Shortest Paths III: Floyd-Warshall

Instructor: Meng-Fen Chiang

COMPCSI220: WEEK 11

THE UNIVERSITY OF AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND

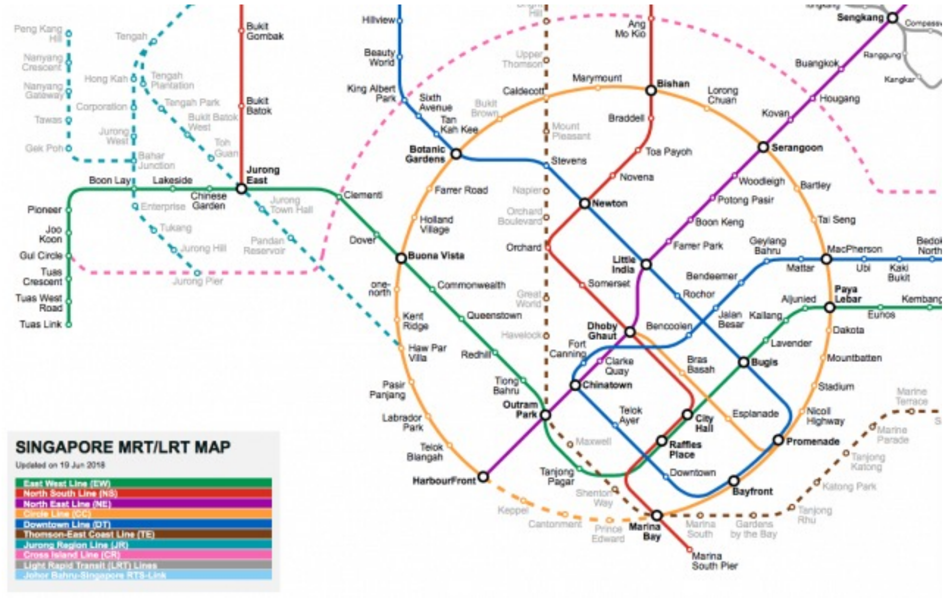
Slides adapted from Mark Wilson, Georgy Gimel'farb, Simone Linz and Tanya Gvozdeva

THE UNIVERSITY OF AUCKLAND
Te Whare Wananga o Tamaki Makaurau
NEW ZEALAND

OUTLINE

- Algorithms on Weighted Graphs
  - Dijkstra
  - Bellman-Ford
  - **Floyd-Warshall**

- All-Pairs Shortest Path



Shortest Path Algorithms

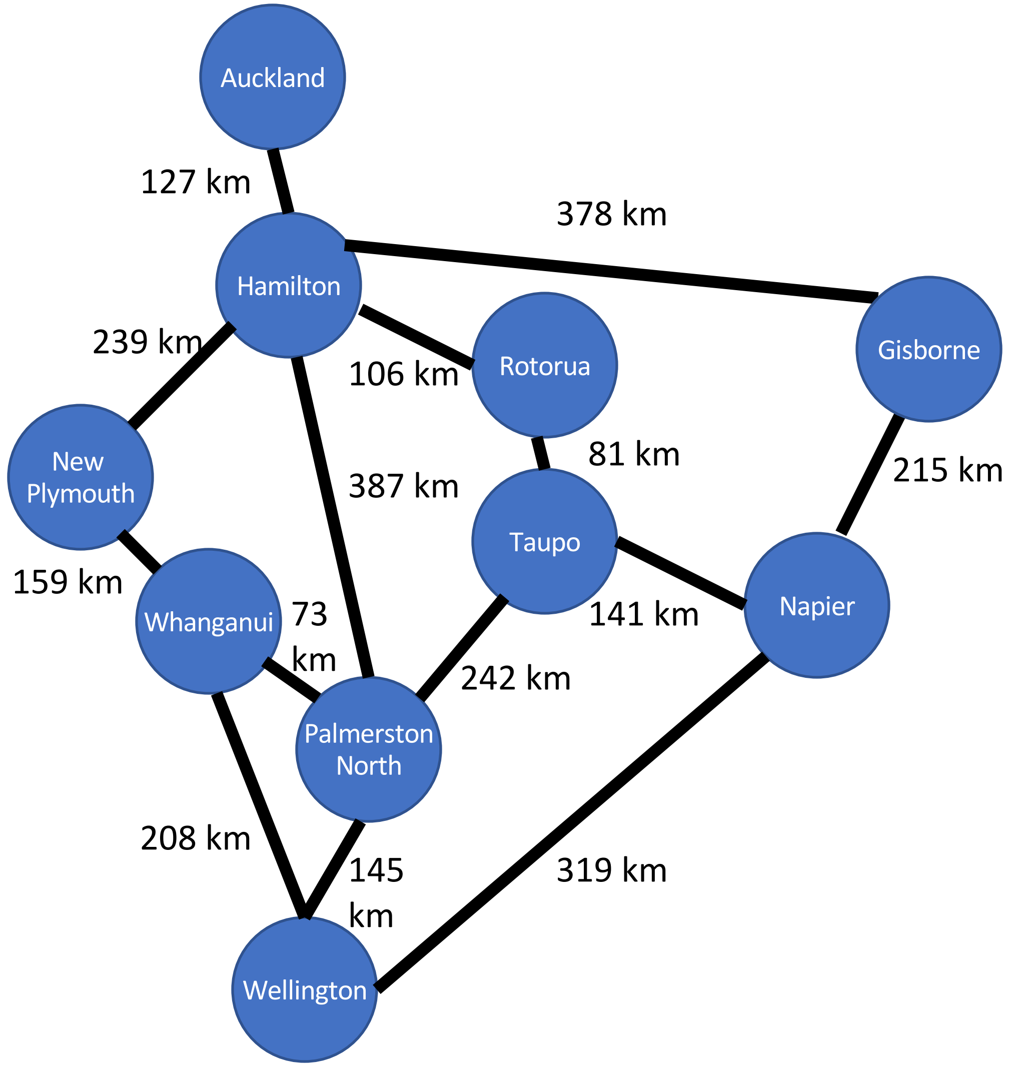
- **Dijkstra** provides the shortest path from **one** node to any other nodes in a graph
- **Bellman-Ford** is similar to Dijkstra but can handle **negative costs**
- **Floyd-Warshall** gives shortest paths between **all** pairs of nodes and can handle **negative costs**

Shortest Path Algorithms (Contd)

- Dijkstra provides the shortest path from **o** [illegible] any other nodes in a graph
- Bellman-Ford similar to Dijkstra bu [illegible] dle **negative costs**
- Floyd-Warshall gives shorte [illegible] etween **all** pairs of nodes and can handle **negative costs**

NO NEGATIVE CYCLE ALLOWED

Example: All-Pair-Shortest-Path



Source of distances: Google Maps

Example: All-Pairs-Shortest-Path

How can we produce a matrix that has the actual lowest costs rather than just ∞?

$$
\begin{array}{l|cccccccccc|}
 & \text{Auckland} & \text{Gisborne} & \text{Hamilton} & \text{Napier} & \text{New Plymouth} & \text{Palmerston North} & \text{Rotorua} & \text{Taupo} & \text{Wellington} & \text{Whanganui} \\
\hline
\text{Auckland} & 0 & \infty & 127 & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\text{Gisborne} & \infty & 0 & 378 & 215 & \infty & \infty & \infty & \infty & \infty & \infty \\
\text{Hamilton} & 127 & 378 & 0 & \infty & 239 & 387 & 106 & \infty & \infty & \infty \\
\text{Napier} & \infty & 215 & \infty & 0 & \infty & \infty & \infty & 141 & 319 & \infty \\
\text{New Plymouth} & \infty & \infty & 239 & \infty & 0 & \infty & \infty & \infty & \infty & 159 \\
\text{Palmerston North} & \infty & \infty & 387 & \infty & \infty & 0 & \infty & 242 & 145 & 73 \\
\text{Rotorua} & \infty & \infty & 106 & \infty & \infty & \infty & 0 & 81 & \infty & \infty \\
\text{Taupo} & \infty & \infty & \infty & 141 & \infty & 242 & 81 & 0 & \infty & \infty \\
\text{Wellington} & \infty & \infty & \infty & 319 & \infty & 145 & \infty & \infty & 0 & 208 \\
\text{Whanganui} & \infty & \infty & \infty & \infty & 159 & 73 & \infty & \infty & 208 & 0 \\
\hline
\end{array}
$$

Example: All-Pairs-Shortest-Path

After running Dijkstra’s or Bellman-Ford from Auckland, the matrix will look like this:

$$
\begin{array}{l|cccccccccc}
 & \text{Auckland} & \text{Gisborne} & \text{Hamilton} & \text{Napier} & \text{New Plymouth} & \text{Palmerston North} & \text{Rotorua} & \text{Taupo} & \text{Wellington} & \text{Whanganui} \\
\hline
\text{Auckland} & 0 & 505 & 127 & 455 & 366 & 514 & 233 & 314 & 659 & 525 \\
\text{Gisborne} & \infty & 0 & 378 & 215 & \infty & \infty & \infty & \infty & \infty & \infty \\
\text{Hamilton} & 127 & 378 & 0 & \infty & 239 & 387 & 106 & \infty & \infty & \infty \\
\text{Napier} & \infty & 215 & \infty & 0 & \infty & \infty & \infty & 141 & 319 & \infty \\
\text{New Plymouth} & \infty & \infty & 239 & \infty & 0 & \infty & \infty & \infty & \infty & 159 \\
\text{Palmerston North} & \infty & \infty & 387 & \infty & \infty & 0 & \infty & 242 & 145 & 73 \\
\text{Rotorua} & \infty & \infty & 106 & \infty & \infty & \infty & 0 & 81 & \infty & \infty \\
\text{Taupo} & \infty & \infty & \infty & 141 & \infty & 242 & 81 & 0 & \infty & \infty \\
\text{Wellington} & \infty & \infty & \infty & 319 & \infty & 145 & \infty & \infty & 0 & 208 \\
\text{Whanganui} & \infty & \infty & \infty & \infty & 159 & 73 & \infty & \infty & 208 & 0
\end{array}
$$

Options for All-Pairs-Shortest-Path

- Run Dijkstra's algorithm starting at each of the n vertices: $\Theta(n)$for iterating through the vertices, and $O(n^2)$ for each Dijkstra run. Total: $O(n^3)$

- Use the Bellman-Ford algorithm n times: $O(n^2m)$ at best, $O(n^4)$ at worst!

- Several algorithms are known; we present one, Floyd’s algorithm. Alternative to running Dijkstra from each node.

All Pairs Shortest Path Problem

- Number nodes (say from 0 to n−1) and at each step k, maintain matrix of shortest distances from node i to node j not passing through nodes higher than k. Update at each step to see whether node k shortens current best distance.

- Need triply nested for loops, so runs in $O(n^3)$ time. Better than Bellman Ford ($O(n^2m)$) for dense graphs.

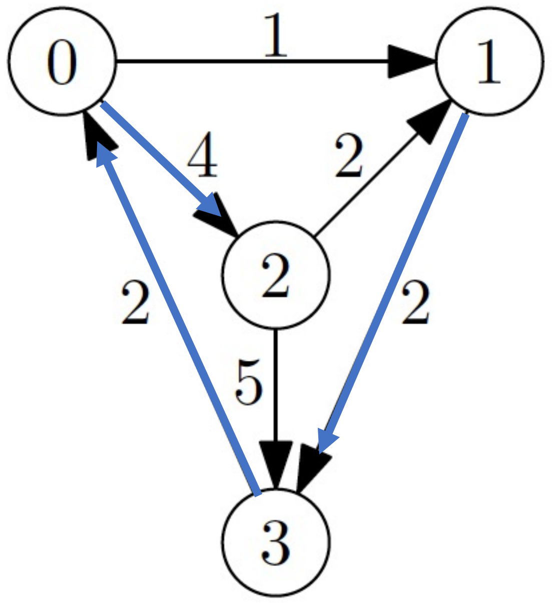
Floyd-Warshall Algorithm

- Definition. In the **all-pairs shortest path problem (APSP)** we are given a weighted (di-)graph (G, c), and must determine for each $u, v \in V(G)$ the weight of a minimum weight path from u to v.

- The solution to the all-pairs shortest path problem can be presented as a distance matrix.

Example 31.2

- For the digraph we have already calculated the all-pairs distance matrix in Example 28.11:



$$\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 0 & 1 & 4 & 3 \\ 1 & 4 & 0 & \boxed{8} & 2 \\ 2 & 6 & 2 & 0 & 4 \\ 3 & 2 & 3 & 6 & 0 \end{array}$$

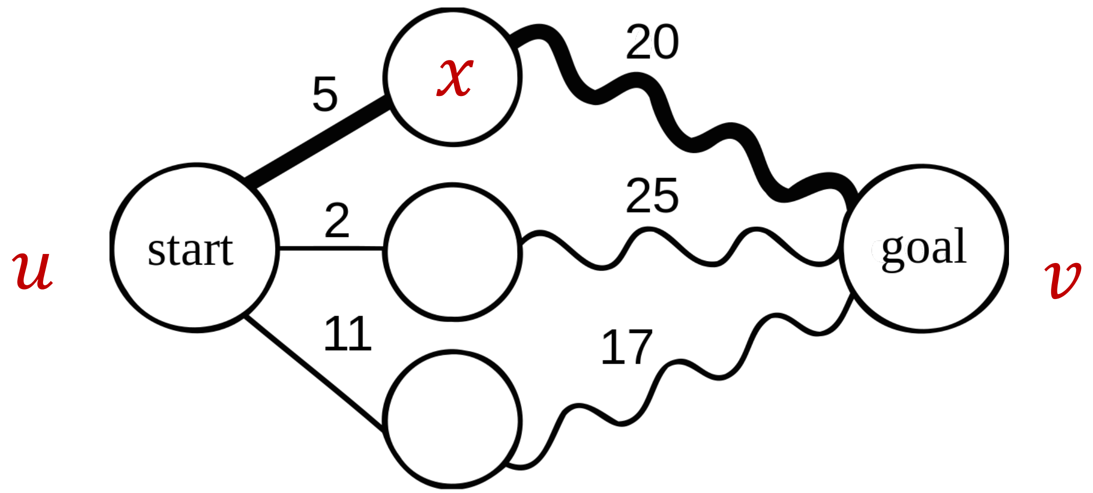
Floyd-Warshall Algorithm

Algorithm 1 Floyd's algorithm.

```
1: function FLOYD(weighted digraph$(G, c)$)
2:     array $d[0..n-1, 0..n-1]$
3:     $d \leftarrow c$
4:     for $x \in V(G)$ do
5:         for $u \in V(G)$ do
6:             for $v \in V(G)$ do
7:                 $d[u,v] \leftarrow \min(d[u,v], d[u,x] + d[x,v])$
8:     return $d$
```

Floyd-Warshall Algorithm

- This algorithm is based on the dynamic programming principles.
- At the bottom of the outer for loop, for each $u, v \in V(G), d[u, v]$ is the length of the shortest path from u to v passing through intermediate nodes that have been seen in for x loop.



Floyd-Warshall Algorithm

Algorithm 1 Floyd's algorithm.

```
1: function FLOYD(weighted digraph$(G, c)$)
2:     array $d[0..n-1, 0..n-1]$
3:     $d \leftarrow c$
4:     for $x \in V(G)$ do
5:         for $u \in V(G)$ do
6:             for $v \in V(G)$ do
7:                 $d[u,v] \leftarrow \min(d[u,v], d[u,x] + d[x,v])$
8:     return $d$
```

$x = 4$

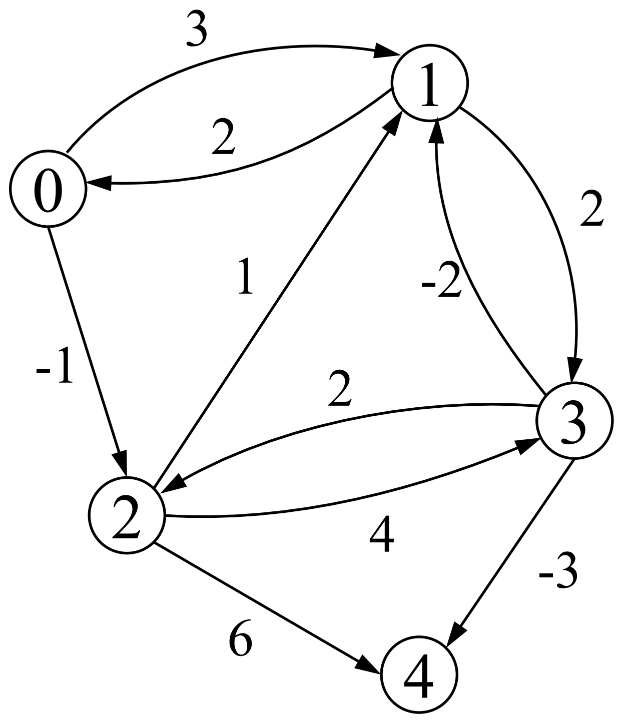
$u = 8$

$v = 2$

$d[8,2] = \min\{d[8,2], d[8,4] + d[4,2]\}$

Then minimum weight of a path: $8 - w_1 - w_2 - \cdots - w_n - 2$ such that $w_i \in \{0,1,2,3,4\}$

