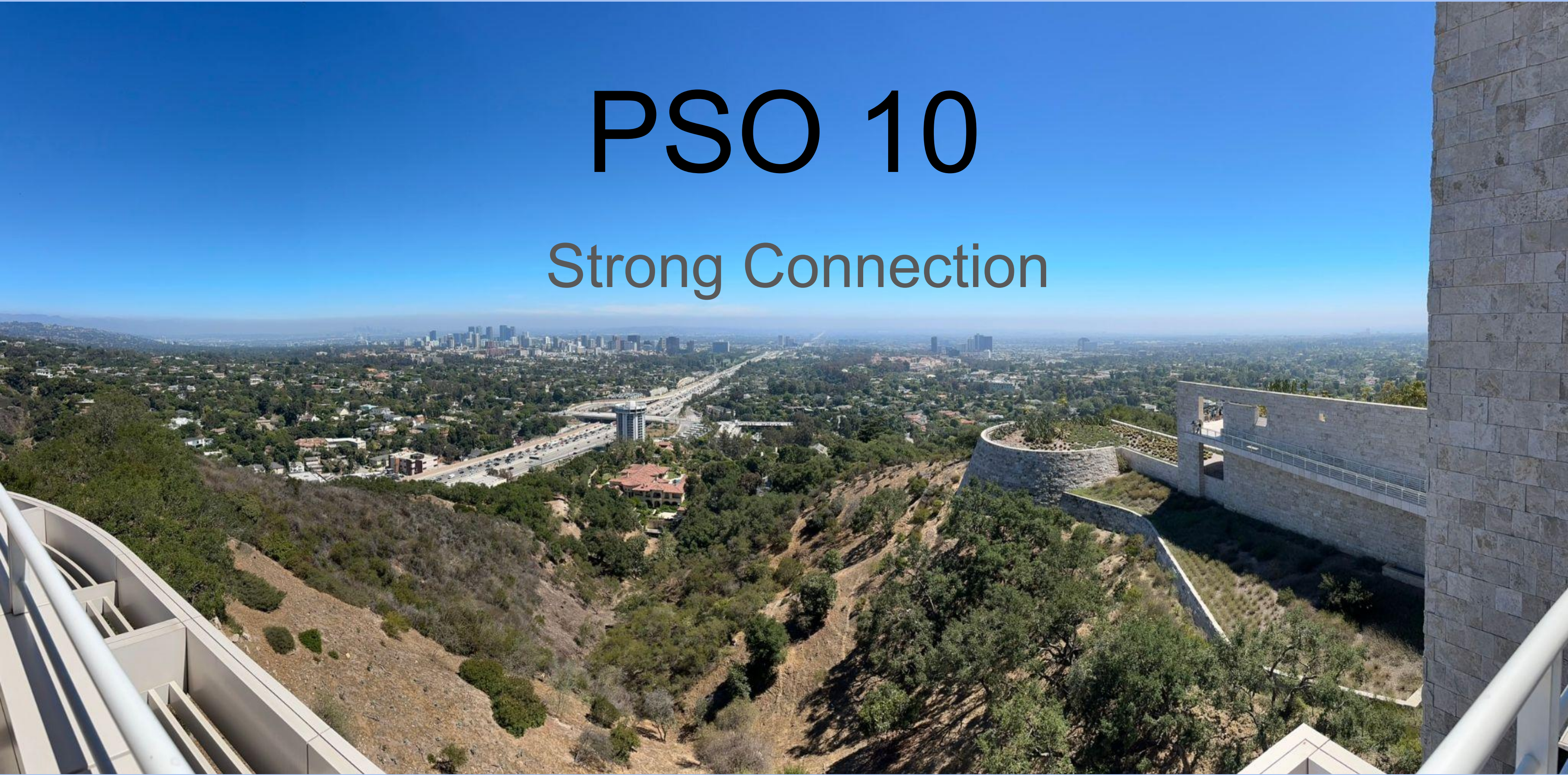
PSO 10

Strong Connection

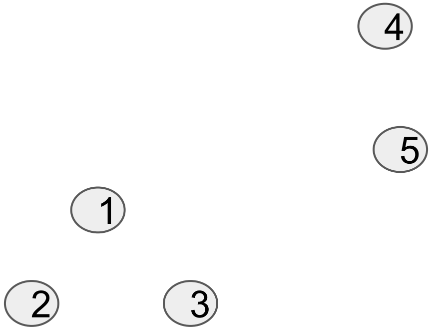


Question 1

(Strongly connected components)

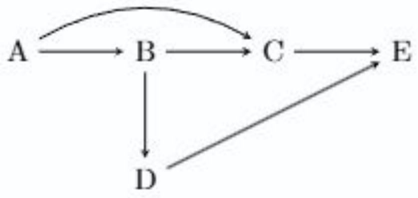
1. How can the number of strongly connected components of a graph change if a new edge is added?
2. **(Euler tour)** An Euler tour of a strongly connected, directed graph $G = (V, E)$ is a cycle that traverses each edge of G exactly once, although it may visit a vertex more than once. Show that G has Euler tour if and only if

$$\text{in-degree}(v) = \text{out-degree}(v), \forall v \in V.$$



Question 2

Consider the directed graph $G = (V, E)$ given below:



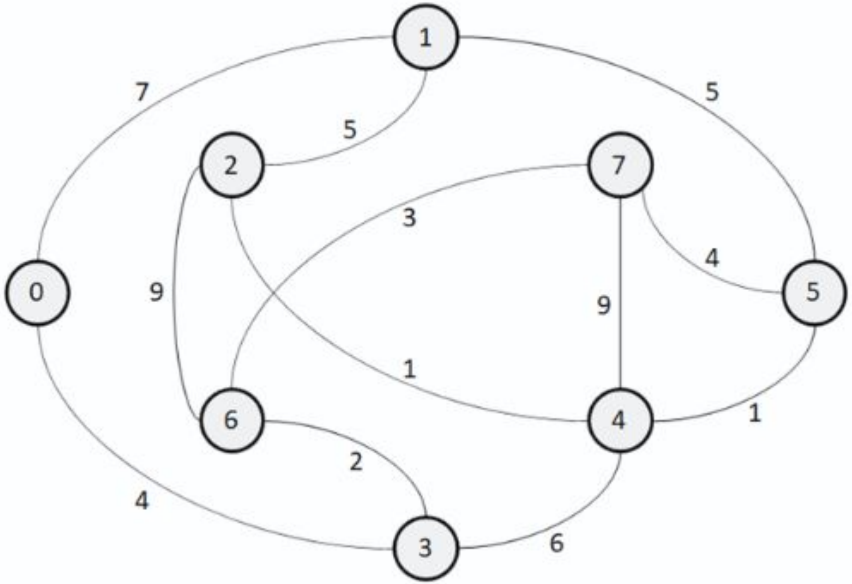
where the set of vertices is $V = \{A, B, C, D, E\}$ and the set of edges is:

$$E = \{(A, B), (A, C), (B, C), (B, D), (C, E), (D, E)\}.$$

1. Construct the adjacency matrix A of G.
2. Compute the transitive closure of G using Warshall's algorithm.
3. Draw the graph representation of the transitive closure of G.
4. Determine the reachability of each node in G.
5. Identify if G is strongly connected. If not, can you add one edge to make G become a strongly connected graph?

Question 3

Consider the following graph G:



Let G_d be a directed graph using the vertices of G. For a pair of vertices u and v connected by an edge in G, their respective directed edge in G_d is as follows:

$$\text{Edge with vertices } u \text{ and } v = \begin{cases} (u, v), & \deg(u) < \deg(v) \vee (\deg(u) = \deg(v) \wedge u < v) \\ (v, u), & \text{Otherwise} \end{cases}$$

1. Is G_d strongly connected? If yes, explain why. Otherwise, list the minimum number of edges required to make G_d strongly connected.
2. Show all the topological orderings of G_d.

Question 1

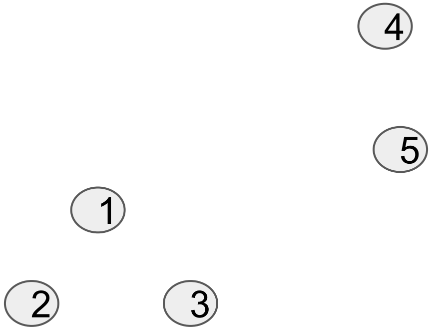
(Strongly connected components)

1. How can the number of strongly connected components of a graph change if a new edge is added?

2. **(Euler tour)** An Euler tour of a strongly connected, directed graph $G = (V, E)$ is a cycle that traverses each edge of G exactly once, although it may visit a vertex more than once. Show that G has Euler tour if and only if

$$\text{in-degree}(v) = \text{out-degree}(v), \forall v \in V.$$

Strongly connected component?



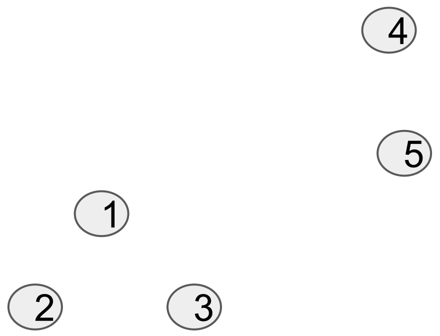
Question 1

(Strongly connected components)

1. How can the number of strongly connected components of a graph change if a new edge is added?

Can either increase/decrease/stay the same.

Can it **increase?**



Question 1

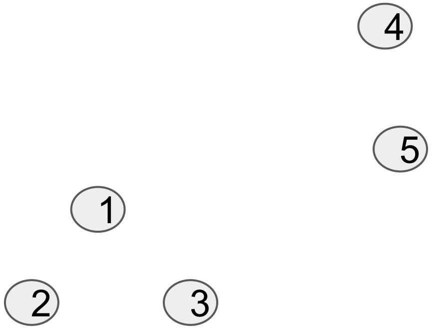
(Strongly connected components)

1. How can the number of strongly connected components of a graph change if a new edge is added?

Can either increase/decrease/stay the same.

Can it **increase? No**

Can it **decrease?**



Question 1

(Strongly connected components)

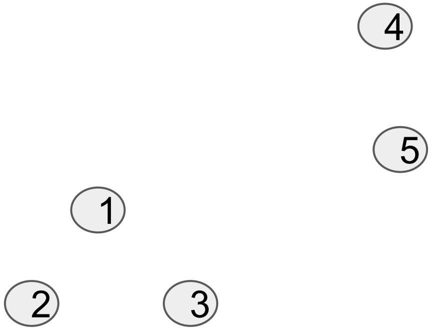
1. How can the number of strongly connected components of a graph change if a new edge is added?

Can either increase/decrease/stay the same.

Can it **increase? No**

Can it **decrease? Yes**

Can it **stay the same**?



Question 1

(Strongly connected components)

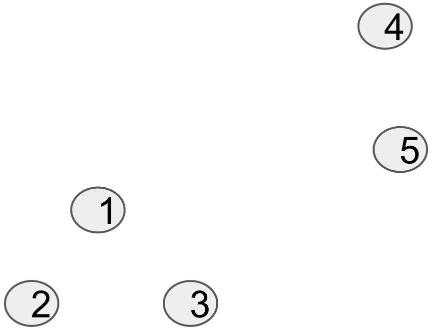
1. How can the number of strongly connected components of a graph change if a new edge is added?

Can either increase/decrease/stay the same.

Can it **increase? No**

Can it **decrease? Yes**

Can it **stay the same**? **Yes**



2. **(Euler tour)** An Euler tour of a strongly connected, directed graph $G = (V, E)$ is a cycle that traverses each edge of G exactly once, although it may visit a vertex more than once. Show that G has Euler tour if and only if

$$\text{in-degree}(v) = \text{out-degree}(v), \forall v \in V.$$

Oh boy, lets start with an informal proof

2. **(Euler tour)** An Euler tour of a strongly connected, directed graph $G = (V, E)$ is a cycle that traverses each edge of G exactly once, although it may visit a vertex more than once. Show that G has Euler tour if and only if

$$\text{in-degree}(v) = \text{out-degree}(v), \forall v \in V.$$

(→) Suppose G has an Euler tour.

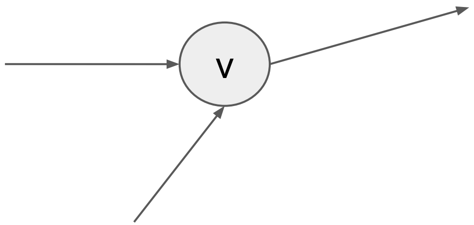
We want to show every vertex v has indeg(v) = outdeg(v).

2. **(Euler tour)** An Euler tour of a strongly connected, directed graph $G = (V, E)$ is a cycle that traverses each edge of G exactly once, although it may visit a vertex more than once. Show that G has Euler tour if and only if

$$\text{in-degree}(v) = \text{out-degree}(v), \forall v \in V.$$

(→) Suppose G has an Euler tour.

We want to show every vertex v has indeg(v) = outdeg(v).



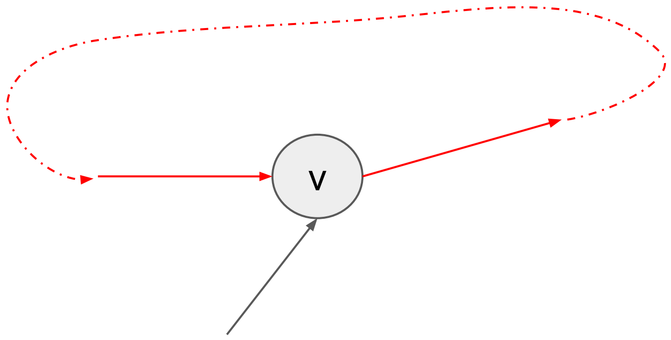
Suppose not, that there is a vertex v with indeg(v) > outdeg(v).

