TA: Jade Cheng

ICS 241

Recitation Lecture Note #9

October 23, 2009

Recitation #9

Question: For each of these problems about a subway system, describe a weighted graph model that can be used to solve the problem. [Chapter 9.6 Review]

a. What is the least amount of time required to travel between two stops?

Answer: A simple graph with weighted edges will do it:

Vertices:	The stops that are presented on the subway map.
Edges:	The adjacent stops are connected with edges.
Weights:	The times required to travel between adjacent stops.

b. What is the minimum distance that can be traveled to reach a stop from another stop?

Answer: A simple graph with weighted edges will do it:

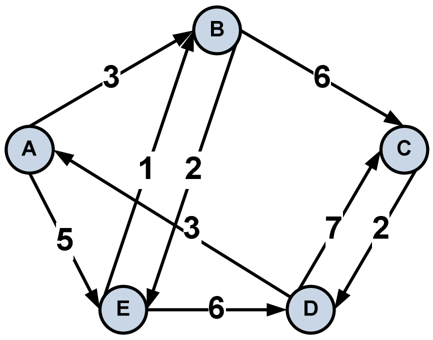
Vertices:	Same as above
Edges:	The adjacent stops are connected with edges.
Weights:	The distances between adjacent stops.

c. What is the least fare required to travel between two stops if fares between stops are added to give the total fare?

Answer: A simple graph with weighted edges will do it:

Vertices:	The stops that are presented on the subway map.
Edges:	The adjacent stops are connected with edges.
Weights:	The fares between adjacent stops.

Question: Find the length of the shortest path in the following edge weighted graphs by tracing Dijkstra's Algorithm. [Chapter 9.6 Review]



Review: The pseudo code for Dijkstra's algorithm:

```
// $G$ has vertices $a = v_0, v_1, \cdots, v\_n = z$ and weights $w(v_i, v_j)$
// where $w(v_i, v_j) = \infty$ if $\{v_i, v_j\}$ is not an edge in $G$
procedure Dijkstra ($G$: weighted connected simple graph, with all weights positive)

    // Initialize All labels are to be infinite.
    for $i := 1$ to $n$
        $L(v_i) := \infty$

    // Initialize the label of $a$, the starting vertex, to be $0$.
    $L(a) := 0$

    // Initialize set $S$ to be an empty set.
    $S := \emptyset$

    // At the end of the main loop, $L(z)$ = length of a shortest path from $a$ to $z$.
    while $z \notin S$
        begin

            // Find the vertex with the minimal label, and add it to set $S$.
            $u :=$ a vertex not in $S$ with $L(u)$ minimal
            $S := S \cup \{u\}$

            // Update the labels of vertices that are not in $S$.
            for all vertices $v$ not in $S$
                if $L(u) + w(u,v) < L(v)$ then $L(v) := L(u) + w(u,v)$
        end
    endwhile
```

a. Find the shortest path between A and D in the given graph, use vertex A as the source.

Answer: Let's start from the original graph:

Step 1:

$S = \emptyset$

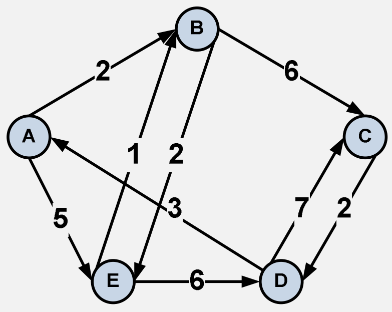
$A = \infty$

$B = \infty$

$C = \infty$

$D = \infty$

$E = \infty$



Step 2:

$S = \{A\}$

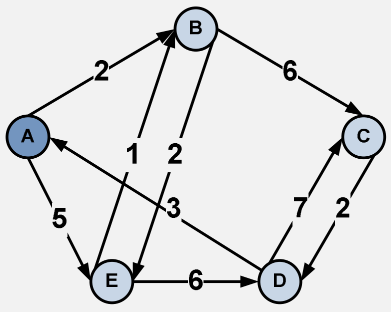
$A = 0$

$B = 0 + 2 = 2$

$E = 0 + 5 = 5$

$C = \infty$

$D = \infty$



Step 3:

$S = \{A, B\}$

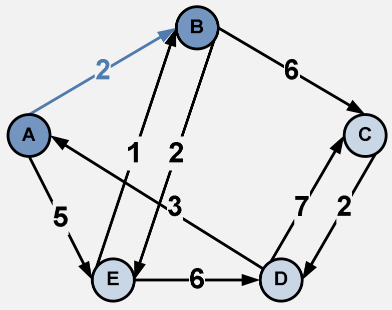
$A = 0$

$B = 2$

$E = 2 + 2 = 4$

$C = 2 + 6 = 8$

$D = \infty$



Step 4:

$S = \{A, B, E\}$

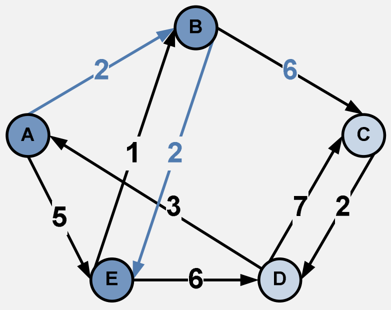
$A = 0$

$B = 2$

$E = 4$

$C = 8$

$D = 4 + 6 = 10$



Step 5:

$S = \{A, B, E, C\}$

$A = 0$

$B = 2$

$E = 4$

$C = 8$

$D = 10$

Step 6:

$S = \{A, B, E, C, D\}$

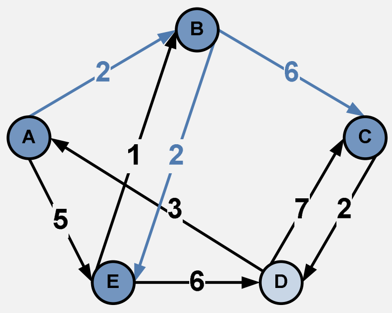
$A = 0$

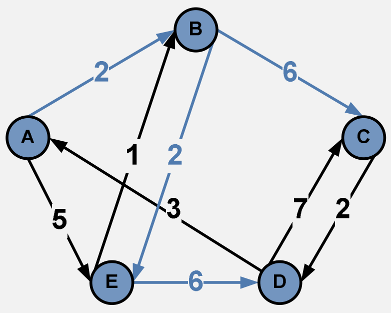
$B = 2$

$E = 4$

$C = 8$

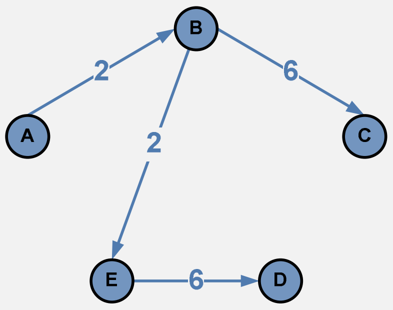
$D = 10$





At this point, vertex z, the destination, in this case D is added to set S, therefore the main loop condition, $z \notin S$, can't be satisfied any more. Program exit the loop and terminate.

Now let's take a look at what we have obtained from the algorithm execution. Obviously we have a set of labels, which represent the lengths of the shortest paths from the starting vertex, in this case A, to reach all other vertices of the input graph. This conclusion is clear by looking at the graph as well:



We learned from this output graph that all pairs of shortest paths starting from vertex A. Namely, the minimal cost of traveling from vertex A to B is 2, from A to C is 8 going through B, from A to D is 10 going through B and E, and from A to E is 4 going through B.

