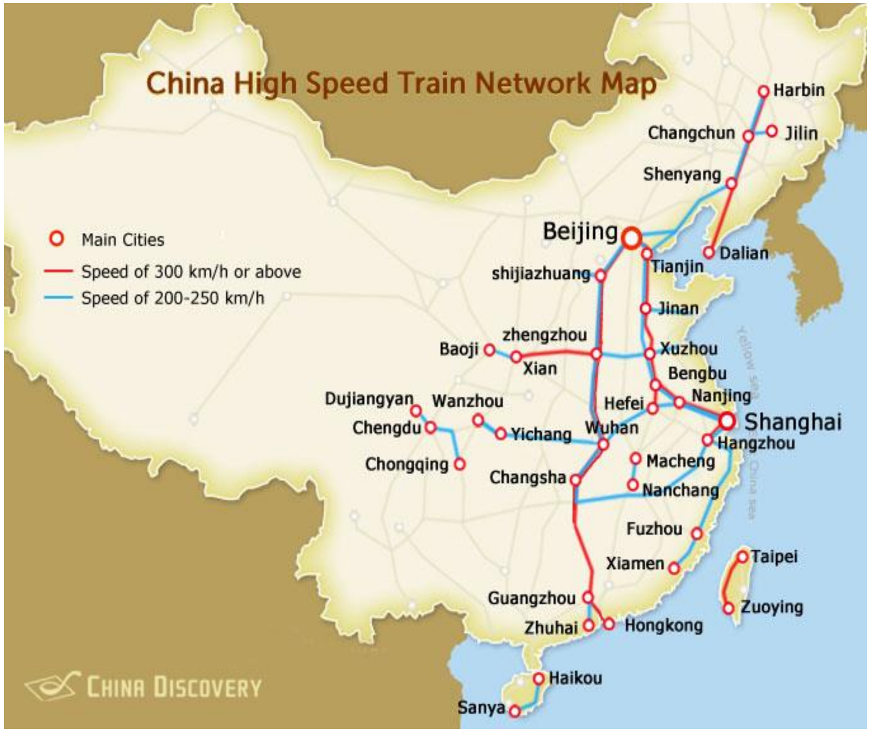
Lecture 9

Graphs

The Speed Train Network

- If you look at a map of the china high speed train network map there are **three components**:
  - **main cities**.
  - **Routes speed between 200-250km/h**
  - **Routes speed 300km/h or above**.



Flight routes

- If you look at where Mandarin Airlines flies to:
  - there is a map of **airports**;
  - and a **route** between airports.

This lecture

- I want to **illustrate how to solve** all kinds of problems using graphs.
- I will **not always give the code** for the algorithms, but I expect you to study the general idea.
- If you continue a degree in Computer Science, **you'll learn all the algorithms** I describe today in greater detail.

Variations

- There are many variations of this simple principle:
  - Does the **direction** of edges matter? (directed or undirected)
  - Is there a “**cost**” assigned with every edge? (weighted or unweighted)
  - Does the graph have a **special shape**?

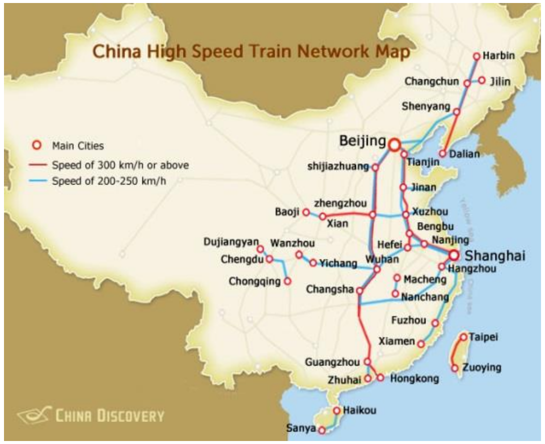
  (bipartite; complete; strongly connected; etc.)
- I’ll try to cover a few of these.

Why graphs?

- Problems involving graphs are very common in **Computer Science**:
  - How should I **drive** from Nottingham to Shanghai?
  - How should I **schedule** who manages my store?
  - How should I **arrange** these electrical components to minimize how much wire I need to connect them?