2. **(Euler tour)** An Euler tour of a strongly connected, directed graph $G = (V, E)$ is a cycle that traverses each edge of G exactly once, although it may visit a vertex more than once. Show that G has Euler tour if and only if

$$\text{in-degree}(v) = \text{out-degree}(v), \forall v \in V.$$

(→) Suppose G has an Euler tour.

We want to show every vertex v has indeg(v) = outdeg(v).



Suppose not, that there is a vertex v with indeg(v) > outdeg(v).

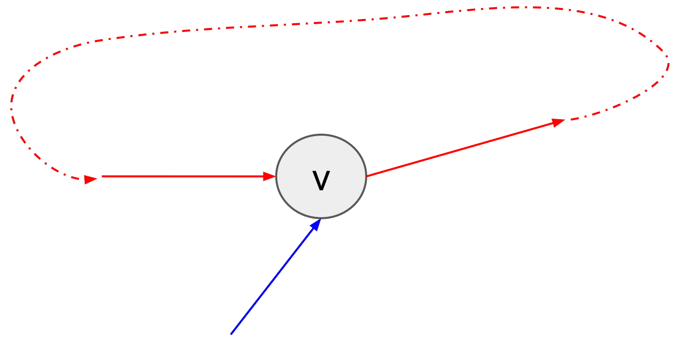
An Euler tour is a cycle i.e. each incoming edge is “paired” with an outgoing edge

2. **(Euler tour)** An Euler tour of a strongly connected, directed graph $G = (V, E)$ is a cycle that traverses each edge of G exactly once, although it may visit a vertex more than once. Show that G has Euler tour if and only if

$$\text{in-degree}(v) = \text{out-degree}(v), \forall v \in V.$$

(→) Suppose G has an Euler tour.

We want to show every vertex v has indeg(v) = outdeg(v).



Suppose not, that there is a vertex v with indeg(v) > outdeg(v).

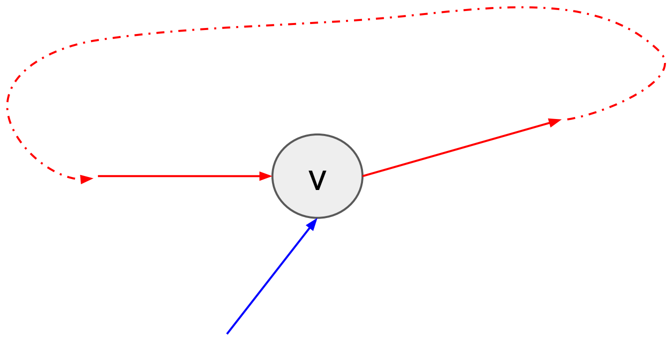
There will be an edge left over!

2. **(Euler tour)** An Euler tour of a strongly connected, directed graph $G = (V, E)$ is a cycle that traverses each edge of G exactly once, although it may visit a vertex more than once. Show that G has Euler tour if and only if

$$\text{in-degree}(v) = \text{out-degree}(v), \forall v \in V.$$

(→) Suppose G has an Euler tour.

We want to show every vertex v has indeg(v) = outdeg(v).



Suppose not, that there is a vertex v with indeg(v) > outdeg(v).

There will be an edge left over!

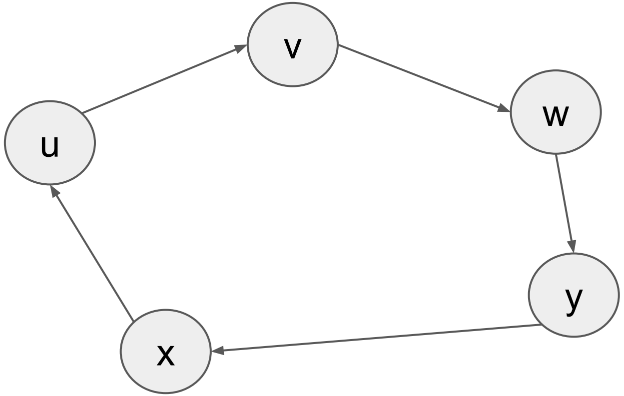
Exercise: show the same holds when indeg(v) < outdeg(v)

2. **(Euler tour)** An Euler tour of a strongly connected, directed graph $G = (V, E)$ is a cycle that traverses each edge of G exactly once, although it may visit a vertex more than once. Show that G has Euler tour if and only if

$$\text{in-degree}(v) = \text{out-degree}(v), \forall v \in V.$$

(←) Suppose indeg(v) = outdeg(v) for all vertices v.

We want to show there is an Euler tour

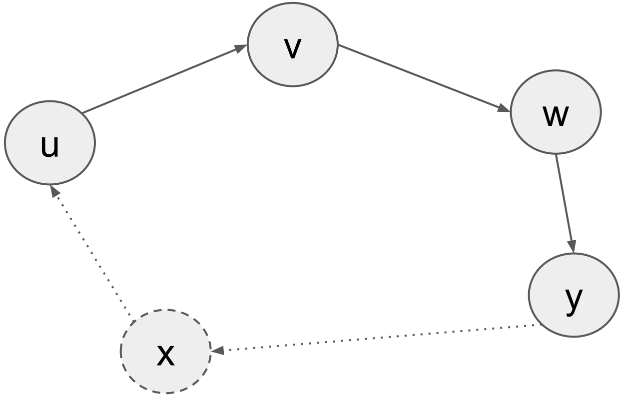


2. **(Euler tour)** An Euler tour of a strongly connected, directed graph $G = (V, E)$ is a cycle that traverses each edge of G exactly once, although it may visit a vertex more than once. Show that G has Euler tour if and only if

$$\text{in-degree}(v) = \text{out-degree}(v), \forall v \in V.$$

(←) Suppose indeg(v) = outdeg(v) for all vertices v.

We want to show there is an Euler tour



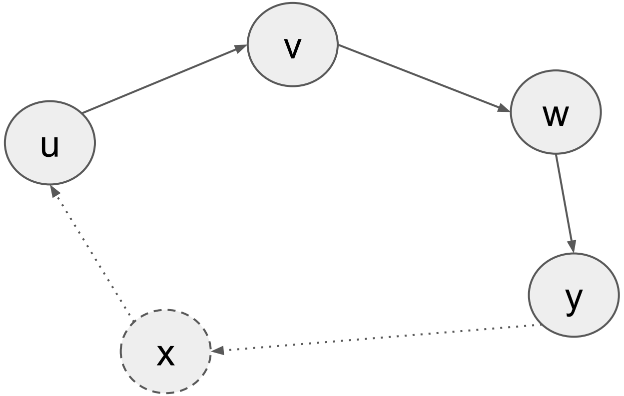
Suppose I delete a vertex (x)

2. **(Euler tour)** An Euler tour of a strongly connected, directed graph $G = (V, E)$ is a cycle that traverses each edge of G exactly once, although it may visit a vertex more than once. Show that G has Euler tour if and only if

$$\text{in-degree}(v) = \text{out-degree}(v), \forall v \in V.$$

(←) Suppose indeg(v) = outdeg(v) for all vertices v.

We want to show there is an Euler tour



Then there are vertices u,y such that:

indeg(y) = outdeg(y) + 1

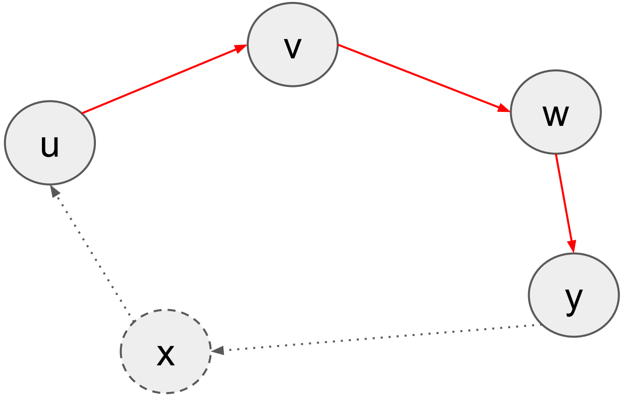
indeg(u) = outdeg(u) - 1

2. **(Euler tour)** An Euler tour of a strongly connected, directed graph $G = (V, E)$ is a cycle that traverses each edge of G exactly once, although it may visit a vertex more than once. Show that G has Euler tour if and only if

$$\text{in-degree}(v) = \text{out-degree}(v), \forall v \in V.$$

(←) Suppose indeg(v) = outdeg(v) for all vertices v.

We want to show there is an Euler tour



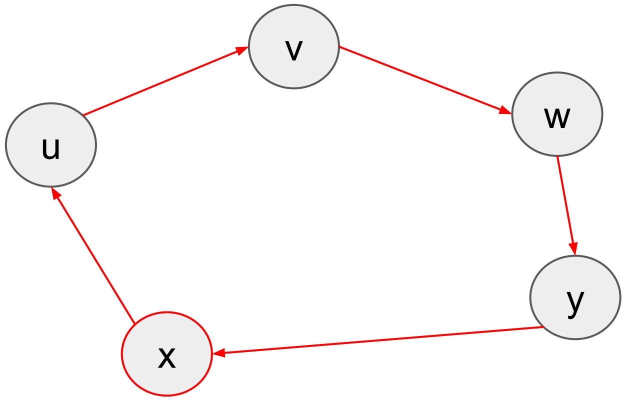
If we instead find an Euler **path** from u → y,

2. **(Euler tour)** An Euler tour of a strongly connected, directed graph $G = (V, E)$ is a cycle that traverses each edge of G exactly once, although it may visit a vertex more than once. Show that G has Euler tour if and only if

$$\text{in-degree}(v) = \text{out-degree}(v), \forall v \in V.$$

(←) Suppose indeg(v) = outdeg(v) for all vertices v.

We want to show there is an Euler tour



If we instead find an Euler **path** from u → y,

We can just add back x to get an Euler tour

2. **(Euler tour)** An Euler tour of a strongly connected, directed graph $G = (V, E)$ is a cycle that traverses each edge of G exactly once, although it may visit a vertex more than once. Show that G has Euler tour if and only if

$$\text{in-degree}(v) = \text{out-degree}(v), \forall v \in V.$$

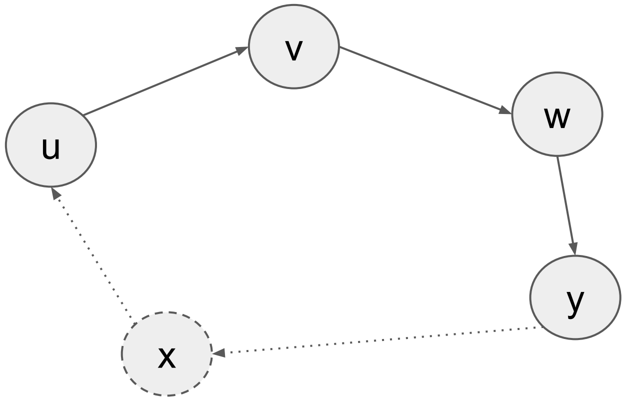
(←) Suppose indeg(v) = outdeg(v) for all vertices v.

We want to show there is an Euler tour

Then there are vertices u,y such that:

indeg(y) = outdeg(y) + 1

indeg(u) = outdeg(u) - 1



So let’s find an Euler path in this graph

2. **(Euler tour)** An Euler tour of a strongly connected, directed graph $G = (V, E)$ is a cycle that traverses each edge of G exactly once, although it may visit a vertex more than once. Show that G has Euler tour if and only if

$$\text{in-degree}(v) = \text{out-degree}(v), \forall v \in V.$$

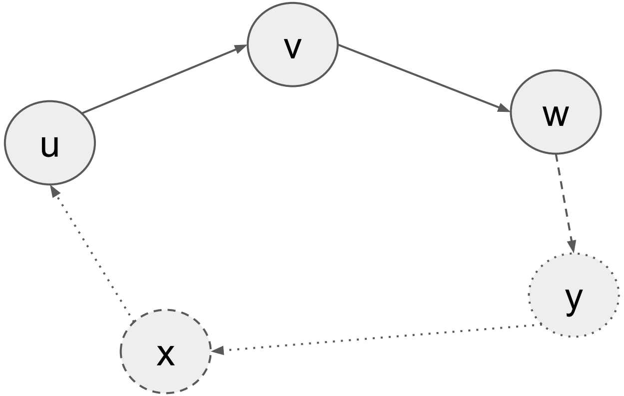
(←) Suppose indeg(v) = outdeg(v) for all vertices v.

Then there are vertices u,y such that:

indeg(y) = outdeg(y) + 1

indeg(u) = outdeg(u) - 1

We want to show there is an Euler tour



Suppose I delete y

2. **(Euler tour)** An Euler tour of a strongly connected, directed graph $G = (V, E)$ is a cycle that traverses each edge of G exactly once, although it may visit a vertex more than once. Show that G has Euler tour if and only if

$$\text{in-degree}(v) = \text{out-degree}(v), \forall v \in V.$$

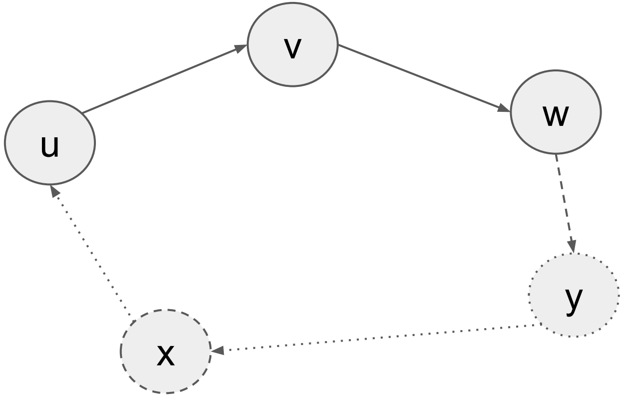
(←) Suppose indeg(v) = outdeg(v) for all vertices v.

