PSO 9

Connected Components, Dijkstra, toposort

Question 1

(Strongly connected components)

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

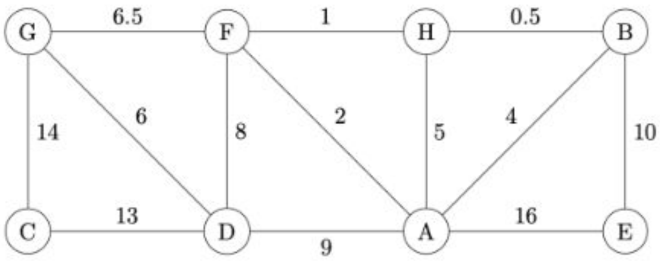
2. **(Euler tour)** An Euler tour of a strongly connected, directed graph $G = (V, E)$ is a cycle that traverses each edge of G exactly once, although it may visit a vertex more than once. Show that G has Euler tour if and only if

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

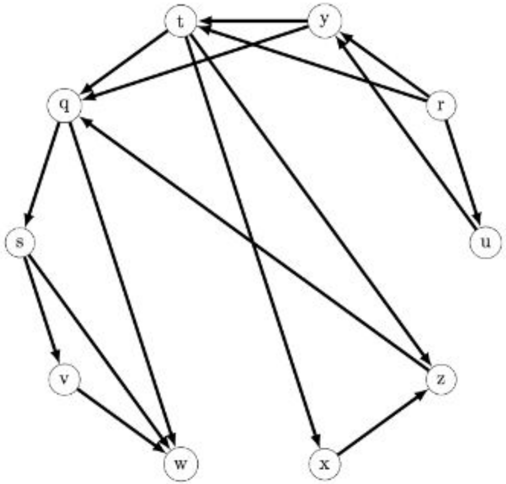
Question 2

(Dijkstra and topological sorting)

1. List the order that edges are added if we run Dijkstra's algorithm starting at node A in the following graph.



2. Execute the topological sort algorithm on the directed graph below.



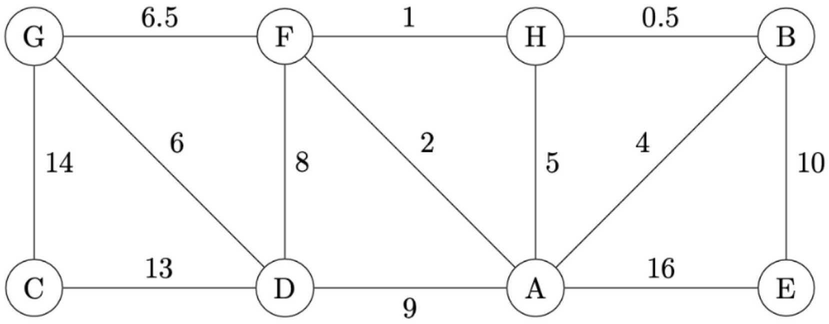
Lets start with this one

Dijkstra

```
algorithm DijkstraShortestPath(G(V,E), s ∈ V)

   let dist:V → ℤ
   let prev:V → V
   let Q be an empty priority queue

   dist[s] ← 0
   for each v ∈ V do
      if v ≠ s then
         dist[v] ← ∞
      end if
      prev[v] ← -1
      Q.add(dist[v], v)
   end for

   while Q is not empty do
      u ← Q.getMin()
      for each w ∈ V adjacent to u still in Q do
         d ← dist[u] + weight(u, w)
         if d < dist[w] then
            dist[w] ← d
            prev[w] ← u
            Q.set(d, w)
         end if
      end for
   end while

   return dist, prev
end algorithm
```

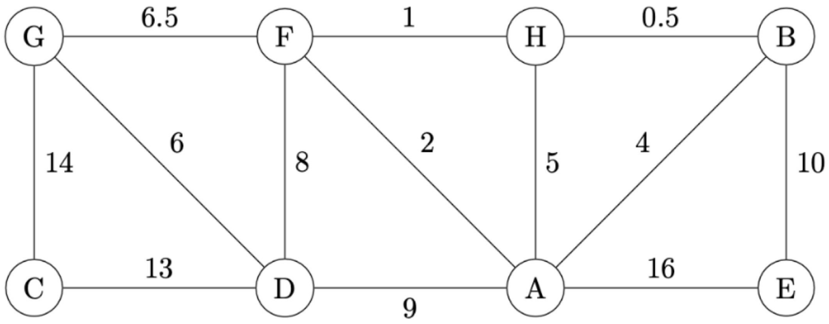
For a vertex s, finds shortest paths to all vertices. At each step..

- Consider current closest vertex u (priority queue)
- Greedily update path lengths to u’s neighbors
- Mark as visited



	A	B	C	D	E	F	G	H
dist	0	∞	∞	∞	∞	∞	∞	∞
prev	-1	-1	-1	-1	-1	-1	-1	-1

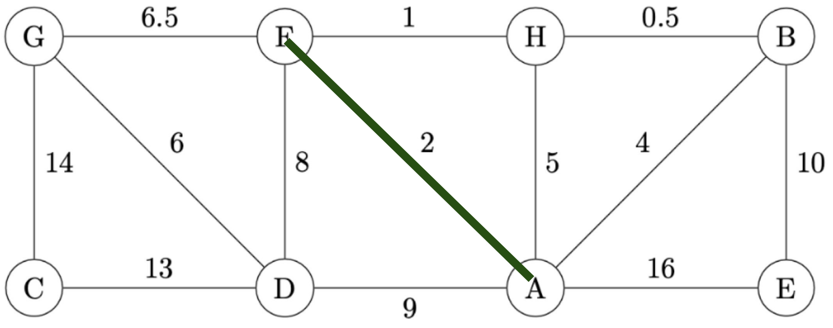
Q
(0, A)
(∞, B)
(∞, C)
(∞, D)
(∞, E)
(∞, F)
(∞, G)



	A	B	C	D	E	F	G	H
dist	0	4	∞	9	16	2	∞	5
prev	-1	A	-1	A	A	A	-1	A

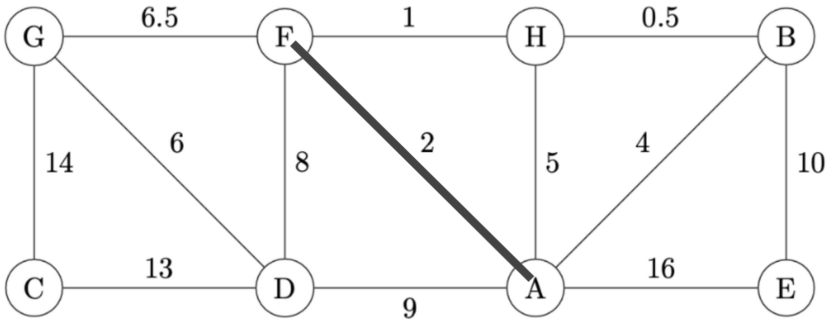
Q

(2, F)
(4, B)
(9, D)
(16, E)
(∞, C)
(∞, G)



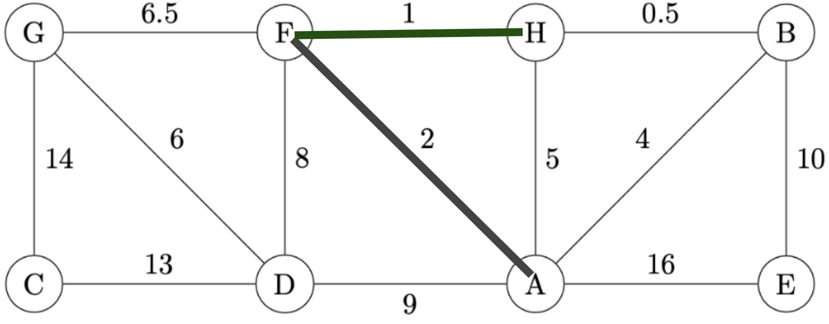
	A	B	C	D	E	F	G	H
dist	0	4	∞	9	16	2	∞	5
prev	-1	A	-1	A	A	A	-1	A

Q
(2, F)
(4, B)
(9, D)
(16, E)
(∞, C)
(∞, G)



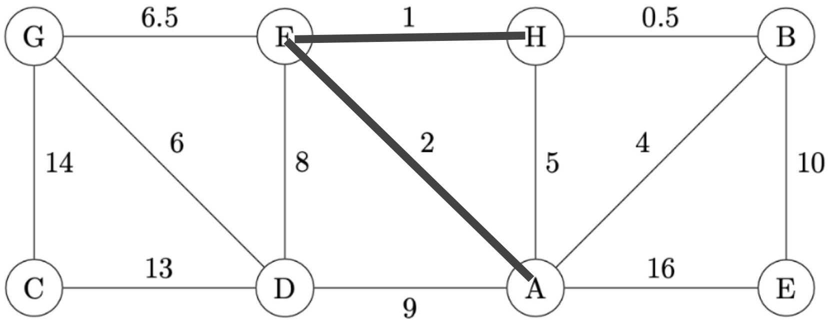
	A	B	C	D	E	F	G	H
dist	0	4	∞	9	16	2	8.5	3
prev	−1	A	−1	A	A	A	F	F

Q
(3, H)
(4, B)
(9, D)
(16, E)
(∞, C)
(8.5, G)



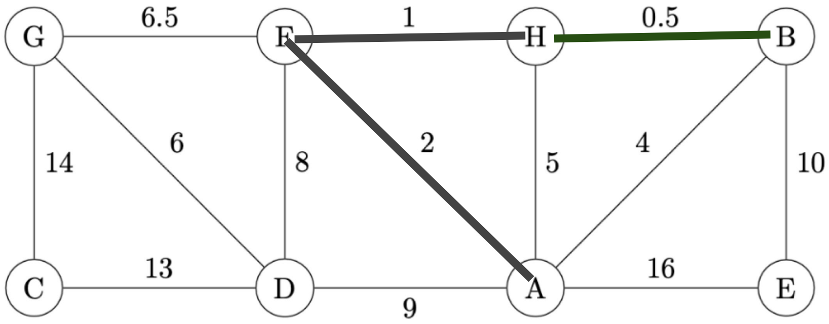
	A	B	C	D	E	F	G	H
dist	0	4	∞	9	16	2	8.5	3
prev	-1	A	-1	A	A	A	F	F

Q
(3, H)
(4, B)
(9, D)
(16, E)
(∞, C)
(8.5, G)



