COMP90038
Algorithms and Complexity

Lecture 19: Warshall and Floyd algorithms

(with thanks to Harald Søndergaard & Michael Kirley)

Andres Munoz-Acosta

munoz.m@unimelb.edu.au

Peter Hall Building G.83

On the previous lecture

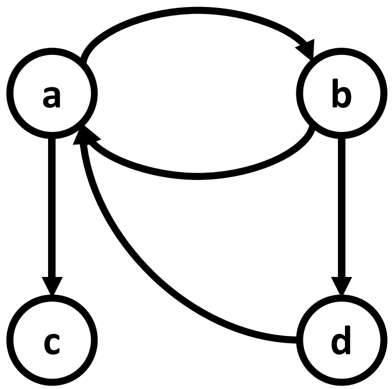
- We discussed **Dynamic Programming**, a bottom-up problem solving technique
  - We divide the problem into smaller, overlapping ones
  - Partial results are stored and used to find the complete solution
  - Algorithms usually involve a recursive relationship
- DP is often used to solve **combinatorial optimization** problems
  - Find the **best** possible **combination** subject to some **constraints**
- We demonstrated algorithms for three problems:
  - Coin row problem
  - Knapsack problem
  - Message passing in a tree problem

Today’s lecture

- We apply dynamic programming principles to two graph problems:
  - Computing the **transitive closure** of a directed graph
  - **Finding shortest distances** in weighted directed graphs

Warshall’s algorithm

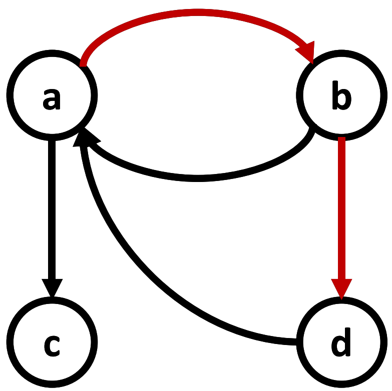
- Warshall's algorithm computes the **transitive closure** of a directed graph
  - An **edge** (a,d) is in the **transitive closure** of graph G if and only if **there is a path** in G from a to d



$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Warshall’s algorithm

- Warshall's algorithm computes the **transitive closure** of a directed graph
  - An **edge** (a,d) is in the **transitive closure** of graph G if and only if **there is a path** in G from a to d

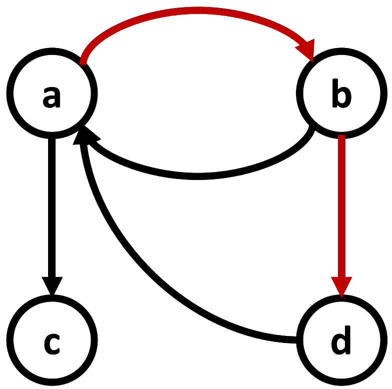


$$\begin{bmatrix} 0 & 1 & 1 & \mathbf{1} \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Warshall’s algorithm

- Warshall's algorithm computes the **transitive closure** of a directed graph
  - An **edge** (a,d) is in the **transitive closure** of graph G if and only if **there is a path** in G from a to d

- Transitive closure is important in applications where we need to reach a “goal state” from some “initial state”



$$\begin{bmatrix} 0 & 1 & 1 & \mathbf{1} \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Warshall's algorithm

- Assuming that the nodes of graph G are numbered from 1 to n, can we answer the question:

Is there a path from node i to node j using nodes $[1 \ \ldots \ k]$ as ***stepping stones***?

Warshall's algorithm

- Assuming that the nodes of graph G are numbered from 1 to n, can we answer the question:

Is there a path from node i to node j using nodes $[1 \ldots k]$ as ***stepping stones***?

- Such path exists if and only if we can:
  - step from i to j using only nodes $[1 \ldots k\text{-}1]$, or
  - step from i to k using only nodes $[1 \ldots k\text{-}1]$, and then step from k to j using only nodes $[1 \ldots k\text{-}1]$

Warshall's Algorithm

- If G's adjacency matrix is A then we can express the recurrence relation as:

$$R[i,j,0] = A[i,j]$$

$$R[i,j,k] = R[i,j,k-1] \textbf{ or } (R[i,k,k-1] \textbf{ and } R[k,j,k-1])$$

Warshall's Algorithm

- If G's adjacency matrix is A then we can express the recurrence relation as:

$$R[i, j, 0] = A[i, j]$$

$$R[i, j, k] = R[i, j, k-1] \textbf{ or } (R[i, k, k-1] \textbf{ and } R[k, j, k-1])$$

Use the existing path created in the previous step

Warshall's Algorithm

- If G's adjacency matrix is A then we can express the recurrence relation as:

$$R[i, j, 0] = A[i, j]$$

$$R[i, j, k] = R[i, j, k-1] \ \mathbf{or} \ (R[i, k, k-1] \ \mathbf{and} \ R[k, j, k-1])$$

Or create a new path using *k* as intermediate step

Warshall's Algorithm

- If G's adjacency matrix is A then we can express the recurrence relation as:

$$R[i, j, 0] = A[i, j]$$

$$R[i, j, k] = R[i, j, k-1] \textbf{ or } (R[i, k, k-1] \textbf{ and } R[k, j, k-1])$$

- This gives us an algorithm with a dynamic programming flavour:

function $\textsc{Warshall}(A[\cdot, \cdot], n)$
  $R[\cdot, \cdot, 0] \leftarrow A$
  for $k \leftarrow 1$ to n **do**
    for $i \leftarrow 1$ to n **do**
      for $j \leftarrow 1$ to n **do**
        $R[i, j, k] \leftarrow R[i, j, k-1]$ **or** $(R[i, k, k-1]$ **and** $R[k, j, k-1])$
  return $R[\cdot, \cdot, n]$

Warshall’s algorithm

- If we allow A to be used for the output, we can simplify things
  - If $R[i,k,k\text{-}1]$ (that is, $A[i,k]$) is 0 then we do nothing

Warshall's algorithm

- If we allow A to be used for the output, we can simplify things
  - If $R[i,k,k\text{-}1]$ (that is, $A[i,k]$) is 0 then we do nothing
  - But if $A[i,k]$ is 1 and if $\mathrm{A}[k,j]$ is also 1, then $A[i,j]$ gets set to 1

$$
\begin{array}{l}
\textbf{for } k \leftarrow 1 \text{ to } n \textbf{ do} \\
\quad \textbf{for } i \leftarrow 1 \text{ to } n \textbf{ do} \\
\quad\quad \textbf{for } j \leftarrow 1 \text{ to } n \textbf{ do} \\
\quad\quad\quad \textbf{if } A[i,k] \textbf{ then} \\
\quad\quad\quad\quad \textbf{if } A[k,j] \textbf{ then} \\
\quad\quad\quad\quad\quad A[i,j] \leftarrow 1
\end{array}
$$

Warshall's algorithm

- If we allow A to be used for the output, we can simplify things
  - If $R[i,k,k\text{-}1]$ (that is, $A[i,k]$) is 0 then we do nothing
  - But if $A[i,k]$ is 1 and if $\mathrm{A}[k,j]$ is also 1, then $A[i,j]$ gets set to 1

- $A[i,k]$ does not depend on j, so testing it can be moved outside the innermost loop

$$
\begin{array}{l}
\textbf{for } k \leftarrow 1 \text{ to } n \textbf{ do} \\
\quad \textbf{for } i \leftarrow 1 \text{ to } n \textbf{ do} \\
\quad\quad \textbf{for } j \leftarrow 1 \text{ to } n \textbf{ do} \\
\quad\quad\quad \boxed{\textbf{if } A[i,k] \textbf{ then}} \\
\quad\quad\quad\quad \textbf{if } A[k,j] \textbf{ then} \\
\quad\quad\quad\quad\quad A[i,j] \leftarrow 1
\end{array}
$$

Warshall's algorithm

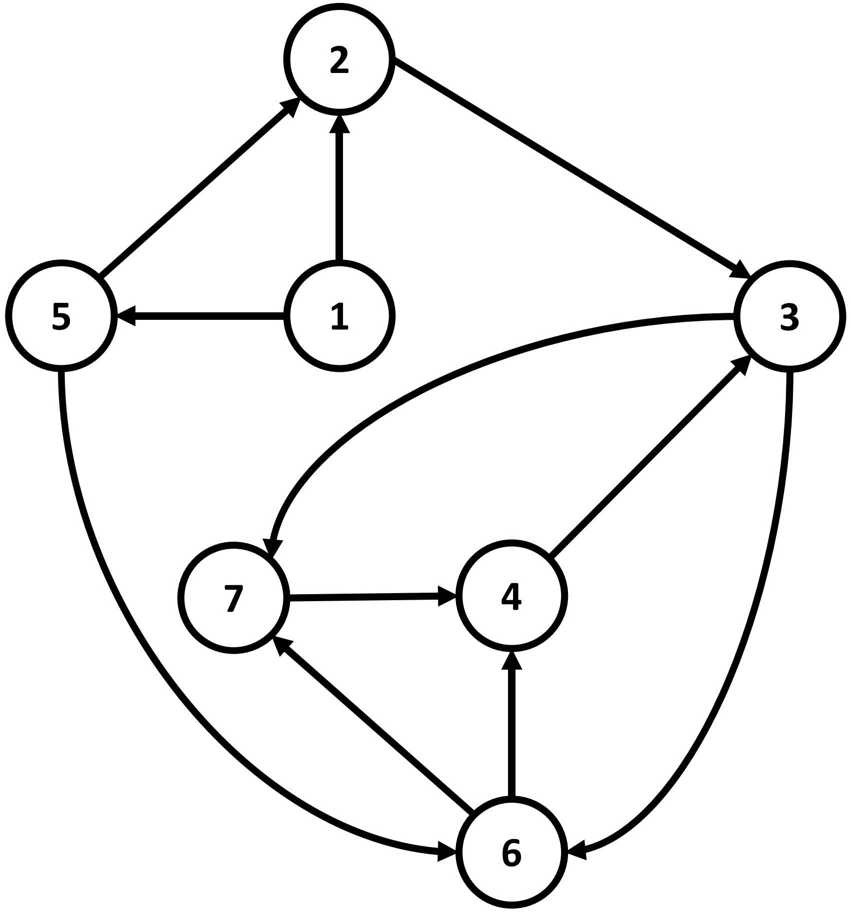
- If we allow A to be used for the output, we can simplify things
  - If $R[i,k,k\text{-}1]$ (that is, $A[i,k]$) is 0 then we do nothing
  - But if $A[i,k]$ is 1 and if $\mathrm{A}[k,j]$ is also 1, then $A[i,j]$ gets set to 1