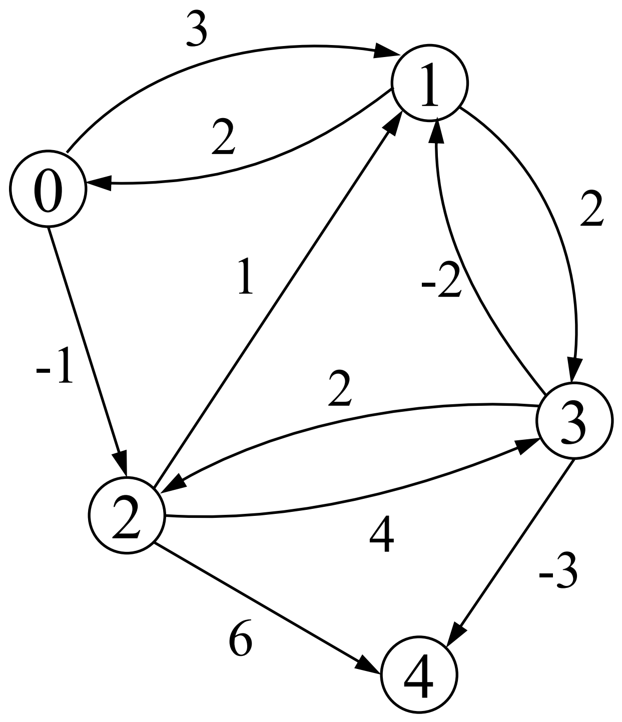
Illustrating Floyd's algorithm



$$\begin{array}{c|ccccc}
 & 0 & 1 & 2 & 3 & 4 \\
\hline
0 & 0 & 3 & -1 & \infty & \infty \\
1 & 2 & 0 & \infty & 2 & \infty \\
2 & \infty & 1 & 0 & 4 & 6 \\
3 & \infty & -2 & 2 & 0 & -3 \\
4 & \infty & \infty & \infty & \infty & 0
\end{array}$$

adj/cost matrix

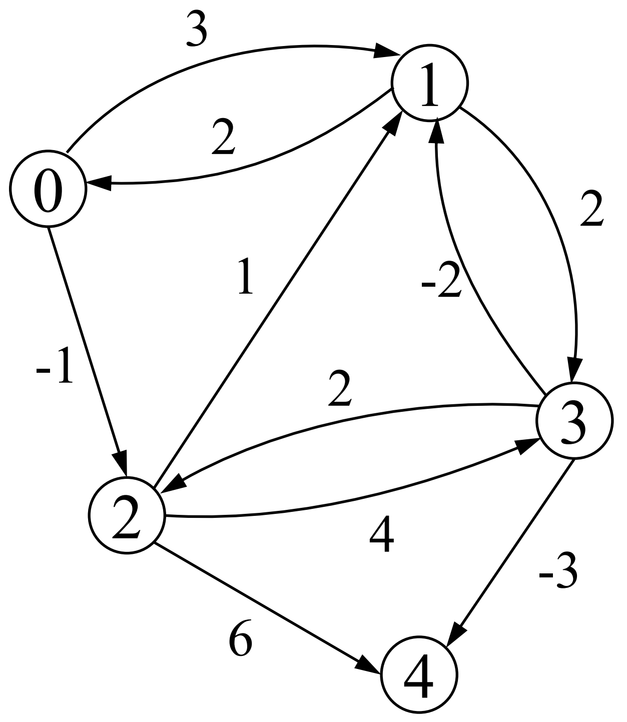
Illustrating Floyd’s algorithm



	0	1	2	3	4
0	0	3	-1	∞	∞
1	2	0	1	2	∞
2	∞	1	0	4	6
3	∞	-2	2	0	-3
4	∞	∞	∞	∞	0

$x = 0$

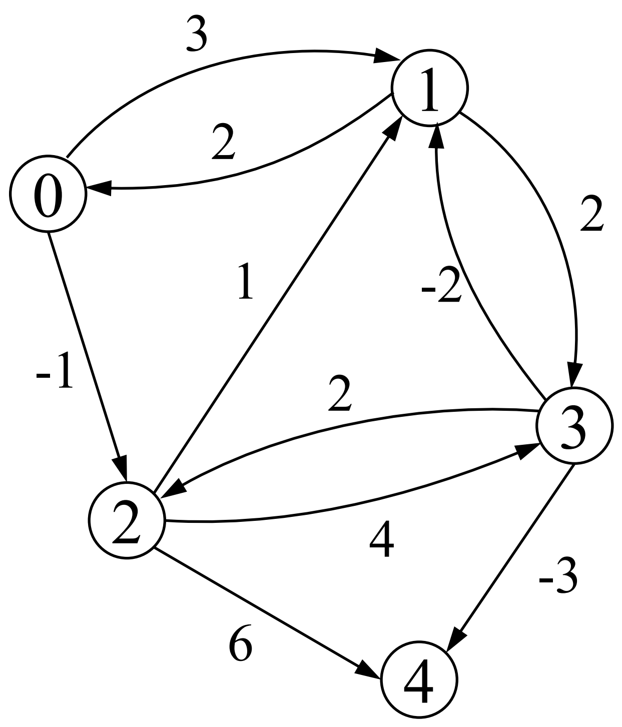
Illustrating Floyd's algorithm



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

$x = 1$

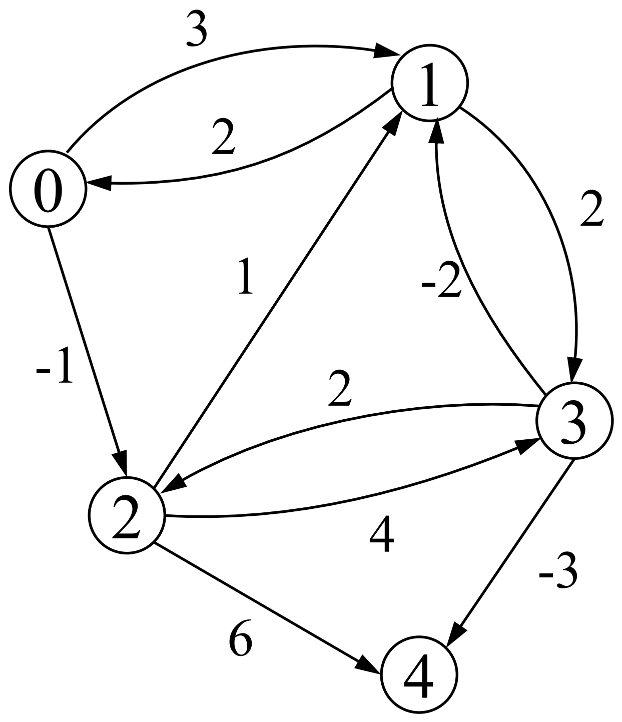
Illustrating Floyd’s algorithm



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

$x = 2$

Illustrating Floyd's algorithm



	0	1	2	3	4
0	0	0	-1	2	-1
1	2	0	1	2	-1
2	3	1	0	3	0
3	0	-2	-1	0	-3
4	∞	∞	∞	∞	0

$x = 3$

Illustrating Floyd's algorithm

$d[u,v] = \min(d[u,v], d[u,x] + d[x,v])$

$x = 0$

$$\begin{array}{c|ccccc}
 & 0 & 1 & 2 & 3 & 4 \\
\hline
0 & 0 & 3 & \boxed{-1} & \infty & \infty \\
1 & \boxed{2} & 0 & \boxed{\infty} & 2 & \infty \\
2 & \infty & 1 & 0 & 4 & 6 \\
3 & \infty & -2 & 2 & 0 & -3 \\
4 & \infty & \infty & \infty & \infty & 0
\end{array}$$

cost matrix

$$\begin{aligned} d[1,2] &= \min(d[1,2], d[1,0] + d[0,2]) \\ &= \min(\infty, 2 + (-1)) \\ &= 1 \end{aligned}$$

$$\begin{aligned} d[3,4] &= \min(d[3,4], d[3,0] + d[0,4]) \\ &= \min(-3, \infty + \infty) \\ &= -3 \end{aligned}$$

If $x = 0$, no update for

- $d[0,v] = \min(d[0,v], d[0,0] + d[0,v])$
  $= \min(d[0,v], d[0,v]) = d[0,v]$
- $d[u,0] = \min(d[u,0], d[u,0] + d[0,0])$
  $= \min(d[u,0], d[u,0]) = d[u,0]$

Illustrating Floyd's algorithm

$d[u,v] = \min(d[u,v], d[u,x] + d[x,v])$

$x = 1$

	0	1	2	3	4
0	0	3	-1	∞	∞
1	2	0	1	2	∞
2	∞	1	0	4	6
3	∞	-2	2	0	-3
4	∞	∞	∞	∞	0

resulting matrix for $x = 0$

$$d[0,3] = \min(d[0,3], d[0,1] + d[1,3]) = \min(\infty, 3 + 2) = 5$$

$$d[2,0] = \min(d[2,0], d[2,1] + d[1,0]) = \min(\infty, 1 + 2) = 3$$

$$d[2,3] = \min(d[2,3], d[2,1] + d[1,3]) = \min(4, 1 + 2) = 3$$

$$d[3,0] = \min(d[3,0], d[3,1] + d[1,0]) = \min(\infty, -2 + 2) = 0$$

$$d[3,2] = \min(d[3,2], d[3,1] + d[1,2]) = \min(2, -2 + 1) = -1$$

Illustrating Floyd's algorithm

$d[u, v] = \min(d[u, v], d[u, x] + d[x, v])$

$x = 2$

$$\begin{array}{c|ccccc}
 & 0 & 1 & 2 & 3 & 4 \\
\hline
0 & 0 & 3 & -1 & 5 & \infty \\
1 & 2 & 0 & 1 & 2 & \infty \\
2 & 3 & 1 & 0 & 3 & 6 \\
3 & 0 & -2 & -1 & 0 & -3 \\
4 & \infty & \infty & \infty & \infty & 0
\end{array}$$

resulting matrix for $x = 1$

$$\begin{aligned} d[0,1] &= \min(d[0,1], d[0,2] + d[2,1]) \\ &= \min(3, -1 + 1) = 0 \end{aligned}$$

$$\begin{aligned} d[0,3] &= \min(d[0,3], d[0,2] + d[2,3]) \\ &= \min(5, -1 + 3) = 2 \end{aligned}$$

$$\begin{aligned} d[0,4] &= \min(d[0,4], d[0,2] + d[2,4]) \\ &= \min(\infty, -1 + 6) = 5 \end{aligned}$$

$$\begin{aligned} d[1,4] &= \min(d[1,4], d[1,2] + d[2,4]) \\ &= \min(\infty, 1 + 6) = 7 \end{aligned}$$

Illustrating Floyd's algorithm

$d[u,v] = \min(d[u,v], d[u,x] + d[x,v])$

$x = 3$

$$\begin{array}{c|ccccc} & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 0 & 0 & -1 & 2 & 5 \\ 1 & 2 & 0 & 1 & 2 & 7 \\ 2 & 3 & 1 & 0 & 3 & 6 \\ 3 & 0 & -2 & -1 & 0 & -3 \\ 4 & \infty & \infty & \infty & \infty & 0 \end{array}$$

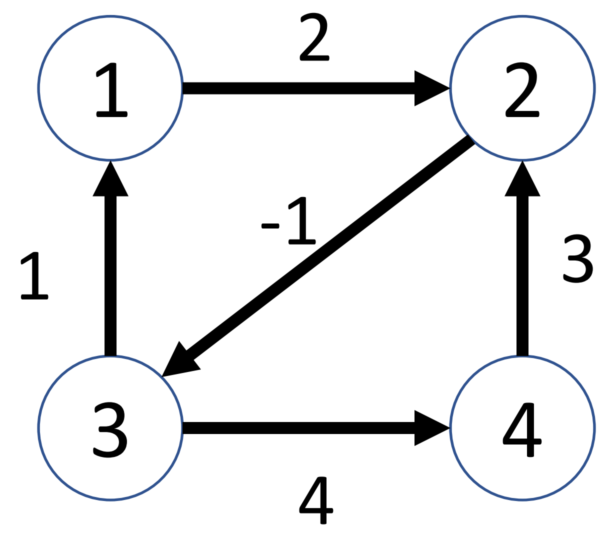
resulting matrix for $x = 2$

$$\begin{aligned} d[0,4] &= \min(d[0,4], d[0,3] + d[3,4]) \\ &= \min(5, 2 + (-3)) = -1 \end{aligned}$$

$$\begin{aligned} d[1,4] &= \min(d[1,4], d[1,3] + d[3,4]) \\ &= \min(7, 2 + (-3)) = -1 \end{aligned}$$

$$\begin{aligned} d[2,4] &= \min(d[2,4], d[2,3] + d[3,4]) \\ &= \min(6, 3 + (-3)) = 0 \end{aligned}$$

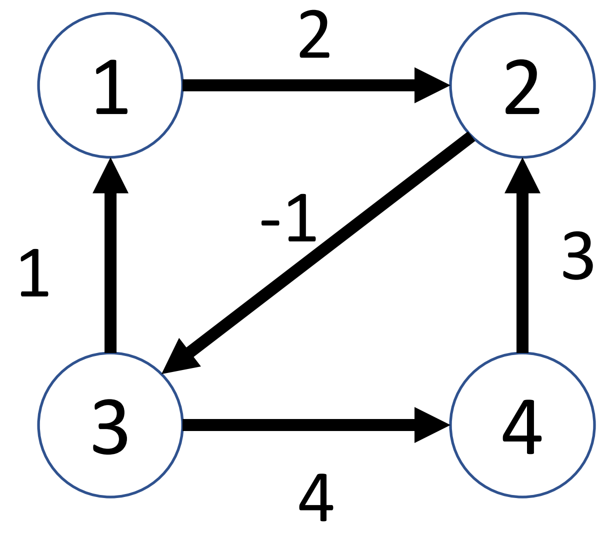
Example: Here is our Distance Matrix



	1	2	3	4
1				
2				
3				
4				

Example: Initialize our Matrix

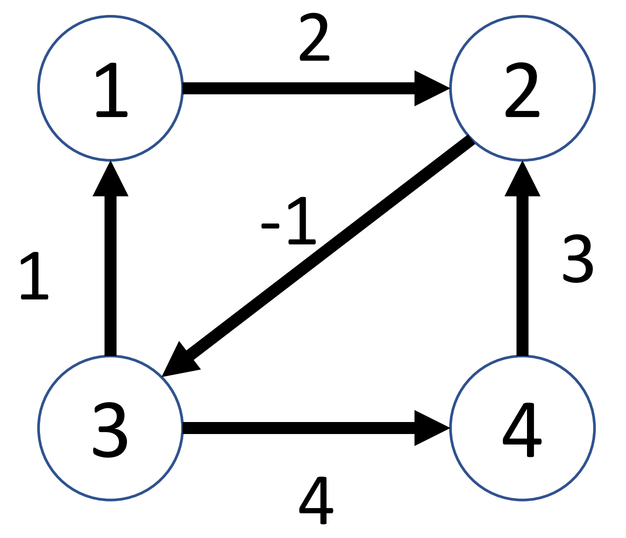
Start: Set the distance cost $d(x,y)$ of all pairs of vertices x and y to either $cost(x,v)$ or $+\infty$, except for $d(s,s)$, which is set to 0.



	1	2	3	4
1				
2				
3				
4				

Example: Initialize our Matrix

Start: Set the distance cost $d(x,y)$ of all pairs of vertices x and y to either $cost(x,v)$ or $+\infty$, except for $d(s,s)$, which is set to 0.

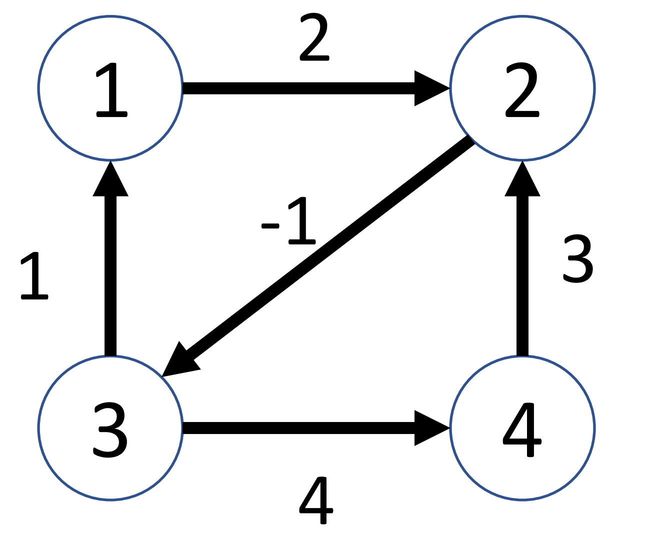


Set d(s,s) = 0

	1	2	3	4
1	0			
2		0		
3			0	
4				0

Example: Initialize our Matrix

Start: Set the distance cost $d(x,y)$ of all pairs of vertices x and y to either $cost(x,v)$ or $+\infty$, except for $d(s,s)$, which is set to 0.

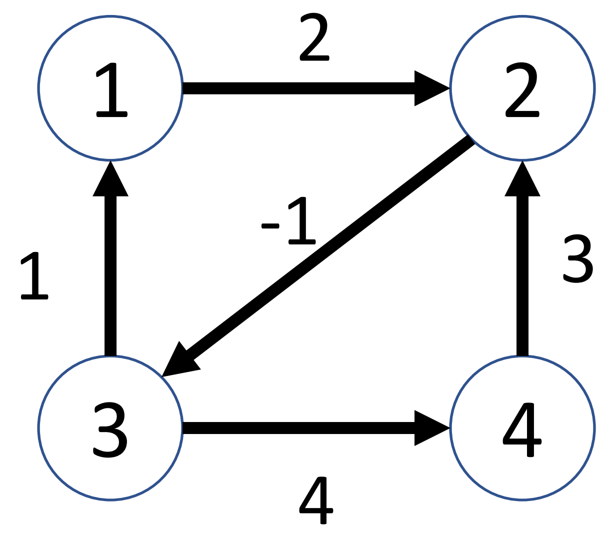


Set d(x,y) = cost(x,y) or +∞

	1	2	3	4
1	0			
2		0		
3			0	
4				0

Example: Initialize our Matrix

Start: Set the distance cost $d(x,y)$ of all pairs of vertices x and y to either $cost(x,v)$ or $+\infty$, except for $d(s,s)$, which is set to 0.