	A	B	C	D	E	F	G	H
dist	0	3.5	∞	9	16	2	8.5	3
prev	-1	H	-1	A	A	A	F	F

Q
(3.5, B)
(8.5, G)
(9, D)
(16, E)
(∞, C)



	A	B	C	D	E	F	G	H
dist	0	3.5	∞	9	16	2	8.5	3
prev	-1	H	-1	A	A	A	F	F

Q

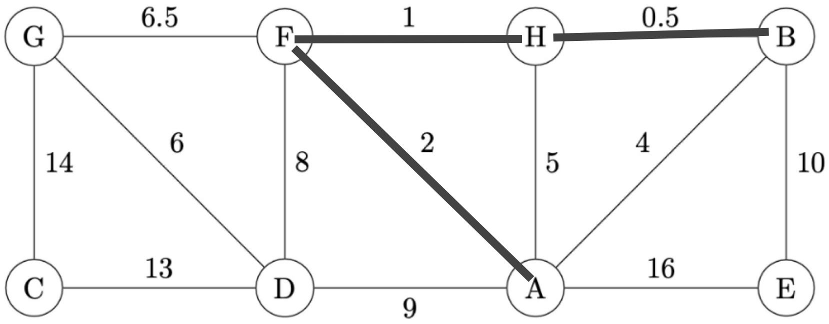
(3.5, B)

(8.5, G)

(9, D)

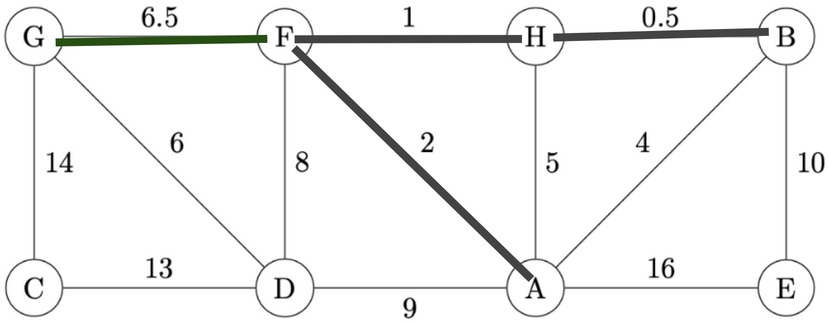
(16, E)

(∞, C)



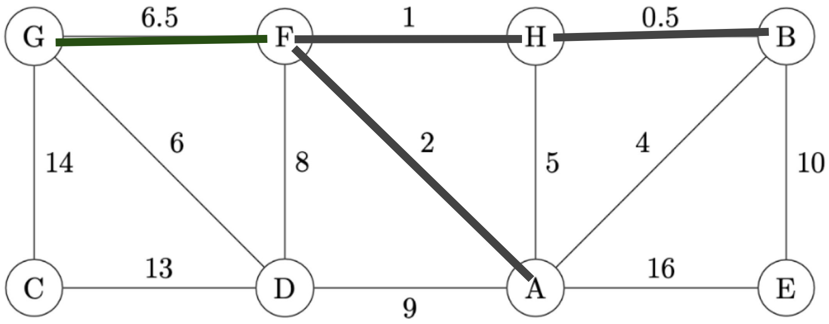
	A	B	C	D	E	F	G	H
dist	0	3.5	∞	9	13.5	2	8.5	3
Prev	-1	H	-1	A	B	A	F	F

Q
(8.5, G)
(9, D)
(13.5, E)
(∞, C)



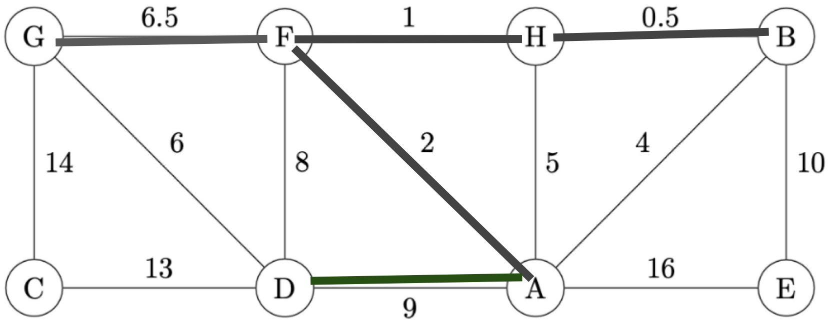
	A	B	C	D	E	F	G	H
dist	0	3.5	∞	9	13.5	2	8.5	3
Prev	-1	H	-1	A	B	A	F	F

Q
(8.5, G)
(9, D)
(13.5, E)
(∞, C)



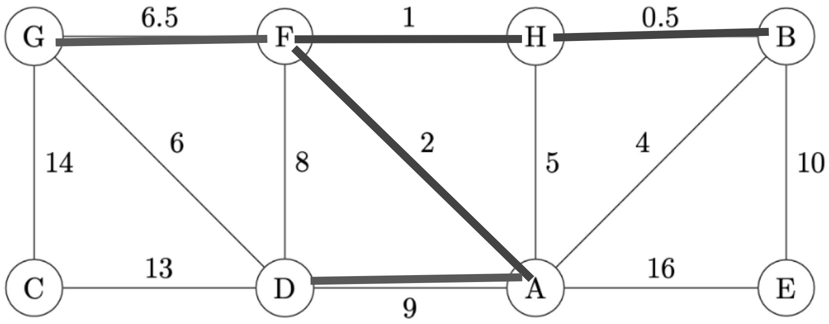
	A	B	C	D	E	F	G	H
dist	0	3.5	22.5	9	13.5	2	8.5	3
Prev	-1	H	G	A	B	A	F	F

Q
(9, D)
(13.5, E)
(22.5, C)



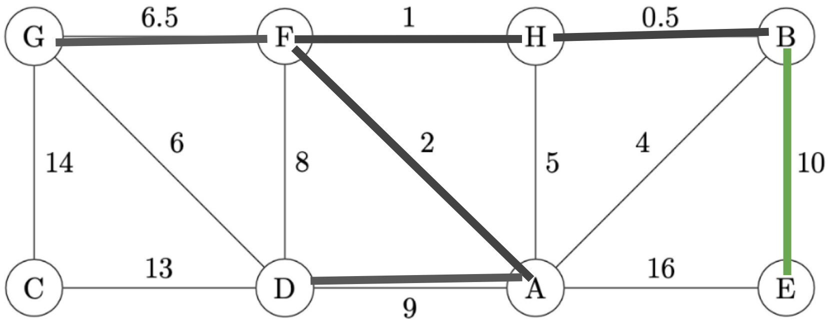
	A	B	C	D	E	F	G	H
dist	0	3.5	22.5	9	13.5	2	8.5	3
Prev	-1	H	G	A	B	A	F	F

Q
(9, D)
(13.5, E)
(22.5, C)



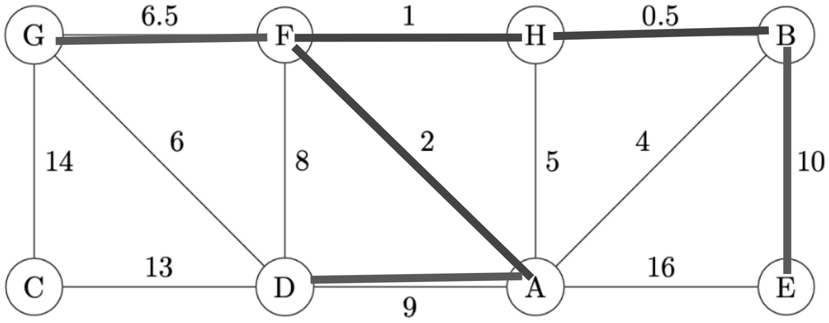
	A	B	C	D	E	F	G	H
dist	0	3.5	22	9	13.5	2	8.5	3
Prev	-1	H	D	A	B	A	F	F

Q
(13.5, E)
(22, C)



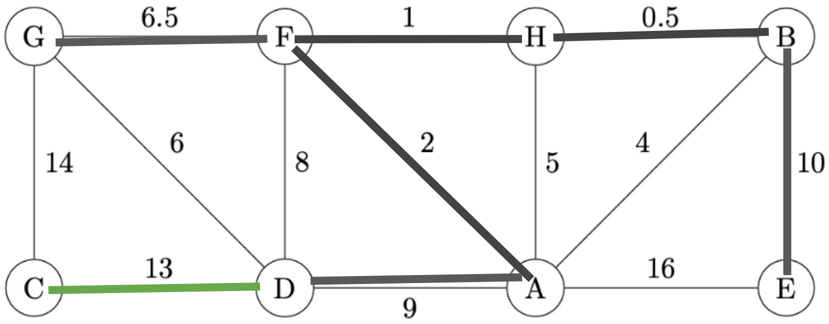
	A	B	C	D	E	F	G	H
dist	0	3.5	22	9	13.5	2	8.5	3
Prev	-1	H	D	A	B	A	F	F

Q
(13.5, E)
(22 , C)



	A	B	C	D	E	F	G	H
dist	0	3.5	22	9	13.5	2	8.5	3
Prev	-1	H	D	A	B	A	F	F

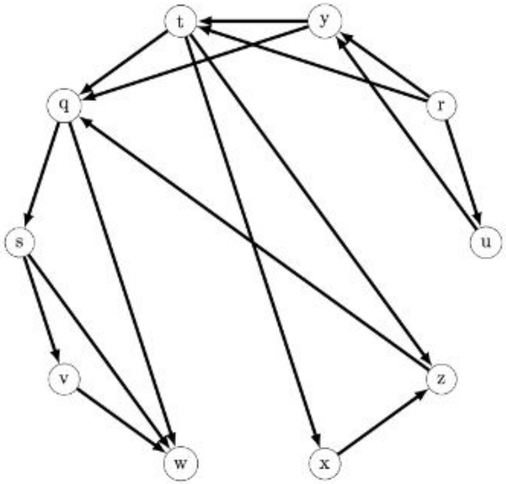
Q
(22 , C)



	A	B	C	D	E	F	G	H
dist	0	3.5	22	9	13.5	2	8.5	3
Prev	-1	H	D	A	B	A	F	F

Q
(22 , C)