We want to show there is an Euler tour

Then there are vertices u,y such that:

indeg(y) = outdeg(y) + 1

indeg(u) = outdeg(u) - 1



Then there are vertices u,w such that:

indeg(w) = outdeg(w) + 1

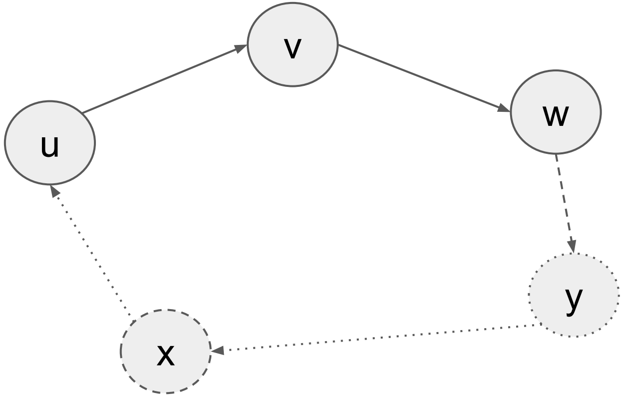
indeg(u) = outdeg(u) - 1

2. **(Euler tour)** An Euler tour of a strongly connected, directed graph $G = (V, E)$ is a cycle that traverses each edge of G exactly once, although it may visit a vertex more than once. Show that G has Euler tour if and only if

$$\text{in-degree}(v) = \text{out-degree}(v), \forall v \in V.$$

(←) Suppose indeg(v) = outdeg(v) for all vertices v.

We want to show there is an Euler tour



Then there are vertices u,y such that:

indeg(y) = outdeg(y) + 1

indeg(u) = outdeg(u) - 1

Then there are vertices u,w such that:

indeg(w) = outdeg(w) + 1

indeg(u) = outdeg(u) - 1

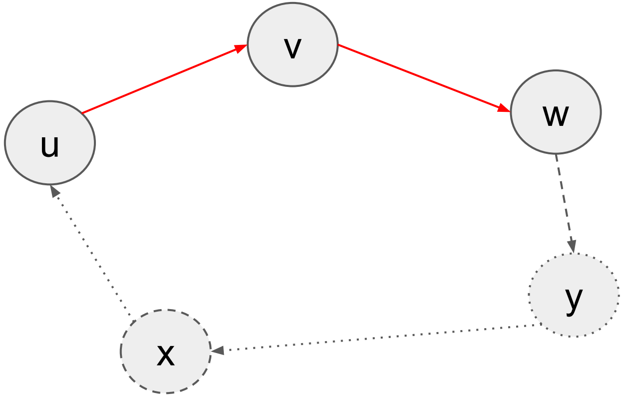
This new graph (deleted y) shares the same structure as the previous graph.. We can induct on the number of edges!

2. **(Euler tour)** An Euler tour of a strongly connected, directed graph $G = (V, E)$ is a cycle that traverses each edge of G exactly once, although it may visit a vertex more than once. Show that G has Euler tour if and only if

$$\text{in-degree}(v) = \text{out-degree}(v), \forall v \in V.$$

(←) Suppose indeg(v) = outdeg(v) for all vertices v.

We want to show there is an Euler tour



Then there are vertices u,y such that:

indeg(y) = outdeg(y) + 1

indeg(u) = outdeg(u) - 1

Then there are vertices u,w such that:

indeg(w) = outdeg(w) + 1

indeg(u) = outdeg(u) - 1

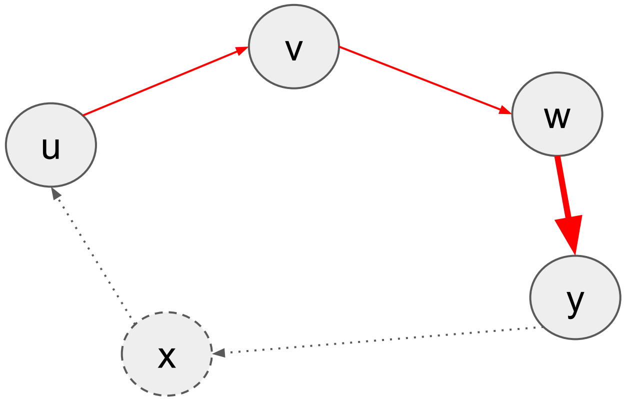
By Induction there is an Euler path from u → w

2. **(Euler tour)** An Euler tour of a strongly connected, directed graph $G = (V, E)$ is a cycle that traverses each edge of G exactly once, although it may visit a vertex more than once. Show that G has Euler tour if and only if

$$\text{in-degree}(v) = \text{out-degree}(v), \forall v \in V.$$

(←) Suppose indeg(v) = outdeg(v) for all vertices v.

We want to show there is an Euler tour



Then there are vertices u,y such that:

indeg(y) = outdeg(y) + 1

indeg(u) = outdeg(u) - 1

Then there are vertices u,w such that:

indeg(w) = outdeg(w) + 1

indeg(u) = outdeg(u) - 1

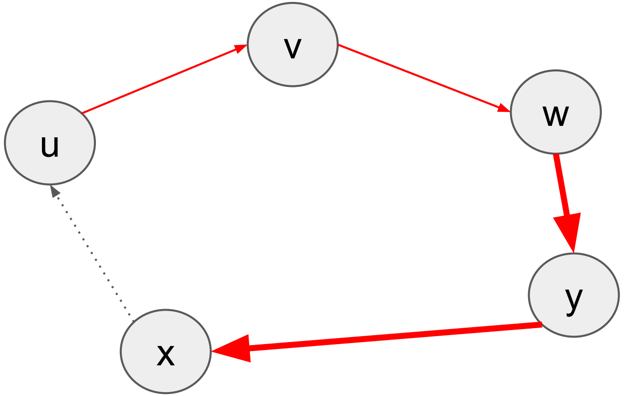
Add back y

2. **(Euler tour)** An Euler tour of a strongly connected, directed graph $G = (V, E)$ is a cycle that traverses each edge of G exactly once, although it may visit a vertex more than once. Show that G has Euler tour if and only if

$$\text{in-degree}(v) = \text{out-degree}(v), \forall v \in V.$$

(←) Suppose indeg(v) = outdeg(v) for all vertices v.

We want to show there is an Euler tour



Then there are vertices u,y such that:

indeg(y) = outdeg(y) + 1

indeg(u) = outdeg(u) - 1

Then there are vertices u,w such that:

indeg(w) = outdeg(w) + 1

indeg(u) = outdeg(u) - 1

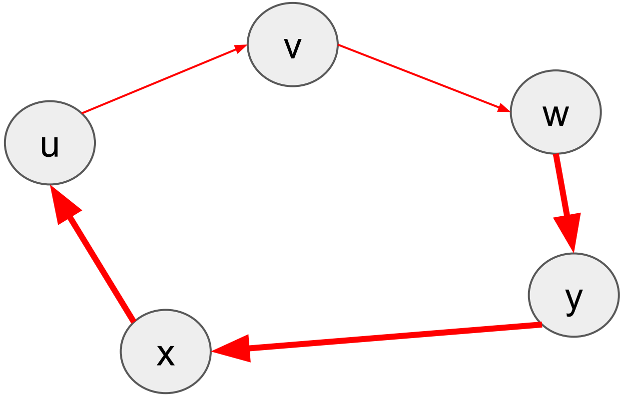
Add back x

2. **(Euler tour)** An Euler tour of a strongly connected, directed graph $G = (V, E)$ is a cycle that traverses each edge of G exactly once, although it may visit a vertex more than once. Show that G has Euler tour if and only if

$$\text{in-degree}(v) = \text{out-degree}(v), \forall v \in V.$$

(←) Suppose indeg(v) = outdeg(v) for all vertices v.

We want to show there is an Euler tour



Then there are vertices u,y such that:

indeg(y) = outdeg(y) + 1

indeg(u) = outdeg(u) - 1

Then there are vertices u,w such that:

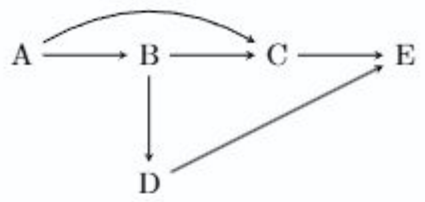
indeg(w) = outdeg(w) + 1

indeg(u) = outdeg(u) - 1

Complete the tour!

Question 2

Consider the directed graph $G = (V, E)$ given below:

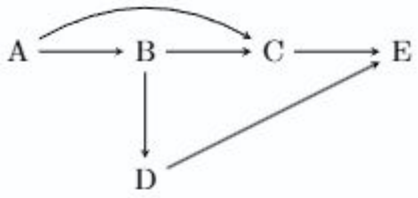


1. Construct the adjacency matrix A of G.

u/v	A	B	C	D	E
A	0				
B		0			
C			0		
D				0	
E					0

Question 2

Consider the directed graph $G = (V, E)$ given below:



2. Compute the transitive closure of G using Warshall's algorithm.

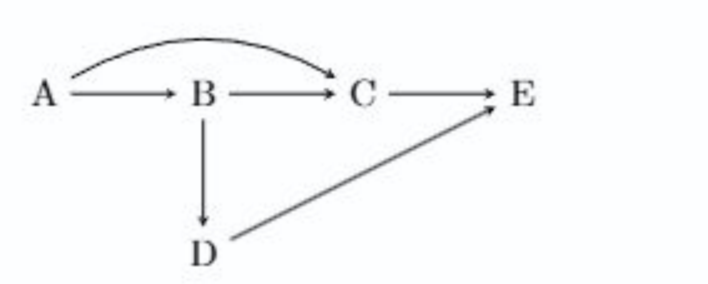
What's your algorithm Warshall/Floyd/Ingerman/Roy/Kleene?

Worst-case space complexity	$\Theta(\lvert V\rvert^2)$

History and naming [edit]

The Floyd–Warshall algorithm is an example of dynamic programming, and was published in its currently recognized form by Robert Floyd in 1962.[3] However, it is essentially the same as algorithms previously published by Bernard Roy in 1959[4] and also by Stephen Warshall in 1962[5] for finding the transitive closure of a graph,[6] and is closely related to Kleene's algorithm (published in 1956) for converting a deterministic finite automaton into a regular expression, with the difference being the use of a min-plus semiring.[7] The modern formulation of the algorithm as three nested for-loops was first described by Peter Ingerman, also in 1962.[8]

```
algorithm Floyd-Warshall(M:adjacency matrix representing G(V,E))

    R^(-1) ← M
    n ← |V|

    for k from 0 to n − 1 do
        for i from 0 to n − 1 do
            for j from 0 to n − 1 do
                R^(k)[i,j] ← R^(k−1)[i,j] or (R^(k−1)[i,k] and R^(k−1)[k,j])
            end for
        end for
    end for

    return R^(n−1)
end algorithm
```



$R^{(-1)} \leftarrow M$
$n \leftarrow |V|$

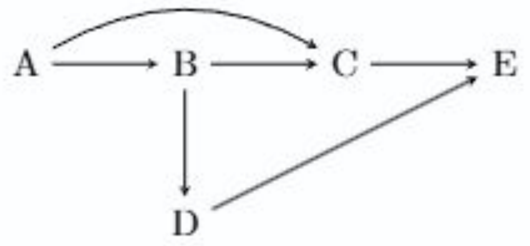
for k from 0 to $n-1$ **do**
  for i from 0 to $n-1$ **do**
    for j from 0 to $n-1$ **do**
      $R^{(k)}[i,j] \leftarrow R^{(k-1)}[i,j]$ or $(R^{(k-1)}[i,k]$ and $R^{(k-1)}[k,j])$
    end for
  end for
end for

return $R^{(n-1)}$
end algorithm

$R^{(-1)}$	A	B	C	D	E
A	0	1	1	0	0
B	0	0	1	1	0
C	0	0	0	0	1
D	0	0	0	0	1
E	0	0	0	0	0

$\xrightarrow{k = A}$

$R^{(0)}$	A	B	C	D	E
A					
B					
C					
D					
E					



$R^{(-1)} \leftarrow M$
$n \leftarrow |V|$

for k **from** 0 **to** $n-1$ **do**
  for i **from** 0 **to** $n-1$ **do**
    for j **from** 0 **to** $n-1$ **do**
      $R^{(k)}[i,j] \leftarrow R^{(k-1)}[i,j]$ **or** $(R^{(k-1)}[i,k]$ **and** $R^{(k-1)}[k,j])$
    end for
  end for
end for

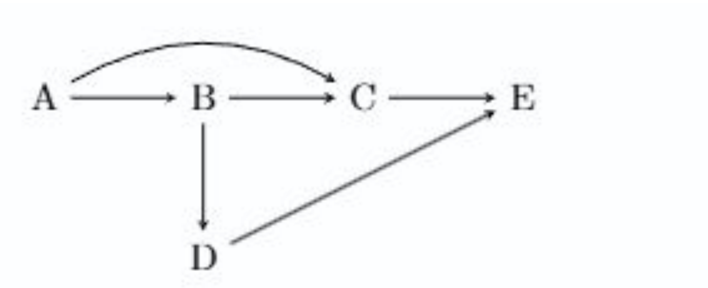
return $R^{(n-1)}$
end algorithm

$R^{(-1)}$	A	B	C	D	E
A	0	1	1	0	0
B	0	0	1	1	0
C	0	0	0	0	1
D	0	0	0	0	1
E	0	0	0	0	0

k = A

i = A
j = A

$R^{(0)}$	A	B	C	D	E
A					
B					
C					
D					
E					

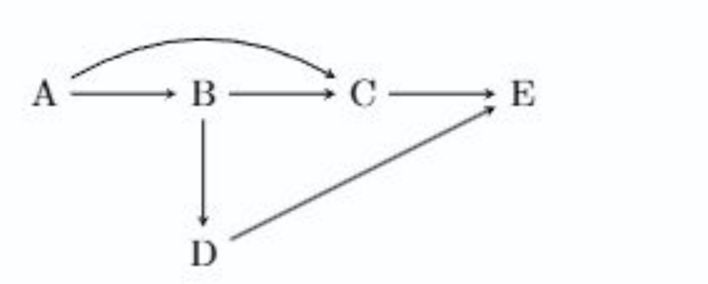



```
R^(-1) ← M
n ← |V|

for k from 0 to n−1 do
   for i from 0 to n−1 do
      for j from 0 to n−1 do
         R^(k)[i,j] ← R^(k−1)[i,j] or (R^(k−1)[i,k] and R^(k−1)[k,j])
      end for
   end for
end for

return R^(n−1)
end algorithm
```