- $A[i,k]$ does not depend on j, so testing it can be moved outside the innermost loop
  - This leads to a simpler version of the algorithm

$$
\begin{array}{l}
\textbf{for } k \leftarrow 1 \text{ to } n \textbf{ do} \\
\quad \textbf{for } i \leftarrow 1 \text{ to } n \textbf{ do} \\
\quad\quad \boxed{\textbf{if } A[i,k] \textbf{ then}} \\
\quad\quad\quad \textbf{for } j \leftarrow 1 \text{ to } n \textbf{ do} \\
\quad\quad\quad\quad \textbf{if } A[k,j] \textbf{ then} \\
\quad\quad\quad\quad\quad A[i,j] \leftarrow 1
\end{array}
$$

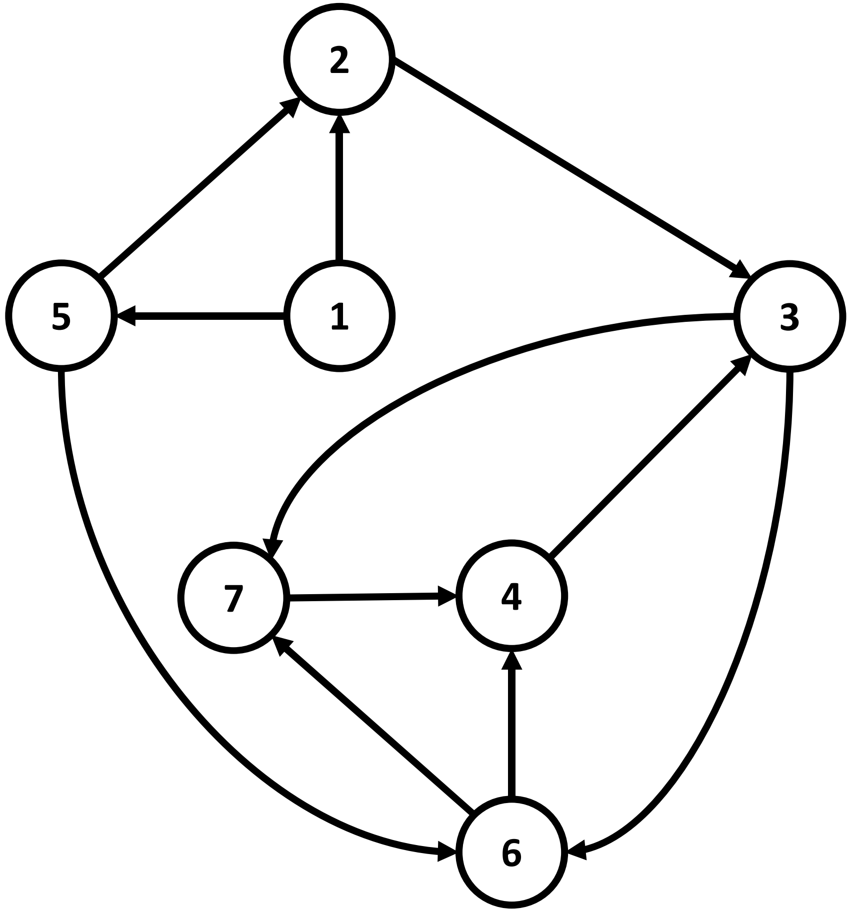
Warshall’s algorithm

- Let’s examine this algorithm. Let our graph be



Warshall’s algorithm

- Let’s examine this algorithm. Let our graph be



- Then, the adjacency matrix is:

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Warshall's algorithm

- For k=1, all the elements in the column are zero, so this **if** statement does nothing.

```
for k ← 1 to n do
    for i ← 1 to n do
        if A[i, k] then
            for j ← 1 to n do
                if A[k, j] then
                    A[i, j] ← 1
```

$$\begin{bmatrix} \mathbf{0} & 1 & 0 & 0 & 1 & 0 & 0 \\ \mathbf{0} & 0 & 1 & 0 & 0 & 0 & 0 \\ \mathbf{0} & 0 & 0 & 0 & 0 & 1 & 1 \\ \mathbf{0} & 0 & 1 & 0 & 0 & 0 & 0 \\ \mathbf{0} & 1 & 0 & 0 & 0 & 1 & 0 \\ \mathbf{0} & 0 & 0 & 1 & 0 & 0 & 1 \\ \mathbf{0} & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Warshall's algorithm

- For k=2, we have $\boldsymbol{A[1,2] = 1}$ and $\boldsymbol{A[5,2] = 1}$, and $\boldsymbol{A[2,3]=1}$

$$
\begin{bmatrix}
0 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0
\end{bmatrix}
$$

for $k \leftarrow 1$ to n **do**
  for $i \leftarrow 1$ to n **do**
    if $A[i,k]$ **then**
      for $j \leftarrow 1$ to n **do**
        if $A[k,j]$ **then**
          $A[i,j] \leftarrow 1$

Warshall's algorithm

- For k=2, we have ***A*[1,2] = 1** and ***A*[5,2] = 1**, and ***A*[2,3]=1**

There are '1's on this same row and column

$$\begin{bmatrix} 0 & \mathbf{1} & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

for $k \leftarrow 1$ to n **do**
    for $i \leftarrow 1$ to n **do**
        if $A[i, k]$ **then**
            for $j \leftarrow 1$ to n **do**
                if $A[k, j]$ **then**
                    $A[i, j] \leftarrow 1$

Warshall's algorithm

- For k=2, we have $\boldsymbol{A[1,2] = 1}$ and $\boldsymbol{A[5,2] = 1}$, and $\boldsymbol{A[2,3]=1}$

$$\begin{array}{l}
\textbf{for } k \leftarrow 1 \text{ to } n \textbf{ do} \\
\quad \textbf{for } i \leftarrow 1 \text{ to } n \textbf{ do} \\
\quad\quad \textbf{if } A[i,k] \textbf{ then} \\
\quad\quad\quad \textbf{for } j \leftarrow 1 \text{ to } n \textbf{ do} \\
\quad\quad\quad\quad \textbf{if } A[k,j] \textbf{ then} \\
\quad\quad\quad\quad\quad A[i,j] \leftarrow 1
\end{array}$$

$$\begin{bmatrix}
0 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0
\end{bmatrix}$$

Same here!!!

Warshall's algorithm

- For k=2, we have $\boldsymbol{A[1,2] = 1}$ and $\boldsymbol{A[5,2] = 1}$, and $\boldsymbol{A[2,3]=1}$
  - Then, we can make $\boldsymbol{A[1,3] = 1}$ and $\boldsymbol{A[5,3] = 1}$

$$
\begin{array}{l}
\textbf{for } k \leftarrow 1 \text{ to } n \textbf{ do} \\
\quad \textbf{for } i \leftarrow 1 \text{ to } n \textbf{ do} \\
\quad\quad \textbf{if } A[i,k] \textbf{ then} \\
\quad\quad\quad \textbf{for } j \leftarrow 1 \text{ to } n \textbf{ do} \\
\quad\quad\quad\quad \boxed{\begin{array}{l} \textbf{if } A[k,j] \textbf{ then} \\ \quad A[i,j] \leftarrow 1 \end{array}}
\end{array}
$$

$$
\begin{bmatrix}
0 & 1 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0
\end{bmatrix}
$$

Warshall's algorithm

- For k=3, we have $\boldsymbol{A[1,3]}$, $\boldsymbol{A[2,3]}$, $\boldsymbol{A[4,3]}$, $\boldsymbol{A[5,3]}$, $\mathbf{A[3,6]}$ and $\boldsymbol{A[3,7]}$ equal to 1

$$
\begin{aligned}
&\textbf{for } k \leftarrow 1 \text{ to } n \textbf{ do}\\
&\quad \textbf{for } i \leftarrow 1 \text{ to } n \textbf{ do}\\
&\quad\quad \textbf{if } A[i,k] \textbf{ then}\\
&\quad\quad\quad \textbf{for } j \leftarrow 1 \text{ to } n \textbf{ do}\\
&\quad\quad\quad\quad \textbf{if } A[k,j] \textbf{ then}\\
&\quad\quad\quad\quad\quad A[i,j] \leftarrow 1
\end{aligned}
$$

$$
\begin{bmatrix}
0 & 1 & 1 & 0 & 1 & 0 & 0\\
0 & 0 & 1 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 1 & 1\\
0 & 0 & 1 & 0 & 0 & 0 & 0\\
0 & 1 & 1 & 0 & 0 & 1 & 0\\
0 & 0 & 0 & 1 & 0 & 0 & 1\\
0 & 0 & 0 & 1 & 0 & 0 & 0
\end{bmatrix}
$$

Warshall's algorithm

- For k=3, we have ***A*[1,3]**, ***A*[2,3]**, ***A*[4,3]**, ***A*[5,3]**, **A[3,6]** and ***A*[3,7]** equal to 1

This block should turn into '1's

$$
\begin{bmatrix}
0 & 1 & \mathbf{1} & 0 & 1 & \boxed{0} & \boxed{0} \\
0 & 0 & \mathbf{1} & 0 & 0 & \boxed{0} & \boxed{0} \\
0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} \\
0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\
0 & 1 & \mathbf{1} & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0
\end{bmatrix}
$$

$$
\begin{array}{l}
\textbf{for } k \leftarrow 1 \text{ to } n \textbf{ do} \\
\quad \textbf{for } i \leftarrow 1 \text{ to } n \textbf{ do} \\
\quad\quad \textbf{if } A[i,k] \textbf{ then} \\
\quad\quad\quad \textbf{for } j \leftarrow 1 \text{ to } n \textbf{ do} \\
\quad\quad\quad\quad \textbf{if } A[k,j] \textbf{ then} \\
\quad\quad\quad\quad\quad A[i,j] \leftarrow 1
\end{array}
$$