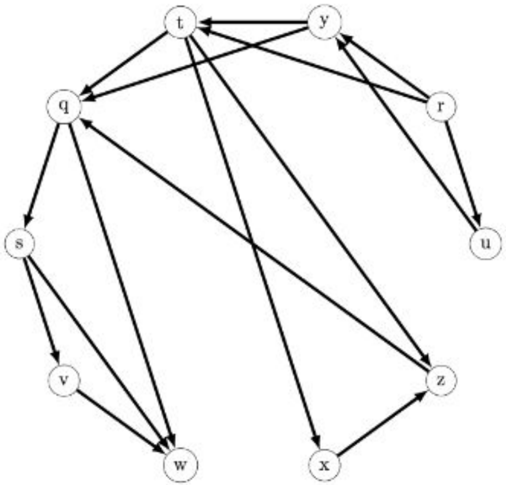
2. Execute the topological sort algorithm on the directed graph below.




```
algorithm TopologicalSort(G(V,E))
    let H be a copy of G
    n ← 0
    let T: v ∈ V → ℤ≥0

    while H is not empty do
        pick v ∈ H.V s.t. indeg(v) = 0
        T[v] ← n
        n ← n + 1
        remove v from H
    end while

    return T
end algorithm
```

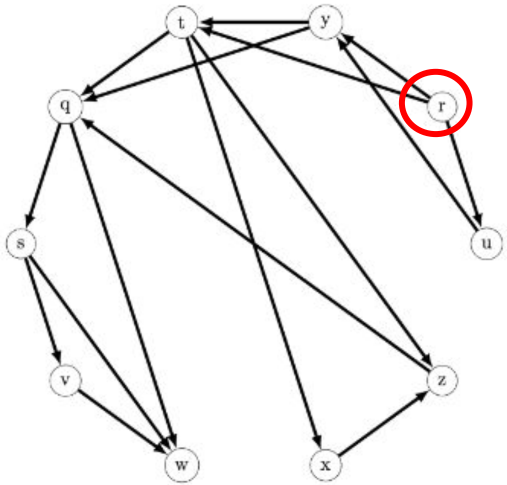
```
algorithm TopologicalSort(G(V,E))
    let H be a copy of G
    n ← 0
    let T: v ∈ V → ℤ≥0

    while H is not empty do
        pick v ∈ H.V s.t. indeg(v) = 0
        T[v] ← n
        n ← n + 1
        remove v from H
    end while

    return T
end algorithm
```

2. Execute the topological sort algorithm on the directed graph below.



Vertex	q	r	s	t	u	v	w	x	y	z
Order										

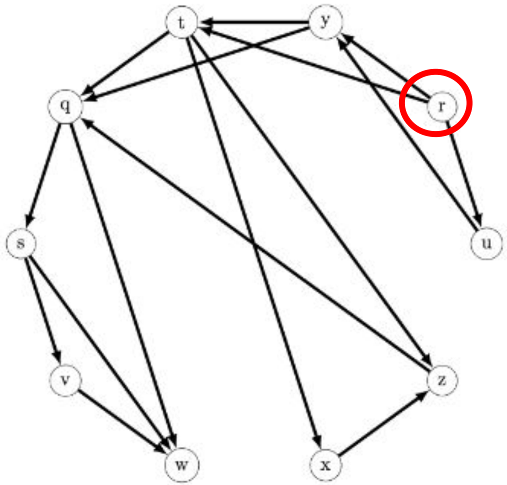
2. Execute the topological sort algorithm on the directed graph below.




```
algorithm TopologicalSort(G(V,E))
    let H be a copy of G
    n ← 0
    let T: v ∈ V → ℤ≥0

    while H is not empty do
        pick v ∈ H.V s.t. indeg(v) = 0
        T[v] ← n
        n ← n + 1
        remove v from H
    end while

    return T
end algorithm
```

Vertex	q	r	s	t	u	v	w	x	y	z
Order										

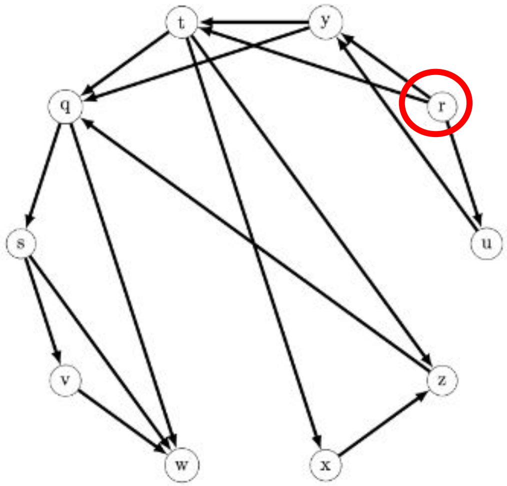
2. Execute the topological sort algorithm on the directed graph below.




```
algorithm TopologicalSort(G(V,E))
    let H be a copy of G
    n ← 0
    let T: v ∈ V → ℤ≥0

    while H is not empty do
        pick v ∈ H.V s.t. indeg(v) = 0
        T[v] ← n
        n ← n + 1
        remove v from H
    end while

    return T
end algorithm
```

Vertex	q	r	s	t	u	v	w	x	y	z
Order		0								

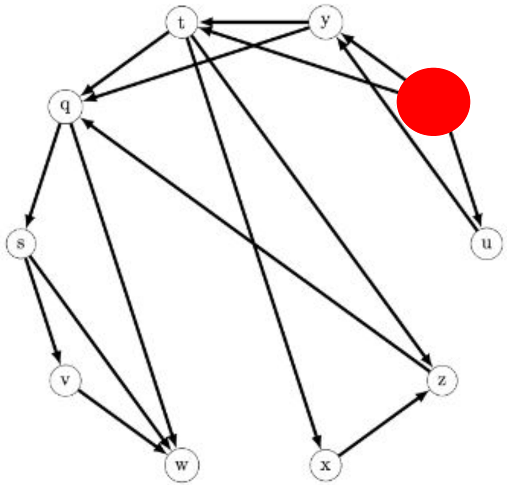
2. Execute the topological sort algorithm on the directed graph below.




```
algorithm TopologicalSort(G(V,E))
    let H be a copy of G
    n ← 1
    let I : v ∈ V → ℤ≥0

    while H is not empty do
        pick v ∈ H.V s.t. indeg(v) = 0
        T[v] ← n
        n ← n + 1
        remove v from H
    end while

    return T
end algorithm
```

Vertex	q	r	s	t	u	v	w	x	y	z
Order		0								

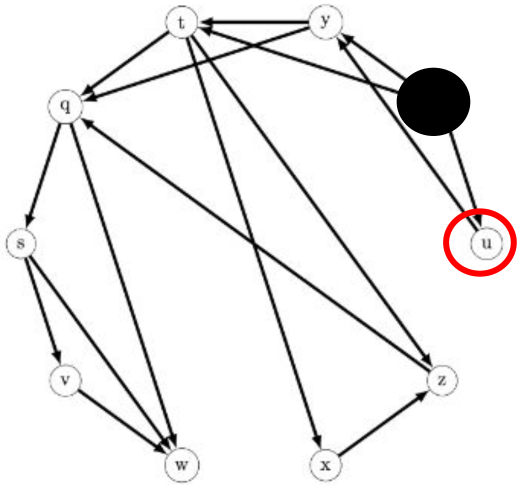
2. Execute the topological sort algorithm on the directed graph below.




```
algorithm TopologicalSort(G(V,E))
    let H be a copy of G
    n ← 1
    let I : v ∈ V → ℤ≥0

    while H is not empty do
        pick v ∈ H.V s.t. indeg(v) = 0
        T[v] ← n
        n ← n + 1
        remove v from H
    end while

    return T
end algorithm
```

Vertex	q	r	s	t	u	v	w	x	y	z
Order		0								

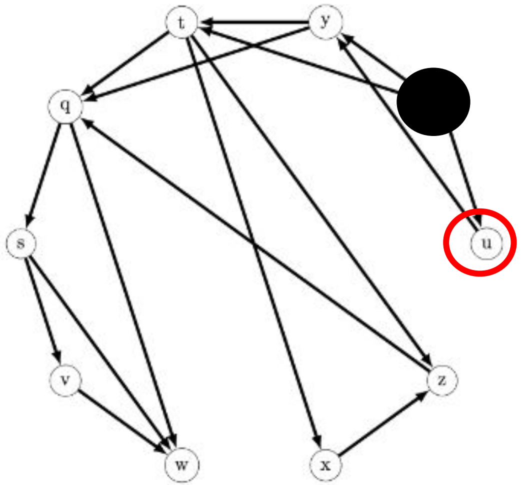
2. Execute the topological sort algorithm on the directed graph below.




```
algorithm TopologicalSort(G(V,E))
    let H be a copy of G
    n ← 1
    let I : v ∈ V → ℤ≥0

    while H is not empty do
        pick v ∈ H.V s.t. indeg(v) = 0
        T[v] ← n
        n ← n + 1
        remove v from H
    end while

    return T
end algorithm
```

Vertex	q	r	s	t	u	v	w	x	y	z
Order		0								

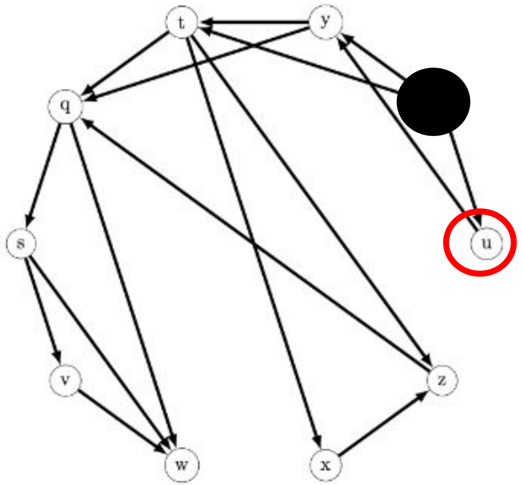
2. Execute the topological sort algorithm on the directed graph below.




```
algorithm TopologicalSort(G(V,E))
    let H be a copy of G
    n ← 1
    let T : v ∈ V → ℤ≥0

    while H is not empty do
        pick v ∈ H.V s.t. indeg(v) = 0
        T[v] ← n
        n ← n + 1
        remove v from H
    end while

    return T
end algorithm
```