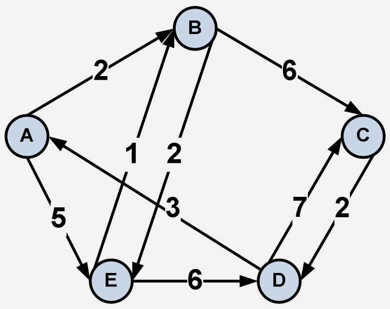
You might wonder, is it necessary to compute them all, as the question asks for only the shortest path from A to D. The answer to this question is, yes! Resolving the "whole world" in order to find one optimal path turned out to be the best way to do it.

b. Find the shortest path between D and B in the given graph, use vertex D as the source

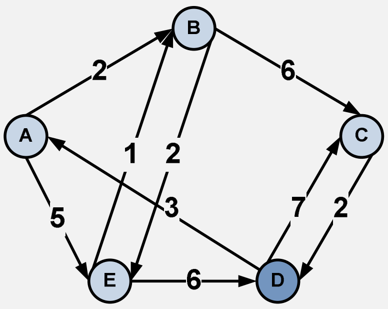
Answer: Let's start from the original graph:

Step 1:	*Step 2:*
$S = \emptyset$	$S = \{D\}$
$A = \infty$	$D = 0$
$B = \infty$	$A = 0 + 3 = 3$
$C = \infty$	$C = 0 + 7 = 7$
$D = \infty$	$B = \infty$

$E = \infty$



$E = \infty$



Step 3:

$S = \{D, A\}$

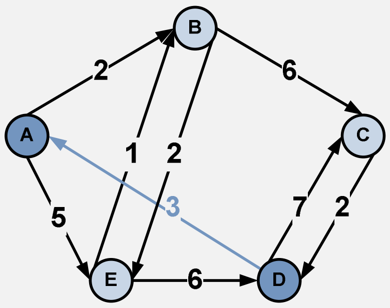
$D = 0$

$A = 3$

$B = 3 + 2 = 5$

$C = 7$

$E = 3 + 5 = 8$



Step 4:

$S = \{D, A, B\}$

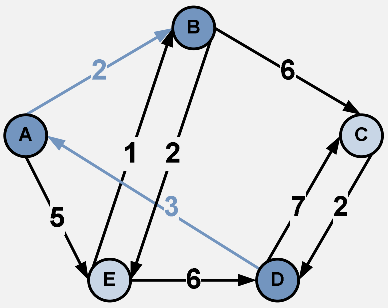
$D = 0$

$A = 3$

$B = 5$

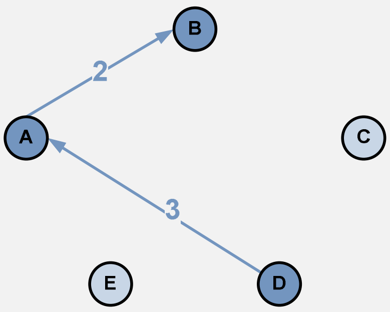
$C = 7$

$E = 5 + 2 = 7$



At this point, vertex z, the destination, in this case B is added to set S, therefore the main loop condition, $z \notin S$, can't be satisfied any more. Program exit the loop and terminate.

Now let's take a look at what we have obtained from the algorithm execution. Obviously we have a set of labels, which represent the lengths of the shortest paths from the starting vertex, in this case D, to reach all other vertices in set S. They are vertices A and B. And we don't know about the shortest paths to the rest of the vertices in this case. This conclusion is clear by looking at the graph as well:



Question: Extend Dijkstra's algorithm for finding the length of a shortest path between two vertices in a weighted simple connected graph so that the length of a shortest path between the vertex a and every other vertex of the graph is found. [Chapter 9.6 Review]

Answer: As you might notice, we've already done this once. In part **a** of the previous question, we trace through the algorithm and resolved the entire graph starting from the specified vertex. This is because the destination vertex happen to be the last vertex that was added in the set S. Before adding the destination, we've already learned the shortest paths from the source to all other vertices.