Warshall's algorithm

- For k=3, we have A[1,3], A[2,3], A[4,3], A[5,3], A[3,6] and A[3,7] equal to 1

$$
\begin{array}{l}
\textbf{for } k \leftarrow 1 \text{ to } n \textbf{ do} \\
\quad \textbf{for } i \leftarrow 1 \text{ to } n \textbf{ do} \\
\quad\quad \textbf{if } A[i,k] \textbf{ then} \\
\quad\quad\quad \textbf{for } j \leftarrow 1 \text{ to } n \textbf{ do} \\
\quad\quad\quad\quad \textbf{if } A[k,j] \textbf{ then} \\
\quad\quad\quad\quad\quad A[i,j] \leftarrow 1
\end{array}
$$

$$
\begin{bmatrix}
0 & 1 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0
\end{bmatrix}
$$

This one too…

Warshall's algorithm

- For k=3, we have ***A*[1,3]**, ***A*[2,3]**, ***A*[4,3]**, ***A*[5,3]**, **A[3,6]** and ***A*[3,7]** equal to 1
  - Then, we can make ***A*[1,6]**, ***A*[2,6]**, ***A*[4,6]**, ***A*[1,7]**, ***A*[2,7]**, ***A*[4,7]**, and ***A*[5,7]** equal to 1

$$
\begin{array}{l}
\textbf{for } k \leftarrow 1 \text{ to } n \textbf{ do} \\
\quad \textbf{for } i \leftarrow 1 \text{ to } n \textbf{ do} \\
\quad\quad \textbf{if } A[i,k] \textbf{ then} \\
\quad\quad\quad \textbf{for } j \leftarrow 1 \text{ to } n \textbf{ do} \\
\quad\quad\quad\quad \textbf{if } A[k,j] \textbf{ then} \\
\quad\quad\quad\quad\quad A[i,j] \leftarrow 1
\end{array}
$$

$$
\begin{bmatrix}
0 & 1 & 1 & 0 & 1 & \mathbf{1} & \mathbf{1} \\
0 & 0 & 1 & 0 & 0 & \mathbf{1} & \mathbf{1} \\
0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 & \mathbf{1} & \mathbf{1} \\
0 & 1 & 1 & 0 & 0 & 1 & \mathbf{1} \\
0 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0
\end{bmatrix}
$$

Warshall's algorithm

- Let's look at the next steps:

$$k=4$$

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & \mathbf{1} & 0 & 0 & \mathbf{1} & \mathbf{1} \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 1 \\ 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \end{bmatrix}$$

Warshall's algorithm

- Let's look at the next steps:

$$k=4$$

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & \mathbf{1} & 0 & 0 & \mathbf{1} & \mathbf{1} \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 1 \\ 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \end{bmatrix}$$

In row 4 there is a '1' on this column

Warshall’s algorithm

- Let’s look at the next steps:

$$k=4$$

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & \mathbf{1} & 0 & 0 & \mathbf{1} & \mathbf{1} \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 1 \\ 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \end{bmatrix}$$

Also on this column

Warshall's algorithm

- Let's look at the next steps:

$$k{=}4$$

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & \mathbf{1} & 0 & 0 & \mathbf{1} & \mathbf{1} \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 1 \\ 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \end{bmatrix}$$

and on this column

Warshall's algorithm

- Let's look at the next steps:

$$k=4$$

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & \mathbf{1} & 0 & 0 & \mathbf{1} & \mathbf{1} \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 1 \\ 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \end{bmatrix}$$

In Column 4 there is a '1' on this row

Warshall's algorithm

- Let's look at the next steps:

$$k=4$$

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & \mathbf{1} & 0 & 0 & \mathbf{1} & \mathbf{1} \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 1 \\ 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \end{bmatrix}$$

Also on this row

Warshall's algorithm

- Let's look at the next steps:

$$k{=}4$$

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & \mathbf{1} & 1 & 0 & \mathbf{1} & 1 \\ 0 & 0 & \mathbf{1} & 1 & 0 & \mathbf{1} & \mathbf{1} \end{bmatrix}$$

Warshall’s algorithm

- Let’s look at the next steps:

k=4

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & \mathbf{1} & 1 & 0 & \mathbf{1} & 1 \\ 0 & 0 & \mathbf{1} & 1 & 0 & \mathbf{1} & \mathbf{1} \end{bmatrix}$$

k=5

$$\begin{bmatrix} 0 & 1 & 1 & 0 & \mathbf{1} & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & \mathbf{1} & \mathbf{1} & 0 & 0 & \mathbf{1} & \mathbf{1} \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Warshall's algorithm

- Let's look at the next steps:

These are done

k=4

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & \mathbf{1} & 1 & 0 & \mathbf{1} & 1 \\ 0 & 0 & \mathbf{1} & 1 & 0 & \mathbf{1} & \mathbf{1} \end{bmatrix}$$

k=5

$$\begin{bmatrix} 0 & \boxed{1 \quad 1} & 0 & \mathbf{1} & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & \mathbf{1} & \mathbf{1} & 0 & 0 & \mathbf{1} & \mathbf{1} \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Warshall's algorithm

- Let's look at the next steps:

k=4

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

k=5

So are these

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Warshall's algorithm

- Let's look at the next steps:

$$k=4 \qquad \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & \mathbf{1} & 1 & 0 & \mathbf{1} & 1 \\ 0 & 0 & \mathbf{1} & 1 & 0 & \mathbf{1} & \mathbf{1} \end{bmatrix}$$

$$k=5 \qquad \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

There are no changes

Warshall's algorithm

- Let's look at the next steps:

k=4

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & \mathbf{1} & 1 & 0 & \mathbf{1} & 1 \\ 0 & 0 & \mathbf{1} & 1 & 0 & \mathbf{1} & \mathbf{1} \end{bmatrix}$$

k=5

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

k=6

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 1 & \mathbf{1} & 1 \\ 0 & 0 & 1 & 0 & 0 & \mathbf{1} & 1 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 1 \\ 0 & 0 & 1 & 0 & 0 & \mathbf{1} & 1 \\ 0 & 1 & 1 & 0 & 0 & \mathbf{1} & 1 \\ 0 & 0 & \mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & \mathbf{1} \\ 0 & 0 & 1 & 1 & 0 & \mathbf{1} & 1 \end{bmatrix}$$

Warshall's algorithm

- Let's look at the next steps:

$$k=4 \qquad \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & \mathbf{1} & 1 & 0 & \mathbf{1} & 1 \\ 0 & 0 & \mathbf{1} & 1 & 0 & \mathbf{1} & \mathbf{1} \end{bmatrix}$$

$$k=5 \qquad \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

$$k=6 \qquad \begin{bmatrix} 0 & 1 & 1 & \mathbf{1} & 1 & 1 & 1 \\ 0 & 0 & 1 & \mathbf{1} & 0 & 1 & 1 \\ 0 & 0 & \mathbf{1} & \mathbf{1} & 0 & 1 & 1 \\ 0 & 0 & 1 & \mathbf{1} & 0 & 1 & 1 \\ 0 & 1 & 1 & \mathbf{1} & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Warshall's algorithm

- Let's look at the next steps:

k=4

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & \mathbf{1} & 1 & 0 & \mathbf{1} & 1 \\ 0 & 0 & \mathbf{1} & 1 & 0 & \mathbf{1} & \mathbf{1} \end{bmatrix}$$

k=5

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

k=6

$$\begin{bmatrix} 0 & 1 & 1 & \mathbf{1} & 1 & 1 & 1 \\ 0 & 0 & 1 & \mathbf{1} & 0 & 1 & 1 \\ 0 & 0 & \mathbf{1} & \mathbf{1} & 0 & 1 & 1 \\ 0 & 0 & 1 & \mathbf{1} & 0 & 1 & 1 \\ 0 & 1 & 1 & \mathbf{1} & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

- For k=7 there are no changes either…

Warshall’s algorithm

- This algorithm’s complexity is $\Theta(n^3)$
  - There is **no difference** between the best, average, and worst cases.

Warshall’s algorithm

- This algorithm’s complexity is $\Theta(n^3)$
  - There is **no difference** between the best, average, and worst cases.

- The algorithm has a tight inner loop, making it **ideal for dense graphs**

- However, it is **not the best** transitive-closure algorithm to use for **sparse graphs**

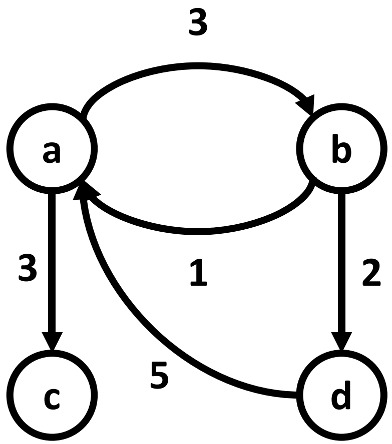
Warshall’s algorithm

- This algorithm’s complexity is $\Theta(n^3)$
  - There is **no difference** between the best, average, and worst cases.
- The algorithm has a tight inner loop, making it **ideal for dense graphs**
- However, it is **not the best** transitive-closure algorithm to use for **sparse graphs**
  - For sparse graphs, it may be better doing DFS from each node v, keeping track of which nodes are reached from v

Floyd's algorithm

- Floyd's algorithm solves the **all-pairs shortest-path** problem for weighted graphs with **positive weights**.
  - It works for **directed** as well as **undirected** graphs