Vertex	q	r	s	t	u	v	w	x	y	z
Order		0			1					

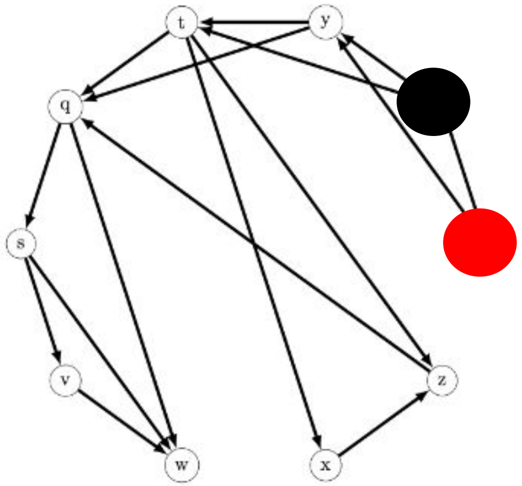
2. Execute the topological sort algorithm on the directed graph below.




```
algorithm TopologicalSort(G(V,E))
    let H be a copy of G
    n ← 2
    let T : v ∈ V → ℤ≥0

    while H is not empty do
        pick v ∈ H.V s.t. indeg(v) = 0
        T[v] ← n
        n ← n + 1
        remove v from H
    end while

    return T
end algorithm
```

Vertex	q	r	s	t	u	v	w	x	y	z
Order		0			1					

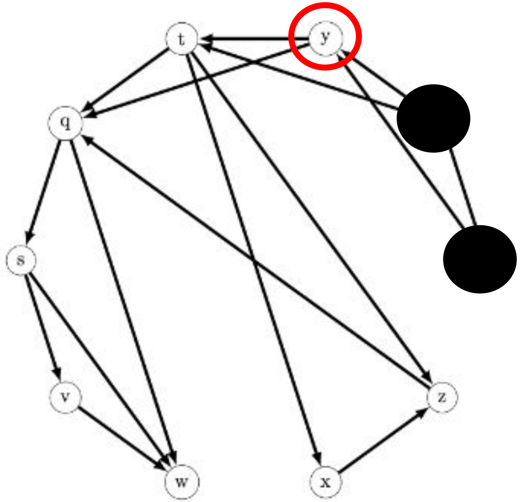
2. Execute the topological sort algorithm on the directed graph below.




```
algorithm TopologicalSort(G(V,E))
    let H be a copy of G
    n ← 2
    let T : v ∈ V → ℤ≥0

    while H is not empty do
        pick v ∈ H.V s.t. indeg(v) = 0
        T[v] ← n
        n ← n + 1
        remove v from H
    end while

    return T
end algorithm
```

Vertex	q	r	s	t	u	v	w	x	y	z
Order		0			1					

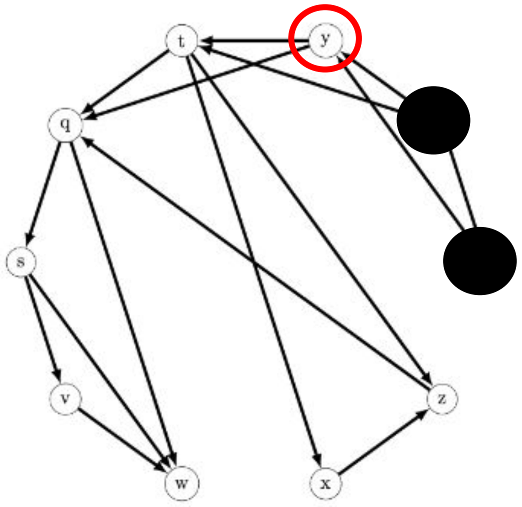
2. Execute the topological sort algorithm on the directed graph below.




```
algorithm TopologicalSort(G(V,E))
    let H be a copy of G
    n ← 2
    let T : v ∈ V → ℤ≥0

    while H is not empty do
        pick v ∈ H.V s.t. indeg(v) = 0
        T[v] ← n
        n ← n + 1
        remove v from H
    end while

    return T
end algorithm
```

Vertex	q	r	s	t	u	v	w	x	y	z
Order		0			1					

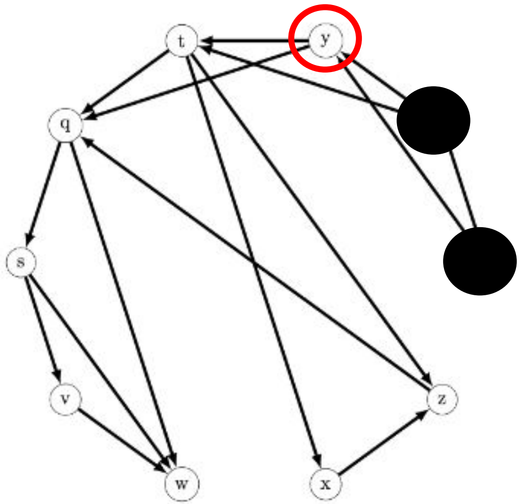
2. Execute the topological sort algorithm on the directed graph below.




```
algorithm TopologicalSort(G(V,E))
    let H be a copy of G
    n ← 2
    let T : v ∈ V → ℤ≥0

    while H is not empty do
        pick v ∈ H.V s.t. indeg(v) = 0
        T[v] ← n
        n ← n + 1
        remove v from H
    end while

    return T
end algorithm
```

Vertex	q	r	s	t	u	v	w	x	y	z
Order		0			1				2	

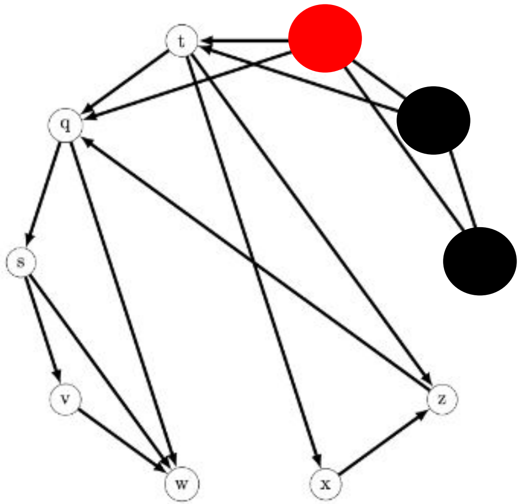
2. Execute the topological sort algorithm on the directed graph below.




```
algorithm TopologicalSort(G(V,E))
    let H be a copy of G
    n ← 3
    let I : v ∈ V → ℤ≥0

    while H is not empty do
        pick v ∈ H.V s.t. indeg(v) = 0
        T[v] ← n
        n ← n + 1
        remove v from H
    end while

    return T
end algorithm
```

Vertex	q	r	s	t	u	v	w	x	y	z
Order		0			1				2	

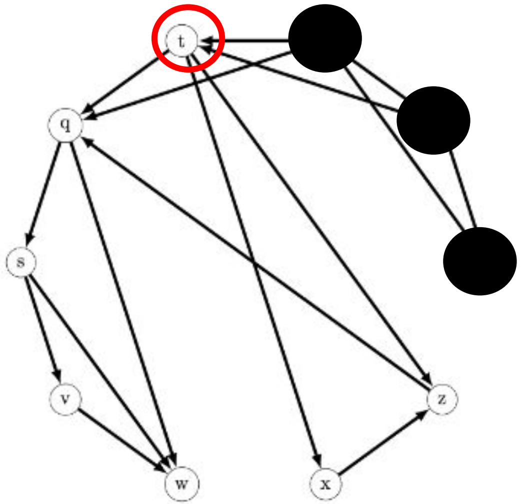
2. Execute the topological sort algorithm on the directed graph below.




```
algorithm TopologicalSort(G(V,E))
    let H be a copy of G
    n ← 3
    let I : v ∈ V → ℤ≥0

    while H is not empty do
        pick v ∈ H.V s.t. indeg(v) = 0
        T[v] ← n
        n ← n + 1
        remove v from H
    end while

    return T
end algorithm
```

Vertex	q	r	s	t	u	v	w	x	y	z
Order		0			1				2	

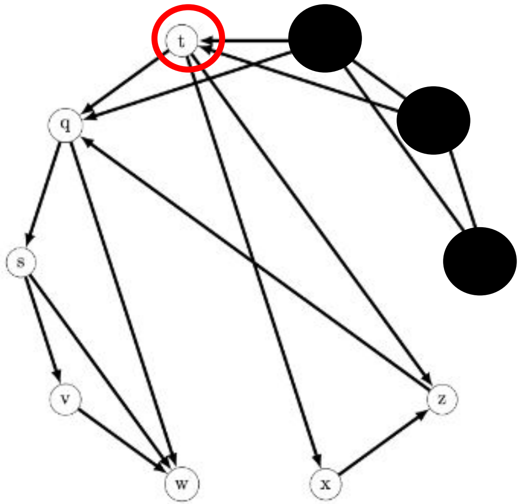
2. Execute the topological sort algorithm on the directed graph below.




```
algorithm TopologicalSort(G(V,E))
    let H be a copy of G
    n ← 3
    let T: v ∈ V → ℤ≥0

    while H is not empty do
        pick v ∈ H.V s.t. indeg(v) = 0
        T[v] ← n
        n ← n + 1
        remove v from H
    end while

    return T
end algorithm
```