Follow the same trend of thoughts, if we let the algorithm continue until it reaches the point that all vertices are in set S, we would end up with a graph connecting all other vertices with the source in the shortest paths.

Modified Dijkstra's algorithm that finds the shortest paths between a and every other vertiex:

```
// $G$ has vertices $a = v_0, v_1, \cdots, v\_n = z$ and weights $w(v_i, v_j)$
// where $w(v_i, v_j) = \infty$ if $\{v_i, v_j\}$ is not an edge in $G$
procedure Dijkstra ($G$: weighted connected simple graph, with all weights positive)

        // Initialize All labels are to be infinite.
        for $i := 1$ to $n$
                $L(v_i) := \infty$

        // Initialize the label of $a$, the starting vertex, to be 0.
        $L(a) := 0$

        // Initialize set $S$ to be an empty set.
        $S := \emptyset$

        // Main loop terminates when $S$ contains all vertices in the input graph.
        while $\bar{S} \neq \emptyset$
                begin

                        // Find the vertex with the minimal label, and add it to set $S$.
                        $u :=$ a vertex not in $S$ with $L(u)$ minimal
                        $S := S \cup \{u\}$

                        // Update the labels of vertices that are not in $S$.
                        for all vertices $v$ not in $S$
                                if $L(u) + w(u,v) < L(v)$ then $L(v) := L(u) + w(u,v)$
                end
        endwhile
```

Let's look at an example, if we apply the modified Dijkstra's algorithm on part **b** of the previous question, we would have two more steps:

Step 5:

$S = \{D, A, B, C\}$

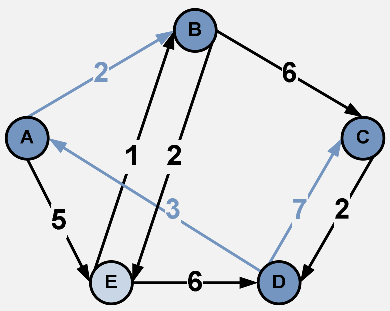
$D = 0$

$A = 3$

$B = 5$

$C = 7$

$E = 7$



Step 6:

$S = \{D, A, B, C, E\}$

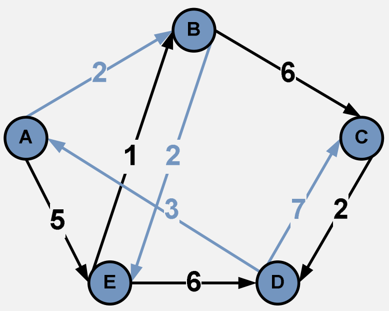
$D = 0$

$A = 3$

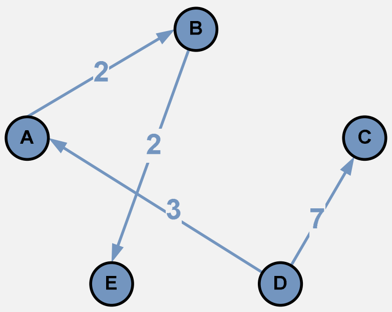
$B = 5$

$C = 7$

$E = 7$



Now let's take a look at what we have. We have a set of labels consisting all vertices, which represent the lengths of the shortest paths from the starting vertex, in this case D, to reach all other vertices of the input graph. This conclusion is clear by looking at the graph as well:



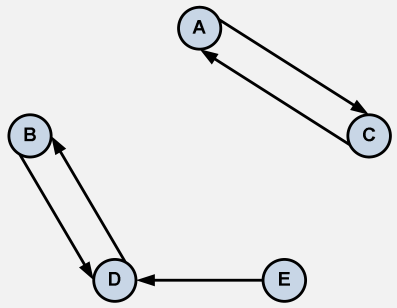
Question: Use Warshall's Algorithm to find the transitive closures of the relation defined on $\{a, b, c, d, e\}$: $\{(a, c), (b, d), (c, a), (d, b), (e, d)\}$. [Chapter 8.4 Review]