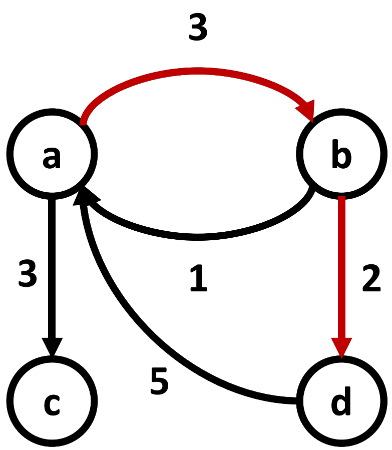
- We assume we are given a **weight matrix** W that holds all the edges' weights
  - If there is no edge from node i to node j, we set $W[i,j] = \infty$



$$\begin{bmatrix} \infty & 3 & 3 & \infty \\ 1 & \infty & \infty & 2 \\ \infty & \infty & \infty & \infty \\ 5 & \infty & \infty & \infty \end{bmatrix}$$

Floyd's algorithm

- Floyd's algorithm solves the **all-pairs shortest-path** problem for weighted graphs with **positive weights**.
  - It works for **directed** as well as **undirected** graphs

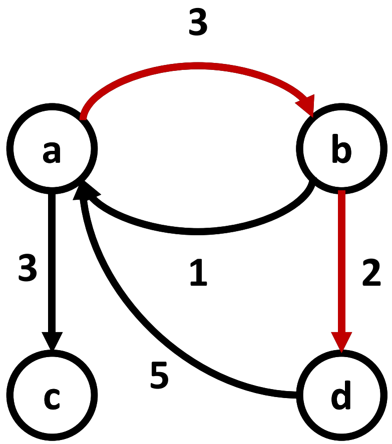


- We assume we are given a **weight matrix** W that holds all the edges' weights
  - If there is no edge from node i to node j, we set $W[i,j] = \infty$

$$\begin{bmatrix} \infty & 3 & 3 & \mathbf{5} \\ 1 & \infty & \infty & 2 \\ \infty & \infty & \infty & \infty \\ 5 & \infty & \infty & \infty \end{bmatrix}$$

Floyd's algorithm

- Floyd's algorithm solves the **all-pairs shortest-path** problem for weighted graphs with **positive weights**.
  - It works for **directed** as well as **undirected** graphs
- We assume we are given a **weight matrix** W that holds all the edges' weights
  - If there is no edge from node i to node j, we set $W[i,j] = \infty$
- We will construct the **distance matrix** D, step by step



$$\begin{bmatrix} \infty & 3 & 3 & \mathbf{5} \\ 1 & \infty & \infty & 2 \\ \infty & \infty & \infty & \infty \\ 5 & \infty & \infty & \infty \end{bmatrix}$$

Floyd's algorithm

- As we did in the Warshall's algorithm, assume nodes are numbered 1 to n. We try to answer the question:

What is the shortest path from node i to node j using nodes $[1 \ldots k]$ as ***stepping stones***?

Floyd's algorithm

- As we did in the Warshall's algorithm, assume nodes are numbered 1 to n. We try to answer the question:

What is the shortest path from node i to node j using nodes $[1 \ldots k]$ as ***stepping stones***?

- Such path will exist if and only if we can:
  - step from i to j using only nodes $[1 \ldots k\text{-}1]$, or
  - step from i to k using only nodes $[1 \ldots k\text{-}1]$, and then step from k to j using only nodes $[1 \ldots k\text{-}1]$.

Floyd's algorithm

- If G's weight matrix is W then we can express the recurrence relation as:

$$D[i, j, 0] = W[i, j]$$

$$D[i, j, k] = \min\left(D[i, j, k-1], D[i, k, k-1] + D[k, j, k-1]\right)$$

Floyd's algorithm

- If G's weight matrix is W then we can express the recurrence relation as:

$$D[i,j,0] = W[i,j]$$

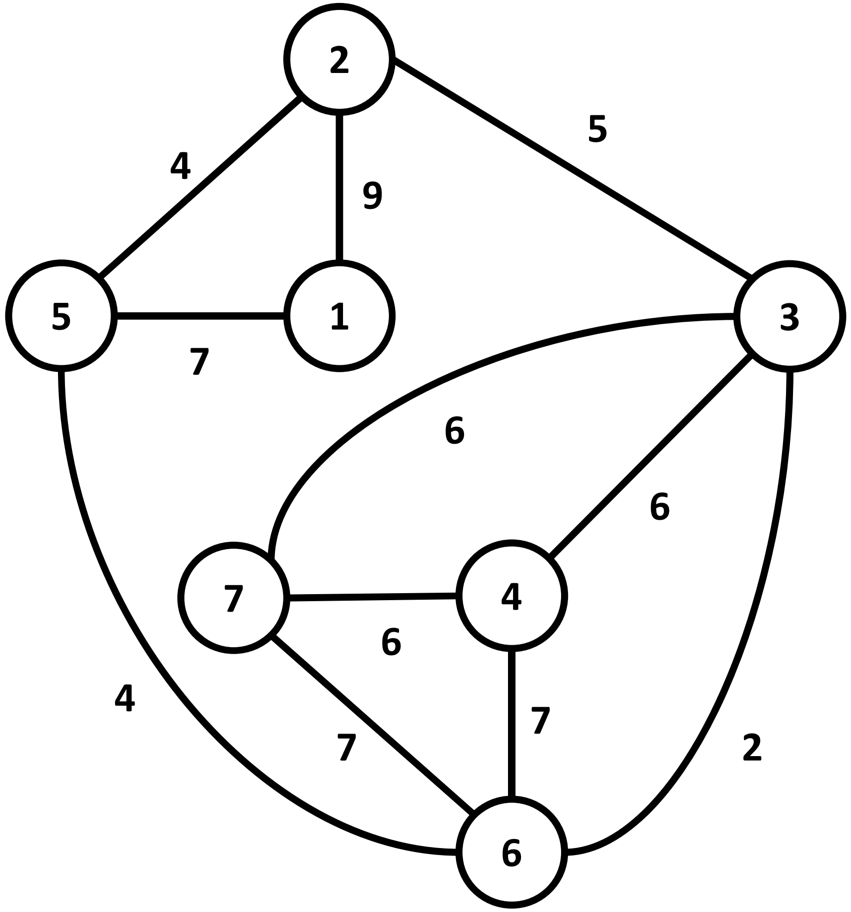
$$D[i,j,k] = \min\left(D[i,j,k-1], D[i,k,k-1] + D[k,j,k-1]\right)$$

- A simpler version updating D:

$$
\begin{array}{l}
\textbf{function } \textsc{Floyd}(W[\cdot,\cdot], n) \\
\quad D \leftarrow W \\
\quad \textbf{for } k \leftarrow 1 \text{ to } n \textbf{ do} \\
\quad\quad \textbf{for } i \leftarrow 1 \text{ to } n \textbf{ do} \\
\quad\quad\quad \textbf{for } j \leftarrow 1 \text{ to } n \textbf{ do} \\
\quad\quad\quad\quad D[i,j] \leftarrow \min\left(D[i,j], D[i,k] + D[k,j]\right) \\
\quad \textbf{return } D
\end{array}
$$

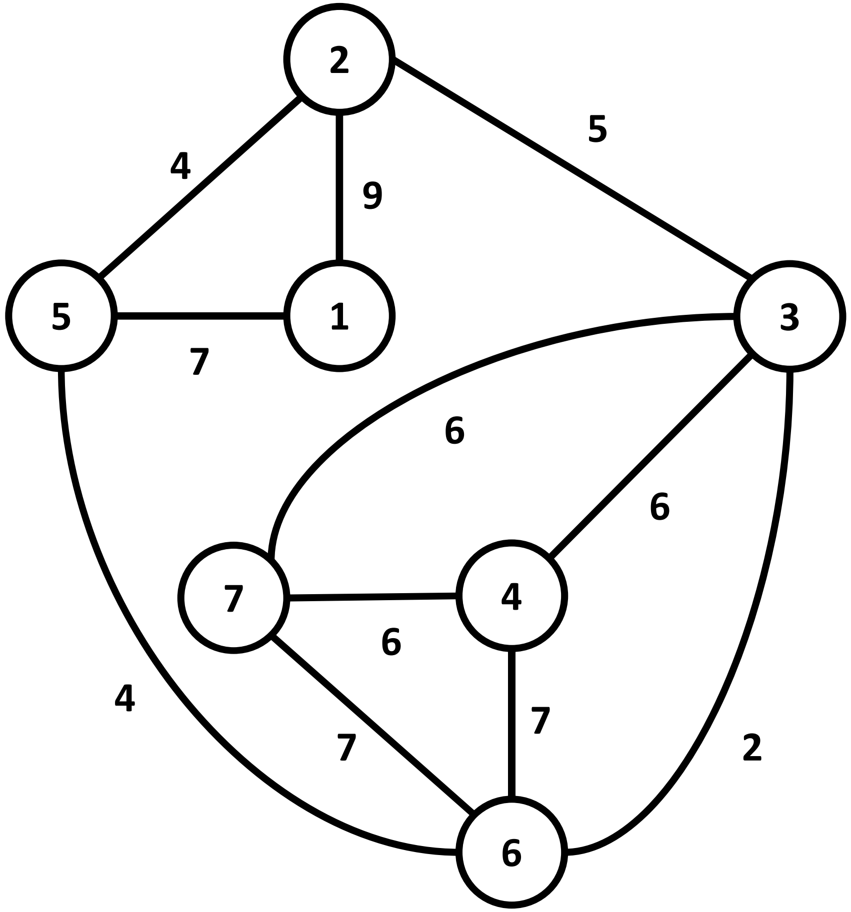
Floyd's algorithm

- Let’s examine this algorithm. Let our graph be



Floyd's algorithm

- Let’s examine this algorithm. Let our graph be



- Then, the weight matrix is:

$$\begin{bmatrix} 0 & 9 & \infty & \infty & 7 & \infty & \infty \\ 9 & 0 & 5 & \infty & 4 & \infty & \infty \\ \infty & 5 & 0 & 6 & \infty & 2 & 6 \\ \infty & \infty & 6 & 0 & \infty & 7 & 6 \\ 7 & 4 & \infty & \infty & 0 & 4 & \infty \\ \infty & \infty & 2 & 7 & 4 & 0 & 7 \\ \infty & \infty & 6 & 6 & \infty & 7 & 0 \end{bmatrix}$$