Vertex	q	r	s	t	u	v	w	x	y	z
Order		0			1				2	

2. Execute the topological sort algorithm on the directed graph below.



algorithm TopologicalSort($G(V,E)$)

  let H be a copy of G

  $n \leftarrow 3$

  let $I : v \in V \rightarrow \mathbb{Z}_{\geq 0}$

  while H is not empty **do**

    pick $v \in H.V$ s.t. $\text{indeg}(v) = 0$

    $T[v] \leftarrow n$

    $n \leftarrow n + 1$

    remove v from H

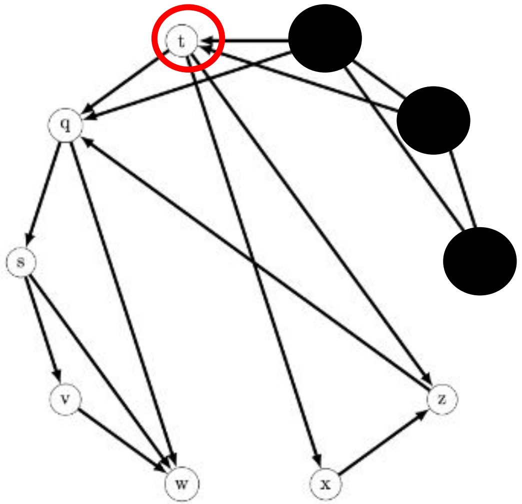
  end while

  return T

end algorithm

Vertex	q	r	s	t	u	v	w	x	y	z
Order		0		3	1				2	

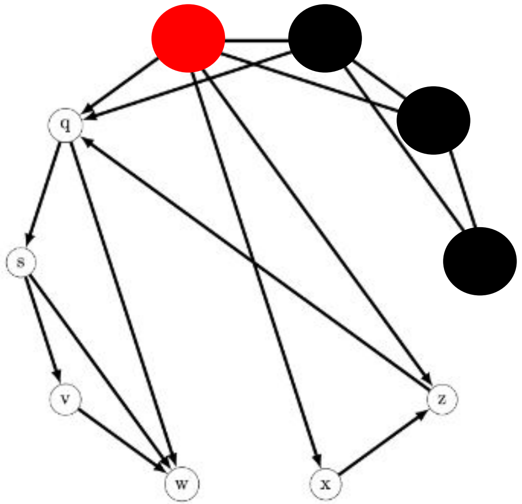
2. Execute the topological sort algorithm on the directed graph below.




```
algorithm TopologicalSort(G(V,E))
    let H be a copy of G
    n ← 4
    let I : v ∈ V → ℤ≥0

    while H is not empty do
        pick v ∈ H.V s.t. indeg(v) = 0
        T[v] ← n
        n ← n + 1
        remove v from H
    end while

    return T
end algorithm
```

Vertex	q	r	s	t	u	v	w	x	y	z
Order		0		3	1				2	

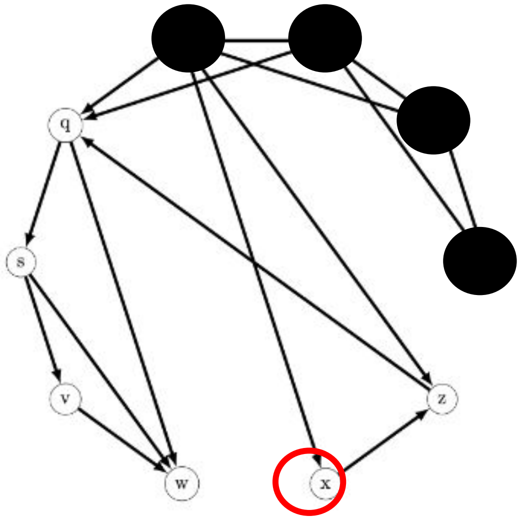
2. Execute the topological sort algorithm on the directed graph below.




```
algorithm TopologicalSort(G(V,E))
    let H be a copy of G
    n ← 4
    let I : v ∈ V → ℤ≥0

    while H is not empty do
        pick v ∈ H.V s.t. indeg(v) = 0
        T[v] ← n
        n ← n + 1
        remove v from H
    end while

    return T
end algorithm
```

Vertex	q	r	s	t	u	v	w	x	y	z
Order		0		3	1				2	

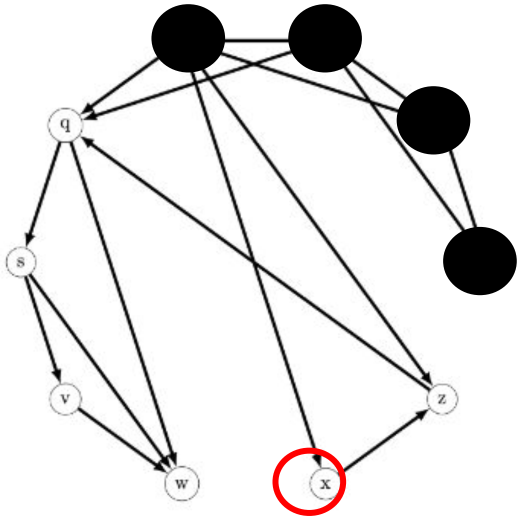
2. Execute the topological sort algorithm on the directed graph below.




```
algorithm TopologicalSort(G(V,E))
    let H be a copy of G
    n ← 4
    let I : v ∈ V → ℤ≥0

    while H is not empty do
        pick v ∈ H.V s.t. indeg(v) = 0
        T[v] ← n
        n ← n + 1
        remove v from H
    end while

    return T
end algorithm
```

Vertex	q	r	s	t	u	v	w	x	y	z
Order		0		3	1				2	

2. Execute the topological sort algorithm on the directed graph below.



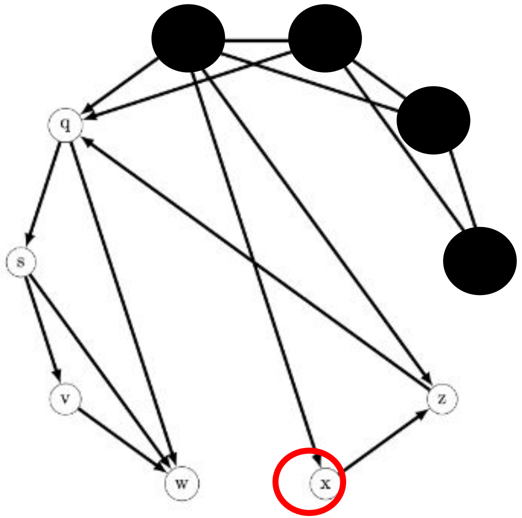
algorithm TopologicalSort($G(V,E)$)
  let H be a copy of G
  $n \leftarrow 4$
  let $I : v \in V \rightarrow \mathbb{Z}_{\geq 0}$

  while H is not empty **do**
    pick $v \in H.V$ s.t. $\text{indeg}(v) = 0$
    $T[v] \leftarrow n$
    $n \leftarrow n + 1$
    remove v from H
  end while

  return T
end algorithm

Vertex	q	r	s	t	u	v	w	x	y	z
Order		0		3	1			4	2	

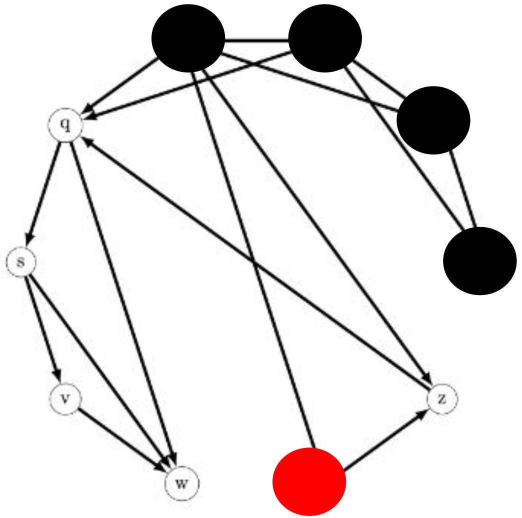
2. Execute the topological sort algorithm on the directed graph below.




```
algorithm TopologicalSort(G(V,E))
    let H be a copy of G
    n ← 5
    let T : v ∈ V → ℤ≥0

    while H is not empty do
        pick v ∈ H.V s.t. indeg(v) = 0
        T[v] ← n
        n ← n + 1
        remove v from H
    end while

    return T
end algorithm
```

Vertex	q	r	s	t	u	v	w	x	y	z
Order		0		3	1			4	2	

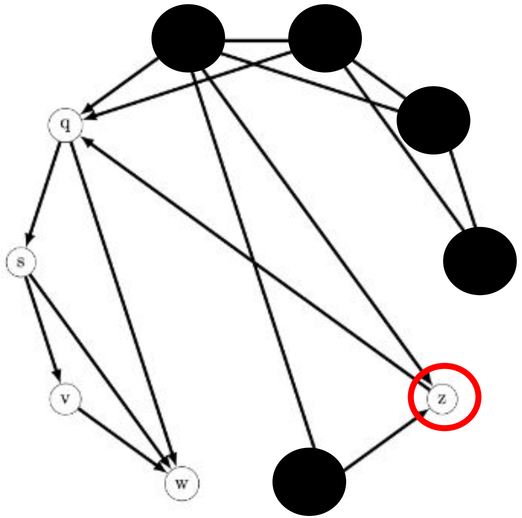
2. Execute the topological sort algorithm on the directed graph below.




```
algorithm TopologicalSort(G(V,E))
    let H be a copy of G
    n ← 5
    let T : v ∈ V → ℤ≥0

    while H is not empty do
        pick v ∈ H.V s.t. indeg(v) = 0
        T[v] ← n
        n ← n + 1
        remove v from H
    end while

    return T
end algorithm
```

Vertex	q	r	s	t	u	v	w	x	y	z
Order		0		3	1			4	2	

2. Execute the topological sort algorithm on the directed graph below.



algorithm TopologicalSort($G(V,E)$)

  let H be a copy of G

  $n \leftarrow$ 5

  let $I : v \in V \rightarrow \mathbb{Z}_{\geq 0}$

  while H is not empty **do**

    pick $v \in H.V$ s.t. $\text{indeg}(v) = 0$

    $T[v] \leftarrow n$

    $n \leftarrow n + 1$

    remove v from H

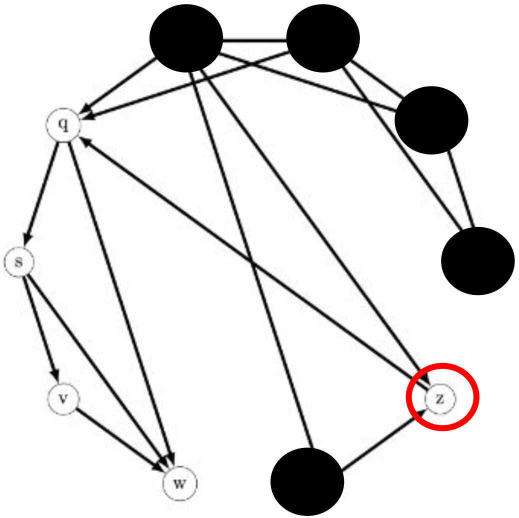
  end while

  return T

end algorithm

Vertex	q	r	s	t	u	v	w	x	y	z
Order		0		3	1			4	2	

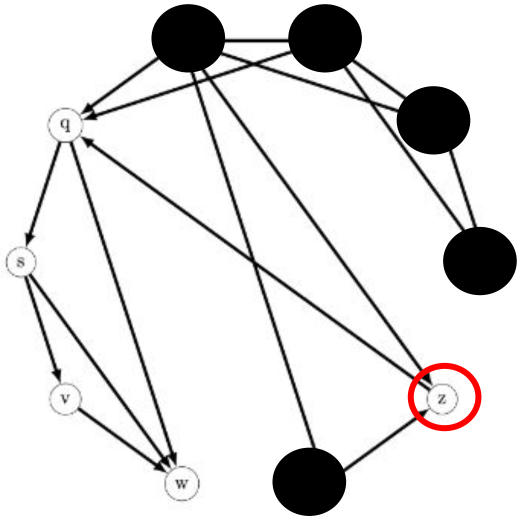
2. Execute the topological sort algorithm on the directed graph below.




```
algorithm TopologicalSort(G(V,E))
    let H be a copy of G
    n ← 5
    let I : v ∈ V → ℤ≥0

    while H is not empty do
        pick v ∈ H.V s.t. indeg(v) = 0
        T[v] ← n
        n ← n + 1
        remove v from H
    end while

    return T
end algorithm
```