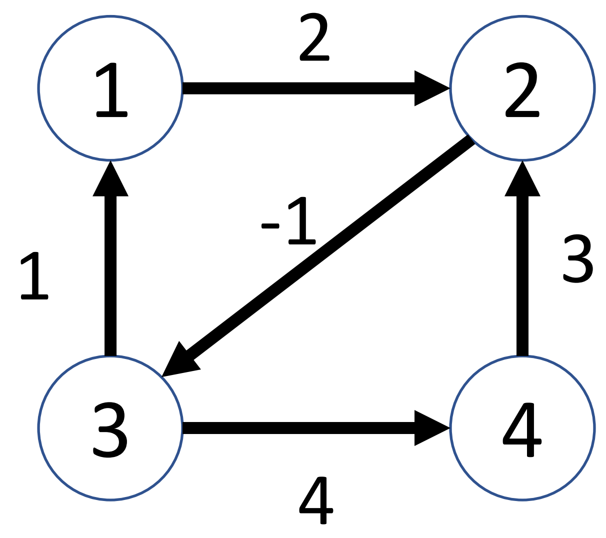


Set d(x,y) = cost(x,y) or +∞

	1	2	3	4
1	0			
2		0		
3			0	
4				0

Example: Initialize our Matrix

Start: Set the distance cost $d(x,y)$ of all pairs of vertices x and y to either $cost(x,v)$ or $+\infty$, except for $d(s,s)$, which is set to 0.

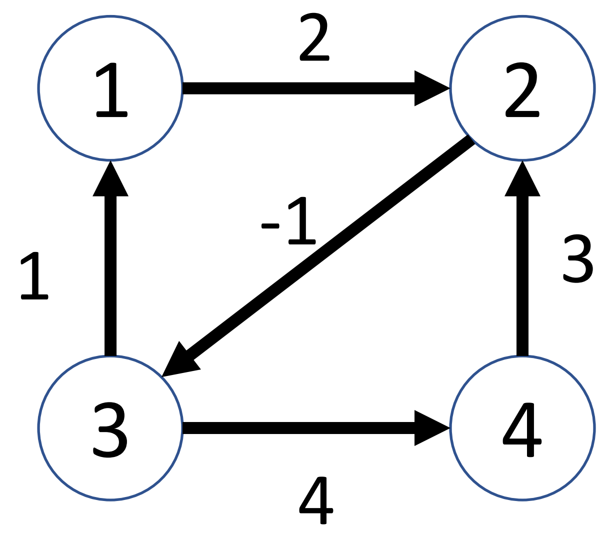


Set d(1,2) = cost(1,2)

	1	2	3	4
1	0	2		
2		0		
3			0	
4				0

Example: Initialize our Matrix

Start: Set the distance cost $d(x,y)$ of all pairs of vertices x and y to either $cost(x,v)$ or $+\infty$, except for $d(s,s)$, which is set to 0.

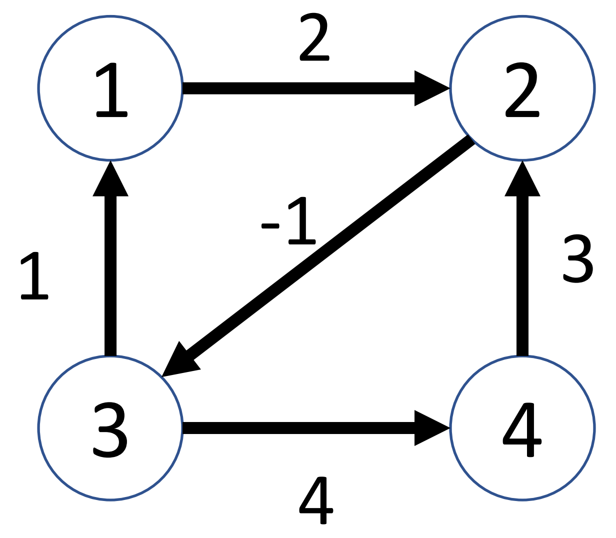


Set d(1,3) = cost(1,3)=+∞

	1	2	3	4
1	0	2	+∞	
2		0		
3			0	
4				0

Example: Initialize our Matrix

Start: Set the distance cost $d(x,y)$ of all pairs of vertices x and y to either $cost(x,v)$ or $+\infty$, except for $d(s,s)$, which is set to 0.

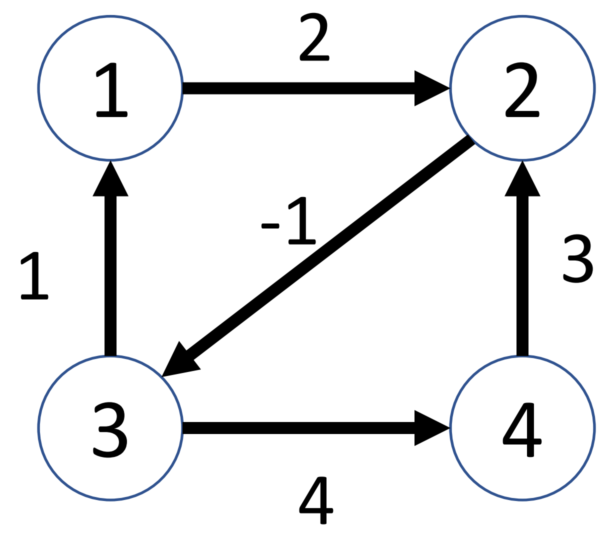


Set d(1,4) = cost(1,4)=+∞

	1	2	3	4
1	0	2	+∞	+∞
2		0		
3			0	
4				0

Example: Initialize our Matrix

Start: Set the distance cost $d(x,y)$ of all pairs of vertices x and y to either $cost(x,v)$ or $+\infty$, except for $d(s,s)$, which is set to 0.



Similarly for the rest of the nodes

	1	2	3	4
1	0	2	$+\infty$	$+\infty$
2	$+\infty$	0	-1	$+\infty$
3	1	$+\infty$	0	4
4	$+\infty$	3	$+\infty$	0

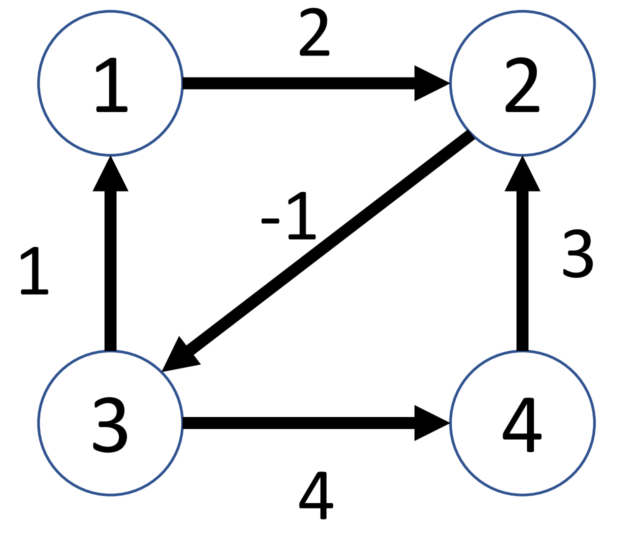
Example: The Nested Loop

k: 1 2 3 4

i: 1 2 3 4

j: 1 2 3 4

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$



	1	2	3	4
1	0	2	+∞	+∞
2	+∞	0	-1	+∞
3	1	+∞	0	4
4	+∞	3	+∞	0

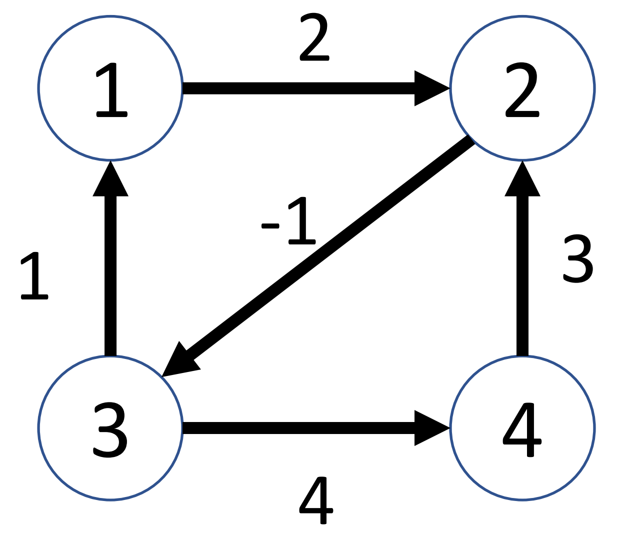
Example: The Nested Loop

k: **1** 2 3 4

i: **1** 2 3 4

j: **1** 2 3 4

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$



	1	2	3	4
1	0	2	+∞	+∞
2	+∞	0	-1	+∞
3	1	+∞	0	4
4	+∞	3	+∞	0

Example: The Nested Loop

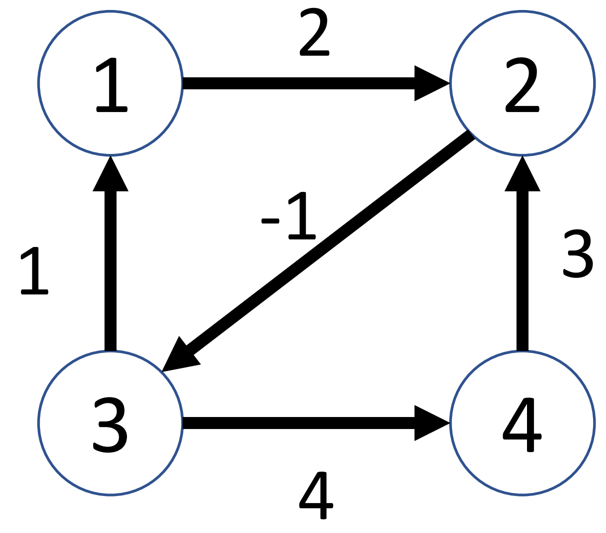
k: **1** 2 3 4

i: **1** 2 3 4

j: **1** 2 3 4

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(1,1) > d(1,1) + d(1,1)$$
$$0 > 0 + 0$$


$$\begin{tikzcd}
1 \arrow[r, "2"] & 2 \arrow[dl, "-1"'] \\
3 \arrow[u, "1"] \arrow[r, "4"'] & 4 \arrow[u, "3"']
\end{tikzcd}$$

	1	2	3	4
1	0	2	+∞	+∞
2	+∞	0	-1	+∞
3	1	+∞	0	4
4	+∞	3	+∞	0

Example: The Nested Loop

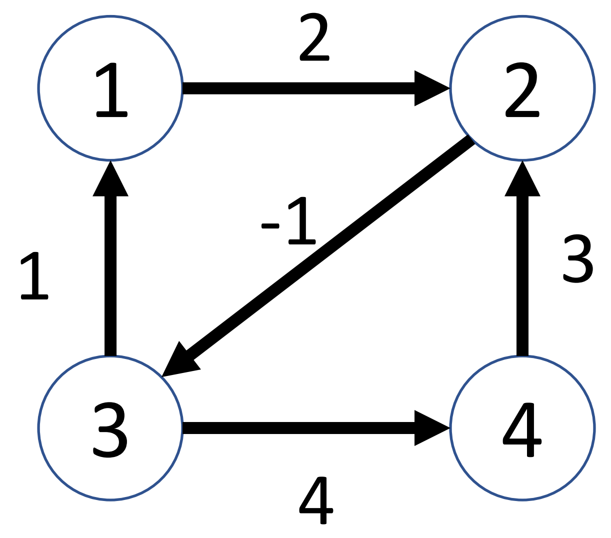
k: **1** 2 3 4

i: **1** 2 3 4

j: 1 **2** 3 4

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(1,2) > d(1,1) + d(1,2)$$
$$2 > 0 + 2$$



	1	2	3	4
1	0	2	+∞	+∞
2	+∞	0	-1	+∞
3	1	+∞	0	4
4	+∞	3	+∞	0

Example: The Nested Loop

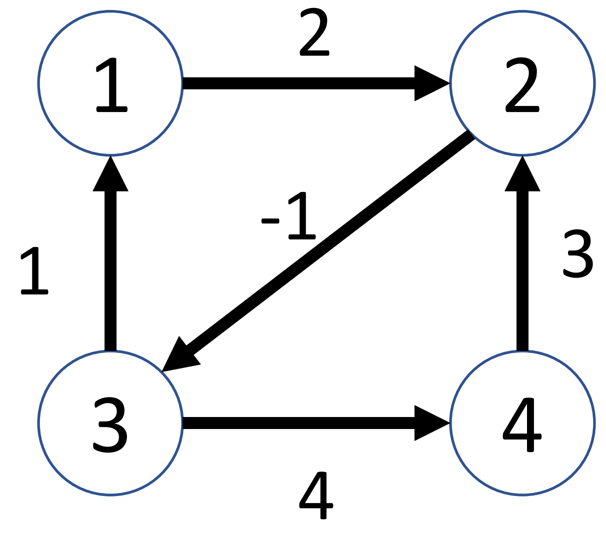
k: **1** 2 3 4

i: **1** 2 3 4

j: 1 2 **3** 4

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(1,3) > d(1,1) + d(1,3)$$
$$+\infty > 0 + +\infty$$



	1	2	3	4
1	0	2	+∞	+∞
2	+∞	0	-1	+∞
3	1	+∞	0	4
4	+∞	3	+∞	0

Example: The Nested Loop

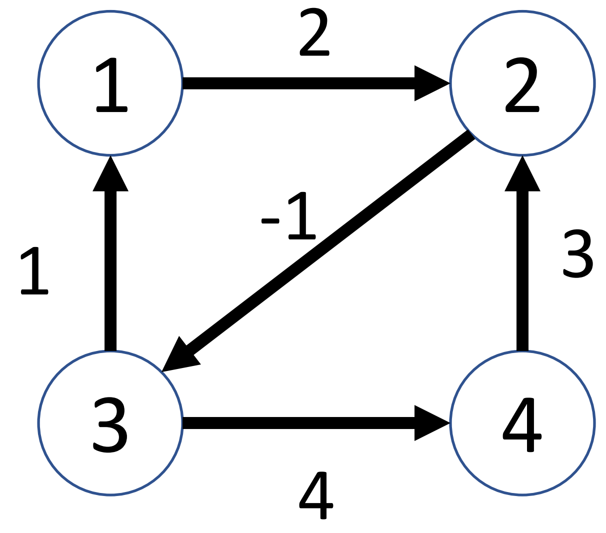
k: **1** 2 3 4

i: **1** 2 3 4

j: 1 2 3 **4**

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(1,4) > d(1,1) + d(1,4)$$
$$+\infty > 0 + +\infty$$



	1	2	3	4
1	0	2	+∞	+∞
2	+∞	0	-1	+∞
3	1	+∞	0	4
4	+∞	3	+∞	0

Example: The Nested Loop

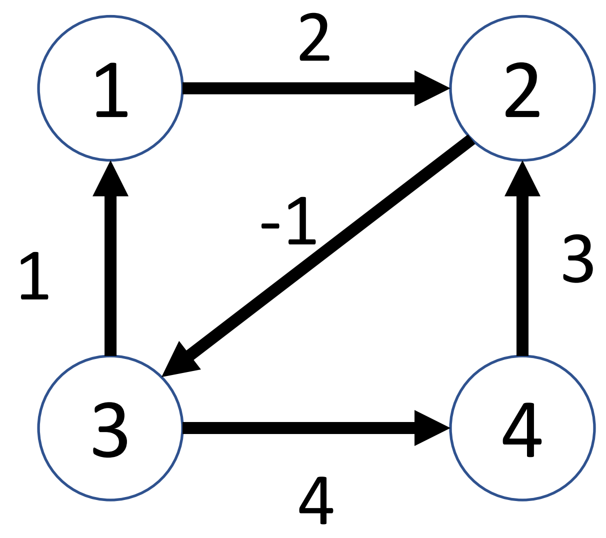
k: **1** 2 3 4

i: 1 **2** 3 4

j: **1** 2 3 4

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(2,1) > d(2,1) + d(1,1)$$
$$+\infty > +\infty + 0$$



	1	2	3	4
1	0	2	+∞	+∞
2	+∞	0	-1	+∞
3	1	+∞	0	4
4	+∞	3	+∞	0

Example: The Nested Loop

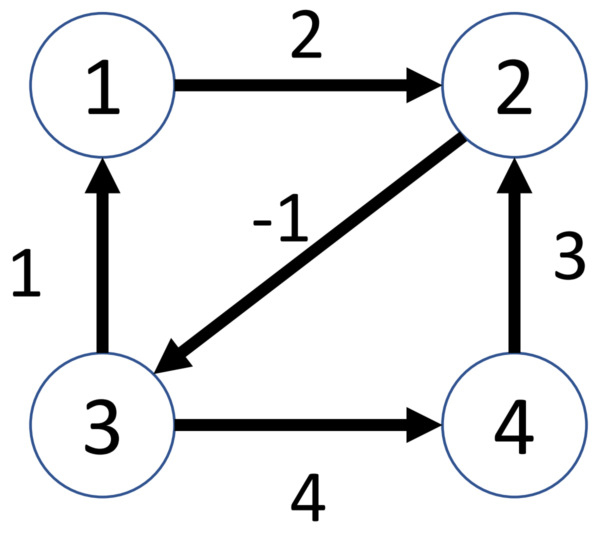
k: **1** 2 3 4

i: 1 **2** 3 4

j: 1 **2** 3 4

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(2,2) > d(2,1) + d(1,2)$$
$$0 > +\infty + 2$$



	1	2	3	4
1	0	2	+∞	+∞
2	+∞	0	-1	+∞
3	1	+∞	0	4
4	+∞	3	+∞	0

Example: The Nested Loop

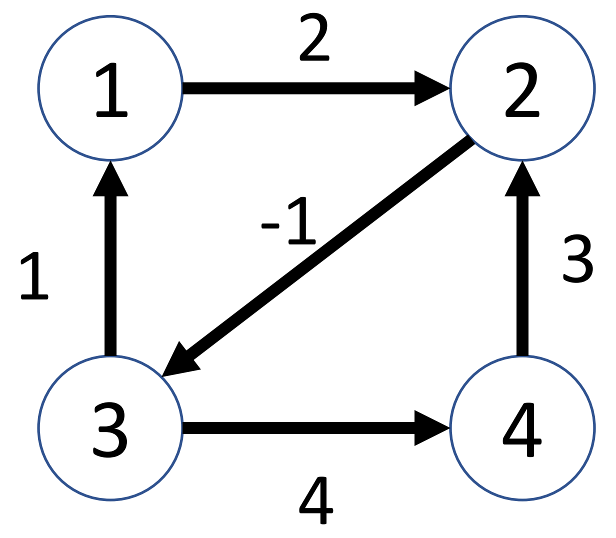
k: **1** 2 3 4

i: 1 **2** 3 4

j: 1 2 **3** 4

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(2,3) > d(2,1) + d(1,3)$$
$$-1 > +\infty + +\infty$$