Floyd's algorithm

- For k=1 there are no changes

$$
\begin{array}{l}
\textbf{function } \textsc{Floyd}(W[\cdot,\cdot], n) \\
\quad D \leftarrow W \\
\quad \textbf{for } k \leftarrow 1 \text{ to } n \textbf{ do} \\
\quad\quad \textbf{for } i \leftarrow 1 \text{ to } n \textbf{ do} \\
\quad\quad\quad \textbf{for } j \leftarrow 1 \text{ to } n \textbf{ do} \\
\quad\quad\quad\quad D[i,j] \leftarrow \min\left(D[i,j], D[i,k] + D[k,j]\right) \\
\quad \textbf{return } D
\end{array}
$$

$$
\begin{bmatrix}
0 & 9 & \infty & \infty & 7 & \infty & \infty \\
9 & 0 & 5 & \infty & 4 & \infty & \infty \\
\infty & 5 & 0 & 6 & \infty & 2 & 6 \\
\infty & \infty & 6 & 0 & \infty & 7 & 6 \\
7 & 4 & \infty & \infty & 0 & 4 & \infty \\
\infty & \infty & 2 & 7 & 4 & 0 & 7 \\
\infty & \infty & 6 & 6 & \infty & 7 & 0
\end{bmatrix}
$$

Floyd's algorithm

- For k=2, ***D*[1,2] = 9** and ***D*[2,3]=5**; and ***D*[5,2] = 4** and ***D*[2,3]=5**

$$\begin{bmatrix} 0 & \mathbf{9} & \infty & \infty & 7 & \infty & \infty \\ \mathbf{9} & 0 & \mathbf{5} & \infty & \mathbf{4} & \infty & \infty \\ \infty & \mathbf{5} & 0 & 6 & \infty & 2 & 6 \\ \infty & \infty & 6 & 0 & \infty & 7 & 6 \\ 7 & \mathbf{4} & \infty & \infty & 0 & 4 & \infty \\ \infty & \infty & 2 & 7 & 4 & 0 & 7 \\ \infty & \infty & 6 & 6 & \infty & 7 & 0 \end{bmatrix}$$

function $\text{FLOYD}(W[\cdot,\cdot], n)$
  $D \leftarrow W$
  for $k \leftarrow 1$ to n **do**
    for $i \leftarrow 1$ to n **do**
      for $j \leftarrow 1$ to n **do**
        $D[i,j] \leftarrow \min\left(D[i,j], D[i,k] + D[k,j]\right)$
  return D

Floyd's algorithm

- For k=2, $D[1,2] = 9$ and $D[2,3]=5$; and $D[5,2] = 4$ and $D[2,3]=5$

We can connect 1 and 3 through 2

$$\begin{bmatrix} 0 & 9 & \boxed{\infty} & \infty & 7 & \infty & \infty \\ 9 & 0 & 5 & \infty & 4 & \infty & \infty \\ \infty & 5 & 0 & 6 & \infty & 2 & 6 \\ \infty & \infty & 6 & 0 & \infty & 7 & 6 \\ 7 & 4 & \infty & \infty & 0 & 4 & \infty \\ \infty & \infty & 2 & 7 & 4 & 0 & 7 \\ \infty & \infty & 6 & 6 & \infty & 7 & 0 \end{bmatrix}$$

function FLOYD($W[\cdot,\cdot], n$)
  $D \leftarrow W$
  for $k \leftarrow 1$ to n **do**
    for $i \leftarrow 1$ to n **do**
      for $j \leftarrow 1$ to n **do**
        $D[i,j] \leftarrow \min(D[i,j], D[i,k] + D[k,j])$
  return D

Floyd's algorithm

- For k=2, ***D*[1,2] = 9** and ***D*[2,3]=5**; and ***D*[5,2] = 4** and ***D*[2,3]=5**

$$
\begin{array}{ccccccc}
 & & & & \text{and 3 and 5 through 2} & & \\
\end{array}
$$

$$
\begin{bmatrix}
0 & 9 & \infty & \infty & 7 & \infty & \infty \\
9 & 0 & 5 & \infty & 4 & \infty & \infty \\
\infty & 5 & 0 & 6 & \boxed{\infty} & 2 & 6 \\
\infty & \infty & 6 & 0 & \infty & 7 & 6 \\
7 & 4 & \infty & \infty & 0 & 4 & \infty \\
\infty & \infty & 2 & 7 & 4 & 0 & 7 \\
\infty & \infty & 6 & 6 & \infty & 7 & 0
\end{bmatrix}
$$

function $\textsc{Floyd}(W[\cdot,\cdot], n)$
    $D \leftarrow W$
    for $k \leftarrow 1$ to n **do**
        for $i \leftarrow 1$ to n **do**
            for $j \leftarrow 1$ to n **do**
                $D[i,j] \leftarrow \min\left(D[i,j], D[i,k] + D[k,j]\right)$
    return D

Floyd's algorithm

- For k=2, ***D*[1,2] = 9** and ***D*[2,3]=5**; and ***D*[5,2] = 4** and ***D*[2,3]=5**
  - Hence, we can make ***D*[1,3]=14** and ***D*[5,3]=9**

$$
\begin{bmatrix}
0 & 9 & 14 & \infty & 7 & \infty & \infty \\
9 & 0 & 5 & \infty & 4 & \infty & \infty \\
14 & 5 & 0 & 6 & 9 & 2 & 6 \\
\infty & \infty & 6 & 0 & \infty & 7 & 6 \\
7 & 4 & 9 & \infty & 0 & 4 & \infty \\
\infty & \infty & 2 & 7 & 4 & 0 & 7 \\
\infty & \infty & 6 & 6 & \infty & 7 & 0
\end{bmatrix}
$$

function $\textsc{Floyd}(W[\cdot,\cdot], n)$
  $D \leftarrow W$
  for $k \leftarrow 1$ to n **do**
    for $i \leftarrow 1$ to n **do**
      for $j \leftarrow 1$ to n **do**
        $D[i,j] \leftarrow \min\left(D[i,j], D[i,k] + D[k,j]\right)$
  return D

Floyd's algorithm

- For k=2, ***D*[1,2] = 9** and ***D*[2,3]=5**; and ***D*[5,2] = 4** and ***D*[2,3]=5**
  - Hence, we can make ***D*[1,3]=14** and ***D*[5,3]=9**
  - Note that the graph is undirected, which makes the matrix symmetric

$$\begin{bmatrix} 0 & 9 & 14 & \infty & 7 & \infty & \infty \\ 9 & 0 & 5 & \infty & 4 & \infty & \infty \\ 14 & 5 & 0 & 6 & 9 & 2 & 6 \\ \infty & \infty & 6 & 0 & \infty & 7 & 6 \\ 7 & 4 & 9 & \infty & 0 & 4 & \infty \\ \infty & \infty & 2 & 7 & 4 & 0 & 7 \\ \infty & \infty & 6 & 6 & \infty & 7 & 0 \end{bmatrix}$$

function $\textsc{Floyd}(W[\cdot,\cdot], n)$

$\quad D \leftarrow W$

$\quad$ **for** $k \leftarrow 1$ to n **do**

$\quad\quad$ **for** $i \leftarrow 1$ to n **do**

$\quad\quad\quad$ **for** $j \leftarrow 1$ to n **do**

$\quad\quad\quad\quad D[i,j] \leftarrow \min\left(D[i,j], D[i,k] + D[k,j]\right)$

$\quad$ **return** D

Floyd's algorithm

- For k=3, we can reach all other nodes in the graph

$$\begin{bmatrix} 0 & 9 & \mathbf{14} & \infty & 7 & \infty & \infty \\ 9 & 0 & \mathbf{5} & \infty & 4 & \infty & \infty \\ \mathbf{14} & \mathbf{5} & 0 & \mathbf{6} & \mathbf{9} & \mathbf{2} & \mathbf{6} \\ \infty & \infty & \mathbf{6} & 0 & \infty & 7 & 6 \\ 7 & 4 & \mathbf{9} & \infty & 0 & 4 & \infty \\ \infty & \infty & \mathbf{2} & 7 & 4 & 0 & 7 \\ \infty & \infty & \mathbf{6} & 6 & \infty & 7 & 0 \end{bmatrix}$$

function $\textsc{Floyd}(W[\cdot,\cdot], n)$
    $D \leftarrow W$
    for $k \leftarrow 1$ to n **do**
        for $i \leftarrow 1$ to n **do**
            for $j \leftarrow 1$ to n **do**
                $D[i,j] \leftarrow \min(D[i,j], D[i,k] + D[k,j])$
    return D

Floyd's algorithm

- For k=3, we can reach all other nodes in the graph

function $\textsc{Floyd}(W[\cdot,\cdot], n)$

  $D \leftarrow W$

  for $k \leftarrow 1$ to n **do**

    for $i \leftarrow 1$ to n **do**

      for $j \leftarrow 1$ to n **do**

        $D[i,j] \leftarrow \min\left(D[i,j], D[i,k] + D[k,j]\right)$

  return D

These and all infinites will be gone…

$$\begin{bmatrix} 0 & 9 & \mathbf{14} & \infty & 7 & \infty & \infty \\ 9 & 0 & \mathbf{5} & \infty & 4 & \infty & \infty \\ \mathbf{14} & \mathbf{5} & 0 & \mathbf{6} & \mathbf{9} & \mathbf{2} & \mathbf{6} \\ \infty & \infty & \mathbf{6} & 0 & \infty & 7 & 6 \\ 7 & 4 & \mathbf{9} & \infty & 0 & 4 & \infty \\ \infty & \infty & \mathbf{2} & 7 & 4 & 0 & 7 \\ \infty & \infty & \mathbf{6} & 6 & \infty & 7 & 0 \end{bmatrix}$$