Vertex	q	r	s	t	u	v	w	x	y	z
Order		0		3	1			4	2	5

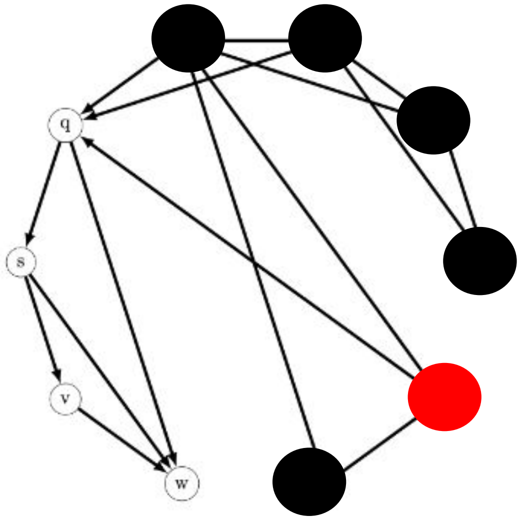
2. Execute the topological sort algorithm on the directed graph below.




```
algorithm TopologicalSort(G(V,E))
    let H be a copy of G
    n ← 6
    let I : v ∈ V → ℤ≥0

    while H is not empty do
        pick v ∈ H.V s.t. indeg(v) = 0
        T[v] ← n
        n ← n + 1
        remove v from H
    end while

    return T
end algorithm
```

Vertex	q	r	s	t	u	v	w	x	y	z
Order		0		3	1			4	2	5

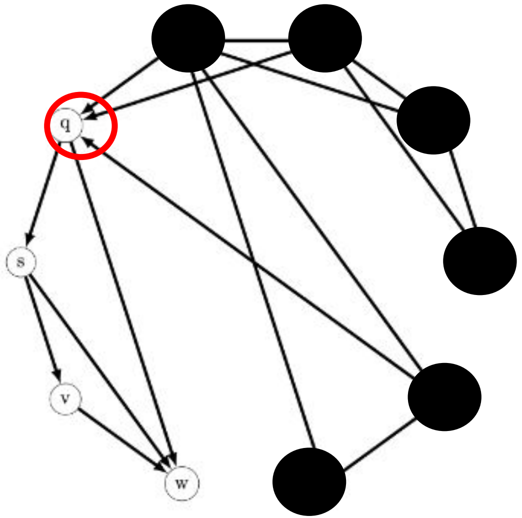
2. Execute the topological sort algorithm on the directed graph below.




```
algorithm TopologicalSort(G(V,E))
    let H be a copy of G
    n ← 6
    let I : v ∈ V → ℤ≥0

    while H is not empty do
        pick v ∈ H.V s.t. indeg(v) = 0
        T[v] ← n
        n ← n + 1
        remove v from H
    end while

    return T
end algorithm
```

Vertex	q	r	s	t	u	v	w	x	y	z
Order		0		3	1			4	2	5

2. Execute the topological sort algorithm on the directed graph below.



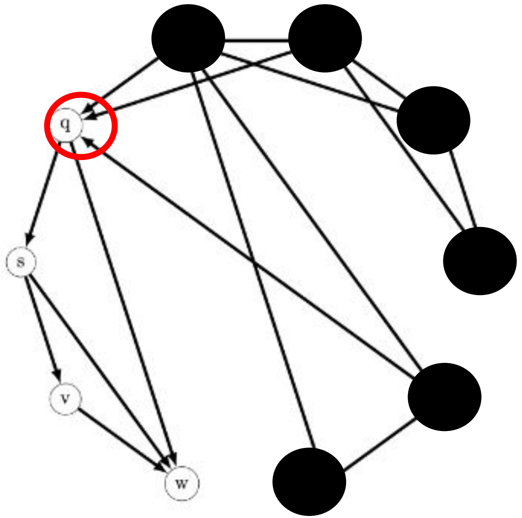
algorithm TopologicalSort($G(V,E)$)
let H be a copy of G
$n \leftarrow$ 6
let $I : v \in V \rightarrow \mathbb{Z}_{\geq 0}$

while H is not empty **do**
pick $v \in H.V$ s.t. $\text{indeg}(v) = 0$
$T[v] \leftarrow n$
$n \leftarrow n + 1$
remove v from H
end while

return T
end algorithm

Vertex	q	r	s	t	u	v	w	x	y	z
Order		0		3	1			4	2	5

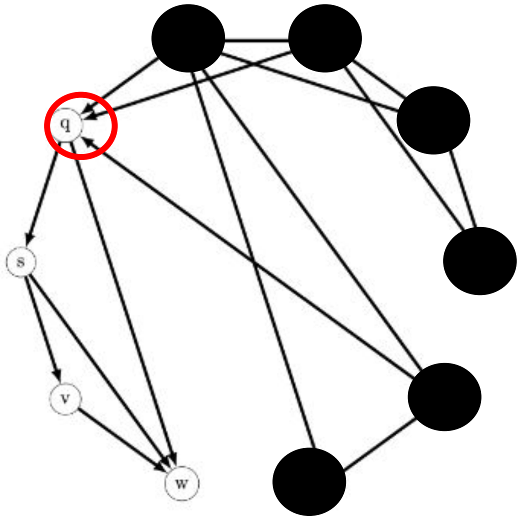
2. Execute the topological sort algorithm on the directed graph below.




```
algorithm TopologicalSort(G(V,E))
    let H be a copy of G
    n ← 6
    let ı : v ∈ V → ℤ≥0

    while H is not empty do
        pick v ∈ H.V s.t. indeg(v) = 0
        T[v] ← n
        n ← n + 1
        remove v from H
    end while

    return T
end algorithm
```

Vertex	q	r	s	t	u	v	w	x	y	z
Order	6	0		3	1			4	2	5

2. Execute the topological sort algorithm on the directed graph below.



algorithm TopologicalSort($G(V,E)$)

  let H be a copy of G

  $n \leftarrow 7$

  let $I : v \in V \rightarrow \mathbb{Z}_{\geq 0}$

  while H is not empty **do**

    pick $v \in H.V$ s.t. indeg$(v) = 0$

    $T[v] \leftarrow n$

    $n \leftarrow n + 1$

    remove v from H

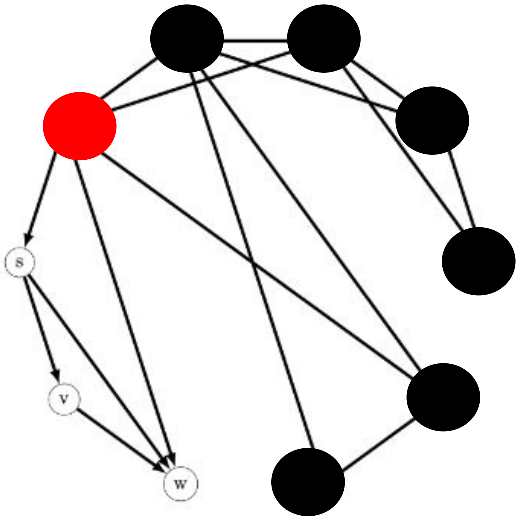
  end while

  return T

end algorithm

Vertex	q	r	s	t	u	v	w	x	y	z
Order	6	0		3	1			4	2	5

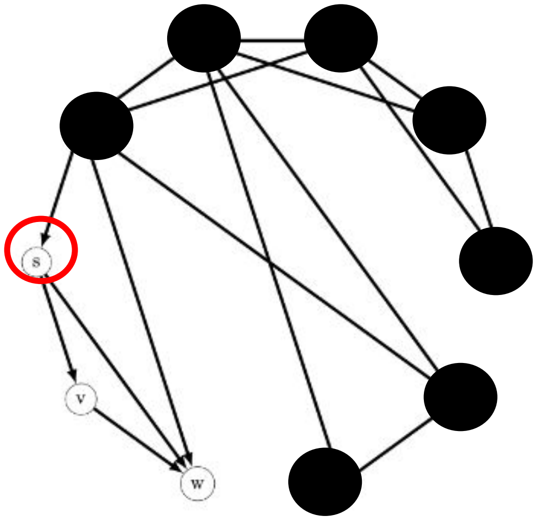
2. Execute the topological sort algorithm on the directed graph below.




```
algorithm TopologicalSort(G(V,E))
    let H be a copy of G
    n ← 7
    let I : v ∈ V → ℤ≥0

    while H is not empty do
        pick v ∈ H.V s.t. indeg(v) = 0
        T[v] ← n
        n ← n + 1
        remove v from H
    end while

    return T
end algorithm
```

Vertex	q	r	s	t	u	v	w	x	y	z
Order	6	0		3	1			4	2	5

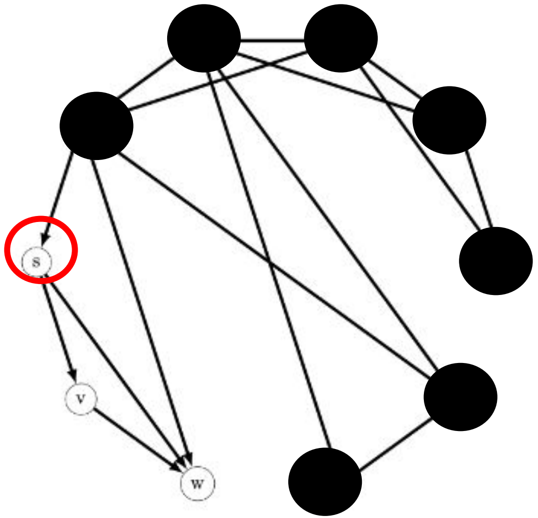
2. Execute the topological sort algorithm on the directed graph below.




```
algorithm TopologicalSort(G(V,E))
    let H be a copy of G
    n ← 7
    let I : v ∈ V → ℤ≥0

    while H is not empty do
        pick v ∈ H.V s.t. indeg(v) = 0
        T[v] ← n
        n ← n + 1
        remove v from H
    end while

    return T
end algorithm
```