	1	2	3	4
1	0	2	+∞	+∞
2	+∞	0	-1	+∞
3	1	+∞	0	4
4	+∞	3	+∞	0

Example: The Nested Loop

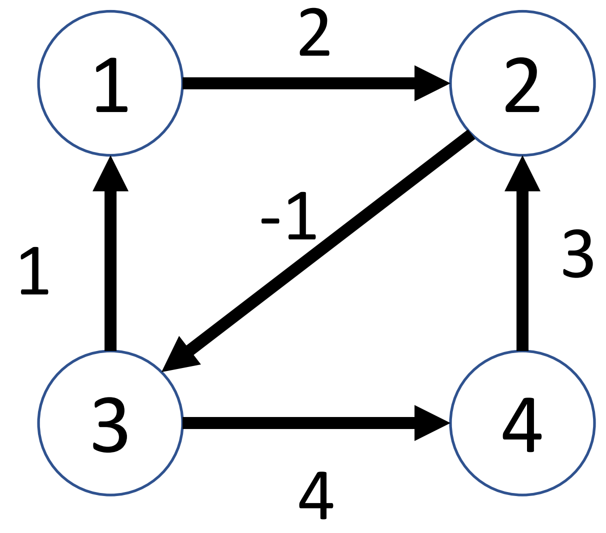
k: **1** 2 3 4

i: 1 **2** 3 4

j: 1 2 3 **4**

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(2,4) > d(2,1) + d(1,4)$$
$$+\infty > +\infty + +\infty$$



	1	2	3	4
1	0	2	+∞	+∞
2	+∞	0	-1	+∞
3	1	+∞	0	4
4	+∞	3	+∞	0

Example: The Nested Loop

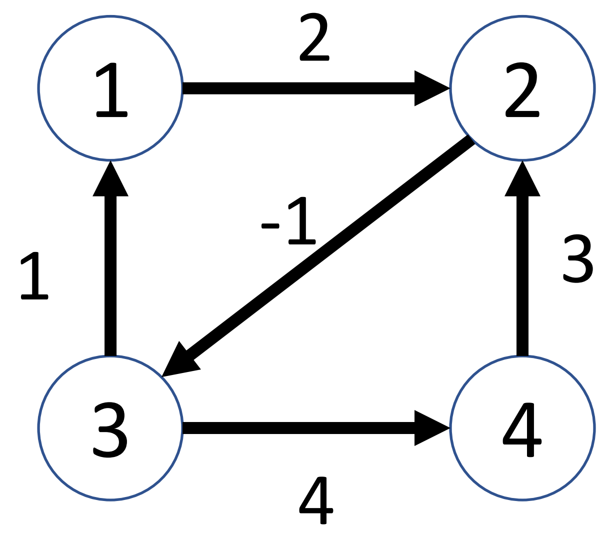
k: **1** 2 3 4

i: 1 2 **3** 4

j: **1** 2 3 4

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(3,1) > d(3,1) + d(1,1)$$
$$1 > 1 + +\infty$$



	1	2	3	4
1	0	2	+∞	+∞
2	+∞	0	-1	+∞
3	1	+∞	0	4
4	+∞	3	+∞	0

Example: The Nested Loop

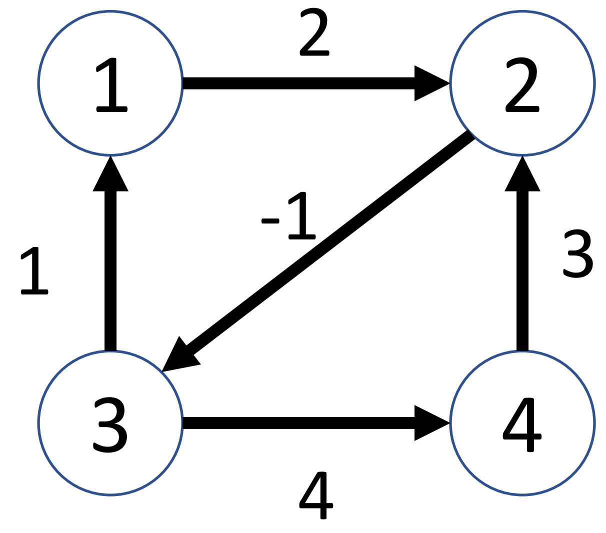
k: **1** 2 3 4

i: 1 2 **3** 4

j: 1 **2** 3 4

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(3,2) > d(3,1) + d(1,2)$$
$$+\infty > 1 + 2 \; \mathit{YES!}$$



	1	2	3	4
1	0	2	+∞	+∞
2	+∞	0	-1	+∞
3	1	+∞	0	4
4	+∞	3	+∞	0

Example: The Nested Loop

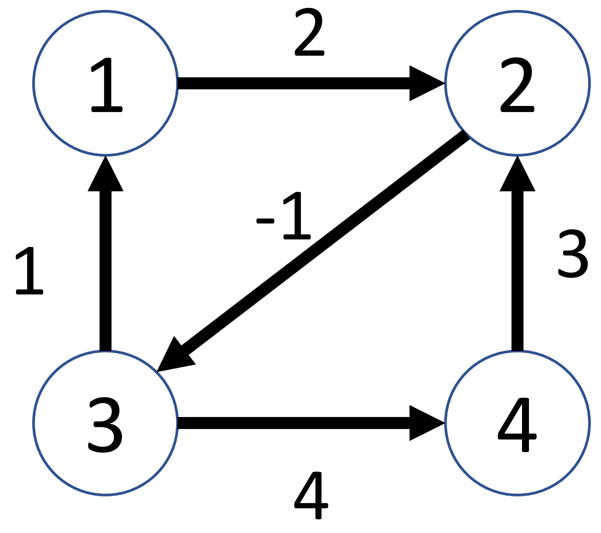
k: **1** 2 3 4

i: 1 2 **3** 4

j: 1 **2** 3 4

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(3,2) > d(3,1) + d(1,2)$$
$$d(3,2) < -1 + 2$$



	1	2	3	4
1	0	2	+∞	+∞
2	+∞	0	-1	+∞
3	1	3	0	4
4	+∞	3	+∞	0

Example: The Nested Loop

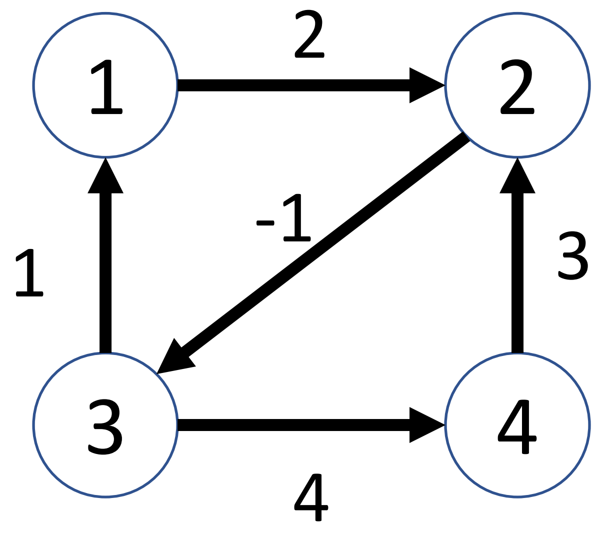
k: **1** 2 3 4

i: 1 2 **3** 4

j: 1 2 **3** 4

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(3,3) > d(3,1) + d(1,3)$$
$$0 > 1 + +\infty$$



	1	2	3	4
1	0	2	+∞	+∞
2	+∞	0	-1	+∞
3	1	3	0	4
4	+∞	3	+∞	0

Example: The Nested Loop

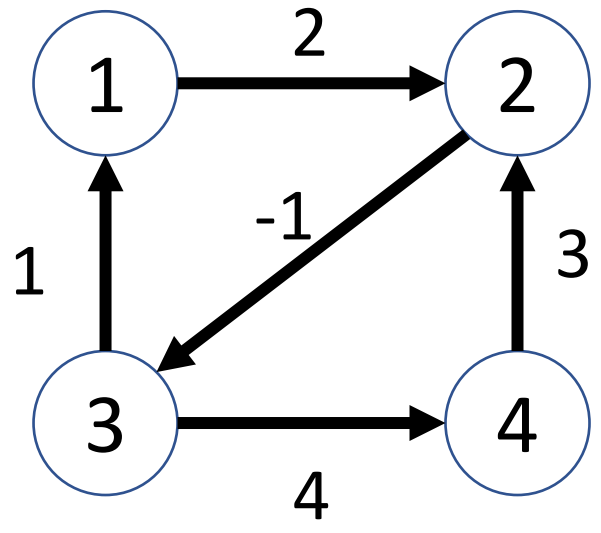
k: **1** 2 3 4

i: 1 2 **3** 4

j: 1 2 3 **4**

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(3,4) > d(3,1) + d(1,4)$$
$$4 > 1 + +\infty$$



	1	2	3	4
1	0	2	+∞	+∞
2	+∞	0	-1	+∞
3	1	3	0	4
4	+∞	3	+∞	0

Example: The Nested Loop

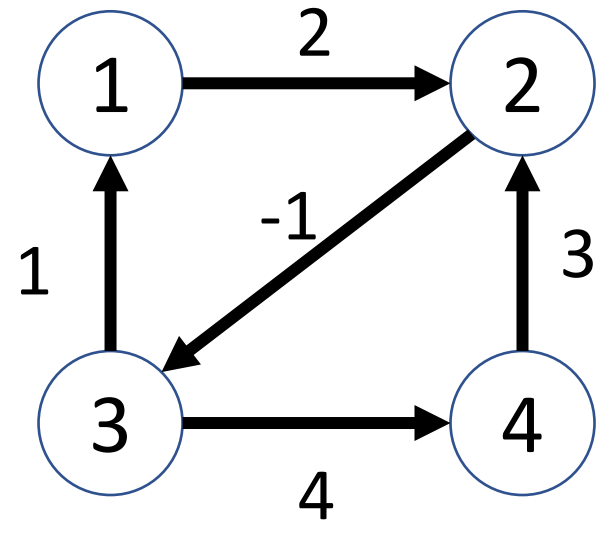
k: **1** 2 3 4

i: 1 2 3 **4**

j: **1** 2 3 4

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(4,1) > d(4,1) + d(1,1)$$
$$+\infty > +\infty + 0$$



	1	2	3	4
1	0	2	+∞	+∞
2	+∞	0	-1	+∞
3	1	3	0	4
4	+∞	3	+∞	0

Example: The Nested Loop

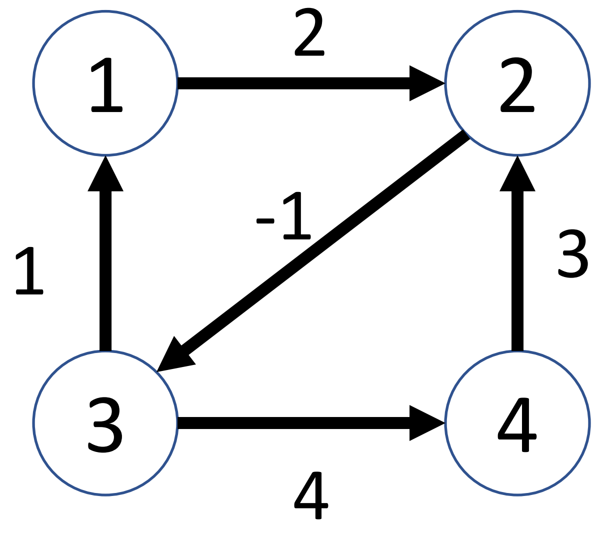
k: **1** 2 3 4

i: 1 2 3 **4**

j: 1 **2** 3 4

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(4,2) > d(4,1) + d(1,2)$$
$$3 > +\infty + 2$$



	1	2	3	4
1	0	2	+∞	+∞
2	+∞	0	-1	+∞
3	1	3	0	4
4	+∞	3	+∞	0

Example: The Nested Loop

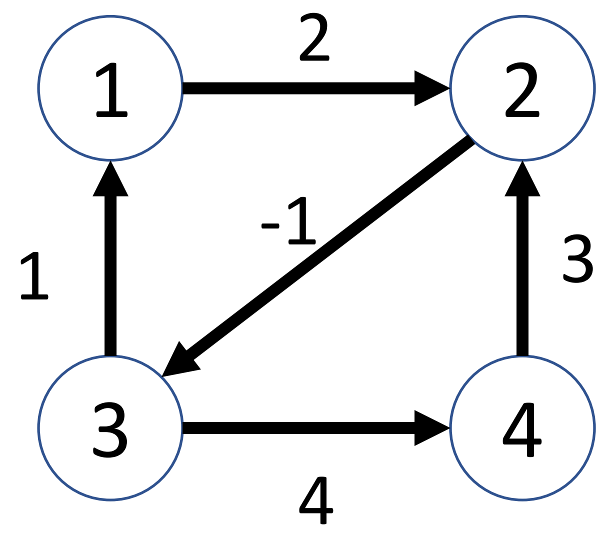
k: **1** 2 3 4

i: 1 2 3 **4**

j: 1 2 **3** 4

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(4,3) > d(4,1) + d(1,3)$$
$$+\infty > +\infty + +\infty$$



	1	2	3	4
1	0	2	+∞	+∞
2	+∞	0	-1	+∞
3	1	3	0	4
4	+∞	3	+∞	0

Example: The Nested Loop

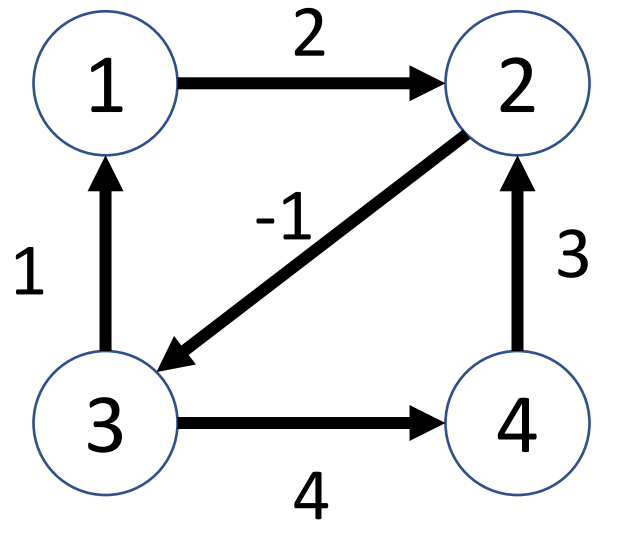
k: **1** 2 3 4

i: 1 2 3 **4**

j: 1 2 3 **4**

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(4,4) > d(4,1) + d(1,4)$$
$$0 > +\infty + +\infty$$



	1	2	3	4
1	0	2	+∞	+∞
2	+∞	0	-1	+∞
3	1	3	0	4
4	+∞	3	+∞	0

Example: The Nested Loop

k: 1 **2** 3 4

i: **1** 2 3 4

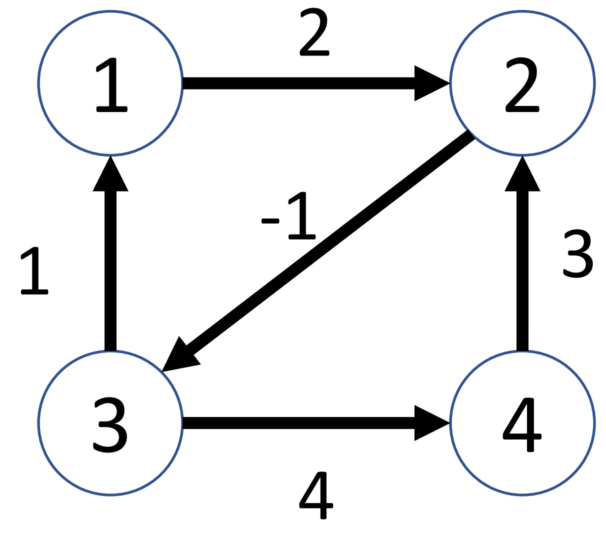
j: **1** 2 3 4

$$d(i,j) > d(i,k) + d(k,j)$$

$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(1,1) > d(1,2) + d(2,1)$$

$$0 > 2 + +\infty$$



	1	2	3	4
1	0	2	+∞	+∞
2	+∞	0	-1	+∞
3	1	3	0	4
4	+∞	3	+∞	0

Example: The Nested Loop

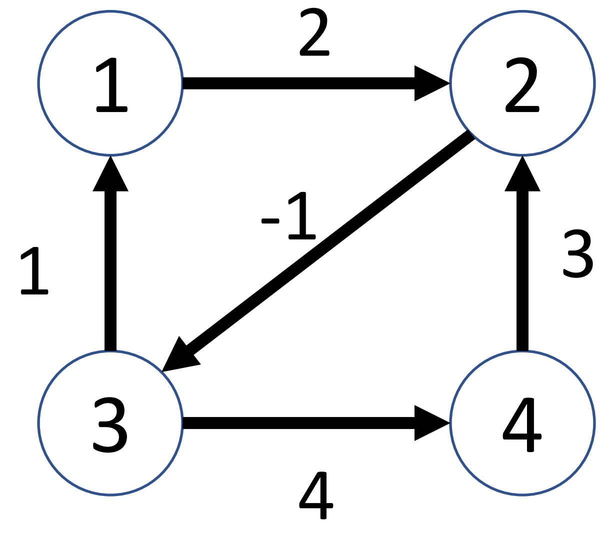
k: 1 **2** 3 4

i: **1** 2 3 4

j: 1 **2** 3 4

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(1,2) > d(1,2) + d(2,2)$$
$$2 > 2 + 0$$



