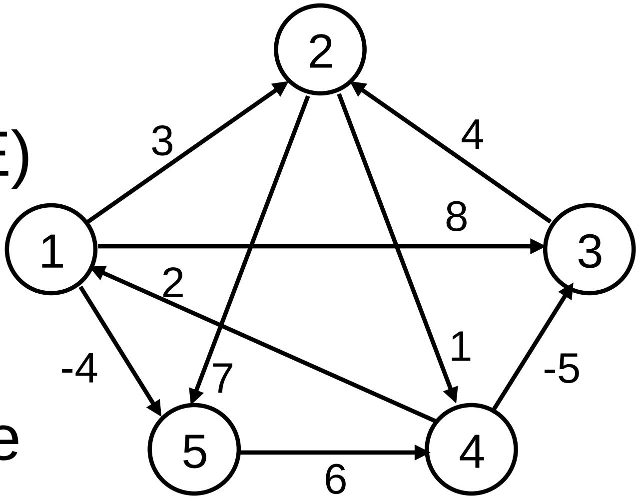
FASTER-ALL-PAIRS-SHORTEST-PATHS(W)

$n = W.rows$
$L^{(1)} = W$
$m = 1$
while $m < n - 1$
    let $L^{(2m)}$ be a new $n \times n$ matrix
    $L^{(2m)} =$ EXTEND-SHORTEST-PATHS$(L^{(m)}, L^{(m)})$
    $m = 2m$
return $L^{(m)}$

- OK to overshoot: products don't change after $L^{(n-1)}$
- **Running Time: $\Theta(n^3 \lg n)$**

The Floyd-Warshall Algorithm

- **Given:**
  - Directed, weighted graph G = (V, E)
  - Negative-weight edges may be present
  - No negative-weight cycles could be present in the graph
- **Compute:**
  - The shortest paths between all pairs of vertices in a graph



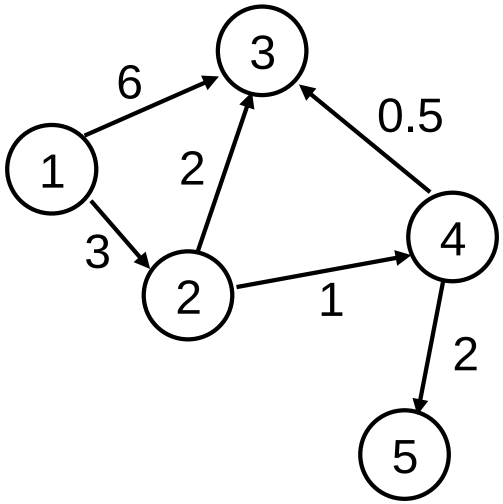
The Structure of a Shortest Path

- Vertices in G are given by

  $V = \{1, 2, \ldots, n\}$

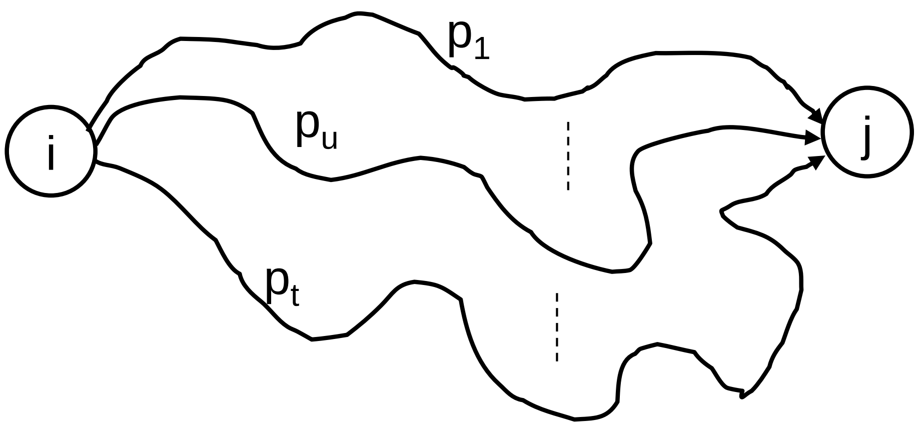
- Consider a path $p = \langle v_1, v_2, \ldots, v_l \rangle$
  - An **intermediate** vertex of p is any vertex in the set $\{v_2, v_3, \ldots, v_{l-1}\}$
  - *E.g.:* $p = \langle 1, 2, 4, 5 \rangle$: $\{2, 4\}$