Vertex	q	r	s	t	u	v	w	x	y	z
Order	6	0		3	1			4	2	5

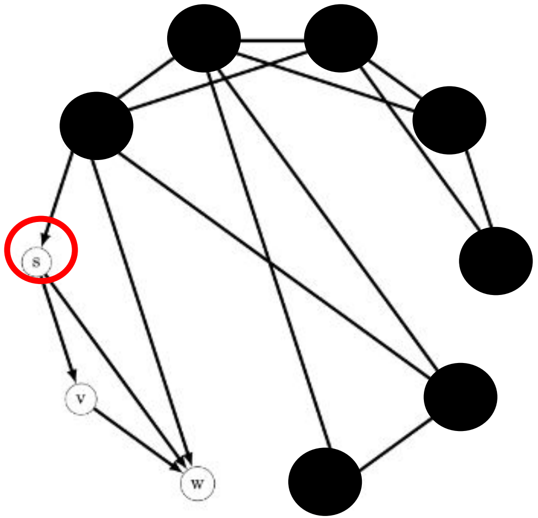
2. Execute the topological sort algorithm on the directed graph below.




```
algorithm TopologicalSort(G(V,E))
    let H be a copy of G
    n ← 7
    let I : v ∈ V → ℤ≥0

    while H is not empty do
        pick v ∈ H.V s.t. indeg(v) = 0
        T[v] ← n
        n ← n + 1
        remove v from H
    end while

    return T
end algorithm
```

Vertex	q	r	s	t	u	v	w	x	y	z
Order	6	0	7	3	1			4	2	5

2. Execute the topological sort algorithm on the directed graph below.

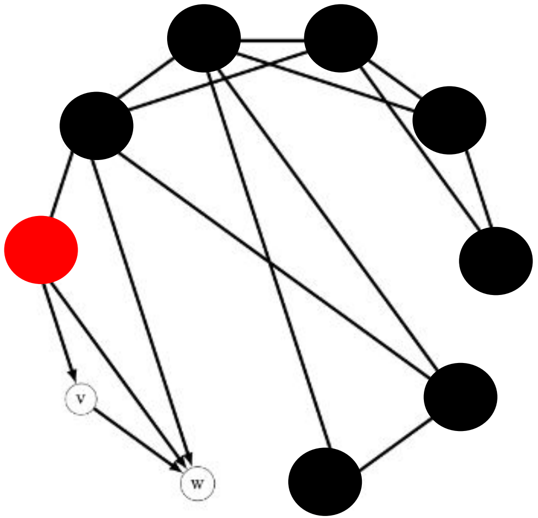



```
algorithm TopologicalSort(G(V,E))
    let H be a copy of G
    n ← 8
    let T : v ∈ V → ℤ≥0

    while H is not empty do
        pick v ∈ H.V s.t. indeg(v) = 0
        T[v] ← n
        n ← n + 1
        remove v from H
    end while

    return T
end algorithm
```

Vertex	q	r	s	t	u	v	w	x	y	z
Order	6	0	7	3	1			4	2	5

The rest follows similarly..

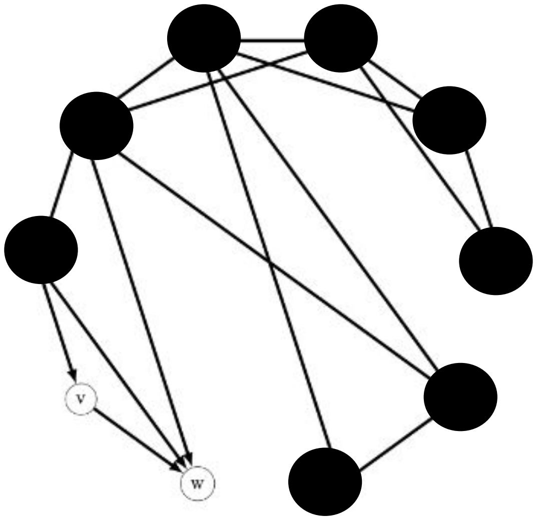
2. Execute the topological sort algorithm on the directed graph below.




```
algorithm TopologicalSort(G(V,E))
    let H be a copy of G
    n ← 8
    let I : v ∈ V → ℤ≥0

    while H is not empty do
        pick v ∈ H.V s.t. indeg(v) = 0
        T[v] ← n
        n ← n + 1
        remove v from H
    end while

    return T
end algorithm
```

Vertex	q	r	s	t	u	v	w	x	y	z
Order	6	0	7	3	1			4	2	5

2. Execute the topological sort algorithm on the directed graph below.




```
algorithm TopologicalSort(G(V,E))
    let H be a copy of G
    n ← 8
    let I : v ∈ V → ℤ≥0

    while H is not empty do
        pick v ∈ H.V s.t. indeg(v) = 0
        T[v] ← n
        n ← n + 1
        remove v from H
    end while

    return T
end algorithm
```

Vertex	q	r	s	t	u	v	w	x	y	z
Order	6	0	7	3	1	8	9	4	2	5

Doing it on paper..

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

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.

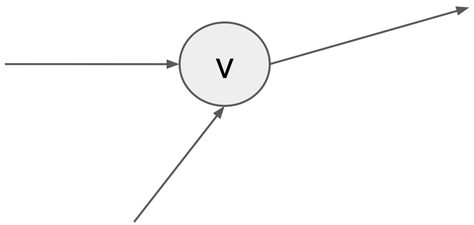
We want to show every vertex v has indeg(v) = outdeg(v).

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).



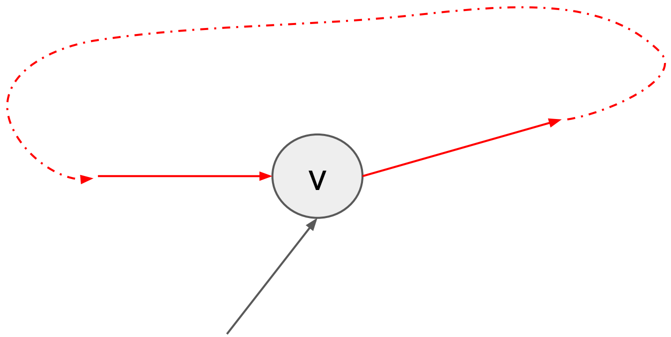
Suppose not, that there is a vertex v with indeg(v) > outdeg(v).

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).

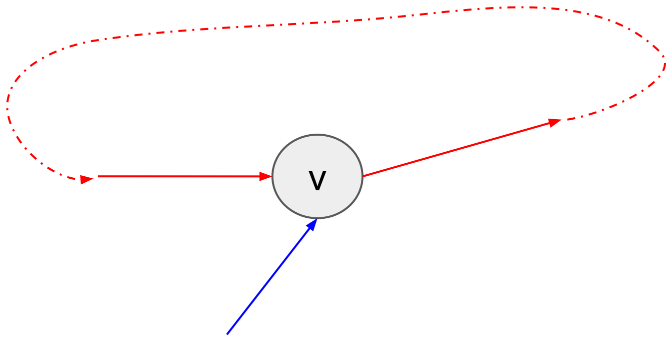
An Euler tour is a cycle i.e. each incoming edge is “paired” with an outgoing edge

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).

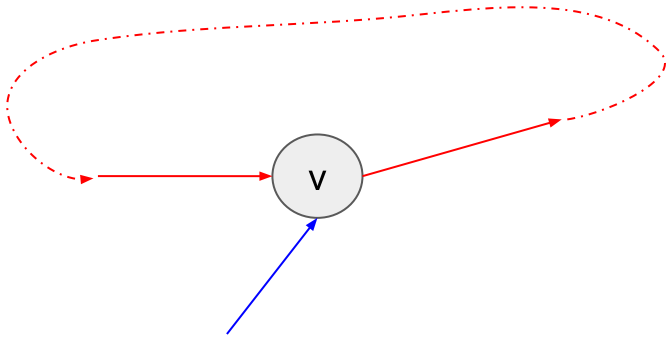
There will be an edge left over!

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!

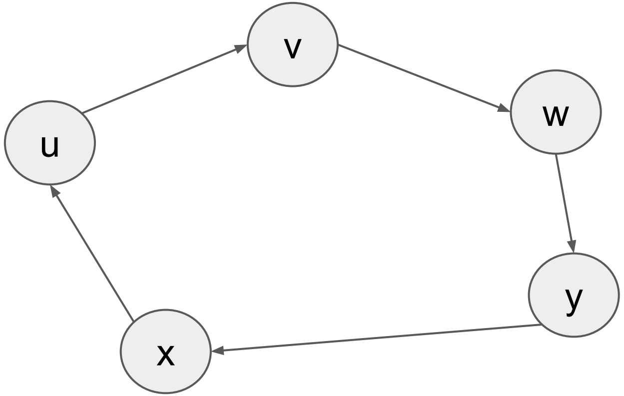
Exercise: show the same holds when indeg(v) < outdeg(v)

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

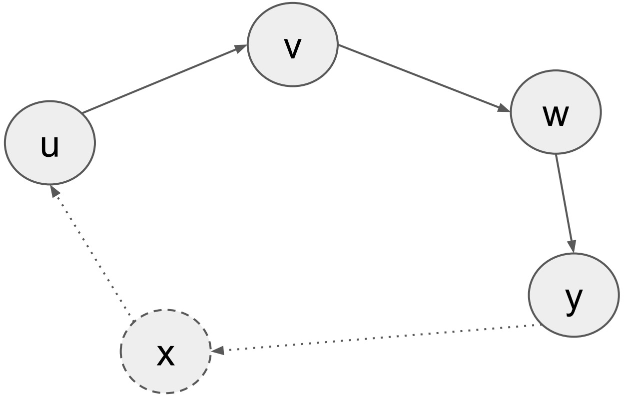


2. **(Euler tour)** An Euler tour of a strongly connected, directed graph $G = (V, E)$ is a cycle that traverses each edge of G exactly once, although it may visit a vertex more than once. Show that G has Euler tour if and only if