$R^{(-1)}$	A	B	C	D	E
A	0	1	1	0	0
B	0	0	1	1	0
C	0	0	0	0	1
D	0	0	0	0	1
E	0	0	0	0	0

$$\xrightarrow[\substack{i = A \\ j = A}]{k = A}$$

$R^{(0)}$	A	B	C	D	E
A	0				
B					
C					
D					
E					



$R^{(-1)} \leftarrow M$
$n \leftarrow |V|$

for k from 0 to $n-1$ **do**
    for i from 0 to $n-1$ **do**
        for j from 0 to $n-1$ **do**
            $R^{(k)}[i,j] \leftarrow R^{(k-1)}[i,j]$ or $(R^{(k-1)}[i,k]$ and $R^{(k-1)}[k,j])$
        end for
    end for
end for

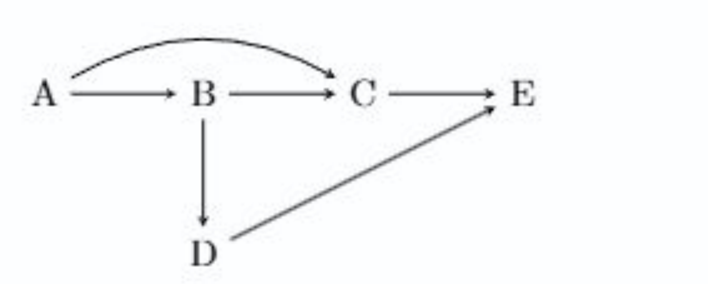
return $R^{(n-1)}$
end algorithm

$R^{(-1)}$	A	B	C	D	E
A	0	1	1	0	0
B	0	0	1	1	0
C	0	0	0	0	1
D	0	0	0	0	1
E	0	0	0	0	0

k = A

i = A
j = B

$R^{(0)}$	A	B	C	D	E
A	0	1			
B					
C					
D					
E					



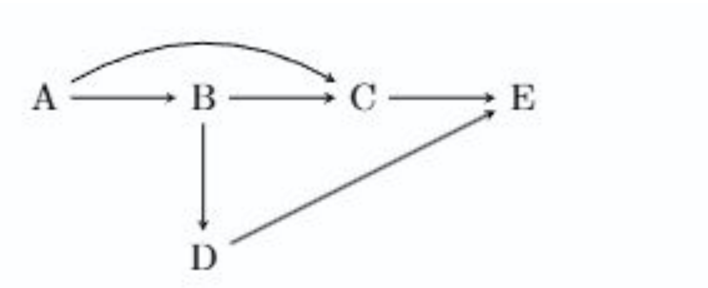

```
R^(−1) ← M
n ← |V|

for k from 0 to n − 1 do
   for i from 0 to n − 1 do
      for j from 0 to n − 1 do
         R^(k)[i,j] ← R^(k−1)[i,j] or (R^(k−1)[i,k] and R^(k−1)[k,j])
      end for
   end for
end for

return R^(n−1)
end algorithm
```

$R^{(-1)}$	A	B	C	D	E
A	0	1	1	0	0
B	0	0	1	1	0
C	0	0	0	0	1
D	0	0	0	0	1
E	0	0	0	0	0

$\xrightarrow{k = A}$

i = A
j = C

$R^{(0)}$	A	B	C	D	E
A	0	1			
B					
C					
D					
E					



$R^{(-1)} \leftarrow M$
$n \leftarrow |V|$

for k from 0 to $n-1$ **do**
  for i from 0 to $n-1$ **do**
    for j from 0 to $n-1$ **do**
      $R^{(k)}[i,j] \leftarrow R^{(k-1)}[i,j]$ or $(R^{(k-1)}[i,k]$ and $R^{(k-1)}[k,j])$
    end for
  end for
end for

return $R^{(n-1)}$
end algorithm

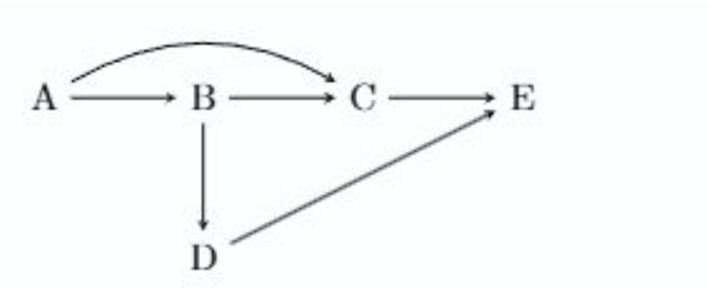
$R^{(-1)}$	A	B	C	D	E
A	0	1	1	0	0
B	0	0	1	1	0
C	0	0	0	0	1
D	0	0	0	0	1
E	0	0	0	0	0

$k = A$

$\longrightarrow$

$i = A$
$j = C$

$R^{(0)}$	A	B	C	D	E
A	0	1	1		
B					
C					
D					
E					



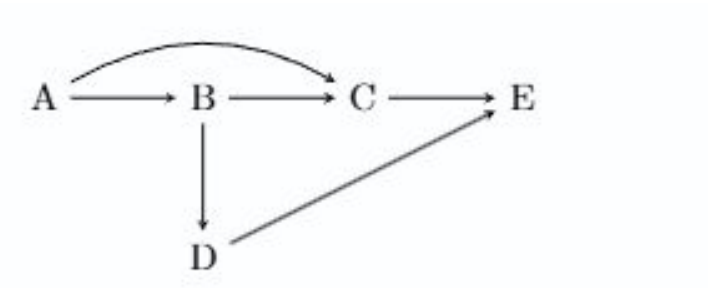

```
    R^(-1) ← M
    n ← |V|

    for k from 0 to n − 1 do
       for i from 0 to n − 1 do
          for j from 0 to n − 1 do
             R^(k)[i,j] ← R^(k−1)[i,j] or (R^(k−1)[i,k] and R^(k−1)[k,j])
          end for
       end for
    end for

    return R^(n−1)
end algorithm
```

$R^{(-1)}$	A	B	C	D	E
A	0	1	1	0	0
B	0	0	1	1	0
C	0	0	0	0	1
D	0	0	0	0	1
E	0	0	0	0	0

$\xrightarrow{k = A}$

i = A
j = D

$R^{(0)}$	A	B	C	D	E
A	0	1	1	0	
B					
C					
D					
E					



$R^{(-1)} \leftarrow M$
$n \leftarrow |V|$

for k from 0 to $n-1$ **do**
  for i from 0 to $n-1$ **do**
    for j from 0 to $n-1$ **do**
      $R^{(k)}[i,j] \leftarrow R^{(k-1)}[i,j]$ or $(R^{(k-1)}[i,k]$ and $R^{(k-1)}[k,j])$
    end for
  end for
end for

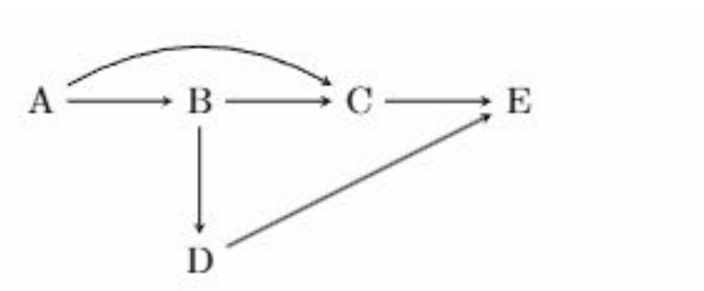
return $R^{(n-1)}$
end algorithm

$R^{(-1)}$	A	B	C	D	E
A	0	1	1	0	0
B	0	0	1	1	0
C	0	0	0	0	1
D	0	0	0	0	1
E	0	0	0	0	0

k = A

i = A
j = E

$R^{(0)}$	A	B	C	D	E
A	0	1	1	0	0
B					
C					
D					
E					




```
R^(-1) ← M
n ← |V|

for k from 0 to n−1 do
   for i from 0 to n−1 do
      for j from 0 to n−1 do
         R^(k)[i,j] ← R^(k−1)[i,j] or (R^(k−1)[i,k] and R^(k−1)[k,j])
      end for
   end for
end for

return R^(n−1)
end algorithm
```

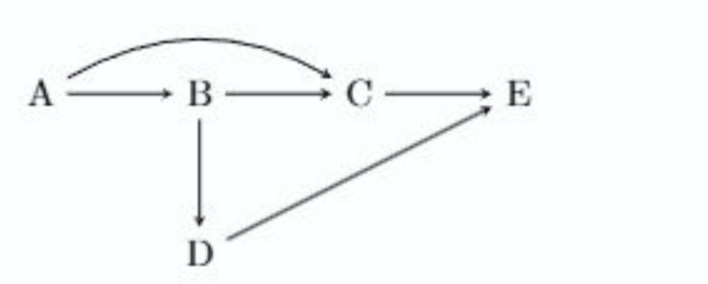
$R^{(-1)}$	A	B	C	D	E
A	0	1	1	0	0
B	0	0	1	1	0
C	0	0	0	0	1
D	0	0	0	0	1
E	0	0	0	0	0

What's the final $R^{(0)}$?

$$\xrightarrow{k = A}$$

i = B
j = A

$R^{(0)}$	A	B	C	D	E
A	0	1	1	0	0
B	0				
C					
D					
E					



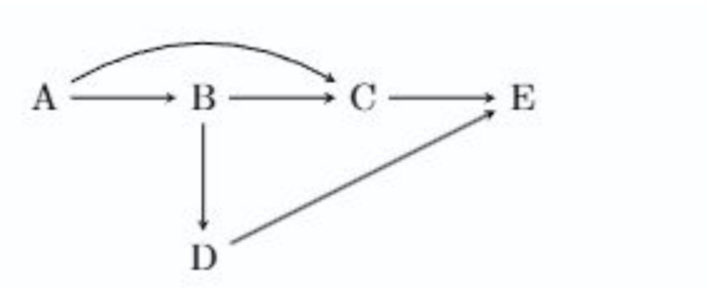

```
R^(-1) ← M
n ← |V|

for k from 0 to n−1 do
   for i from 0 to n−1 do
      for j from 0 to n−1 do
         R^(k)[i,j] ← R^(k−1)[i,j] or (R^(k−1)[i,k] and R^(k−1)[k,j])
      end for
   end for
end for

return R^(n−1)
end algorithm
```

$R^{(-1)}$	A	B	C	D	E
A	0	1	1	0	0
B	0	0	1	1	0
C	0	0	0	0	1
D	0	0	0	0	1
E	0	0	0	0	0

Whats the final $R^{(0)}$? Same as $R^{(-1)}$, why?

$$\xrightarrow[i = B,\ j = A]{k = A}$$

$R^{(0)}$	A	B	C	D	E
A	0	1	1	0	0
B	0	0	1	1	0
C	0	0	0	0	1
D	0	0	0	0	1
E	0	0	0	0	0



$R^{(-1)} \leftarrow M$
$n \leftarrow |V|$

for k from 0 to $n-1$ **do**
  for i from 0 to $n-1$ **do**
    for j from 0 to $n-1$ **do**
      $R^{(k)}[i,j] \leftarrow R^{(k-1)}[i,j]$ or $(R^{(k-1)}[i,k]$ and $R^{(k-1)}[k,j])$
    end for
  end for
end for

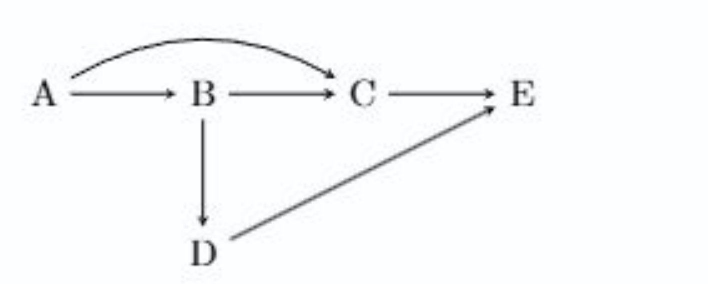
return $R^{(n-1)}$
end algorithm

$R^{(0)}$	A	B	C	D	E
A	0	1	1	0	0
B	0	0	1	1	0
C	0	0	0	0	1
D	0	0	0	0	1
E	0	0	0	0	0

k = B

i = A
j = A

$R^{(1)}$	A	B	C	D	E
A	0	1	1		
B			1	1	
C					1
D					1
E					



$R^{(-1)} \leftarrow M$
$n \leftarrow |V|$

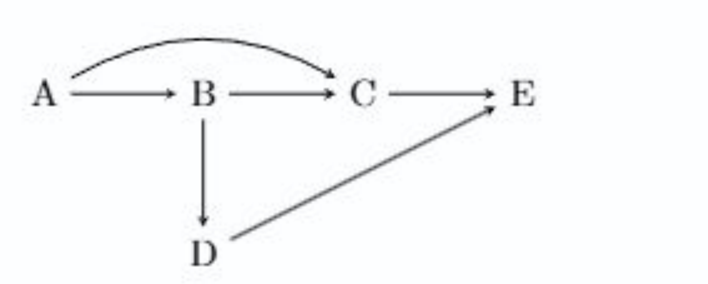
for k from 0 to $n-1$ **do**
  for i from 0 to $n-1$ **do**
    for j from 0 to $n-1$ **do**
      $R^{(k)}[i,j] \leftarrow R^{(k-1)}[i,j]$ or $(R^{(k-1)}[i,k]$ and $R^{(k-1)}[k,j])$
    end for
  end for
end for

return $R^{(n-1)}$
end algorithm

$R^{(0)}$	A	B	C	D	E
A	0	1	1	0	0
B	0	0	1	1	0
C	0	0	0	0	1
D	0	0	0	0	1
E	0	0	0	0	0

$$\xrightarrow[i = A,\; j = B]{k = B}$$

$R^{(1)}$	A	B	C	D	E
A	0	1	1		
B			1	1	
C					1
D					1
E					



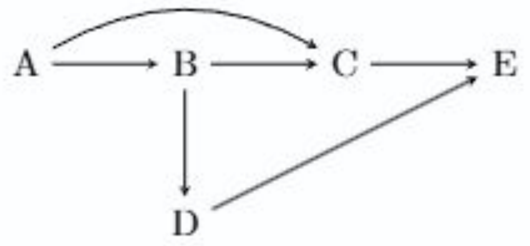

```
    R^(-1) ← M
    n ← |V|

    for k from 0 to n−1 do
       for i from 0 to n−1 do
          for j from 0 to n−1 do
             R^(k)[i,j] ← R^(k−1)[i,j] or (R^(k−1)[i,k] and R^(k−1)[k,j])
          end for
       end for
    end for

    return R^(n−1)
end algorithm
```