Graphs

- This non linear data structure pops up again and again in **Computer Science**.

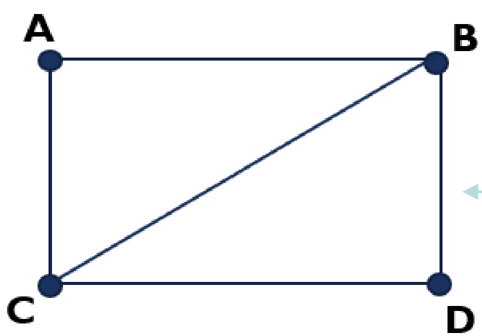


- A **graph** consists of
  - a collection of **vertices *V*** (the points");
  - a collection of **edges *E*** (the "lines") between two vertices.

Graph basics

A graph is a non-linear data structure. It is also a mathematical concept.

Graph is a collection of Vertices and Edges. $G = (V, E)$



Vertex B

Edge BD (or DB)

Vertices: A, B, C, D

Edges: AB, CD, AC, BD, BC

Degree of a vertex: the number of edges joining each vertex

degree(A)=2 degree(C)=3

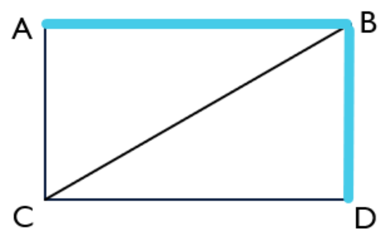
	A	B	C	D	degree
A	0	1	1	0	2
B	1	0	1	1	3
C	1	1	0	1	3
D	0	1	1	0	2

(a matrix representation)

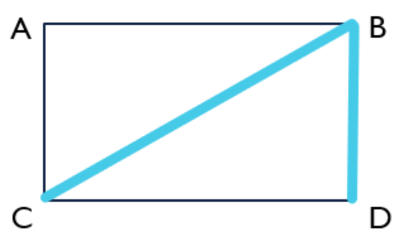
Path

A path on a graph is represented by a set of edges that joins a set of (distinct) vertices.

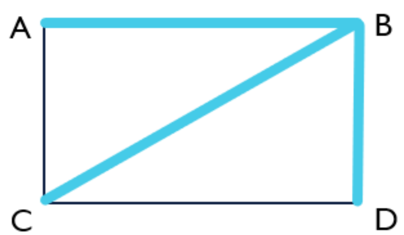
Examples:



Path: A-B-D (or using edges: AB-BD)



Path: C-B-D



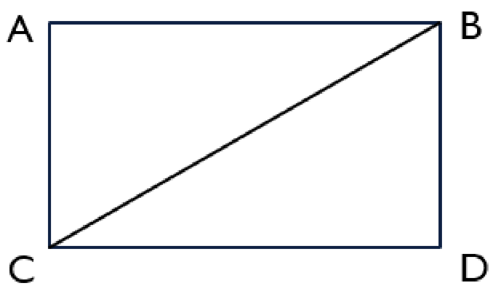
A-B-C-B-D Not a path!

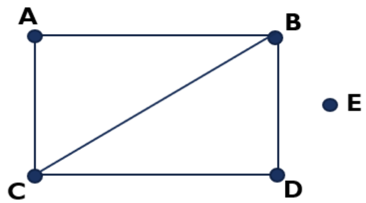
Edge BC is visited twice.

Cycle and Connected graph

A cycle is a path where it ends at the same starting vertex.

Examples:

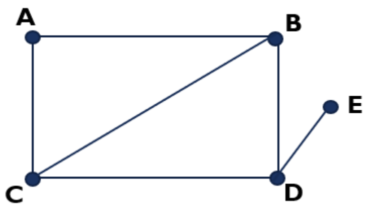




(not a connected graph)

The path A-B-C-A is also a cycle.