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

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

We want to show there is an Euler tour



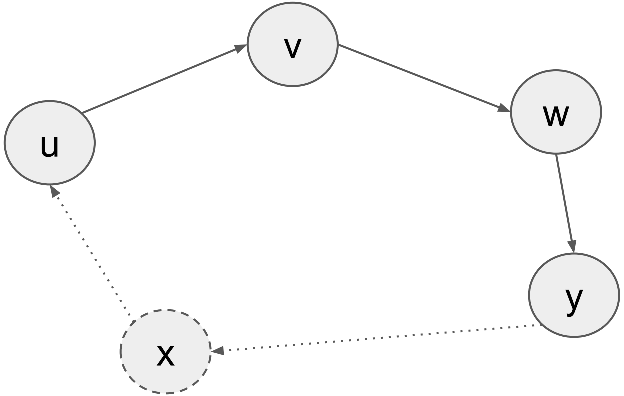
Suppose I delete a vertex (x)

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

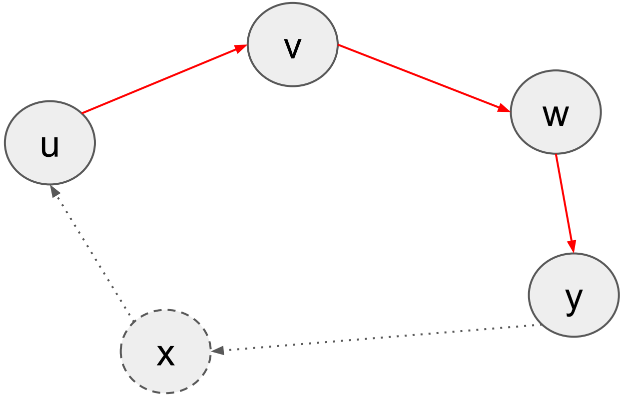
indeg(u) = outdeg(u) - 1

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

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

($\leftarrow$) Suppose indeg(v) = outdeg(v) for all vertices v.

We want to show there is an Euler tour



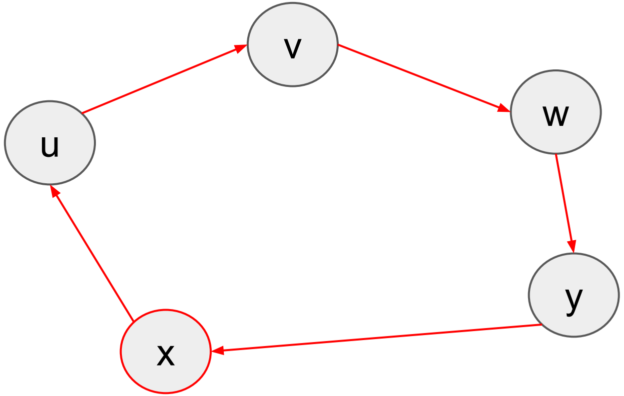
If we instead find an Euler **path** from u $\rightarrow$ y,

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

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

($\leftarrow$) 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 $\rightarrow$ 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

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

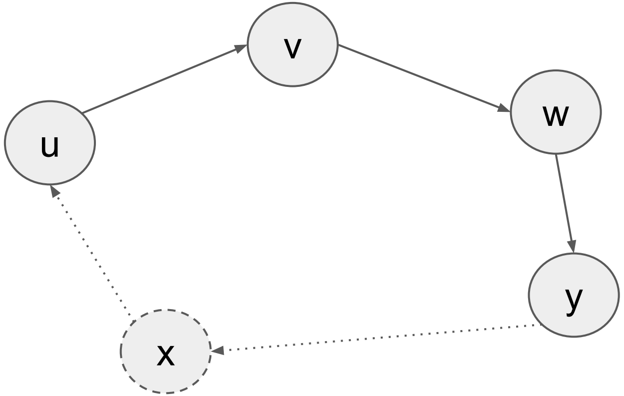
(←) Suppose indeg(v) = outdeg(v) for all vertices v.

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 tour 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

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

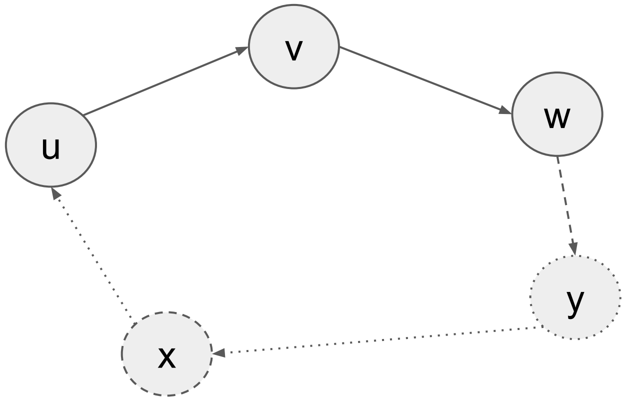
(←) Suppose indeg(v) = outdeg(v) for all vertices v.

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



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

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

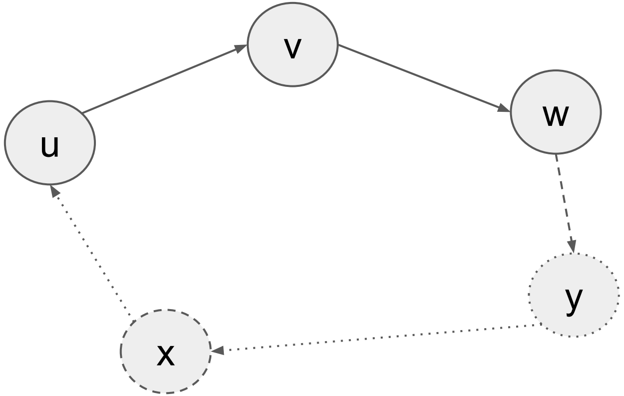
(←) Suppose indeg(v) = outdeg(v) for all vertices v.

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

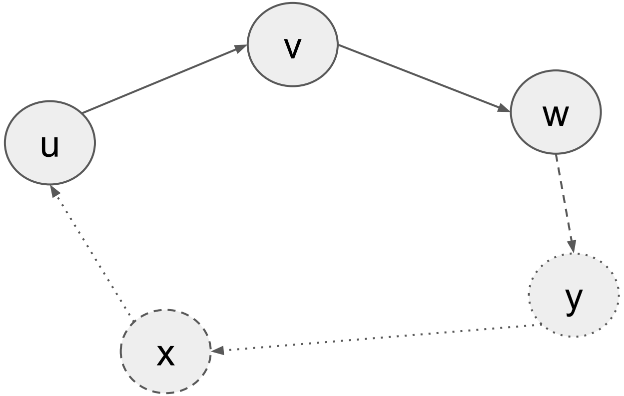
indeg(u) = outdeg(u) - 1

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

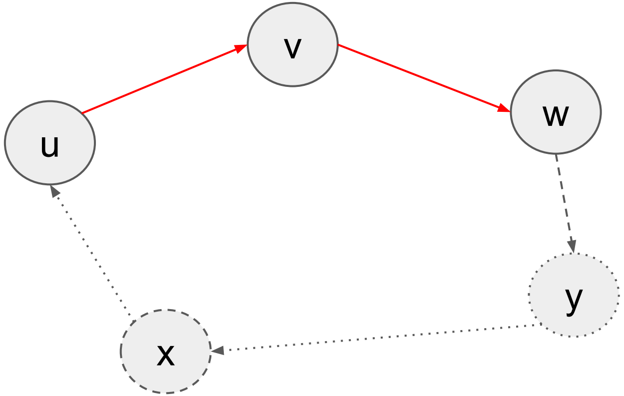
This new graph (deleted y) shares the same structure as the previous graph.. We can induct on the number of edges!

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

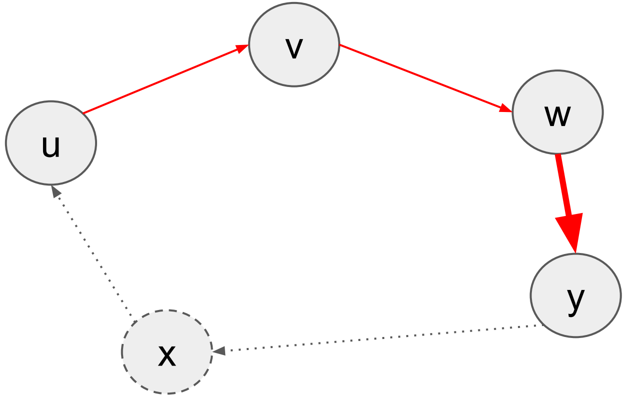
By Induction there is an Euler path from u → w

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

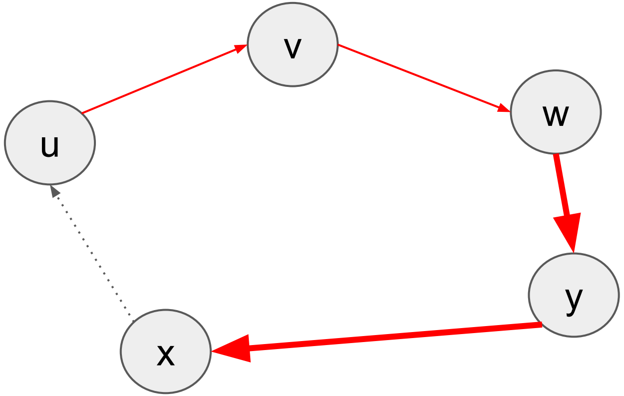
Add back y

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

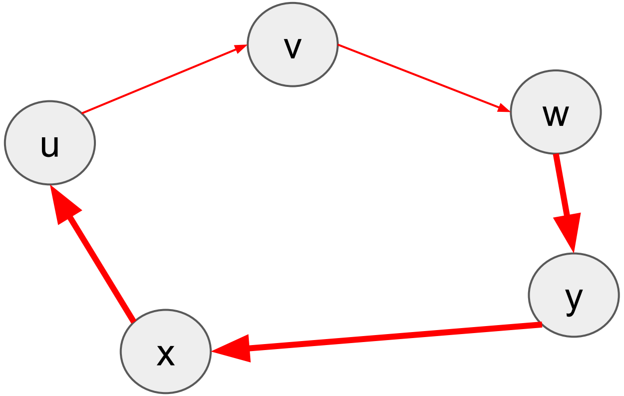
Add back x

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!