	1	2	3	4
1	0	2	+∞	+∞
2	+∞	0	-1	+∞
3	1	3	0	4
4	+∞	3	+∞	0

Example: The Nested Loop

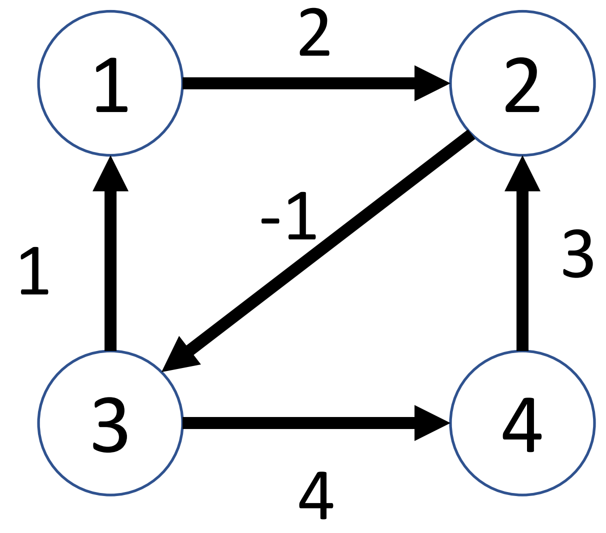
k: 1 **2** 3 4

i: **1** 2 3 4

j: 1 2 **3** 4

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(1,3) > d(1,2) + d(2,3)$$
$$+\infty > 2 - 1\ \mathit{YES!}$$



	1	2	3	4
1	0	2	+∞	+∞
2	+∞	0	-1	+∞
3	1	3	0	4
4	+∞	3	+∞	0

Example: The Nested Loop

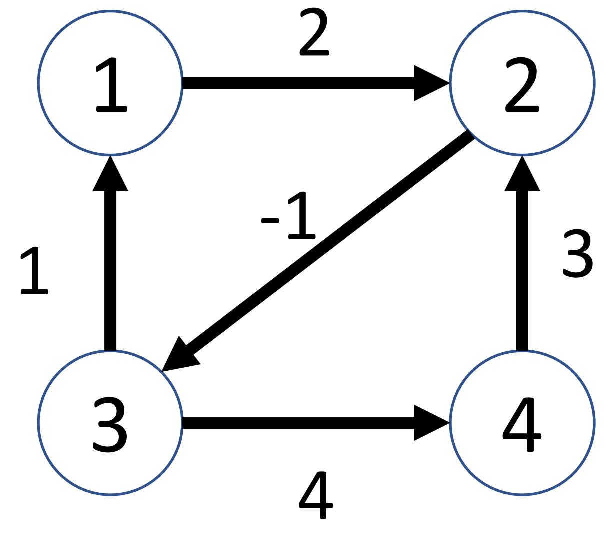
k: 1 **2** 3 4

i: **1** 2 3 4

j: 1 2 **3** 4

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(1,3) > d(1,2) + d(2,3)$$
$$d(1,3) < -2 - 1 = 1$$



	1	2	3	4
1	0	2	1	+∞
2	+∞	0	-1	+∞
3	1	3	0	4
4	+∞	3	+∞	0

Example: The Nested Loop

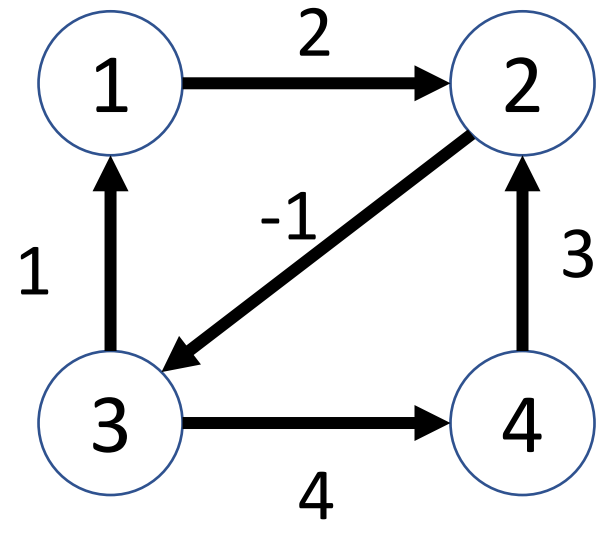
k: 1 **2** 3 4

i: **1** 2 3 4

j: 1 2 3 **4**

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(1,4) > d(1,2) + d(2,4)$$
$$+\infty > 2 + +\infty$$



	1	2	3	4
1	0	2	1	+∞
2	+∞	0	-1	+∞
3	1	3	0	4
4	+∞	3	+∞	0

Example: The Nested Loop

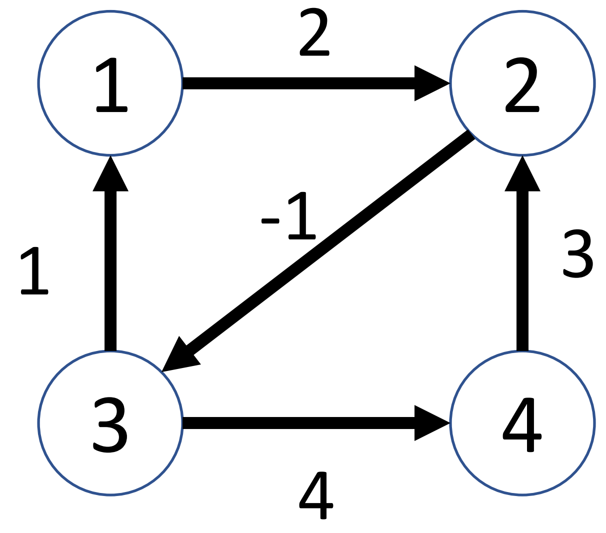
k: 1 **2** 3 4

i: 1 **2** 3 4

j: **1** 2 3 4

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(2,1) > d(2,2) + d(2,1)$$
$$+\infty > 0 + +\infty$$



	1	2	3	4
1	0	2	1	+∞
2	+∞	0	-1	+∞
3	1	3	0	4
4	+∞	3	+∞	0

Example: The Nested Loop

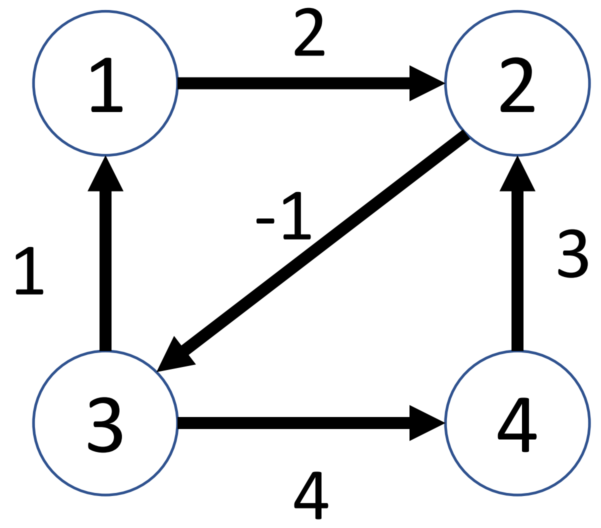
k: 1 **2** 3 4

i: 1 **2** 3 4

j: 1 **2** 3 4

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(2,2) > d(2,2) + d(2,2)$$
$$0 > 0 + 0$$



	1	2	3	4
1	0	2	1	+∞
2	+∞	0	-1	+∞
3	1	3	0	4
4	+∞	3	+∞	0

Example: The Nested Loop

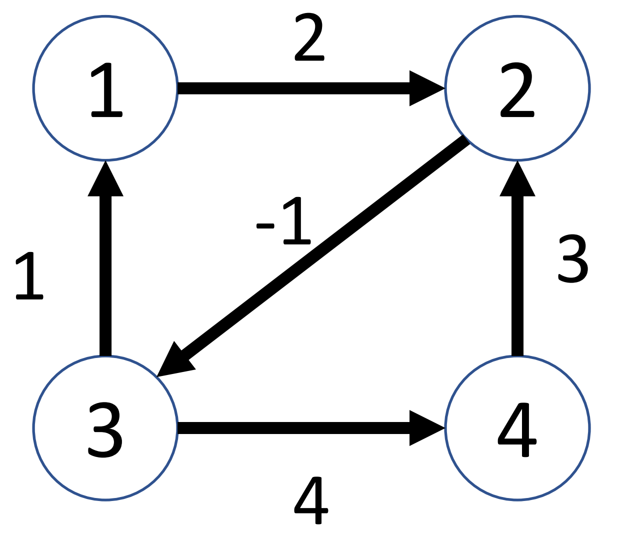
k: 1 **2** 3 4

i: 1 **2** 3 4

j: 1 2 **3** 4

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(2,3) > d(2,2) + d(2,3)$$
$$-1 > 0 + (-1)$$



	1	2	3	4
1	0	2	1	+∞
2	+∞	0	-1	+∞
3	1	3	0	4
4	+∞	3	+∞	0

Example: The Nested Loop

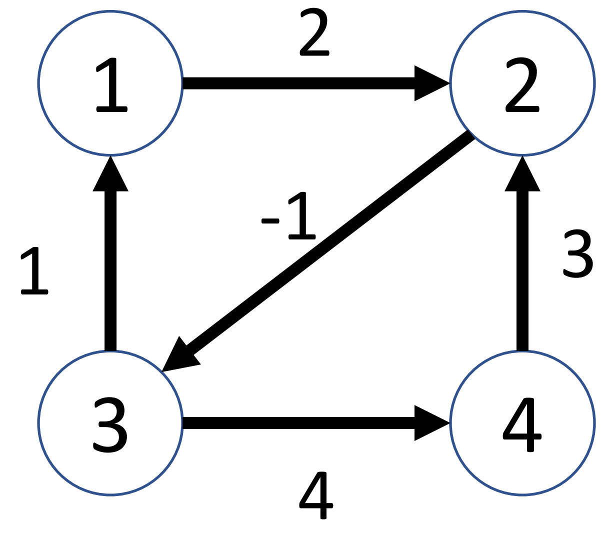
k: 1 **2** 3 4

i: 1 **2** 3 4

j: 1 2 3 **4**

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(2,4) > d(2,2) + d(2,4)$$
$$+\infty > 0 + +\infty$$



	1	2	3	4
1	0	2	1	+∞
2	+∞	0	-1	+∞
3	1	3	0	4
4	+∞	3	+∞	0

Example: The Nested Loop

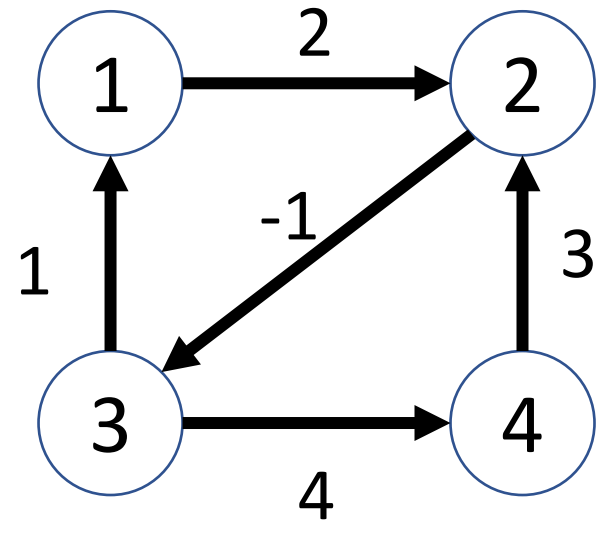
k: 1 **2** 3 4

i: 1 2 **3** 4

j: **1** 2 3 4

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(3,1) > d(3,2) + d(2,1)$$
$$1 > 3 + +\infty$$



	1	2	3	4
1	0	2	1	+∞
2	+∞	0	-1	+∞
3	1	3	0	4
4	+∞	3	+∞	0

Example: The Nested Loop

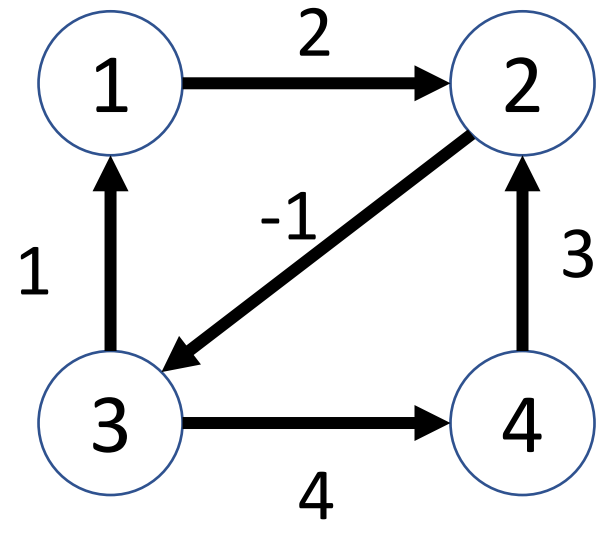
k: 1 **2** 3 4

i: 1 2 **3** 4

j: 1 **2** 3 4

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(3,2) > d(3,2) + d(2,2)$$
$$3 > 3 + 0$$



	1	2	3	4
1	0	2	1	+∞
2	+∞	0	-1	+∞
3	1	3	0	4
4	+∞	3	+∞	0

Example: The Nested Loop

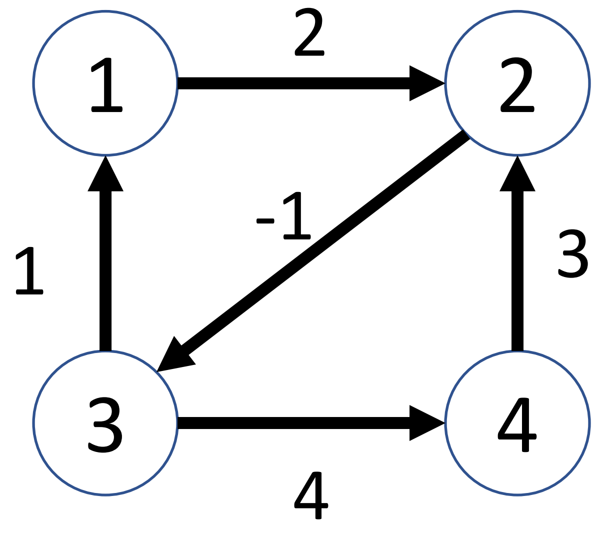
k: 1 **2** 3 4

i: 1 2 **3** 4

j: 1 2 **3** 4

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(3,3) > d(3,2) + d(2,3)$$
$$0 > 3 + (-1)$$



	1	2	3	4
1	0	2	1	+∞
2	+∞	0	-1	+∞
3	1	3	0	4
4	+∞	3	+∞	0

Example: The Nested Loop

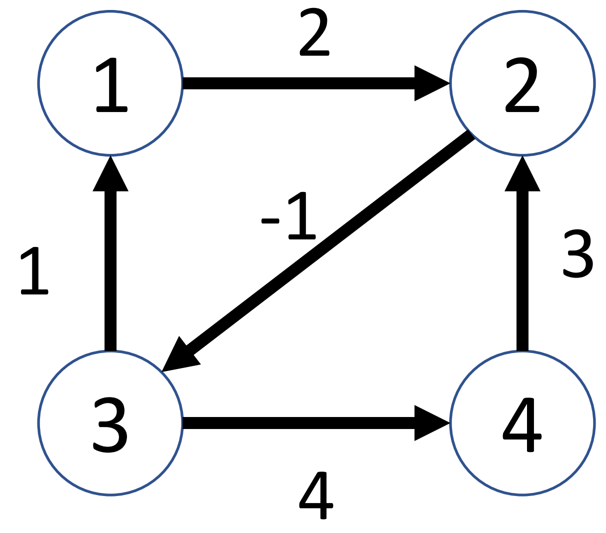
k: 1 **2** 3 4

i: 1 2 **3** 4

j: 1 2 3 **4**

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(3,4) > d(3,2) + d(2,4)$$
$$4 > 3 + +\infty$$



	1	2	3	4
1	0	2	1	+∞
2	+∞	0	-1	+∞
3	1	3	0	4
4	+∞	3	+∞	0

Example: The Nested Loop

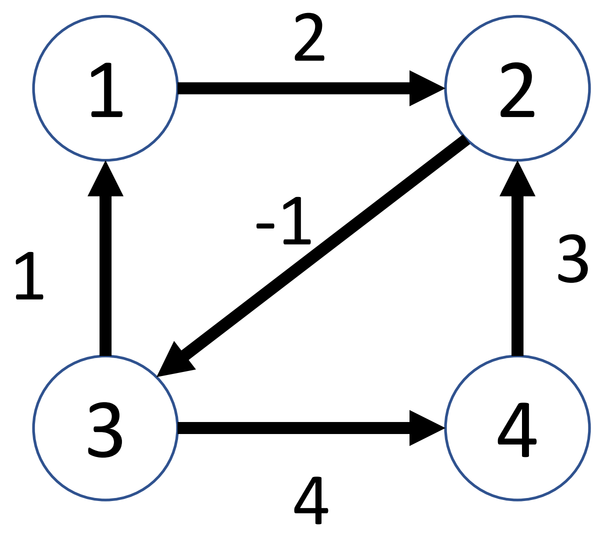
k: 1 **2** 3 4

i: 1 2 3 **4**

j: **1** 2 3 4

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(4,1) > d(4,2) + d(2,1)$$
$$+\infty > 3 + +\infty$$



	1	2	3	4
1	0	2	1	+∞
2	+∞	0	-1	+∞
3	1	3	0	4
4	+∞	3	+∞	0

Example: The Nested Loop

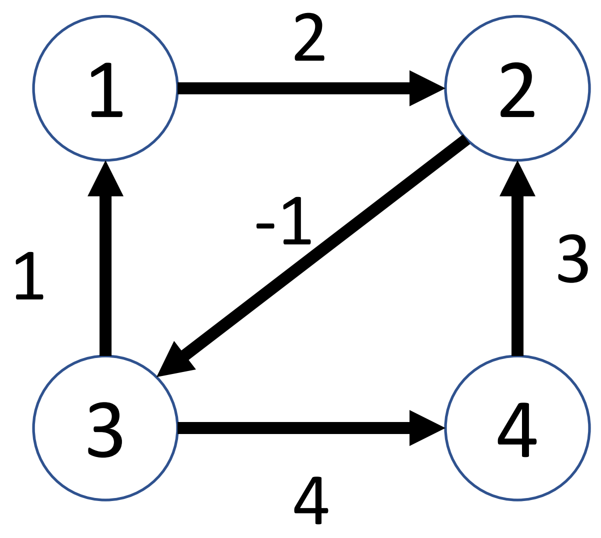
k: 1 **2** 3 4

i: 1 2 3 **4**

j: 1 **2** 3 4

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(4,2) > d(4,2) + d(2,2)$$
$$3 > 3 + 0$$