$R^{(0)}$	A	B	C	D	E
A	0	1	1	0	0
B	0	0	1	1	0
C	0	0	0	0	1
D	0	0	0	0	1
E	0	0	0	0	0

k = B

i = A
j = C

$R^{(1)}$	A	B	C	D	E
A	0	1	1		
B			1	1	
C					1
D					1
E					

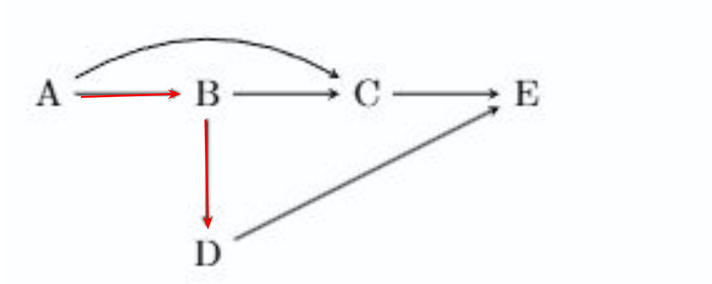



```
R^(-1) ← M
n ← |V|

for k from 0 to n−1 do
   for i from 0 to n−1 do
      for j from 0 to n−1 do
         R^(k)[i,j] ← R^(k−1)[i,j] or (R^(k−1)[i,k] and R^(k−1)[k,j])
      end for
   end for
end for

return R^(n−1)
end algorithm
```

$R^{(0)}$	A	B	C	D	E
A	0	1	1	0	0
B	0	0	1	1	0
C	0	0	0	0	1
D	0	0	0	0	1
E	0	0	0	0	0

$$\xrightarrow[\substack{i = A \\ j = D}]{k = B}$$

$R^{(1)}$	A	B	C	D	E
A	0	1	1		
B			1	1	
C					1
D					1
E					



$R^{(-1)} \leftarrow M$
$n \leftarrow |V|$

for k from 0 to $n-1$ **do**
  for i from 0 to $n-1$ **do**
    for j from 0 to $n-1$ **do**
      $R^{(k)}[i,j] \leftarrow R^{(k-1)}[i,j]$ or $(R^{(k-1)}[i,k]$ and $R^{(k-1)}[k,j])$
    end for
  end for
end for

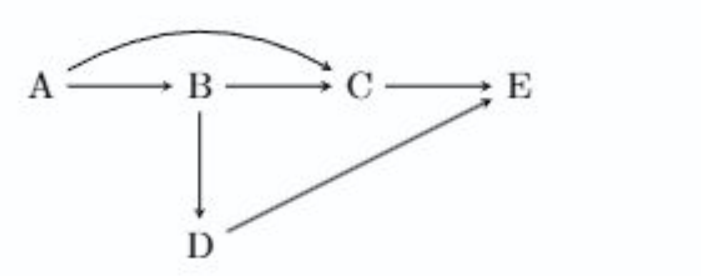
return $R^{(n-1)}$
end algorithm

$R^{(0)}$	A	B	C	D	E
A	0	1	1	0	0
B	0	0	1	1	0
C	0	0	0	0	1
D	0	0	0	0	1
E	0	0	0	0	0

k = B

i = A
j = D

$R^{(1)}$	A	B	C	D	E
A	0	1	1	1	
B			1	1	
C					1
D					1
E					



$R^{(-1)} \leftarrow M$
$n \leftarrow |V|$

for k **from** 0 **to** $n-1$ **do**
  for i **from** 0 **to** $n-1$ **do**
    for j **from** 0 **to** $n-1$ **do**
      $R^{(k)}[i,j] \leftarrow R^{(k-1)}[i,j]$ **or** $(R^{(k-1)}[i,k]$ **and** $R^{(k-1)}[k,j])$
    end for
  end for
end for

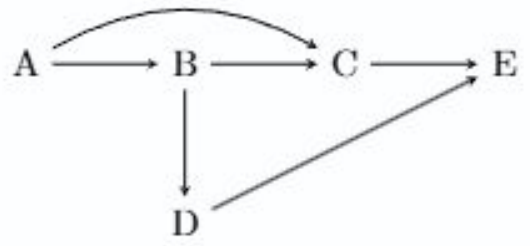
return $R^{(n-1)}$
end algorithm

$R^{(0)}$	A	B	C	D	E
A	0	1	1	0	0
B	0	0	1	1	0
C	0	0	0	0	1
D	0	0	0	0	1
E	0	0	0	0	0

$\xrightarrow{k = B}$

i = A
j = E

$R^{(1)}$	A	B	C	D	E
A	0	1	1	1	?
B			1	1	
C					1
D					1
E					



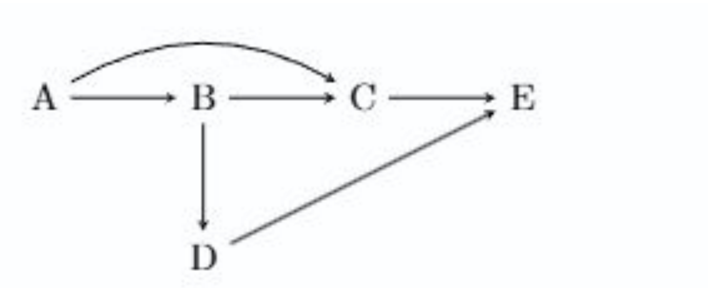

```
R^(-1) ← M
n ← |V|

for k from 0 to n−1 do
   for i from 0 to n−1 do
      for j from 0 to n−1 do
         R^(k)[i,j] ← R^(k−1)[i,j] or (R^(k−1)[i,k] and R^(k−1)[k,j])
      end for
   end for
end for

return R^(n−1)
end algorithm
```

$R^{(0)}$	A	B	C	D	E
A	0	1	1	0	0
B	0	0	1	1	0
C	0	0	0	0	1
D	0	0	0	0	1
E	0	0	0	0	0

$\xrightarrow{k = B}$

i = A
j = E

$R^{(1)}$	A	B	C	D	E
A	0	1	1	1	0
B			1	1	
C					1
D					1
E					



$R^{(-1)} \leftarrow M$

$n \leftarrow |V|$

for k from 0 to $n-1$ **do**
  for i from 0 to $n-1$ **do**
    for j from 0 to $n-1$ **do**
      $R^{(k)}[i,j] \leftarrow R^{(k-1)}[i,j]$ or $(R^{(k-1)}[i,k]$ and $R^{(k-1)}[k,j])$
    end for
  end for
end for

return $R^{(n-1)}$
end algorithm

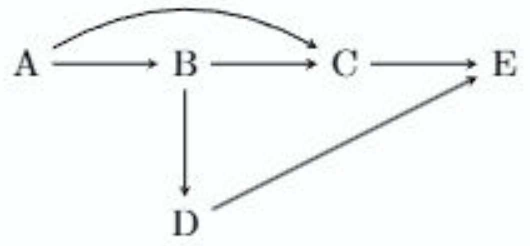
$R^{(0)}$	A	B	C	D	E
A	0	1	1	0	0
B	0	0	1	1	0
C	0	0	0	0	1
D	0	0	0	0	1
E	0	0	0	0	0

What's the final $R^{(1)}$?

$\xrightarrow{k = B}$

$i = B$
$j = A$

$R^{(1)}$	A	B	C	D	E
A	0	1	1	1	0
B			1	1	
C					1
D					1
E					




```
R^(-1) ← M
n ← |V|

for k from 0 to n−1 do
   for i from 0 to n−1 do
      for j from 0 to n−1 do
         R^(k)[i,j] ← R^(k−1)[i,j] or (R^(k−1)[i,k] and R^(k−1)[k,j])
      end for
   end for
end for

return R^(n−1)
end algorithm
```

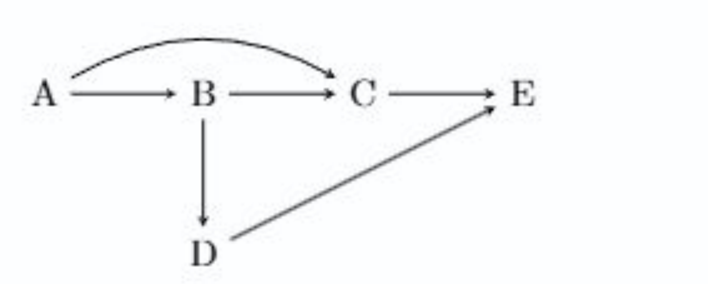
$R^{(0)}$	A	B	C	D	E
A	0	1	1	0	0
B	0	0	1	1	0
C	0	0	0	0	1
D	0	0	0	0	1
E	0	0	0	0	0

What's the final $R^{(1)}$?

k = B

i = B
j = A

$R^{(1)}$	A	B	C	D	E
A	0	1	1	1	0
B	0	0	1	1	0
C	0	0	0	0	1
D	0	0	0	0	1
E	0	0	0	0	0



$$R^{(-1)} \leftarrow M$$
$$n \leftarrow |V|$$

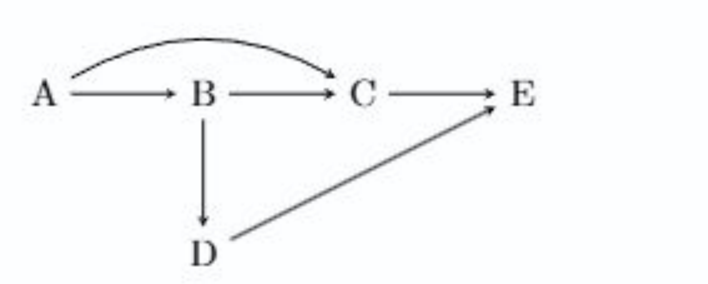
```
for k from 0 to n−1 do
    for i from 0 to n−1 do
        for j from 0 to n−1 do
            R^(k)[i,j] ← R^(k−1)[i,j] or (R^(k−1)[i,k] and R^(k−1)[k,j])
        end for
    end for
end for

return R^(n−1)
end algorithm
```

$R^{(1)}$	A	B	C	D	E
A	0	1	1	1	0
B	0	0	1	1	0
C	0	0	0	0	1
D	0	0	0	0	1
E	0	0	0	0	0

What's the final $R^{(2)}$?

k = C

$R^{(2)}$	A	B	C	D	E
A					
B					
C					
D					
E					



$R^{(-1)} \leftarrow M$
$n \leftarrow |V|$

for k **from** 0 **to** $n-1$ **do**
  for i **from** 0 **to** $n-1$ **do**
    for j **from** 0 **to** $n-1$ **do**
      $R^{(k)}[i,j] \leftarrow R^{(k-1)}[i,j]$ **or** $(R^{(k-1)}[i,k]$ **and** $R^{(k-1)}[k,j])$
    end for
  end for
end for

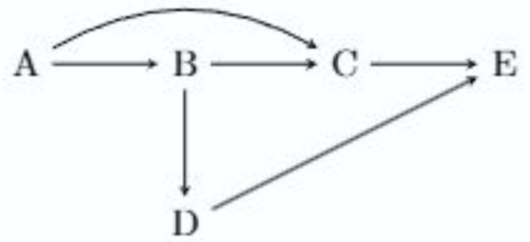
return $R^{(n-1)}$
end algorithm

$R^{(1)}$	A	B	C	D	E
A	0	1	1	1	0
B	0	0	1	1	0
C	0	0	0	0	1
D	0	0	0	0	1
E	0	0	0	0	0

What's the final $R^{(2)}$?

k = C

$R^{(2)}$	A	B	C	D	E
A	0	1	1	1	1
B	0	0	1	1	1
C	0	0	0	0	1
D	0	0	0	0	1
E	0	0	0	0	0



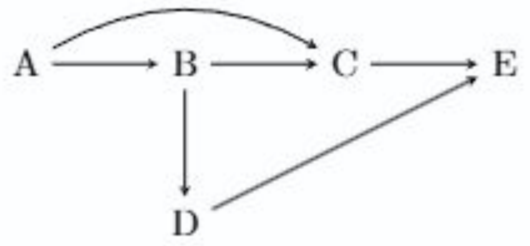

```
R^(-1) ← M
n ← |V|

for k from 0 to n−1 do
   for i from 0 to n−1 do
      for j from 0 to n−1 do
         R^(k)[i,j] ← R^(k−1)[i,j] or (R^(k−1)[i,k] and R^(k−1)[k,j])
      end for
   end for
end for

return R^(n−1)
end algorithm
```

$R^{(2)}$	A	B	C	D	E
A	0	1	1	1	1
B	0	0	1	1	1
C	0	0	0	0	1
D	0	0	0	0	1
E	0	0	0	0	0

What's the final $R^{(3)}$?

k = D

$R^{(3)}$	A	B	C	D	E
A					
B					
C					
D					
E					



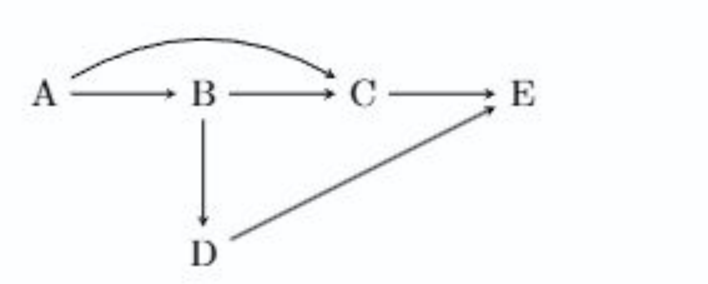

```
    R^(-1) ← M
    n ← |V|

    for k from 0 to n-1 do
        for i from 0 to n-1 do
            for j from 0 to n-1 do
                R^(k)[i,j] ← R^(k-1)[i,j] or (R^(k-1)[i,k] and R^(k-1)[k,j])
            end for
        end for
    end for

    return R^(n-1)
end algorithm
```

$R^{(2)}$	A	B	C	D	E
A	0	1	1	1	1
B	0	0	1	1	1
C	0	0	0	0	1
D	0	0	0	0	1
E	0	0	0	0	0

What's the final $R^{(3)}$?