Review:

```
Procedure Warshall ($M_R : n \times n$ zero-one matrix)
$W := M_R$
for $k := 1$ to $n$
begin
        for $i := 1$ to $n$
        begin
                for $j := 1$ to $n$
                $w_{ij} := w_{ij} \vee (w_{ik} \wedge w_{kj})$
        end
end $\{W = [w_{ij}]$ is $M_R\}$
```

Answer: We talked about Warshall's Algorithm tracing by going through the loop counters. This is a time-consuming procedure, although we understand that it takes the computer no time to accomplish the task. Here we'll introduce a simpler "graphical" way to trace this algorithm. Maybe we'll understand the code better by looking at it from a different perspective.

The given relation can be represented as an adjacent matrix: $W_0 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$, or as a directed graph:

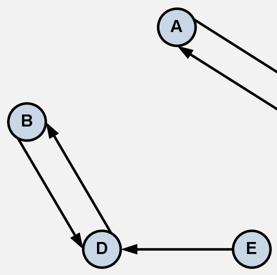


$w_{ij} := w_{ij} \vee (w_{ik} \wedge w_{kj})$, tells us that the algorithm goes through the 1's and 0's in the matrix and turns on some of the 0 bits according to the conditions. This can also be seen as extra edges added to the directed graph, of course. We are examining edge $w_{ij} = 0$, if $w_{ik} \wedge w_{kj}$ holds, we turn on $w_{ij} = 1$, which is saying if vertices i, k and vertices k, j are both connected, we connect vertices i, j. This is the transitivity property by definition. If two vertices are connected through the vertex k, this edge would be connected after this round k execution.

Step 0:

The original input matrix and graph

$$W_0 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

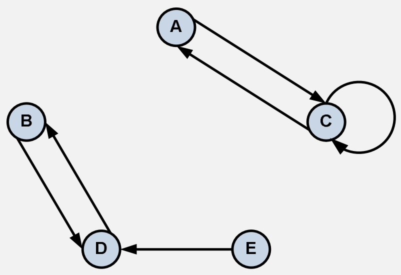


Step 1: done with $k = 1$

Any new nodes connected through A?

Yes, $C \to A \to C$, so (C, C)

$$W_1 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

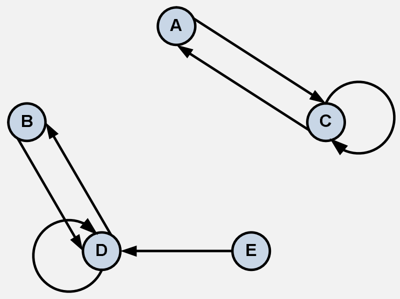


Step 2: done with $k = 2$

Any new nodes connected through B

Yes, $D \to B \to D$, so (D, D)

$$W_2 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

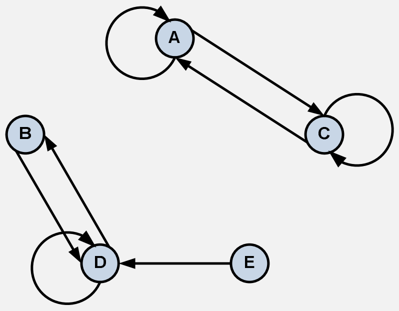


Step 3: done with $k = 3$

Any new nodes connected through C?