	1	2	3	4
1	0	2	1	+∞
2	+∞	0	-1	+∞
3	1	3	0	4
4	+∞	3	+∞	0

Example: The Nested Loop

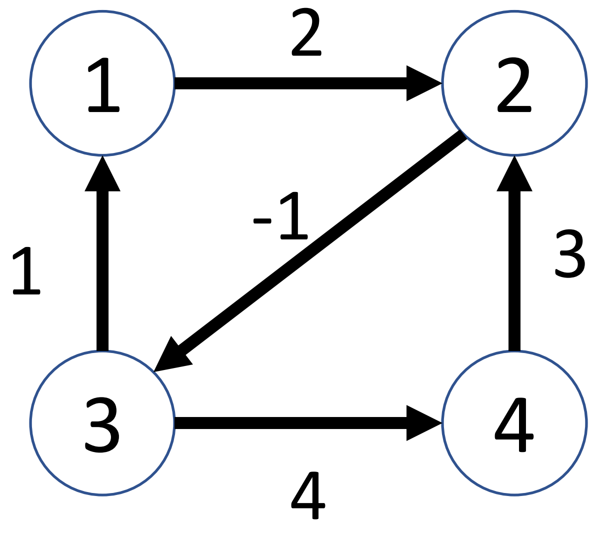
k: 1 **2** 3 4

i: 1 2 3 **4**

j: 1 2 **3** 4

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(4,3) > d(4,2) + d(2,3)$$
$$+\infty > 3 + (-1)\ YES$$



	1	2	3	4
1	0	2	1	+∞
2	+∞	0	-1	+∞
3	1	3	0	4
4	+∞	3	+∞	0

Example: The Nested Loop

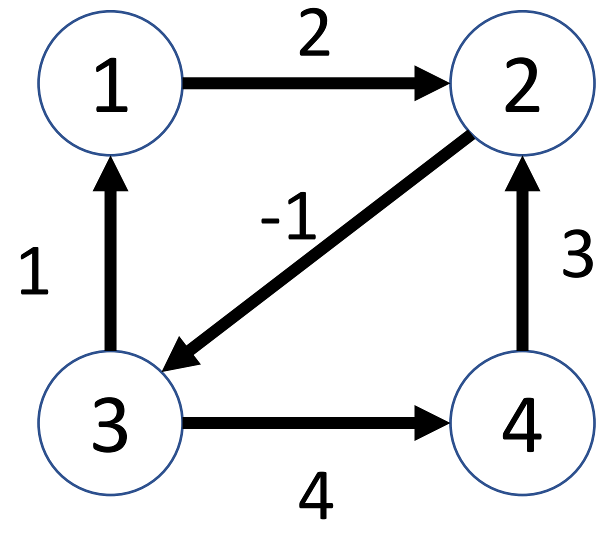
k: 1 **2** 3 4

i: 1 2 3 **4**

j: 1 2 **3** 4

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(4,3) > d(4,2) + d(2,3)$$
$$d(4,3) = -3 + (-1) = 2$$



	1	2	3	4
1	0	2	1	+∞
2	+∞	0	-1	+∞
3	1	3	0	4
4	+∞	3	2	0

Example: The Nested Loop

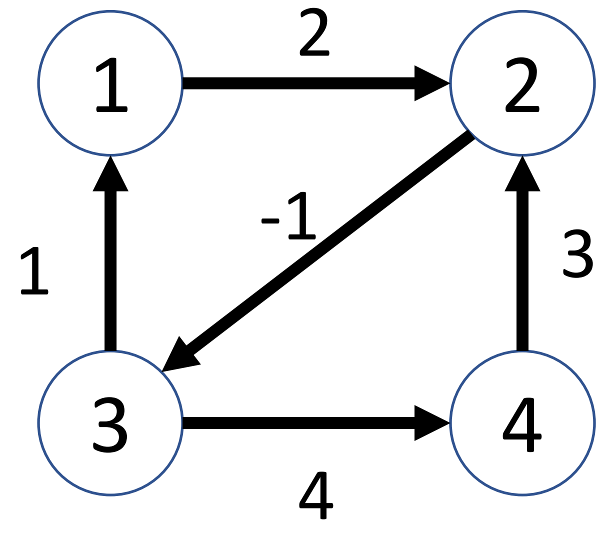
k: 1 **2** 3 4

i: 1 2 3 **4**

j: 1 2 3 **4**

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(4,4) > d(4,2) + d(2,4)$$
$$0 > 3 + +\infty$$



	1	2	3	4
1	0	2	1	+∞
2	+∞	0	-1	+∞
3	1	3	0	4
4	+∞	3	2	0

Example: The Nested Loop

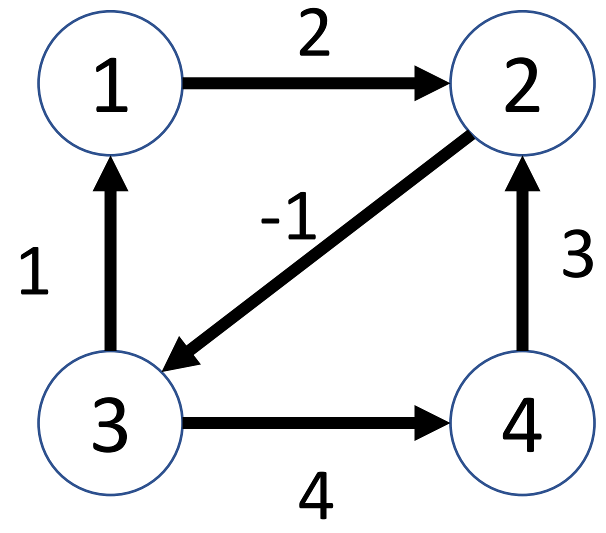
k: 1 2 **3** 4

i: **1** 2 3 4

j: 1 2 3 **4**

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(1,4)$$



	1	2	3	4
1	0	2	1	+∞
2	+∞	0	-1	+∞
3	1	3	0	4
4	+∞	3	2	0

Example: The Nested Loop

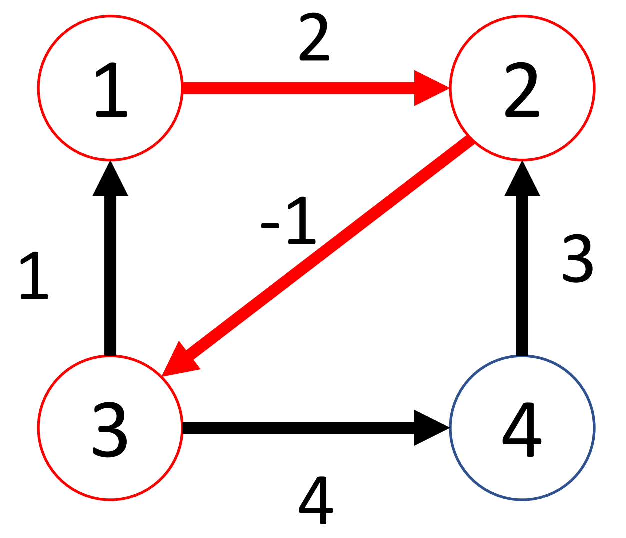
k: 1 2 **3** 4

i: **1** 2 3 4

j: 1 2 3 **4**

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(1,4)$$



	1	2	3	4
1	0	2	1	+∞
2	+∞	0	-1	+∞
3	1	3	0	4
4	+∞	3	2	0

Example: The Nested Loop

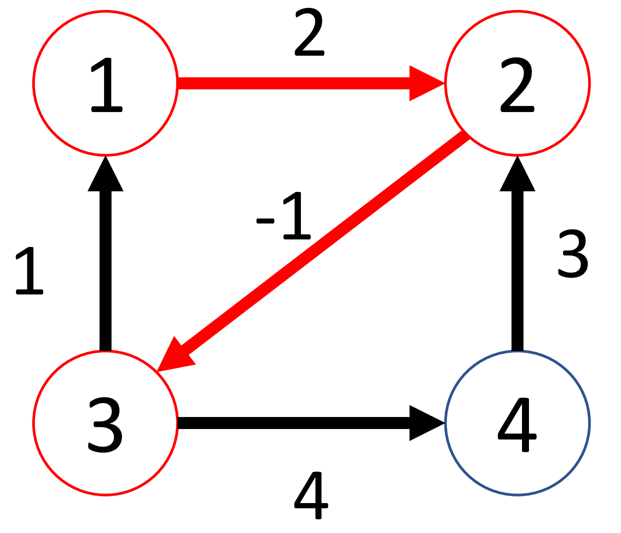
k: 1 2 **3** 4

i: **1** 2 3 4

j: 1 2 3 **4**

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(1,4) > d(1,3)$$



	1	2	3	4
1	0	2	1	+∞
2	+∞	0	-1	+∞
3	1	3	0	4
4	+∞	3	2	0

Example: The Nested Loop

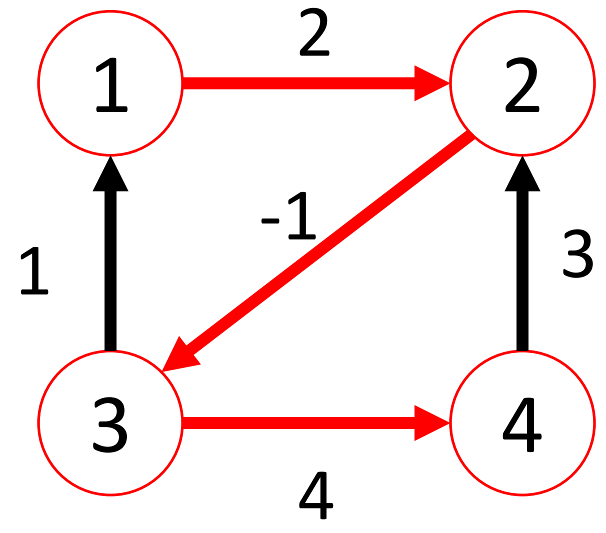
k: 1 2 **3** 4

i: **1** 2 3 4

j: 1 2 3 **4**

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(1,4) > d(1,3) + d(3,4)$$



	1	2	3	4
1	0	2	1	+∞
2	+∞	0	-1	+∞
3	1	3	0	4
4	+∞	3	2	0

Example: The Nested Loop

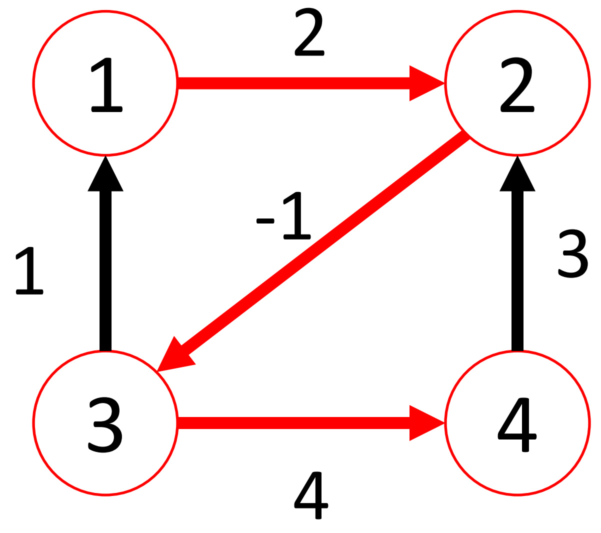
k: 1 2 **3** 4

i: **1** 2 3 4

j: 1 2 3 **4**

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(1,4) > d(1,3) + d(3,4)$$
$$+\infty > 1 + 4$$



	1	2	3	4
1	0	2	1	+∞
2	+∞	0	-1	+∞
3	1	3	0	4
4	+∞	3	2	0

Example: The Nested Loop

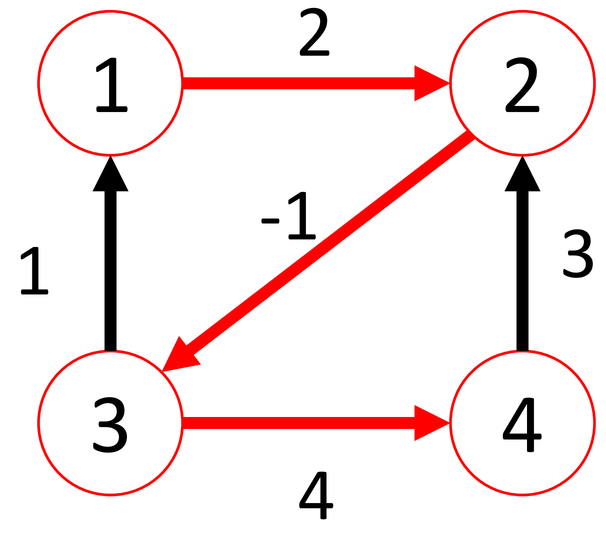
k: 1 2 **3** 4

i: **1** 2 3 4

j: 1 2 3 **4**

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(1,4) > d(1,3) + d(3,4)$$
$$d(1,4) = -5$$



	1	2	3	4
1	0	2	1	5
2	+∞	0	-1	+∞
3	1	3	0	4
4	+∞	3	2	0

Example: The Nested Loop

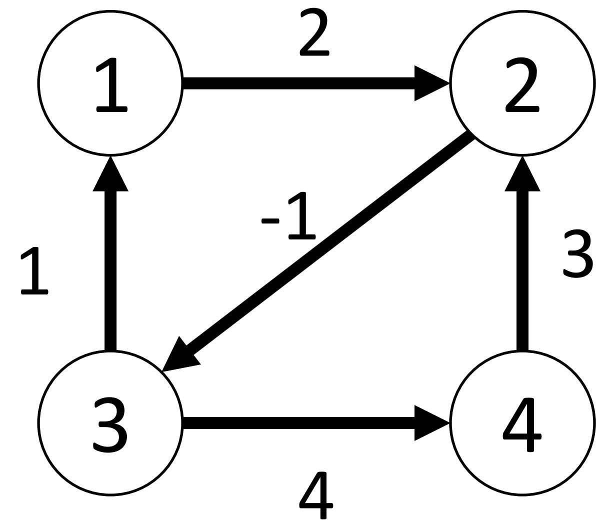
k: 1 2 **3** 4

i: 1 **2** 3 4

j: **1** 2 3 4

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(2,1)$$



	1	2	3	4
1	0	2	1	5
2	+∞	0	-1	+∞
3	1	3	0	4
4	+∞	3	2	0

Example: The Nested Loop

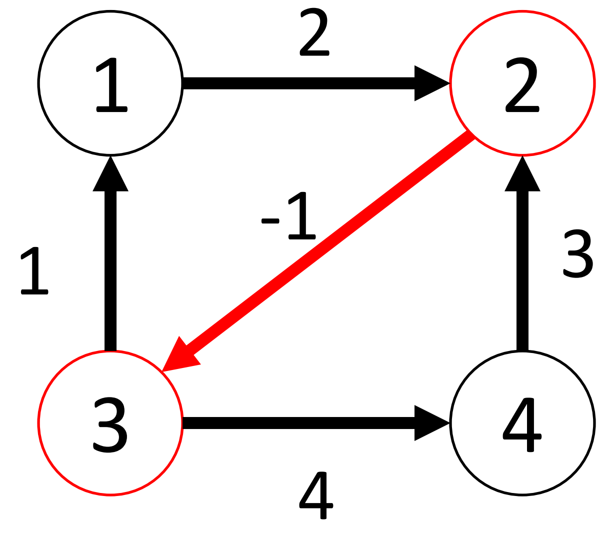
k: 1 2 **3** 4

i: 1 **2** 3 4

j: **1** 2 3 4

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(2,1) > d(2,3)$$



	1	2	3	4
1	0	2	1	5
2	+∞	0	-1	+∞
3	1	3	0	4
4	+∞	3	2	0

Example: The Nested Loop

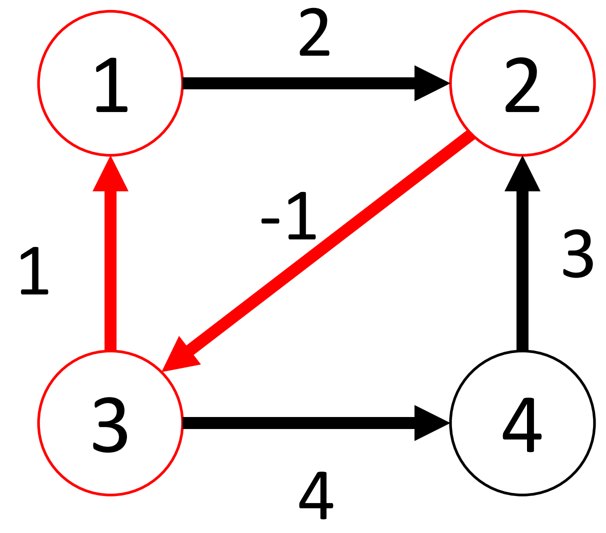
k: 1 2 **3** 4

i: 1 **2** 3 4

j: **1** 2 3 4

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(2,1) > d(2,3) + d(3,1)$$



	1	2	3	4
1	0	2	1	5
2	+∞	0	-1	+∞
3	1	3	0	4
4	+∞	3	2	0

Example: The Nested Loop

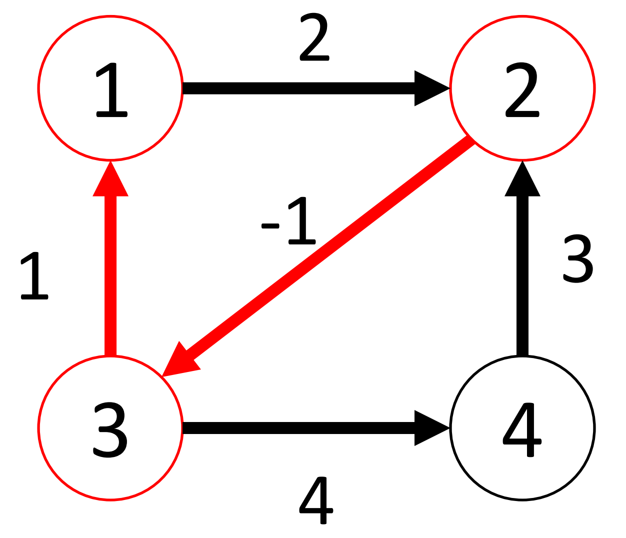
k: 1 2 **3** 4

i: 1 **2** 3 4

j: **1** 2 3 4

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(2,1) < -(-1) + 1$$



	1	2	3	4
1	0	2	1	5
2	0	0	-1	+∞
3	1	3	0	4
4	+∞	3	2	0

Example: The Nested Loop

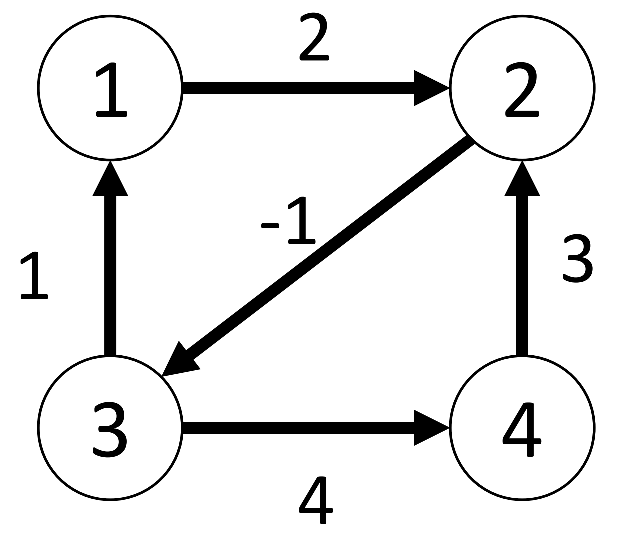
k: 1 2 **3** 4

i: 1 **2** 3 4

j: 1 2 3 **4**

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(2,4)$$


$$\begin{tikzcd}
1 \arrow[r, "2"] & 2 \arrow[dl, "-1"'] \\
3 \arrow[u, "1"] \arrow[r, "4"'] & 4 \arrow[u, "3"']
\end{tikzcd}$$