$\xrightarrow{k = D}$

$R^{(3)}$	A	B	C	D	E
A	0	1	1	1	1
B	0	0	1	1	1
C	0	0	0	0	1
D	0	0	0	0	1
E	0	0	0	0	0



$R^{(-1)} \leftarrow M$
$n \leftarrow |V|$

for k **from** 0 **to** $n-1$ **do**
  for i **from** 0 **to** $n-1$ **do**
    for j **from** 0 **to** $n-1$ **do**
      $R^{(k)}[i,j] \leftarrow R^{(k-1)}[i,j]$ **or** $(R^{(k-1)}[i,k]$ **and** $R^{(k-1)}[k,j])$
    end for
  end for
end for

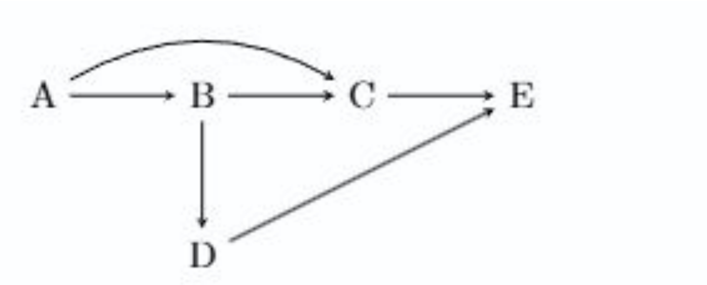
return $R^{(n-1)}$
end algorithm

$R^{(3)}$	A	B	C	D	E
A	0	1	1	1	1
B	0	0	1	1	1
C	0	0	0	0	1
D	0	0	0	0	1
E	0	0	0	0	0

What's the final $R^{(4)}$?

k = E

$R^{(4)}$	A	B	C	D	E
A	0	1	1	1	1
B	0	0	1	1	1
C	0	0	0	0	1
D	0	0	0	0	1
E	0	0	0	0	0

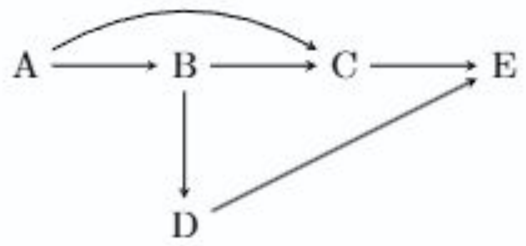


$R^{(4)}$	A	B	C	D	E
A	0	1	1	1	1
B	0	0	1	1	1
C	0	0	0	0	1
D	0	0	0	0	1
E	0	0	0	0	0

Summary

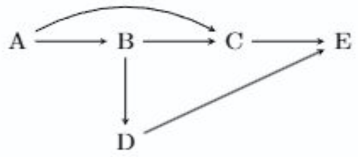
For each k = A,B,C,D,E:
    For each i = …:
        For each j = …:
            Check if there is a path between i and j through k

$R^{(-1)}$: Adj Matrix
$R^{(0/A)}$: $R^{(-1)}$ + (paths through A)
$R^{(1/B)}$: $R^{(0/A)}$+ (paths through B)
$R^{(2/C)}$: $R^{(1/B)}$ + (paths through C)
$R^{(3/D)}$: $R^{(2/C)}$+ (paths through D)
$R^{(4/E)}$: $R^{(3/D)}$+ (paths through E)



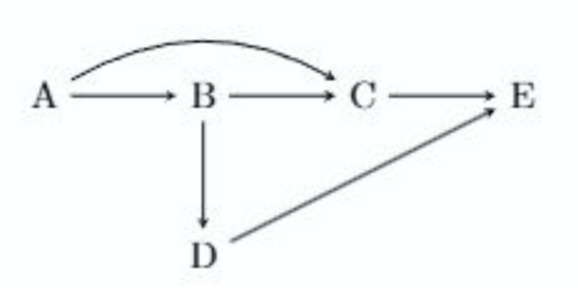
Question 2

Consider the directed graph $G = (V, E)$ given below:



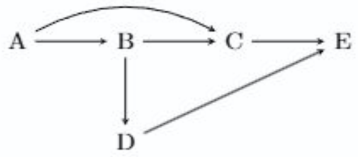
3. Draw the graph representation of the transitive closure of G.

$R^{(4)}$	A	B	C	D	E
A	0	1	1	1	1
B	0	0	1	1	1
C	0	0	0	0	1
D	0	0	0	0	1
E	0	0	0	0	0



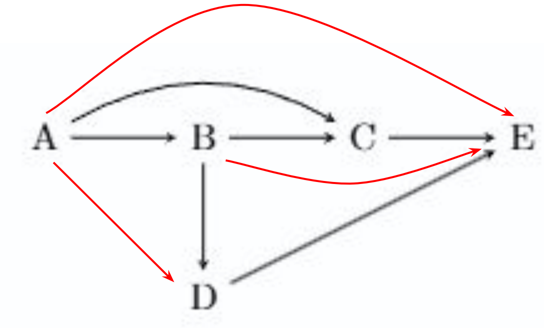
Question 2

Consider the directed graph $G = (V, E)$ given below:

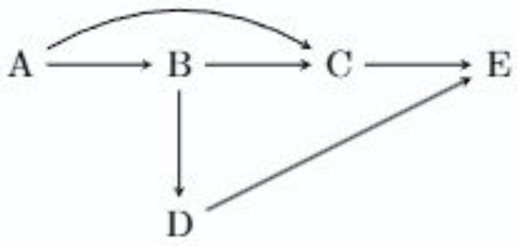


4. Determine the reachability of each node in G.

$R^{(4)}$	A	B	C	D	E
A	0	1	1	1	1
B	0	0	1	1	1
C	0	0	0	0	1
D	0	0	0	0	1
E	0	0	0	0	0



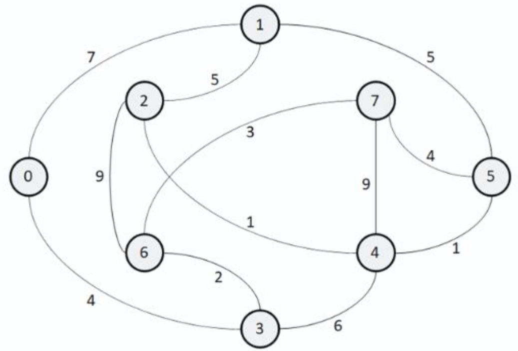
The transitive closure can help here



5. Identify if G is strongly connected. If not, can you add one edge to make G become a strongly connected graph?

Question 3

Consider the following graph G:



Let G_d be a directed graph using the vertices of G. For a pair of vertices u and v connected by an edge in G, their respective directed edge in G_d is as follows:

$$\text{Edge with vertices } u \text{ and } v = \begin{cases} (u,v), & \deg(u) < \deg(v) \vee (\deg(u) = \deg(v) \wedge u < v) \\ (v,u), & \text{Otherwise} \end{cases}$$

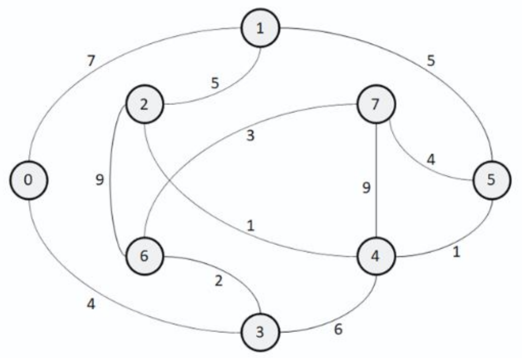
Let's draw G_d

First, calculate deg(v)

v	deg(v)
1	
2	
3	
4	
5	
6	
7	

Question 3

Consider the following graph G:

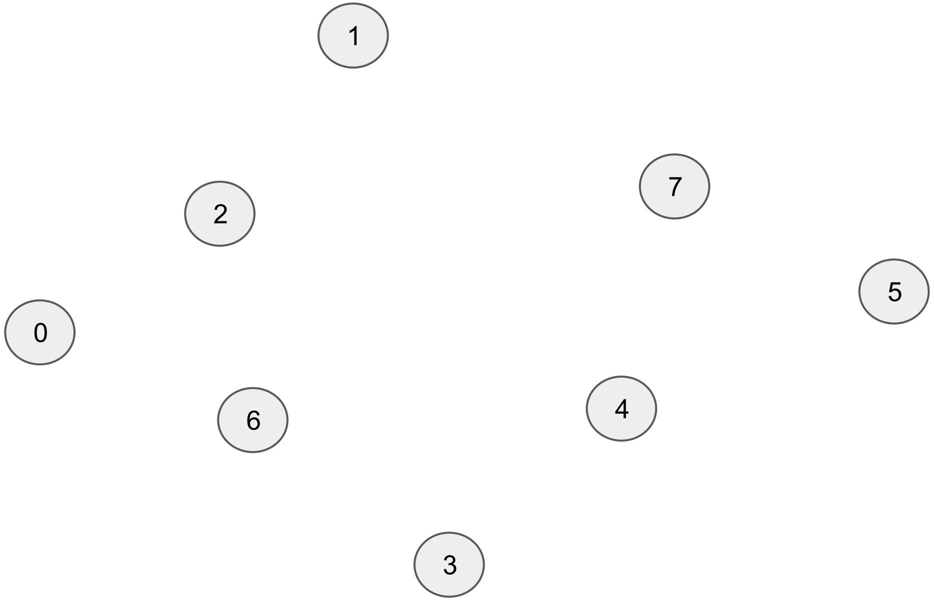


v	deg(v)
1	3
2	3
3	3
4	4
5	3
6	3
7	3

Let G_d be a directed graph using the vertices of G. For a pair of vertices u and v connected by an edge in G, their respective directed edge in G_d is as follows:

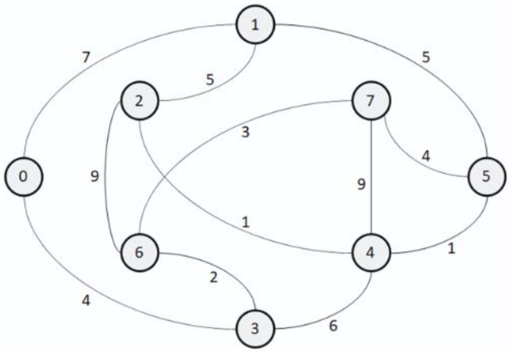
$$\text{Edge with vertices } u \text{ and } v = \begin{cases} (u,v), & \deg(u) < \deg(v) \lor (\deg(u) = \deg(v) \land u < v) \\ (v,u), & \text{Otherwise} \end{cases}$$

Let's draw G_d



Question 3

Consider the following graph G:

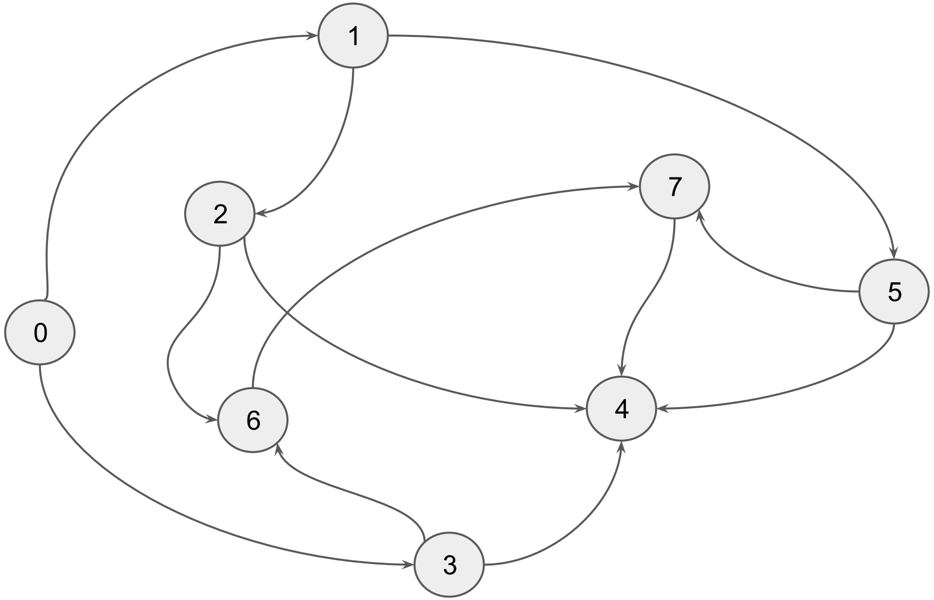


Let G_d be a directed graph using the vertices of G. For a pair of vertices u and v connected by an edge in G, their respective directed edge in G_d is as follows:

$$\text{Edge with vertices } u \text{ and } v = \begin{cases} (u, v), & \deg(u) < \deg(v) \vee (\deg(u) = \deg(v) \wedge u < v) \\ (v, u), & \text{Otherwise} \end{cases}$$