Floyd's algorithm

- For k=3, we can reach all other nodes in the graph

$$\begin{bmatrix} 0 & 9 & 14 & \mathbf{20} & 7 & \mathbf{16} & \mathbf{20} \\ 9 & 0 & 5 & \mathbf{11} & 4 & \mathbf{7} & \mathbf{11} \\ 14 & 5 & 0 & 6 & 9 & 2 & 6 \\ \mathbf{20} & \mathbf{11} & 6 & 0 & \mathbf{15} & 7 & 6 \\ 7 & 4 & 9 & \mathbf{15} & 0 & 4 & \mathbf{15} \\ \mathbf{16} & \mathbf{7} & 2 & 7 & 4 & 0 & 7 \\ \mathbf{20} & \mathbf{11} & 6 & 6 & \mathbf{15} & 7 & 0 \end{bmatrix}$$

function $\textsc{Floyd}(W[\cdot,\cdot], n)$
  $D \leftarrow W$
  for $k \leftarrow 1$ to n **do**
    for $i \leftarrow 1$ to n **do**
      for $j \leftarrow 1$ to n **do**
        $D[i,j] \leftarrow \min\left(D[i,j], D[i,k] + D[k,j]\right)$
  return D

Floyd's algorithm

- For k=3, we can reach all other nodes in the graph
  - However, these may not be the shortest paths

$$
\begin{bmatrix}
0 & 9 & 14 & \mathbf{20} & 7 & \mathbf{16} & \mathbf{20} \\
9 & 0 & 5 & \mathbf{11} & 4 & \mathbf{7} & \mathbf{11} \\
14 & 5 & 0 & 6 & 9 & 2 & 6 \\
\mathbf{20} & \mathbf{11} & 6 & 0 & \mathbf{15} & 7 & 6 \\
7 & 4 & 9 & \mathbf{15} & 0 & 4 & \mathbf{15} \\
\mathbf{16} & \mathbf{7} & 2 & 7 & 4 & 0 & 7 \\
\mathbf{20} & \mathbf{11} & 6 & 6 & \mathbf{15} & 7 & 0
\end{bmatrix}
$$

function $\text{FLOYD}(W[\cdot,\cdot], n)$
    $D \leftarrow W$
    for $k \leftarrow 1$ to n **do**
        for $i \leftarrow 1$ to n **do**
            for $j \leftarrow 1$ to n **do**
                $D[i,j] \leftarrow \min\left(D[i,j], D[i,k] + D[k,j]\right)$
    return D

Floyd's algorithm

- For k=3, we can reach all other nodes in the graph
  - However, these may not be the shortest paths
- There will be no changes for k=4

$$\begin{bmatrix} 0 & 9 & 14 & \mathbf{20} & 7 & \mathbf{16} & \mathbf{20} \\ 9 & 0 & 5 & \mathbf{11} & 4 & \mathbf{7} & \mathbf{11} \\ 14 & 5 & 0 & 6 & 9 & 2 & 6 \\ \mathbf{20} & \mathbf{11} & 6 & 0 & \mathbf{15} & 7 & 6 \\ 7 & 4 & 9 & \mathbf{15} & 0 & 4 & \mathbf{15} \\ \mathbf{16} & \mathbf{7} & 2 & 7 & 4 & 0 & 7 \\ \mathbf{20} & \mathbf{11} & 6 & 6 & \mathbf{15} & 7 & 0 \end{bmatrix}$$

function $\textsc{Floyd}(W[\cdot,\cdot], n)$
$\quad D \leftarrow W$
$\quad$**for** $k \leftarrow 1$ to n **do**
$\quad\quad$**for** $i \leftarrow 1$ to n **do**
$\quad\quad\quad$**for** $j \leftarrow 1$ to n **do**
$\quad\quad\quad\quad D[i,j] \leftarrow \min\left(D[i,j], D[i,k] + D[k,j]\right)$
$\quad$**return** D

Floyd's algorithm

- Let's look at the next steps:

k=5

$$\begin{bmatrix} 0 & 9 & 14 & 20 & \mathbf{7} & 16 & 20 \\ 9 & 0 & 5 & 11 & \mathbf{4} & 7 & 11 \\ 14 & 5 & 0 & 6 & \mathbf{9} & 2 & 6 \\ 20 & 11 & 6 & 0 & \mathbf{15} & 7 & 6 \\ \mathbf{7} & \mathbf{4} & \mathbf{9} & \mathbf{15} & 0 & \mathbf{4} & \mathbf{15} \\ 16 & 7 & 2 & 7 & \mathbf{4} & 0 & 7 \\ 20 & 11 & 6 & 6 & \mathbf{15} & 7 & 0 \end{bmatrix}$$

Floyd's algorithm

- Let’s look at the next steps:

Taking 7+4 is better than 16

k=5

$$\begin{bmatrix} 0 & 9 & 14 & 20 & \mathbf{7} & \boxed{16} & 20 \\ 9 & 0 & 5 & 11 & \mathbf{4} & 7 & 11 \\ 14 & 5 & 0 & 6 & \mathbf{9} & 2 & 6 \\ 20 & 11 & 6 & 0 & \mathbf{15} & 7 & 6 \\ \mathbf{7} & \mathbf{4} & \mathbf{9} & \mathbf{15} & 0 & \mathbf{4} & \mathbf{15} \\ 16 & 7 & 2 & 7 & \mathbf{4} & 0 & 7 \\ 20 & 11 & 6 & 6 & \mathbf{15} & 7 & 0 \end{bmatrix}$$

Floyd's algorithm

- Let’s look at the next steps:

$$k\!=\!5$$

$$\begin{bmatrix} 0 & 9 & 14 & 20 & 7 & \mathbf{11} & 20 \\ 9 & 0 & 5 & 11 & 4 & 7 & 11 \\ 14 & 5 & 0 & 6 & 9 & 2 & 6 \\ 20 & 11 & 6 & 0 & 15 & 7 & 6 \\ 7 & 4 & 9 & 15 & 0 & 4 & 15 \\ \mathbf{11} & 7 & 2 & 7 & 4 & 0 & 7 \\ 20 & 11 & 6 & 6 & 15 & 7 & 0 \end{bmatrix}$$

Floyd's algorithm

- Let's look at the next steps:

k=5 k=6

$$\left[\begin{array}{ccccccc} 0 & 9 & 14 & 20 & 7 & 11 & 20 \\ 9 & 0 & 5 & 11 & 4 & 7 & 11 \\ 14 & 5 & 0 & 6 & 9 & 2 & 6 \\ 20 & 11 & 6 & 0 & 15 & 7 & 6 \\ 7 & 4 & 9 & 15 & 0 & 4 & 15 \\ 11 & 7 & 2 & 7 & 4 & 0 & 7 \\ 20 & 11 & 6 & 6 & 15 & 7 & 0 \end{array}\right] \left[\begin{array}{ccccccc} 0 & 9 & 14 & 20 & 7 & \mathbf{11} & 20 \\ 9 & 0 & 5 & 11 & 4 & \mathbf{7} & 11 \\ 14 & 5 & 0 & 6 & 9 & \mathbf{2} & 6 \\ 20 & 11 & 6 & 0 & 15 & \mathbf{7} & 6 \\ 7 & 4 & 9 & 15 & 0 & \mathbf{4} & 15 \\ \mathbf{11} & \mathbf{7} & \mathbf{2} & \mathbf{7} & \mathbf{4} & 0 & \mathbf{7} \\ 20 & 11 & 6 & 6 & 15 & \mathbf{7} & 0 \end{array}\right]$$

Floyd's algorithm

- Let’s look at the next steps:

Taking 11+2 is better than 14

$$k=5 \qquad\qquad k=6$$

$$\begin{bmatrix} 0 & 9 & 14 & 20 & 7 & 11 & 20 \\ 9 & 0 & 5 & 11 & 4 & 7 & 11 \\ 14 & 5 & 0 & 6 & 9 & 2 & 6 \\ 20 & 11 & 6 & 0 & 15 & 7 & 6 \\ 7 & 4 & 9 & 15 & 0 & 4 & 15 \\ 11 & 7 & 2 & 7 & 4 & 0 & 7 \\ 20 & 11 & 6 & 6 & 15 & 7 & 0 \end{bmatrix} \begin{bmatrix} 0 & 9 & \boxed{14} & 20 & 7 & \mathbf{11} & 20 \\ 9 & 0 & 5 & 11 & 4 & \mathbf{7} & 11 \\ 14 & 5 & 0 & 6 & 9 & \mathbf{2} & 6 \\ 20 & 11 & 6 & 0 & 15 & \mathbf{7} & 6 \\ 7 & 4 & 9 & 15 & 0 & \mathbf{4} & 15 \\ \mathbf{11} & \mathbf{7} & \mathbf{2} & \mathbf{7} & \mathbf{4} & 0 & \mathbf{7} \\ 20 & 11 & 6 & 6 & 15 & \mathbf{7} & 0 \end{bmatrix}$$

Floyd's algorithm

- Let’s look at the next steps:

Taking 11+7 is better than 20

$$k=5 \qquad\qquad k=6$$

$$\begin{bmatrix} 0 & 9 & 14 & 20 & 7 & 11 & 20 \\ 9 & 0 & 5 & 11 & 4 & 7 & 11 \\ 14 & 5 & 0 & 6 & 9 & 2 & 6 \\ 20 & 11 & 6 & 0 & 15 & 7 & 6 \\ 7 & 4 & 9 & 15 & 0 & 4 & 15 \\ 11 & 7 & 2 & 7 & 4 & 0 & 7 \\ 20 & 11 & 6 & 6 & 15 & 7 & 0 \end{bmatrix} \begin{bmatrix} 0 & 9 & 14 & 20 & 7 & \mathbf{11} & \boxed{20} \\ 9 & 0 & 5 & 11 & 4 & \mathbf{7} & 11 \\ 14 & 5 & 0 & 6 & 9 & \mathbf{2} & 6 \\ 20 & 11 & 6 & 0 & 15 & \mathbf{7} & 6 \\ 7 & 4 & 9 & 15 & 0 & \mathbf{4} & 15 \\ \mathbf{11} & \mathbf{7} & \mathbf{2} & \mathbf{7} & \mathbf{4} & 0 & \mathbf{7} \\ 20 & 11 & 6 & 6 & 15 & \mathbf{7} & 0 \end{bmatrix}$$