	1	2	3	4
1	0	2	1	5
2	0	0	-1	+∞
3	1	3	0	4
4	+∞	3	2	0

Example: The Nested Loop

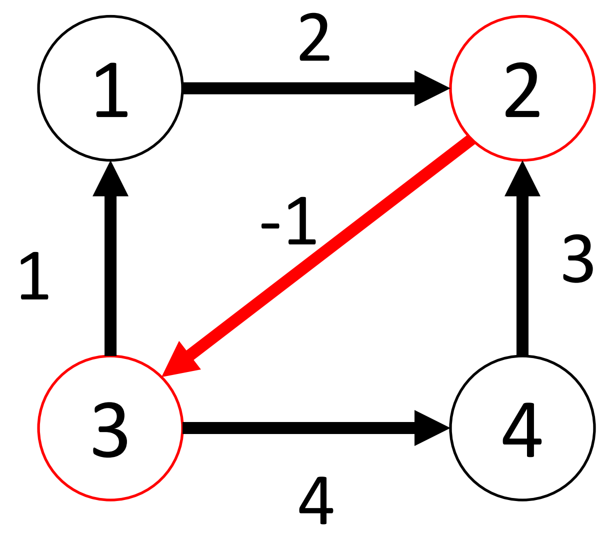
k: 1 2 **3** 4

i: 1 **2** 3 4

j: 1 2 3 **4**

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(2,4) > d(2,3)$$



	1	2	3	4
1	0	2	1	5
2	0	0	-1	+∞
3	1	3	0	4
4	+∞	3	2	0

Example: The Nested Loop

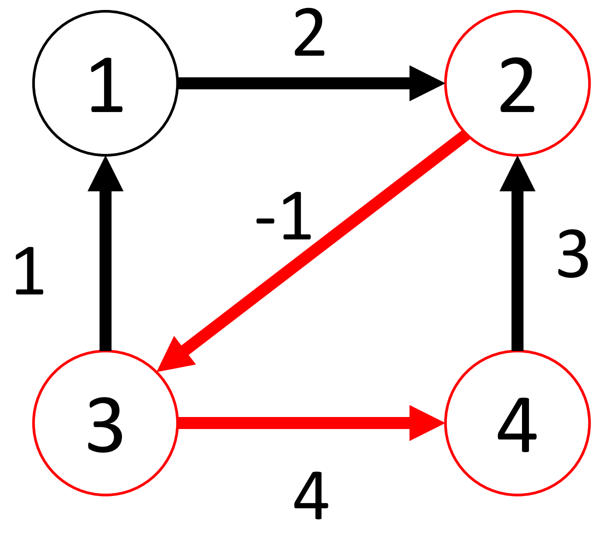
k: 1 2 **3** 4

i: 1 **2** 3 4

j: 1 2 3 **4**

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(2,4) > d(2,3) + d(3,4)$$
$$+\infty > (-1) + 4$$



	1	2	3	4
1	0	2	1	5
2	0	0	-1	+∞
3	1	3	0	4
4	+∞	3	2	0

Example: The Nested Loop

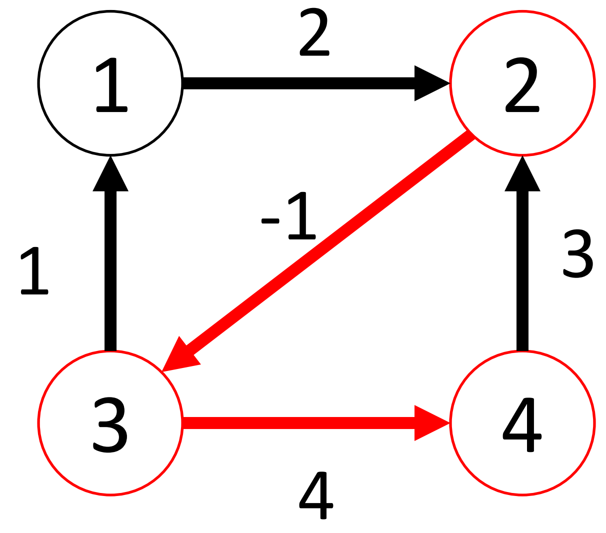
k: 1 2 **3** 4

i: 1 **2** 3 4

j: 1 2 3 **4**

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(2,4) > d(2,3) + d(3,4)$$
$$d(2,4) = -3$$



	1	2	3	4
1	0	2	1	5
2	0	0	-1	3
3	1	3	0	4
4	+∞	3	2	0

Example: The Nested Loop

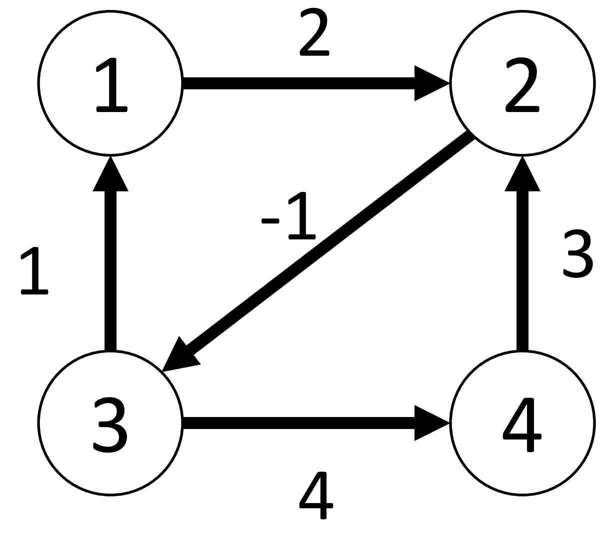
k: 1 2 **3** 4

i: 1 2 3 **4**

j: **1** 2 3 4

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(4,1)$$



	1	2	3	4
1	0	2	1	5
2	0	0	-1	3
3	1	3	0	4
4	+∞	3	2	0

Example: The Nested Loop

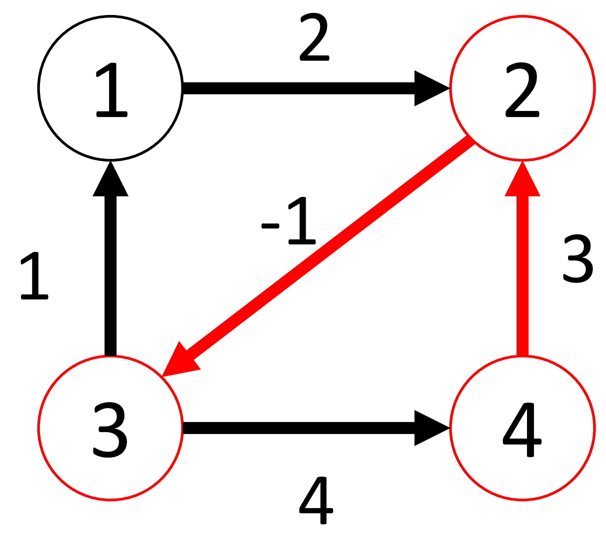
k: 1 2 **3** 4

i: 1 2 3 **4**

j: **1** 2 3 4

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(4,1) > d(4,3)$$



	1	2	3	4
1	0	2	1	5
2	0	0	-1	3
3	1	3	0	4
4	+∞	3	2	0

Example: The Nested Loop

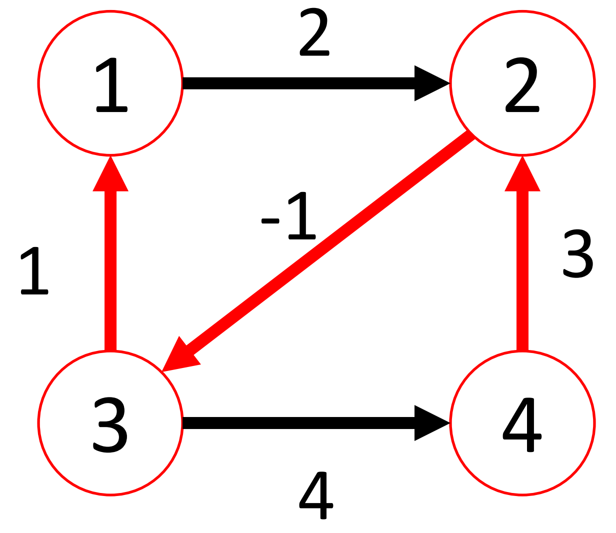
k: 1 2 **3** 4

i: 1 2 3 **4**

j: **1** 2 3 4

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(4,1) > d(4,3) + d(3,1)$$



	1	2	3	4
1	0	2	1	5
2	0	0	-1	3
3	1	3	0	4
4	+∞	3	2	0

Example: The Nested Loop

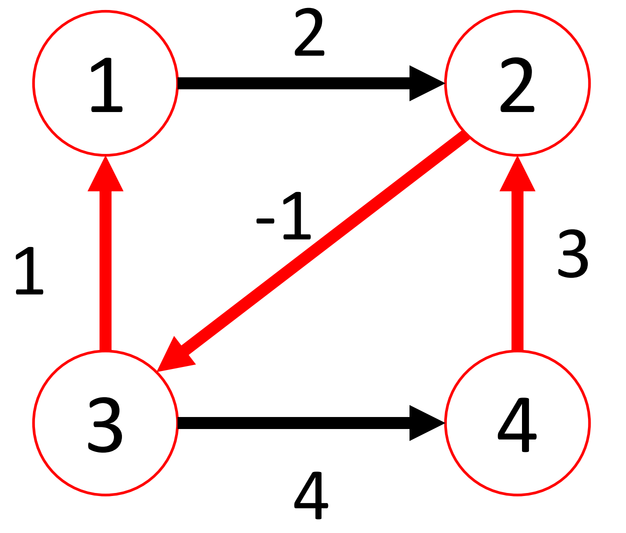
k: 1 2 **3** 4

i: 1 2 3 **4**

j: **1** 2 3 4

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

$$d(4,1) > d(4,3) + d(3,1)$$
$$+\infty > 2 + 1$$



	1	2	3	4
1	0	2	1	5
2	0	0	-1	3
3	1	3	0	4
4	+∞	3	2	0

Example: The Nested Loop

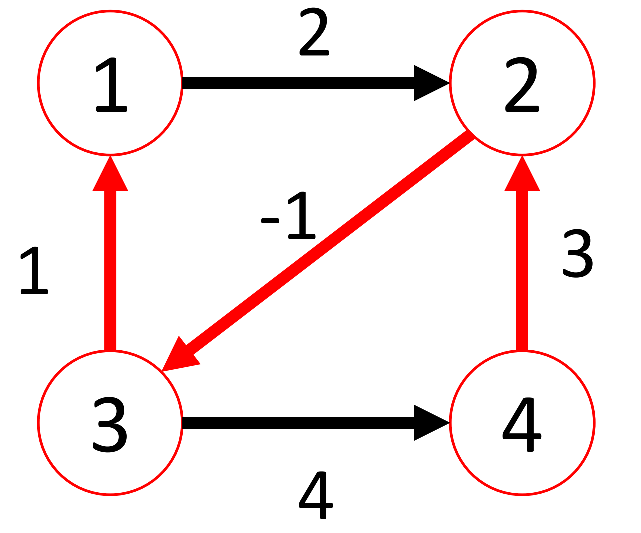
k: 1 2 **3** 4

i: 1 2 3 **4**

j: **1** 2 3 4

$$d(i,j) > d(i,k) + d(k,j)$$
$$d(i,j) \leftarrow d(i,k) + d(k,j)$$

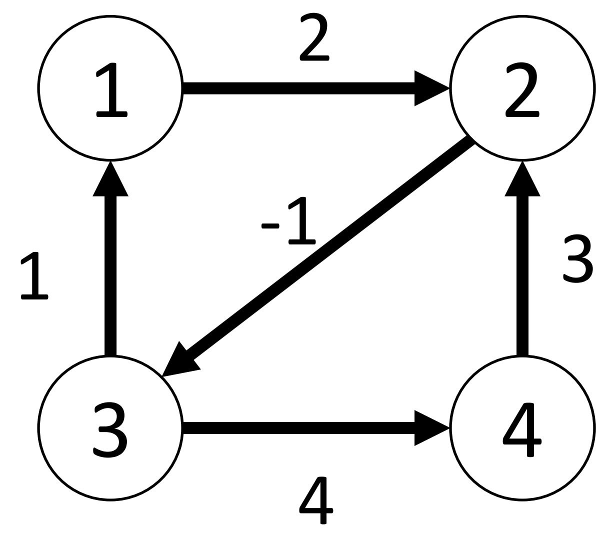
$$d(4,1) > d(4,3) + d(3,1)$$
$$d(4,1) = 3$$



	1	2	3	4
1	0	2	1	5
2	0	0	-1	3
3	1	3	0	4
4	3	3	2	0

Example: The Final Result

	1	**2**	**3**	**4**
1	0	2	1	5
2	0	0	-1	3
3	1	3	0	4
4	3	3	2	0



Once done, the matrix gives us the shortest path between each pair of nodes

Why Floyd's algorithm Works

- **Theorem**: at the bottom of the outer **for-loop**, for all nodes u and v,

- $d[u,v]$ contains the minimum length of all paths from u to v that are restricted to using only intermediate nodes that have been seen in the outer for-loop.

- When algorithm terminates, all nodes have been seen and $d[u,v]$ is the length of the shortest u-to-v path.

Why Floyd's algorithm Works (Contd.)

- Notation: S_k − the set of nodes seen after k passes through this loop; S_k -path – one with all intermediate nodes in S_k; d_k −the corresponding value of d. Induction on the outer **for-loop**:

- **Base case**: $k = 0$; $S_0 = \emptyset$, and the result holds.
- **Induction hypothesis**: It holds after $k \geq 0$ times through the loop.
- **Inductive step**: To show that $d_{k+1}[u, v]$ after k+1 passes through this loop is the minimum length of an u-to-v S_{k+1} -path.

Why Floyd’s algorithm Works (Contd.)

Inductive step:

- Suppose that x is the last node seen in the loop, so $S_{k+1} = S_k \cup \{x\}$.
- Fix an arbitrary pair of nodes $u, v \in V(G)$ and let L be the min-length of an u-to-v S_{k+1} -path, so that obviously $L \leq d_{k+1}[u, v]$.
- To show that also $d_{k+1}[u, v] \leq L$, choose an u-to-v S_{k+1} -path γ of length L.
  - If $x \notin \gamma$, the result follows from the induction hypothesis.
  - If $x \in \gamma$, let γ_1 and γ_2 be, respectively, the u-to-x and x-to-v subpaths. Then γ_1 and γ_2 are S_k -paths and by the inductive hypothesis,

$$L = |\gamma_1| + |\gamma_2| \underset{\text{IH}}{\geq} d_k[u, x] + d_k[x, v] \underset{\text{update formula}}{\geq} d_{k+1}[u, v]$$

Non-negativity of the weights is not used in the proof, and Floyd’s algorithm works for negative weights (but negative weight cycles should not be present).