    $p = \langle 2, 4, 5 \rangle$: $\{4\}$



The Structure of a Shortest Path

- For any pair of vertices $i, j \in V$, consider all paths from i to j whose intermediate vertices are all drawn from a subset {1, 2, …, k}
  - Let p be a minimum-weight path from these paths

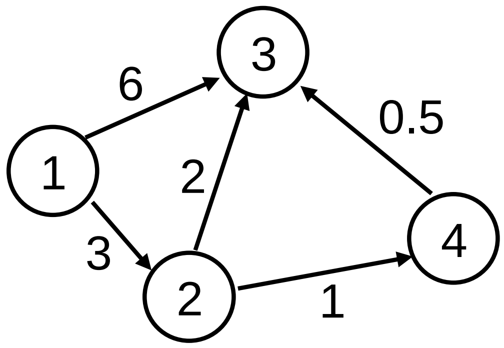


No vertex on these paths has index > k

Example

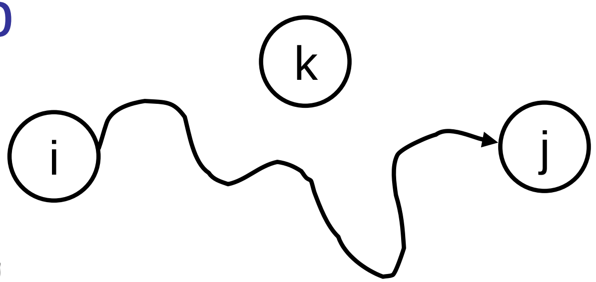
$d_{ij}^{(k)}$ = the weight of a shortest path from vertex i to vertex j with all intermediary vertices drawn from $\{1, 2, \ldots, k\}$

- $d_{13}^{(0)} =$ 6
- $d_{13}^{(1)} =$ 6
- $d_{13}^{(2)} =$ 5
- $d_{13}^{(3)} =$ 5
- $d_{13}^{(4)} =$ 4.5

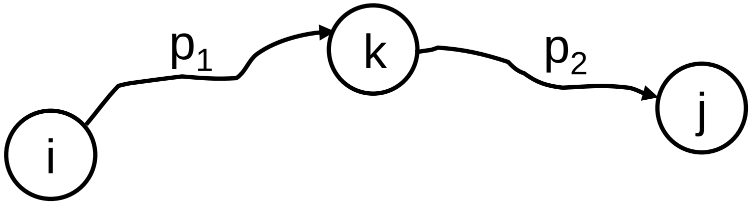


The Structure of a Shortest Path

- k is not an intermediate vertex of path p
  - Shortest path from i to j with intermediate vertices from {1, 2, …, k} is a shortest path from i to j with intermediate vertices from {1, 2, …, k - 1}



- k is an intermediate vertex of path p
  - p_1 is a shortest path from i to k
  - p_2 is a shortest path from k to j
  - k is not intermediary vertex of p_1, p_2
  - p_1 and p_2 are shortest paths from i to k with vertices from {1, 2, …, k - 1}



A Recursive Solution (cont.)

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

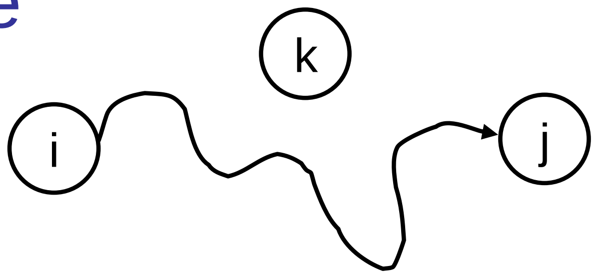
the weight of a shortest path from vertex i to vertex j with all intermediary vertices drawn from $\{1, 2, \ldots, k\}$

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

A Recursive Solution (cont.)