Floyd's algorithm

- Let's look at the next steps:

$$k=5 \qquad\qquad k=6$$

$$\begin{bmatrix} 0 & 9 & 14 & 20 & 7 & \mathbf{11} & 20 \\ 9 & 0 & 5 & 11 & 4 & 7 & 11 \\ 14 & 5 & 0 & 6 & 9 & 2 & 6 \\ 20 & 11 & 6 & 0 & 15 & 7 & 6 \\ 7 & 4 & 9 & 15 & 0 & 4 & 15 \\ \mathbf{11} & 7 & 2 & 7 & 4 & 0 & 7 \\ 20 & 11 & 6 & 6 & 15 & 7 & 0 \end{bmatrix} \begin{bmatrix} 0 & 9 & \mathbf{13} & \mathbf{18} & 7 & 11 & \mathbf{18} \\ 9 & 0 & 5 & 11 & 4 & 7 & 11 \\ \mathbf{13} & 5 & 0 & 6 & \mathbf{6} & 2 & 6 \\ \mathbf{18} & 11 & 6 & 0 & \mathbf{11} & 7 & 6 \\ 7 & 4 & \mathbf{6} & \mathbf{11} & 0 & 4 & \mathbf{11} \\ 11 & 7 & 2 & 7 & 4 & 0 & 7 \\ \mathbf{18} & 11 & 6 & 6 & \mathbf{11} & 7 & 0 \end{bmatrix}$$

Floyd's algorithm

- Let's look at the next steps:

$$
\begin{array}{ccc}
k=5 & k=6 & k=7 \\
\begin{bmatrix}
0 & 9 & 14 & 20 & 7 & \mathbf{11} & 20 \\
9 & 0 & 5 & 11 & 4 & 7 & 11 \\
14 & 5 & 0 & 6 & 9 & 2 & 6 \\
20 & 11 & 6 & 0 & 15 & 7 & 6 \\
7 & 4 & 9 & 15 & 0 & 4 & 15 \\
\mathbf{11} & 7 & 2 & 7 & 4 & 0 & 7 \\
20 & 11 & 6 & 6 & 15 & 7 & 0
\end{bmatrix}
&
\begin{bmatrix}
0 & 9 & \mathbf{13} & \mathbf{18} & 7 & 11 & \mathbf{18} \\
9 & 0 & 5 & 11 & 4 & 7 & 11 \\
\mathbf{13} & 5 & 0 & 6 & \mathbf{6} & 2 & 6 \\
\mathbf{18} & 11 & 6 & 0 & \mathbf{11} & 7 & 6 \\
7 & 4 & \mathbf{6} & \mathbf{11} & 0 & 4 & \mathbf{11} \\
11 & 7 & 2 & 7 & 4 & 0 & 7 \\
\mathbf{18} & 11 & 6 & 6 & \mathbf{11} & 7 & 0
\end{bmatrix}
&
\begin{bmatrix}
0 & 9 & 13 & 18 & 7 & 11 & 18 \\
9 & 0 & 5 & 11 & 4 & 7 & 11 \\
13 & 5 & 0 & 6 & 6 & 2 & 6 \\
18 & 11 & 6 & 0 & 11 & 7 & 6 \\
7 & 4 & 6 & 11 & 0 & 4 & 11 \\
11 & 7 & 2 & 7 & 4 & 0 & 7 \\
18 & 11 & 6 & 6 & 11 & 7 & 0
\end{bmatrix}
\end{array}
$$

Floyd's algorithm

- Let’s look at the next steps:

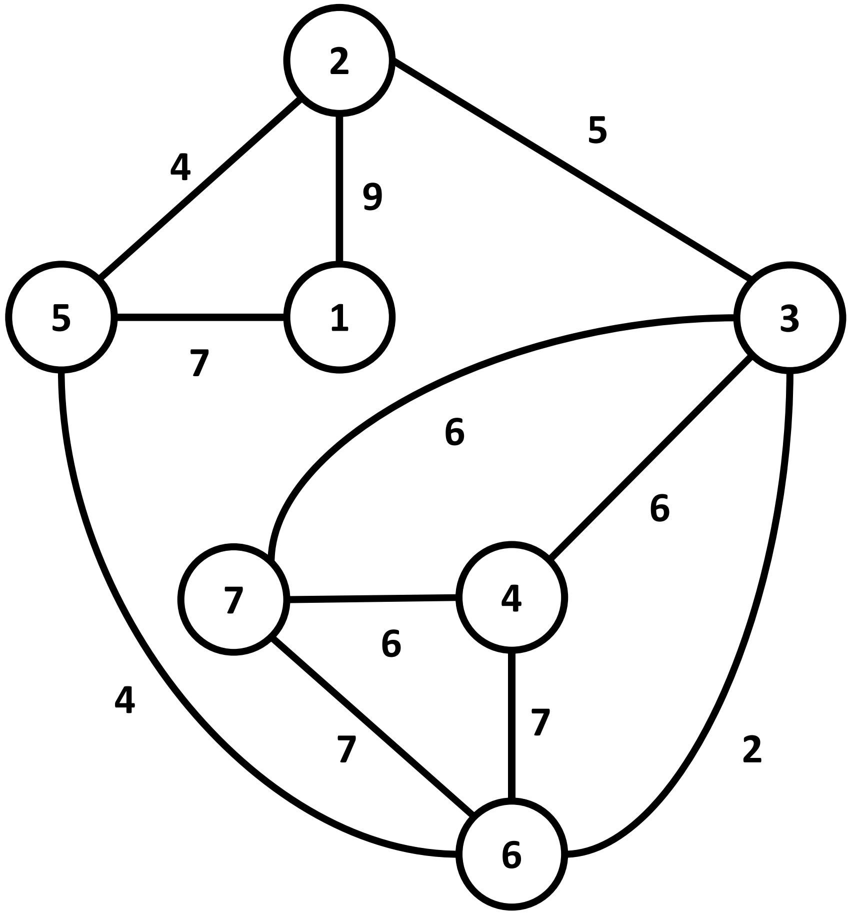
$$k=5 \qquad\qquad k=6 \qquad\qquad k=7$$

$$\begin{bmatrix} 0 & 9 & 14 & 20 & 7 & \mathbf{11} & 20 \\ 9 & 0 & 5 & 11 & 4 & 7 & 11 \\ 14 & 5 & 0 & 6 & 9 & 2 & 6 \\ 20 & 11 & 6 & 0 & 15 & 7 & 6 \\ 7 & 4 & 9 & 15 & 0 & 4 & 15 \\ \mathbf{11} & 7 & 2 & 7 & 4 & 0 & 7 \\ 20 & 11 & 6 & 6 & 15 & 7 & 0 \end{bmatrix} \begin{bmatrix} 0 & 9 & \mathbf{13} & \mathbf{18} & 7 & 11 & \mathbf{18} \\ 9 & 0 & 5 & 11 & 4 & 7 & 11 \\ \mathbf{13} & 5 & 0 & 6 & \mathbf{6} & 2 & 6 \\ \mathbf{18} & 11 & 6 & 0 & \mathbf{11} & 7 & 6 \\ 7 & 4 & \mathbf{6} & \mathbf{11} & 0 & 4 & \mathbf{11} \\ 11 & 7 & 2 & 7 & 4 & 0 & 7 \\ \mathbf{18} & 11 & 6 & 6 & \mathbf{11} & 7 & 0 \end{bmatrix} \begin{bmatrix} 0 & 9 & 13 & 18 & 7 & 11 & 18 \\ 9 & 0 & 5 & 11 & 4 & 7 & 11 \\ 13 & 5 & 0 & 6 & 6 & 2 & 6 \\ 18 & 11 & 6 & 0 & 11 & 7 & 6 \\ 7 & 4 & 6 & 11 & 0 & 4 & 11 \\ 11 & 7 & 2 & 7 & 4 & 0 & 7 \\ 18 & 11 & 6 & 6 & 11 & 7 & 0 \end{bmatrix}$$

- For k=7, D is unchanged. So we have found the best paths

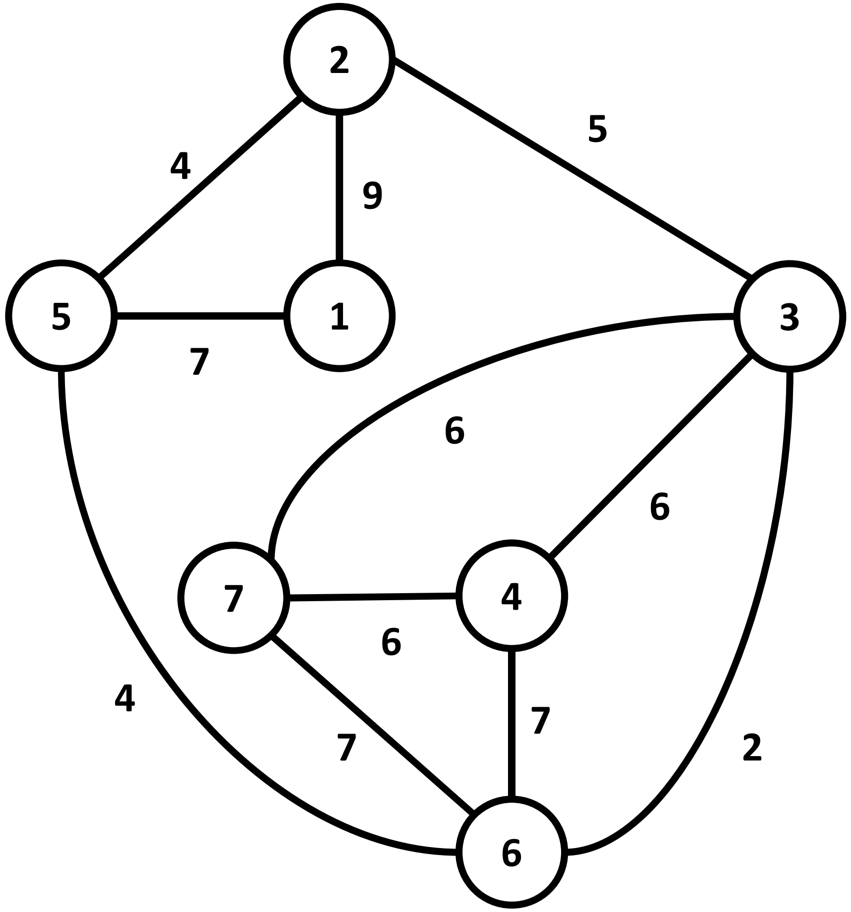
A sub-structure property

- For DP to be applicable, the problem must have a **sub-structure** that allows for a compositional solution