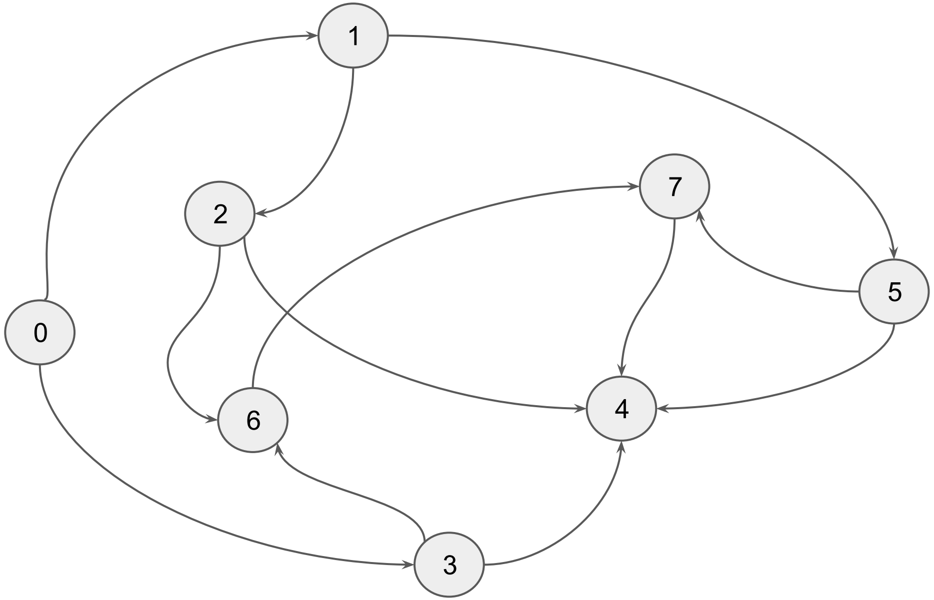
v	deg(v)
1	3
2	3
3	3
4	4
5	3
6	3
7	3

Let's draw G_d



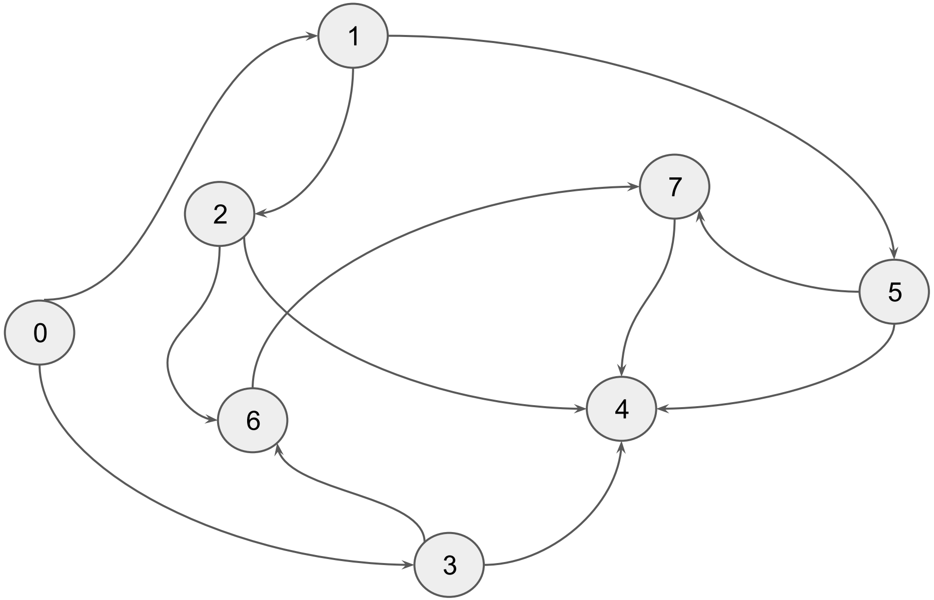
1. Is G_d strongly connected? If yes, explain why. Otherwise, list the minimum number of edges required to make G_d strongly connected.

v	deg(v)
1	3
2	3
3	3
4	4
5	3
6	3
7	3



1. Is G_d strongly connected? If yes, explain why. Otherwise, list the minimum number of edges required to make G_d strongly connected.

If we run Warshall..

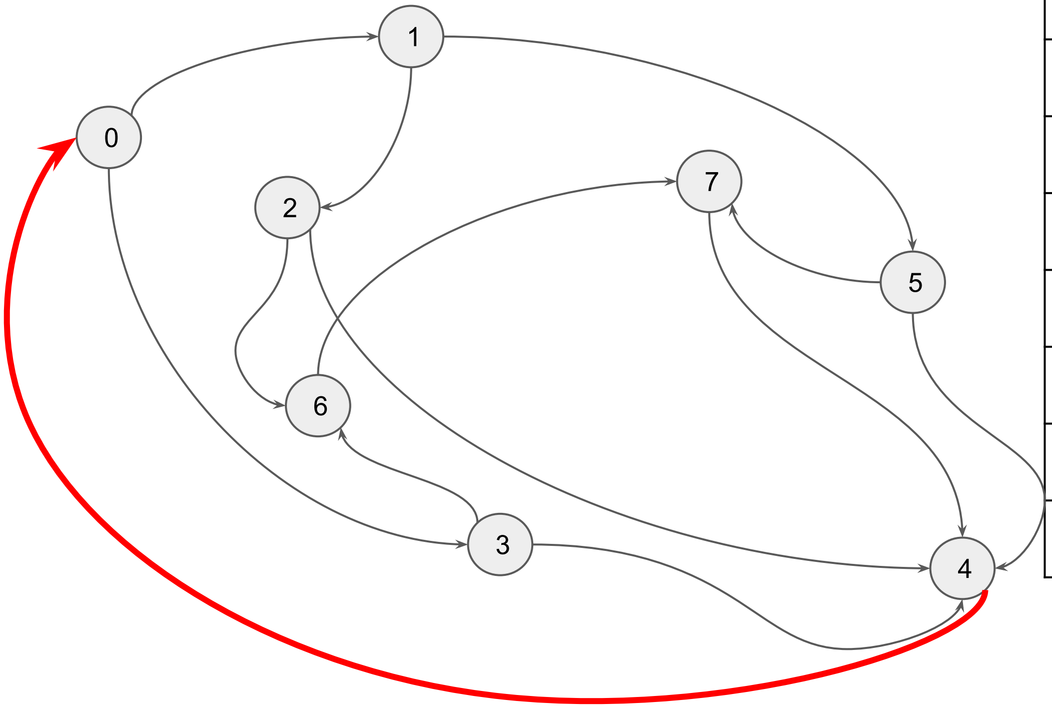


u/v	0	1	2	3	4	5	6	7
0		1	1	1	1	1	1	1
1			1		1	1	1	1
2					1		1	1
3					1		1	1
4								
5					1			1
6					1			1
7					1			

Observations:

1. Is G_d strongly connected? If yes, explain why. Otherwise, list the minimum number of edges required to make G_d strongly connected.

If we run Warshall..

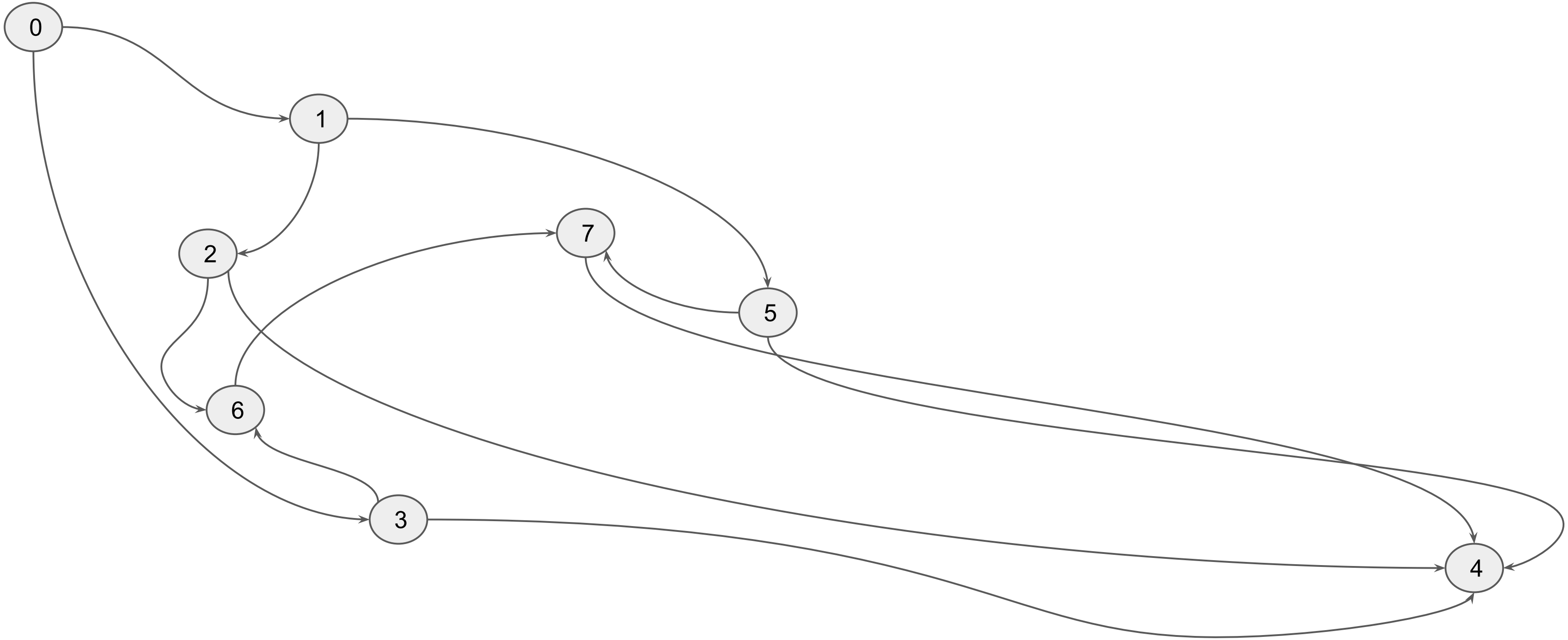


u/v	0	1	2	3	4	5	6	7
0		1	1	1	1	1	1	1
1			1		1	1	1	1
2					1		1	1
3					1		1	1
4								
5					1			1
6					1			1
7					1			

Adding (4,0) makes strongly connected

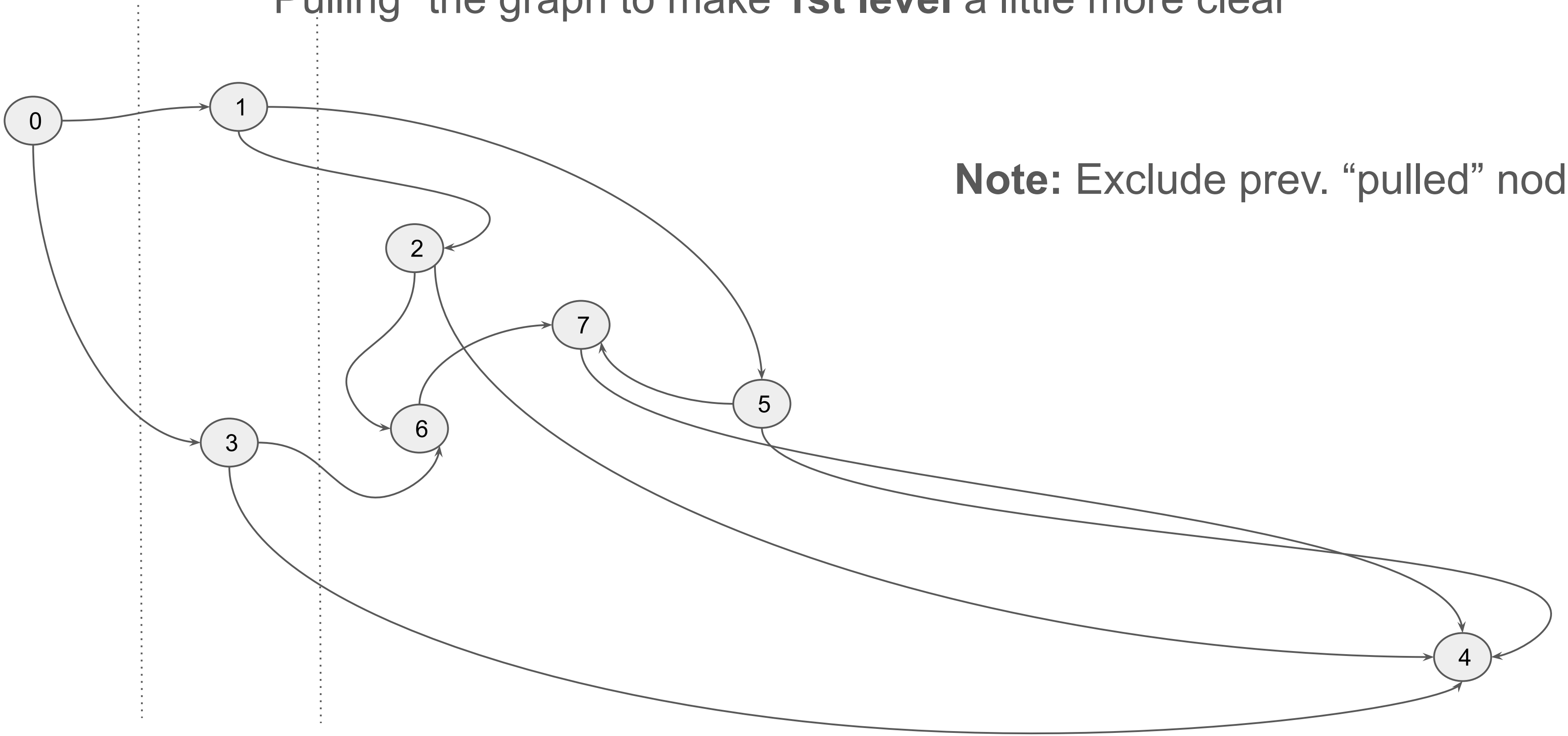
2. Show all the topological orderings of G_d.