Yes, $A \to C \to A$, so (A, A)

$$W_3 = \begin{bmatrix} \mathbf{1} & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$



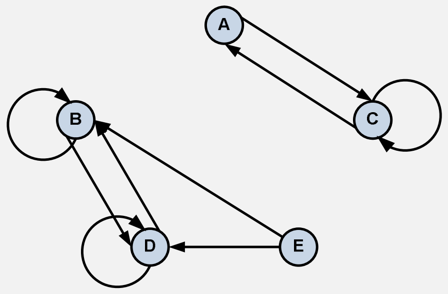
Step 4: done with $k = 4$

Any new nodes connected through D

Yes, $B \to D \to B$, so (B, B)

and $E \to D \to B$, so (E, B)

$$W_4 = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & \mathbf{1} & 0 & 1 & 0 \end{bmatrix}$$

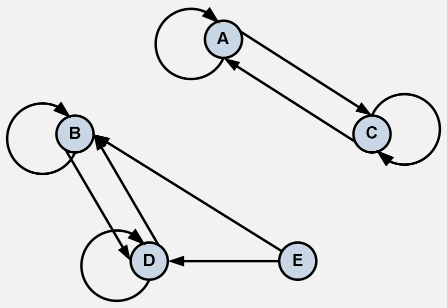


Step 5: done with $k = 5$

Any new nodes connected through E?

No

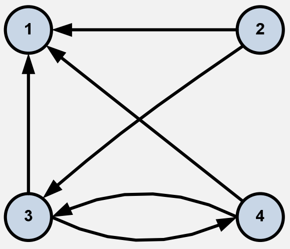
$$W_5 = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$



At this point we've looped from 1 to $n = 5$ for the outer loop counter k. So we are done, and we have the final adjacent matrix and the graph representation. They both demonstrate the transitive closure property of the original relation on $\{a, b, c, d, e\}$.

Question: Use Warshall's Algorithm to find the transitive closure of the relations on the relation $\{(2,1), (2,3), (3,1), (3,4), (4,1), (4,3)\}$ on $\{1, 2, 3, 4\}$. [Chapter 8.4 Review]

Answer: Let's follow the same steps as we did for the previous question. The given relation can be represented as an adjacent matrix: $W_0 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$, or as a directed graph:

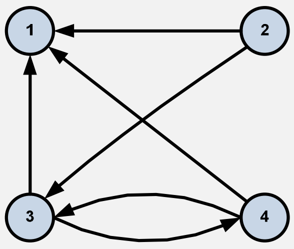


Step 1: done with $k = 1$

Any new nodes connected through 1?

No

$$W_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

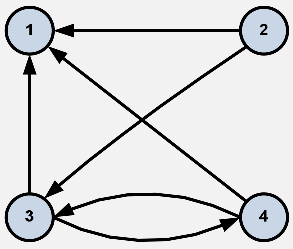


Step 2: done with $k = 2$

Any new nodes connected through 2?

No

$$W_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$



Step 3: done with $k = 3$

Any new nodes connected through 3?

Yes, $4 \rightarrow 3 \rightarrow 4$, so $(4, 4)$

and $2 \rightarrow 3 \rightarrow 4$, so $(2, 4)$

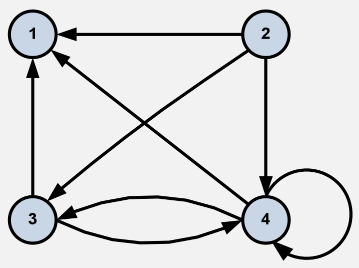
$$W_3 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & \mathbf{1} \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & \mathbf{1} \end{bmatrix}$$

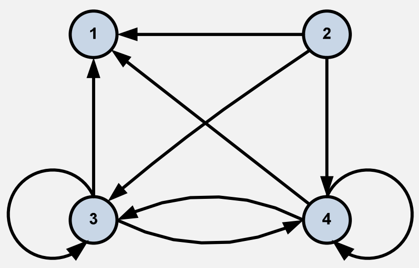
Step 4: done with $k = 4$

Any new nodes connected through 4?

Yes, $3 \rightarrow 4 \rightarrow 3$, so $(3, 3)$

$$W_4 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & \mathbf{1} & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}$$





At this point we've looped from 1 to $n = 4$ for the outer loop counter k. So we are done, and we have the final adjacent matrix and the graph representation. They both demonstrate the transitive closure property of the original relation on $\{1, 2, 3, 4\}$.