$d_{ij}^{(k)}$ = the weight of a shortest path from vertex i to vertex j with all intermediary vertices drawn from $\{1, 2, \ldots, k\}$

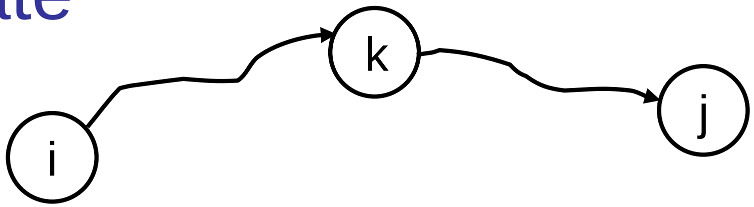
- $k \geq 1$
- **Case 1:** k is not an intermediate vertex of path p
- $d_{ij}^{(k)} = \quad d_{ij}^{(k-1)}$



A Recursive Solution (cont.)

$d_{ij}^{(k)}$ = the weight of a shortest path from vertex i to vertex j with all intermediary vertices drawn from $\{1, 2, \ldots, k\}$

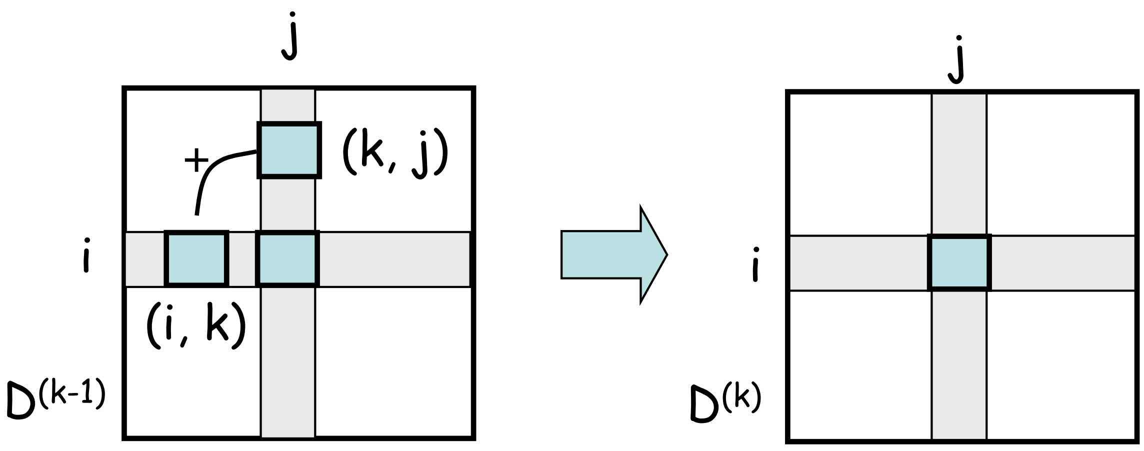
- $k \geq 1$
- **Case 2:** k is an intermediate vertex of path p
- $d_{ij}^{(k)} = d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$



Computing the Shortest Path Weights

- $d_{ij}^{(k)} = \begin{cases} w_{ij} & \text{if } k = 0 \\ \min\{d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\} & \text{if } k \geq 1 \end{cases}$
- The final solution: $D^{(n)} = (d_{ij}^{(n)})$:

$$d_{ij}^{(n)} = \delta(i, j) \quad \forall\, i, j \in V$$



FLOYD-WARSHALL(W)

FLOYD-WARSHALL(W)

$n = W.rows$

$D^{(0)} = W$

for $k = 1$ **to** n

    let $D^{(k)} = \left(d_{ij}^{(k)}\right)$ be a new $n \times n$ matrix

    for $i = 1$ **to** n

        for $j = 1$ **to** n

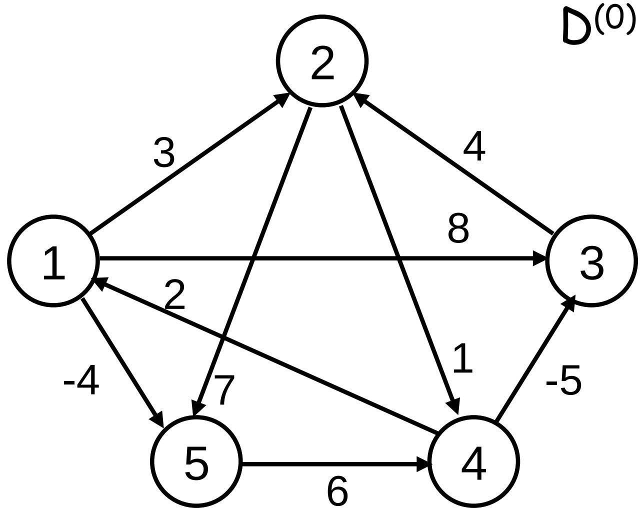
            $d_{ij}^{(k)} = \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right)$