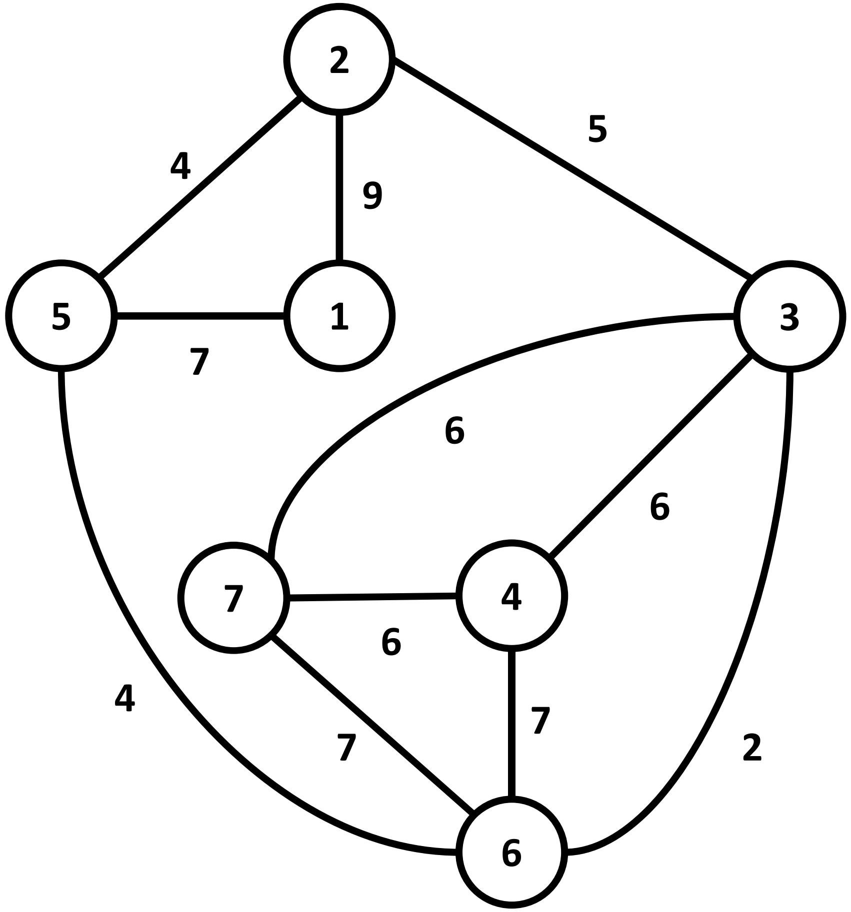
A sub-structure property

- For DP to be applicable, the problem must have a **sub-structure** that allows for a compositional solution
  - Shortest-path problems have this property



A sub-structure property

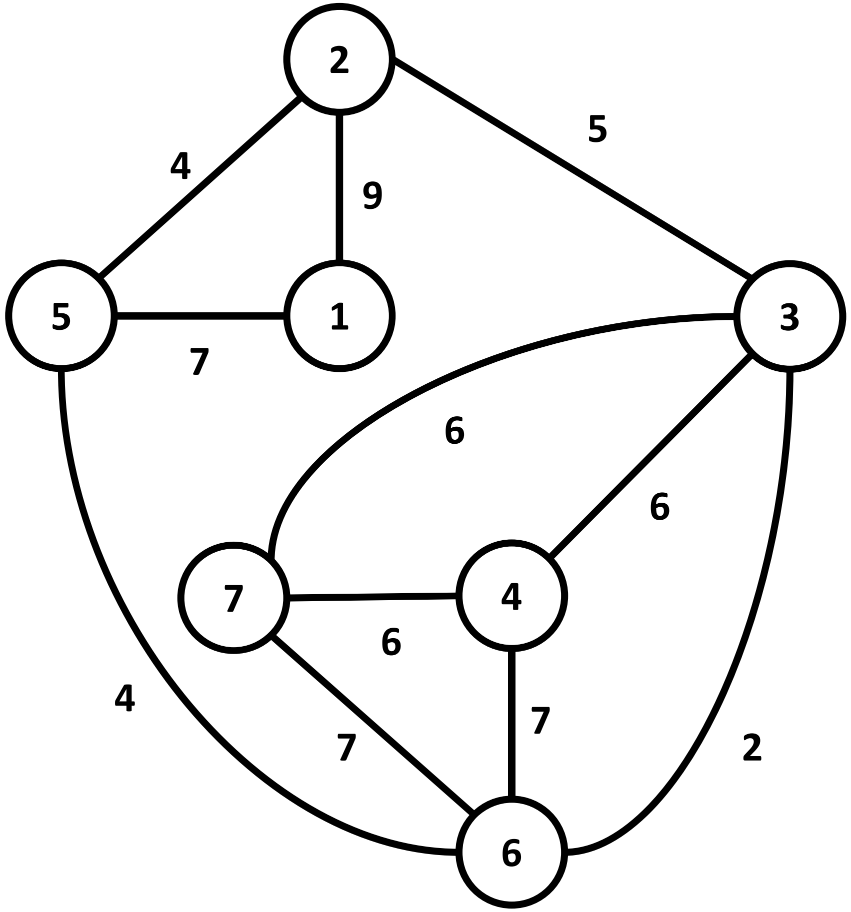
- For DP to be applicable, the problem must have a **sub-structure** that allows for a compositional solution
  - Shortest-path problems have this property
  - For example, if $\{x_1, x_2, \ldots, x_i, \ldots, x_n\}$ is a shortest path from x_1 to x_n then $\{x_1, x_2, \ldots, x_i\}$ is a shortest path from x_1 to x_i



A sub-structure property

- For DP to be applicable, the problem must have a **sub-structure** that allows for a compositional solution
  - Shortest-path problems have this property
  - For example, if $\{x_1, x_2, \ldots, x_i, \ldots, x_n\}$ is a shortest path from x_1 to x_n then $\{x_1, x_2, \ldots, x_i\}$ is a shortest path from x_1 to x_i

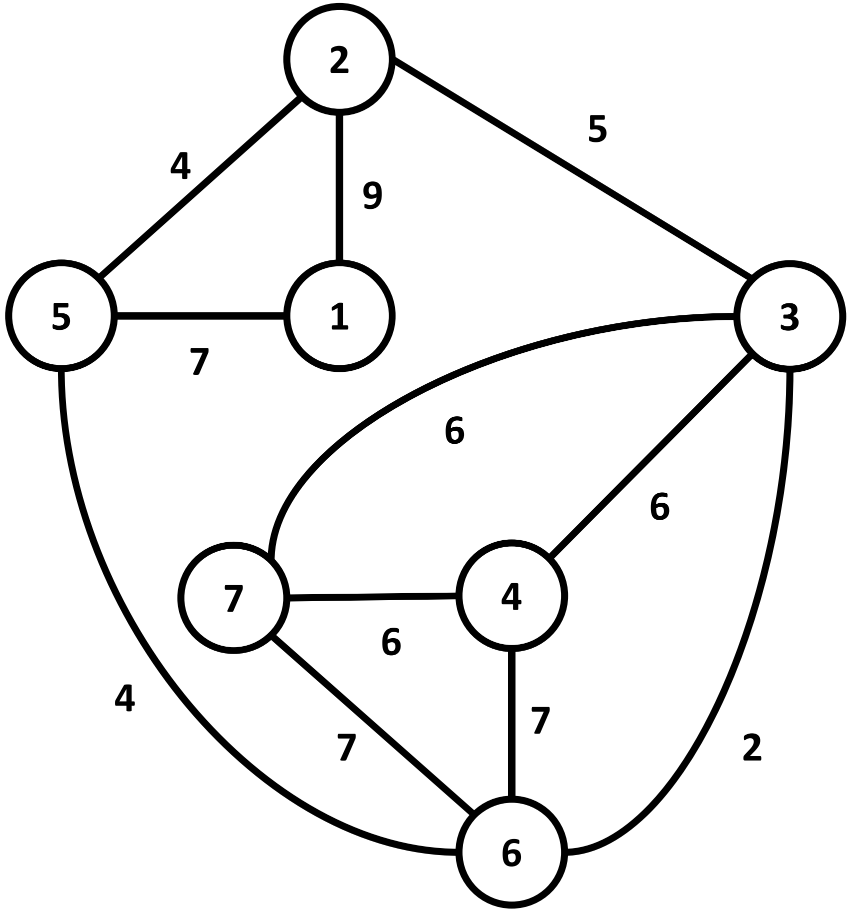
- Longest-path problems don't have this property



A sub-structure property

- For DP to be applicable, the problem must have a **sub-structure** that allows for a compositional solution
  - Shortest-path problems have this property
  - For example, if $\{x_1, x_2, \ldots, x_i, \ldots, x_n\}$ is a shortest path from x_1 to x_n then $\{x_1, x_2, \ldots, x_i\}$ is a shortest path from x_1 to x_i

- Longest-path problems don't have this property
  - For example, $\{1,2,5,6,7,4,3\}$ is a longest path from 1 to 3, but $\{1,2\}$ is not a longest path from 1 to 2, i.e., $\{1,5,6,7,4,3,2\}$ is longer



Next lecture

- Greedy techniques
  - Prim’s algorithm (Levitin Section 9.1)
  - Dijkstra’s algorithm (Levitin Section 9.3)