Computing Actual Shortest Paths

- In addition to knowing the shortest distances, the shortest paths are often to be reconstructed.
- The Floyd's algorithm can be enhanced to compute also the predecessor matrix $\Pi = [\pi_{ij}]_{i,j=1,1}^{n,n}$ where vertex $\pi_{i,j}$ precedes vertex j on a shortest path from vertex i; $1 \leq i, j \leq n$

<u>Computing a sequence $\Pi^{(0)}, \Pi^{(1)}, \dots, \Pi^{(n)}$,</u>

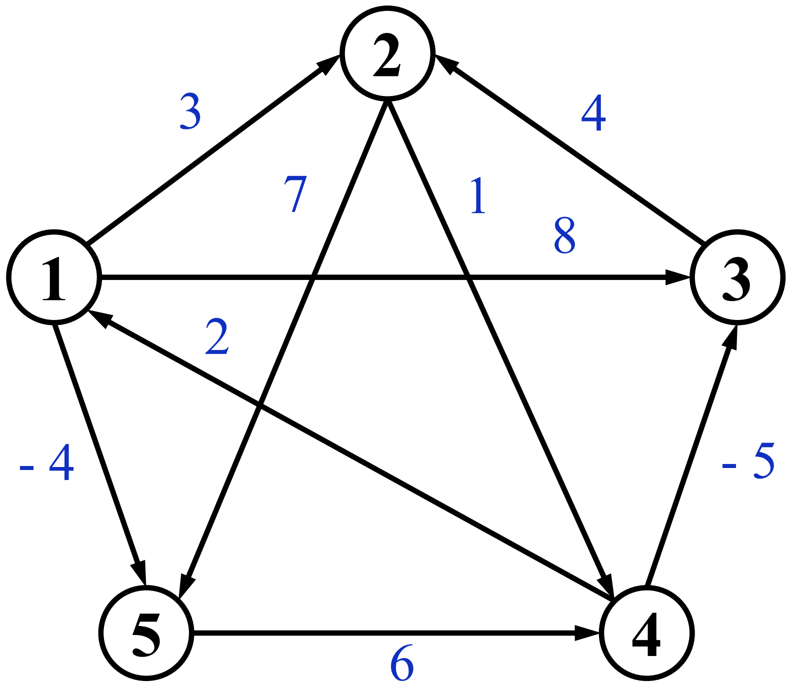
- where vertex $\pi_{i,j}^{(k)}$ precedes the vertex j on a shortest path from vertex i with all intermediate vertices in $V_{(k)} = \{1,2, \dots, k\}$
- For case of no intermediate vertices: $\pi_{i,j}^{(0)} = \begin{cases} \text{NIL} & \text{if } i = j \text{ or } c[i,j] = \infty \\ i & \text{if } i \neq j \text{ and } c[i,j] < \infty \end{cases}$

Floyd's Algorithm with Predecessors

Algorithm 1 Floyd's algorithm with Predecessors.

```
1: function FloydPred(weighted digraph($(G, c)$))
2:     D ← $c$                          ▷ Create initial distance matrix from weights.
3:     $\Pi \leftarrow \Pi^{(0)}$                   ▷ Initialize predecessors from $c$
4:     for $k$ from 1 to $n$ do
5:         for $i$ from 1 to $n$ do
6:             for $j$ from 1 to $n$ do
7:                 if $D[i,j] > D[i,k] + D[k,j]$ then
8:                     $D[i,j] \leftarrow D[i,k] + D[k,j]$
9:                     $\Pi[i,j] \leftarrow \Pi[k,j]$
```

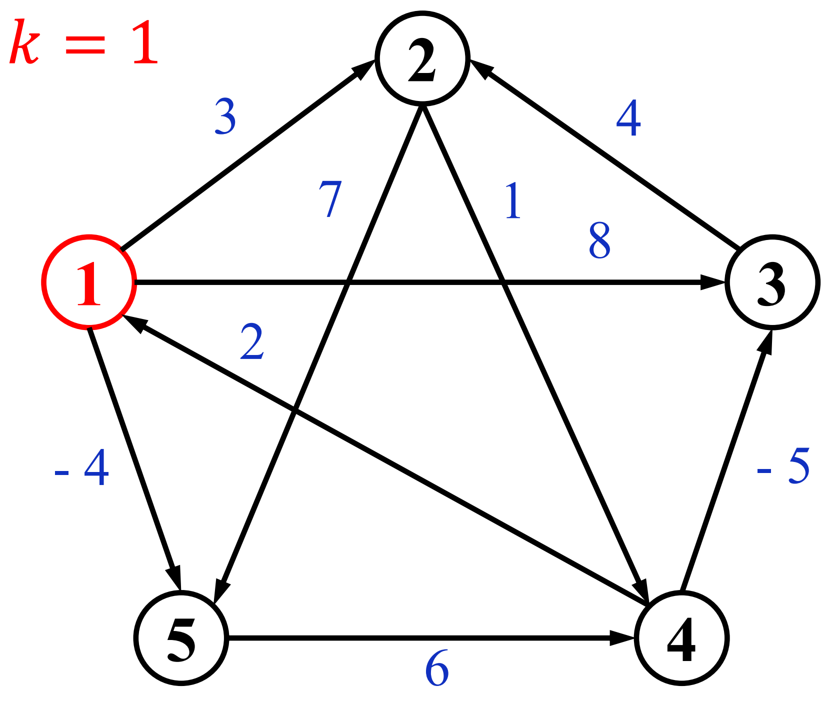
Illustrating Floyd's algorithm with Predecessors



$$D^{(0)} = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 3 & 8 & \infty & -4 \\ 2 & \infty & 0 & \infty & 1 & 7 \\ 3 & \infty & 4 & 0 & \infty & \infty \\ 4 & 2 & \infty & -5 & 0 & \infty \\ 5 & \infty & \infty & \infty & 6 & 0 \end{array}$$

$$\Pi^{(0)} = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & \mathrm{NIL} & 1 & 1 & \mathrm{NIL} & 1 \\ 2 & \mathrm{NIL} & \mathrm{NIL} & \mathrm{NIL} & 2 & 2 \\ 3 & \mathrm{NIL} & 3 & \mathrm{NIL} & \mathrm{NIL} & \mathrm{NIL} \\ 4 & 4 & \mathrm{NIL} & 4 & \mathrm{NIL} & \mathrm{NIL} \\ 5 & \mathrm{NIL} & \mathrm{NIL} & \mathrm{NIL} & 5 & \mathrm{NIL} \end{array}$$

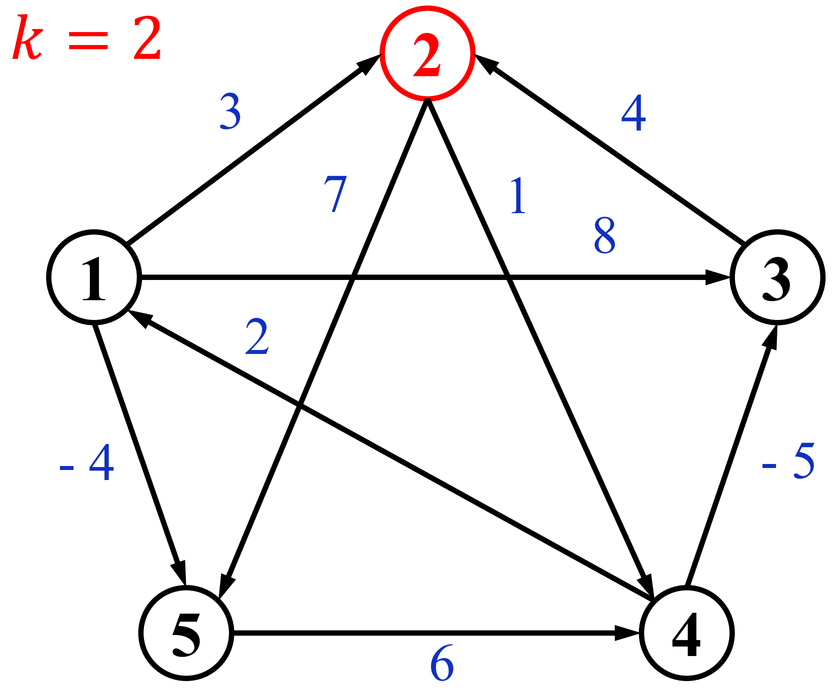
Illustrating Floyd’s algorithm with Predecessors



$$D^{(0)} = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 3 & 8 & \infty & -4 \\ 2 & \infty & 0 & \infty & 1 & 7 \\ 3 & \infty & 4 & 0 & \infty & \infty \\ 4 & 2 & 5 & -5 & 0 & -2 \\ 5 & \infty & \infty & \infty & 6 & 0 \end{array}$$

$$\Pi^{(0)} = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ 2 & \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ 3 & \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & 4 & 1 & 4 & \text{NIL} & 1 \\ 5 & \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{array}$$

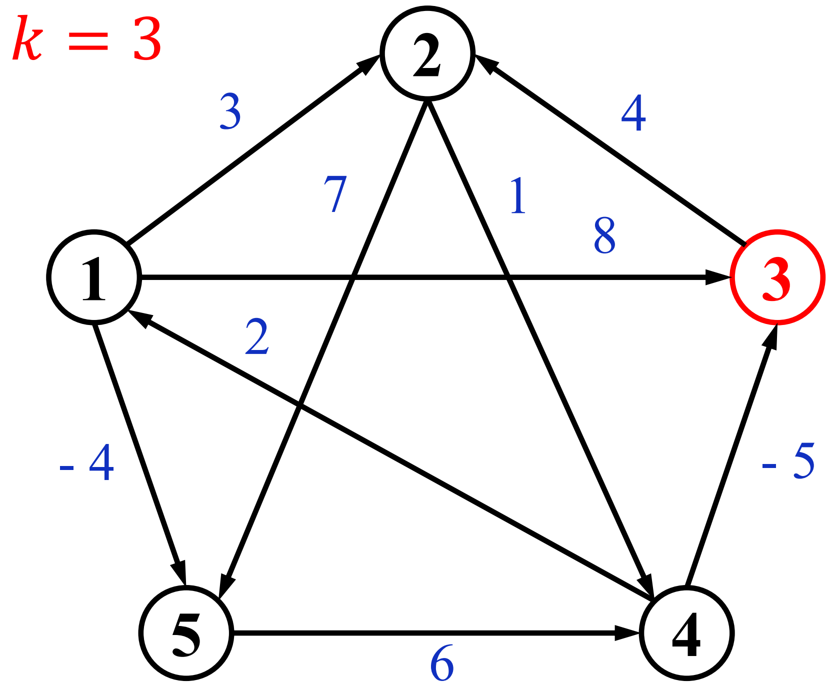
Illustrating Floyd’s algorithm with Predecessors



$$D^{(0)} = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 3 & 8 & 4 & -4 \\ 2 & \infty & 0 & \infty & 1 & 7 \\ 3 & \infty & 4 & 0 & 5 & 11 \\ 4 & 2 & 5 & -5 & 0 & -2 \\ 5 & \infty & \infty & \infty & 6 & 0 \end{array}$$

$$\Pi^{(0)} = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & \text{NIL} & 1 & 1 & 2 & 1 \\ 2 & \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ 3 & \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & 4 & 1 & 4 & \text{NIL} & 1 \\ 5 & \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{array}$$

Illustrating Floyd's algorithm with Predecessors

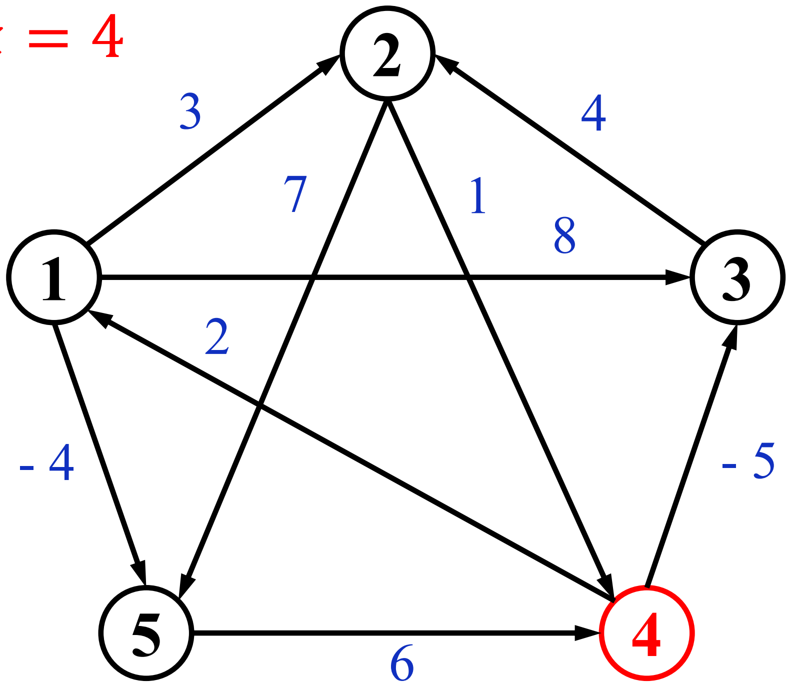


$$D^{(0)} = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 3 & 8 & 4 & -4 \\ 2 & \infty & 0 & \infty & 1 & 7 \\ 3 & \infty & 4 & 0 & 5 & 11 \\ 4 & 2 & -1 & -5 & 0 & -2 \\ 5 & \infty & \infty & \infty & 6 & 0 \end{array}$$

$$\Pi^{(0)} = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & \text{NIL} & 1 & 1 & 2 & 1 \\ 2 & \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ 3 & \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & 4 & 3 & 4 & \text{NIL} & 1 \\ 5 & \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{array}$$

Illustrating Floyd’s algorithm with Predecessors

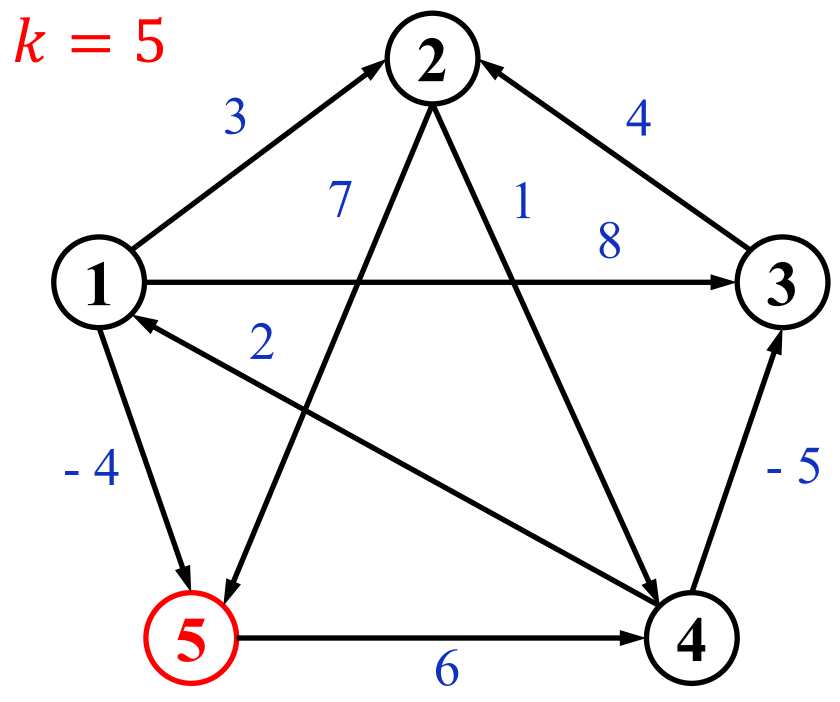
$k = 4$



$$D^{(0)} = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 3 & -1 & 4 & -4 \\ 2 & 3 & 0 & -4 & 1 & -1 \\ 3 & 7 & 4 & 0 & 5 & 3 \\ 4 & 2 & -1 & -5 & 0 & -2 \\ 5 & 8 & 5 & 1 & 6 & 0 \end{array}$$

$$\Pi^{(0)} = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & \text{NIL} & 1 & 4 & 2 & 1 \\ 2 & 4 & \text{NIL} & 4 & 2 & 1 \\ 3 & 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 4 & 3 & 4 & \text{NIL} & 1 \\ 5 & 4 & 3 & 4 & 5 & \text{NIL} \end{array}$$

Illustrating Floyd's algorithm with Predecessors



$$D^{(0)} = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 1 & -3 & 2 & -4 \\ 2 & 3 & 0 & -4 & 1 & -1 \\ 3 & 7 & 4 & 0 & 5 & 3 \\ 4 & 2 & -1 & -5 & 0 & -2 \\ 5 & 8 & 5 & 1 & 6 & 0 \end{array}$$

$$\Pi^{(0)} = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & \text{NIL} & 3 & 4 & 5 & 1 \\ 2 & 4 & \text{NIL} & 4 & 2 & 1 \\ 3 & 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 4 & 3 & 4 & \text{NIL} & 1 \\ 5 & 4 & 3 & 4 & 5 & \text{NIL} \end{array}$$

Getting Shortest Paths from Π Matrix

- The recursive algorithm using the predecessor matrix $\Pi = \Pi^{(n)}$ to print the shortest path between vertices i and j:

Algorithm 1 Getting shortest paths from Π matrix.

```
1: function PrintPath($\Pi, i, j$))
2:     if $i = j$ then print $i$
3:     else
4:         if $\pi_{i,j}$ = NIL then print “no path from $i$ to $j$”
5:         else
6:             PrintPath($\Pi, i, \pi_{i,j}$)
7:             print $j$
```

Comparison

- Summary of how BFS, Djikstra, Bellman-Ford and Floyd can be used to solve the SSSP and APSP problems for weighted and unweighted graphs and digraphs with or without negative arcs.

	SSSP			APSP		
	weighted	unweighted	Complexity	weighted	unweighted	Complexity
BFS	no	yes	$O(m+n)$	no	(yes)	$O(mn+n^2)$
Dijkstra	yes	yes	$O\big((m+n)\log n\big)$	(yes)	(yes)	$O\big((mn+n^2)\log n\big)$
Bellman-Ford	yes	yes	$O(mn)$	(yes)	(yes)	$O(mn^2)$
Floyd	yes	yes	$O(n^3)$	yes	yes	$O(n^3)$

Floyd and Bellman-Ford can detect negative weighted cycles.

(yes) – need to run for n times

Notes on Floyd-Warshall's algorithm

- Essentially just three nested loops: $\Theta(n^3)$.
- The outer loop specifies the potential "via" vertex, the inner two loops the end points of the path.
- Generally faster than a Bellman-Ford algorithm repeated n times.
- Not necessarily faster than Dijkstra but handles negative edge weights correctly.

SUMMARY

- Algorithms on Weighted Graphs
  - Dijkstra
  - Bellman-Ford
  - Floyd-Warshall
- All-Pairs Shortest Path