return $D^{(n)}$

Running time: $\Theta(n^3)$

Example

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



$D^{(0)} = W$

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	∞	-5	0	∞
5	∞	∞	∞	6	0

$D^{(1)}$

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	5	-5	0	-2
5	∞	∞	∞	6	0

$D^{(2)}$

	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2	5	-5	0	-2
5	∞	∞	∞	6	0

$D^{(3)}$

0	3	8	4	-4
∞	0	∞	1	7
∞	4	0	5	11
2	-1	-5	0	-2
∞	∞	∞	6	0

$D^{(4)}$

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

Record all Shortest Paths

shortest-paths tree with root i. For each vertex $i \in V$, we define the ***predecessor subgraph*** of G for i as $G_{\pi,i} = (V_{\pi,i}, E_{\pi,i})$, where

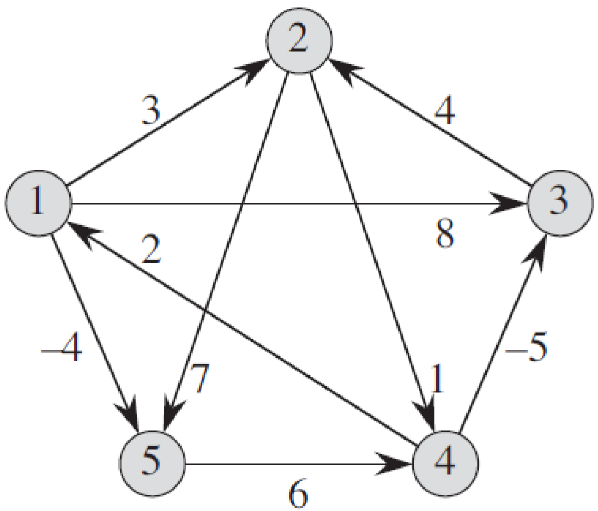
$$V_{\pi,i} = \{j \in V : \pi_{ij} \neq \text{NIL}\} \cup \{i\}$$

and

$$E_{\pi,i} = \{(\pi_{ij}, j) : j \in V_{\pi,i} - \{i\}\} .$$

```
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
```

Constructing a shortest path

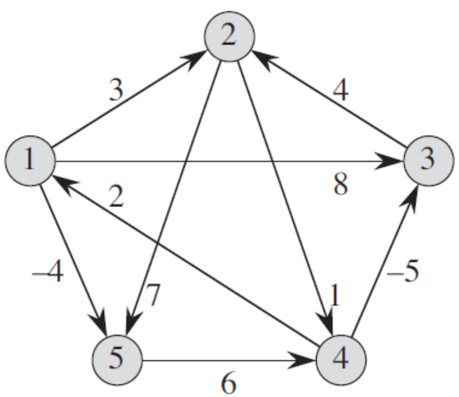


$$\pi_{ij}^{(0)} = \begin{cases} \text{NIL} & \text{if } i = j \text{ or } w_{ij} = \infty , \\ i & \text{if } i \neq j \text{ and } w_{ij} < \infty . \end{cases}$$

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

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

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



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

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

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

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

Transitive closure of a directed graph

Transitive closure of G is the graph G*=(V,E*), where E*={(i,j):there is a path from vertex i to vertex j in G*}:

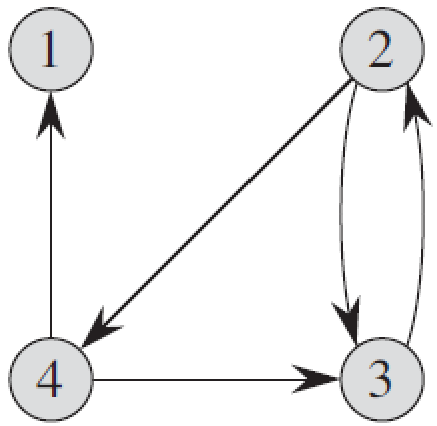
$$t_{ij}^{(0)} = \begin{cases} 0 & \text{if } i \neq j \text{ and } (i,j) \notin E , \\ 1 & \text{if } i = j \text{ or } (i,j) \in E , \end{cases}$$

and for $k \geq 1$,

$$t_{ij}^{(k)} = t_{ij}^{(k-1)} \vee \left(t_{ik}^{(k-1)} \wedge t_{kj}^{(k-1)} \right) .$$

TRANSITIVE-CLOSURE(G)

$n = |G.V|$
let $T^{(0)} = \left(t_{ij}^{(0)}\right)$ be a new $n \times n$ matrix
for $i = 1$ **to** n
$\quad$ **for** $j = 1$ **to** n
$\quad\quad$ **if** $i == j$ or $(i, j) \in G.E$
$\quad\quad\quad$ $t_{ij}^{(0)} = 1$
$\quad\quad$ **else** $t_{ij}^{(0)} = 0$
for $k = 1$ **to** n
$\quad$ let $T^{(k)} = \left(t_{ij}^{(k)}\right)$ be a new $n \times n$ matrix
$\quad$ **for** $i = 1$ **to** n
$\quad\quad$ **for** $j = 1$ **to** n
$\quad\quad\quad$ $t_{ij}^{(k)} = t_{ij}^{(k-1)} \vee \left(t_{ik}^{(k-1)} \wedge t_{kj}^{(k-1)}\right)$
return $T^{(n)}$



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

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

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

Johnson's algorithm for sparse graphs

- Uses both Dijkstra's algorithm and Bellman-Ford algorithm
- To use DijkstraAlgo:
  - Can get $O(V^2 \lg V + VE)$
  - But cannot allow negative weights
- Use Bellman-Ford Algo to **reweight** :
  - Add a resource node
  - Run Bellman-Ford algorithm
  - Assign weight to each node, reweight all edges

Preserving shortest paths by reweighting

1. For all pairs of vertices $u, v \in V$, a path p is a shortest path from u to v using weight function w if and only if p is also a shortest path from u to v using weight function $\hat{w}$.
2. For all edges (u, v), the new weight $\hat{w}(u, v)$ is nonnegative.