“Pulling” the graph to make source/sink a little more clear



2. Show all the topological orderings of G_d.

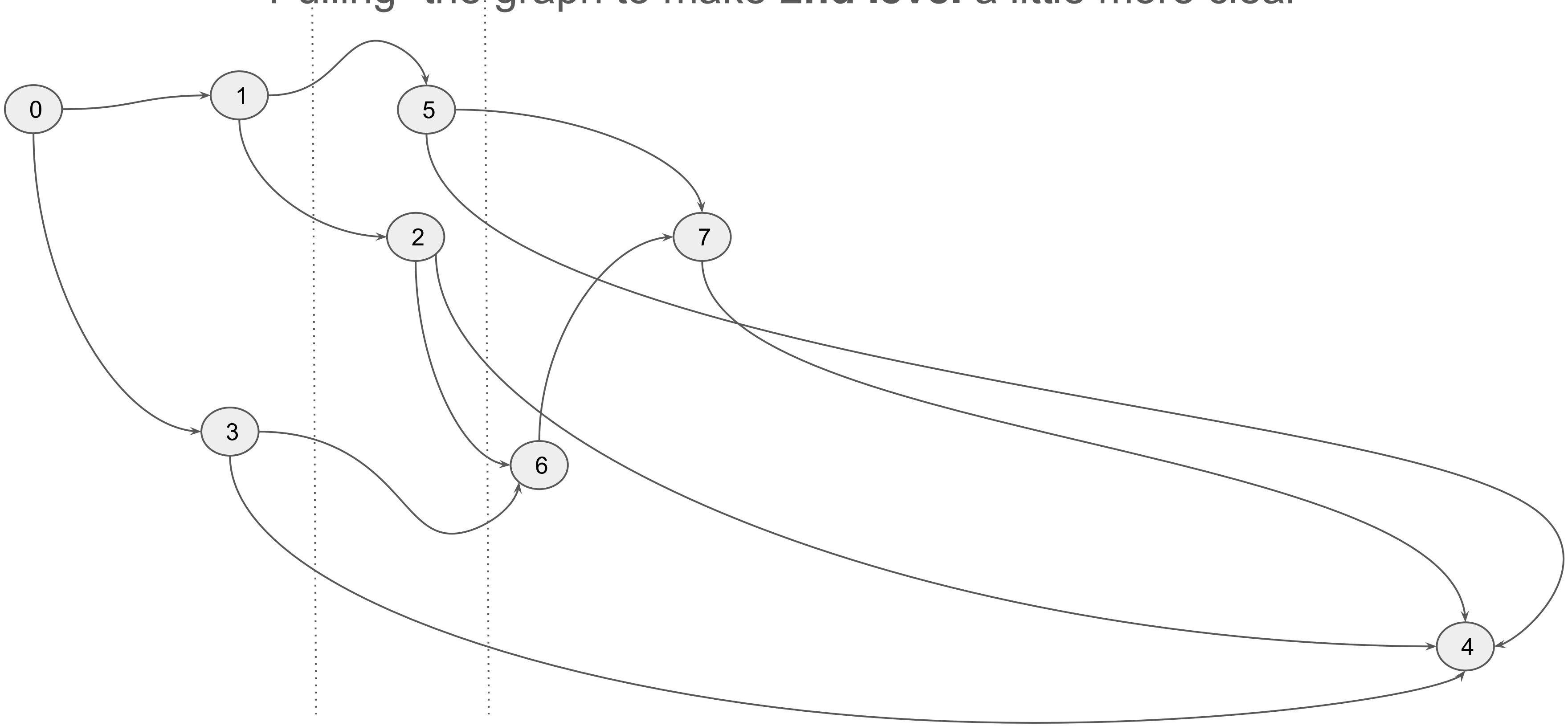
“Pulling” the graph to make **1st level** a little more clear



Note: Exclude prev. “pulled” nodes

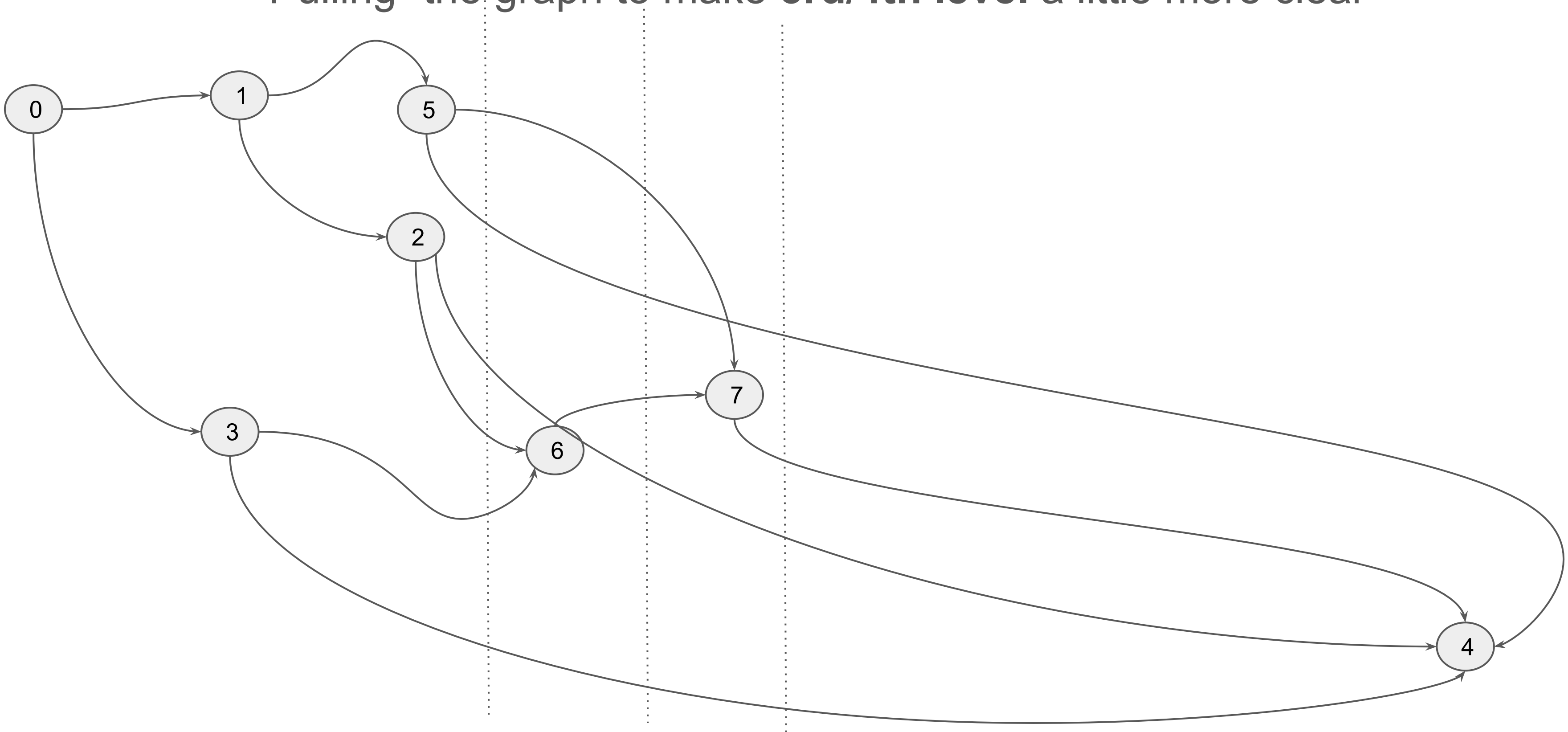
2. Show all the topological orderings of G_d.

"Pulling" the graph to make **2nd level** a little more clear



2. Show all the topological orderings of G_d.

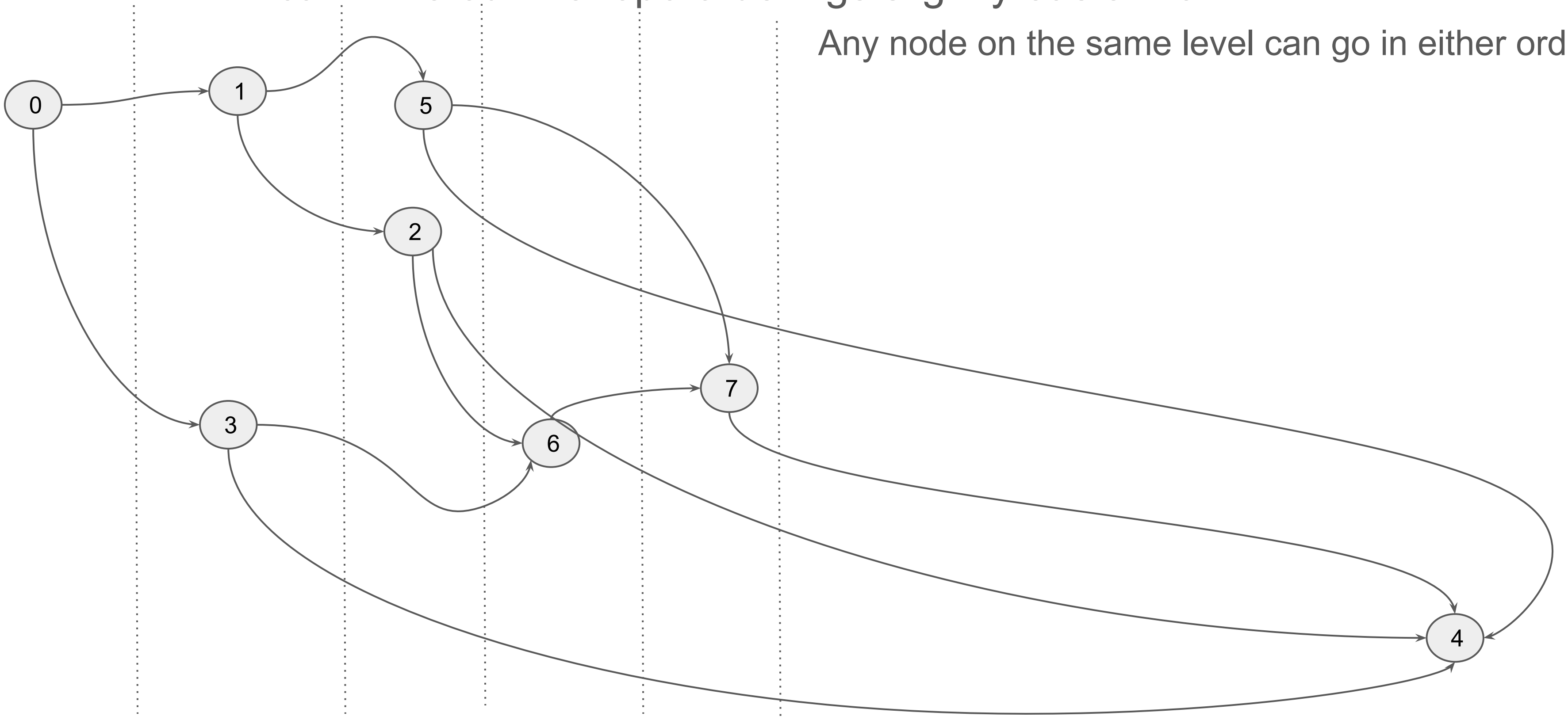
"Pulling" the graph to make **3rd/4th level** a little more clear



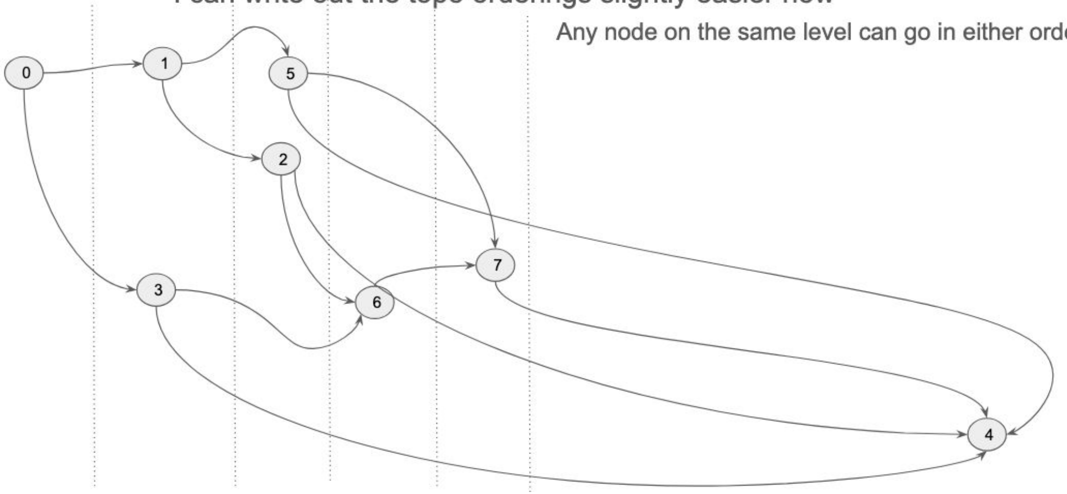
2. Show all the topological orderings of G_d.

I can write out the topo orderings slightly easier now

Any node on the same level can go in either order



I can write out the topo orderings slightly easier now



Any node on the same level can go in either order

Idea: chain by chain