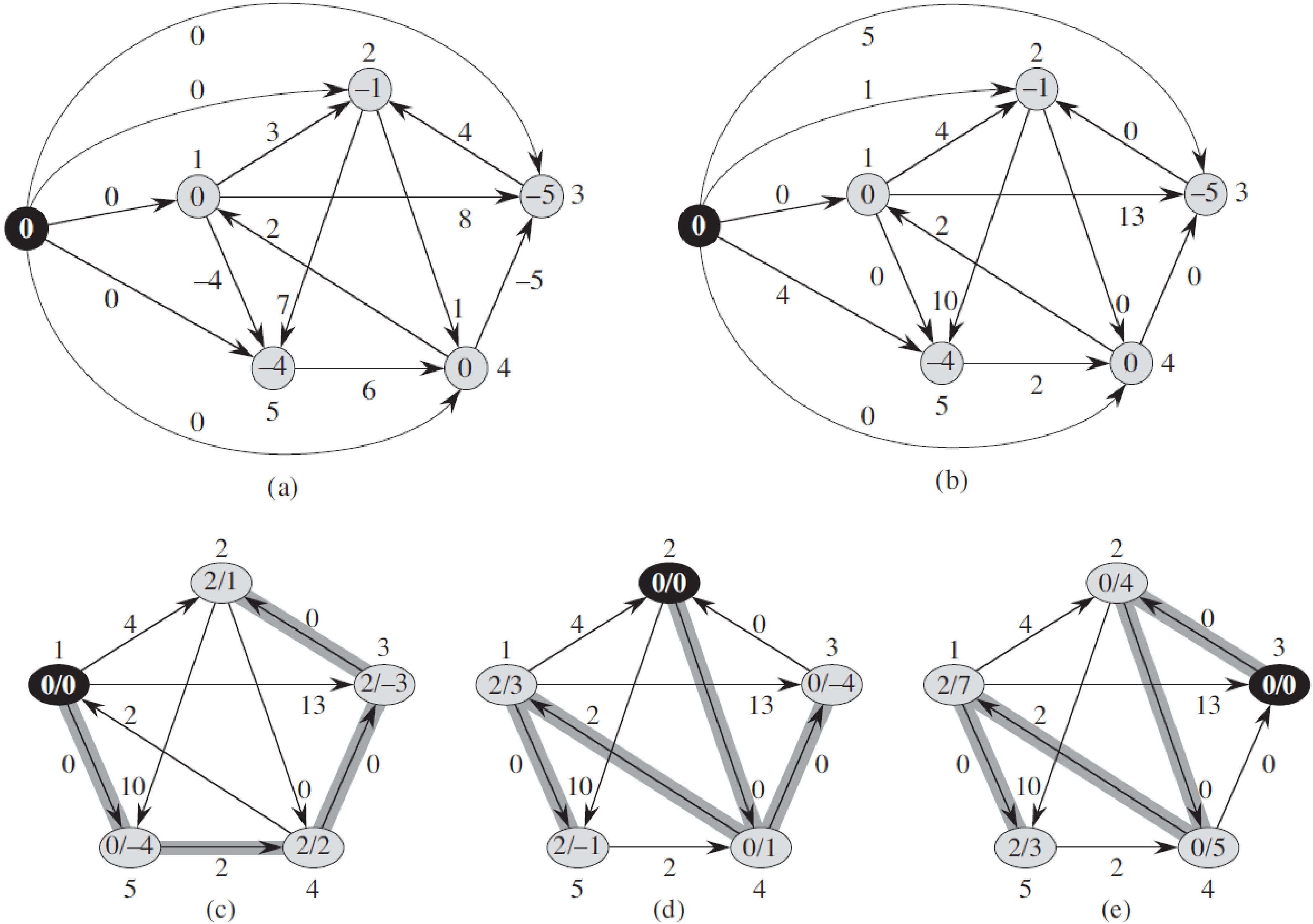
Lemma 25.1 (Reweighting does not change shortest paths)
Given a weighted, directed graph $G = (V, E)$ with weight function $w : E \to \mathbb{R}$, let $h : V \to \mathbb{R}$ be any function mapping vertices to real numbers. For each edge $(u, v) \in E$, define

$$\hat{w}(u, v) = w(u, v) + h(u) - h(v) . \qquad (25.9)$$

Let $p = \langle v_0, v_1, \ldots, v_k \rangle$ be any path from vertex v_0 to vertex v_k. Then p is a shortest path from v_0 to v_k with weight function w if and only if it is a shortest path with weight function $\hat{w}$. That is, $w(p) = \delta(v_0, v_k)$ if and only if $\hat{w}(p) = \hat{\delta}(v_0, v_k)$. Furthermore, G has a negative-weight cycle using weight function w if and only if G has a negative-weight cycle using weight function $\hat{w}$.

JOHNSON(G, w)

compute G', where $G'.V = G.V \cup \{s\}$,
    $G'.E = G.E \cup \{(s, v) : v \in G.V\}$, and
    $w(s, v) = 0$ for all $v \in G.V$
if BELLMAN-FORD(G', w, s) == FALSE
    print "the input graph contains a negative-weight cycle"
else for each vertex $v \in G'.V$
        set $h(v)$ to the value of $\delta(s, v)$
            computed by the Bellman-Ford algorithm
    for each edge $(u, v) \in G'.E$
        $\widehat{w}(u, v) = w(u, v) + h(u) - h(v)$
    let $D = (d_{uv})$ be a new $n \times n$ matrix
    for each vertex $u \in G.V$
        run DIJKSTRA$(G, \widehat{w}, u)$ to compute $\widehat{\delta}(u, v)$ for all $v \in G.V$
        for each vertex $v \in G.V$
            $d_{uv} = \widehat{\delta}(u, v) + h(v) - h(u)$
    return D



(a)
0
1
2
3
4
5
0
–1
–5
–4
0
3
4
8
2
–4
7
1
–5
6
(b)
13
10
(c)
0/0
2/1
2/–3
2/2
0/–4
(d)
0/0
2/3
0/–4
0/1
2/–1
(e)
0/4
2/7
0/0
0/5
2/3