(A-C-B-A, B-C-A-B are considered as equivalent cycle here)



(connected graph)

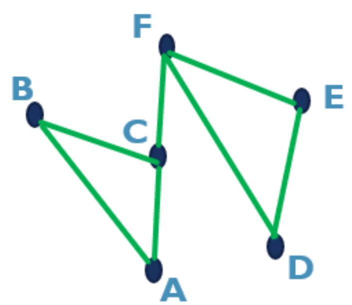
Other cycles: B-C-D-B A-B-D-C-A

If there is a path between any two vertices, the graph is said to be a connected graph.

Tree graph

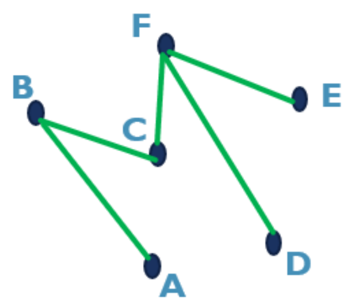
A tree is a connected graph with no cycles,

in which any two vertices are connected by exactly one path.



graph with cycles

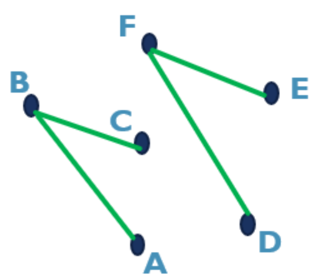
connected, not a tree



graph with no cycle

connected, also a tree

(any vertex can be root node)



graph with no cycles

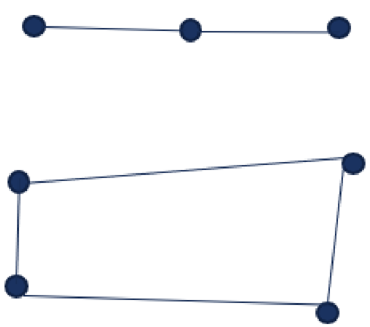
not connected, not a tree

(it is a forest)

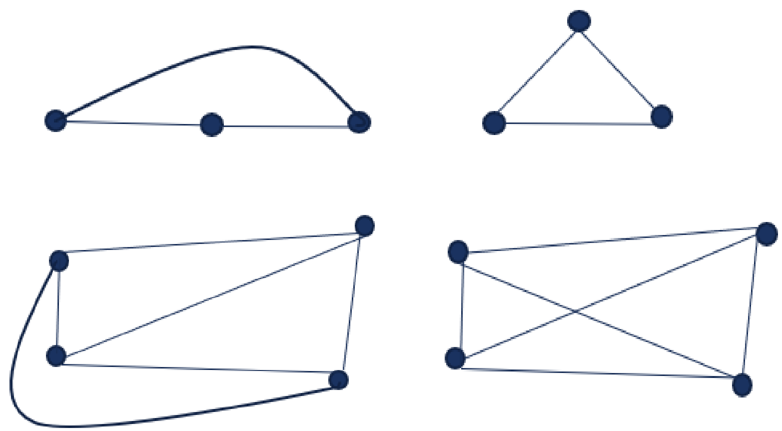
Complete graph

If there any two vertices are connected by a unique edge, the graph is said to be a complete graph.

Not complete



Complete



Number of Vertices and Edges:

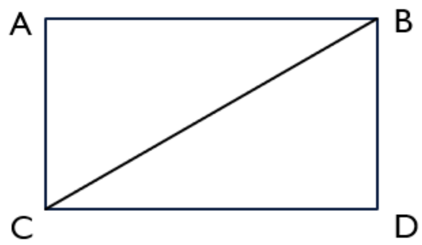
$|V| = 3, |E| = 3$

$|V| = 4, |E| = 6$

For a complete graph with n vertices ($|V| = n$), the total number of edges:

$$|E| = \binom{n}{2} = \frac{n!}{2!(n-2)!} = \frac{n(n-1)}{2}$$

Directed graph



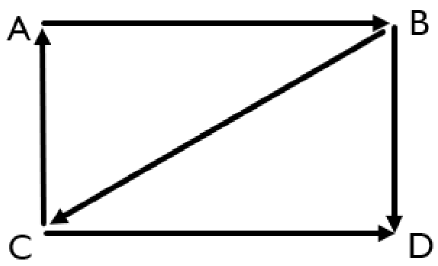
Undirected graph

Edge direction does not matter:

Edge AB (or BA) Path A-B-D (or D-B-A)

Cycle A-B-C-A (or A-C-B-A)

3 different cycles in the example



Directed graph

Edge direction does matter:

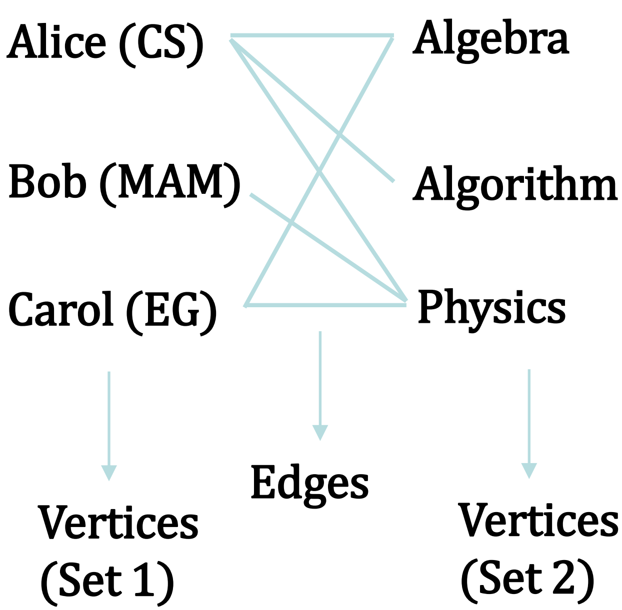
Edge AB Path A-B-D

Cycle A-B-C-A

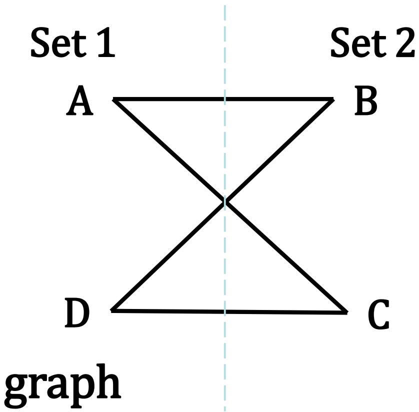
only 1 cycle in the example

Bipartite graph

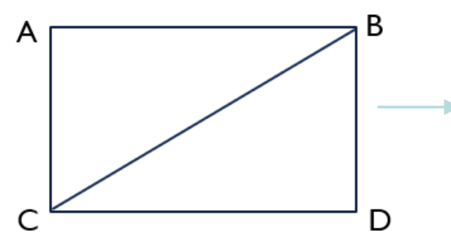
A bipartite graph is a graph in which the vertex set can be partitioned into two sets, so that there is no edge joining vertices within each set.



(bipartite graph application)



Bipartite graph



Not a bipartite graph

If graph is bipartite we can color all vertex of the graph using two colours .
A graph with a triangular cycle cannot be bipartite graph.

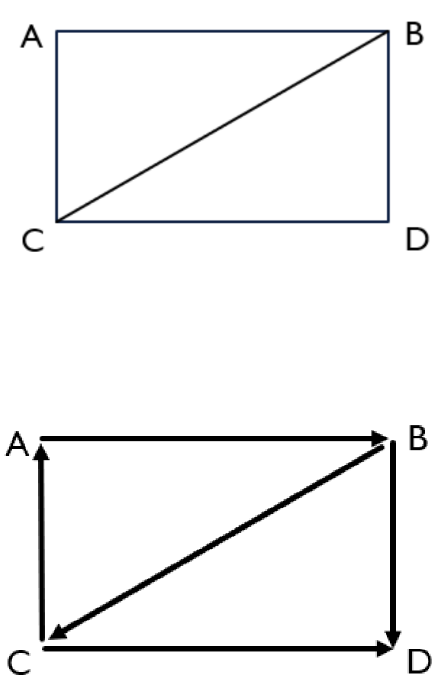
Simple scheduling

- Suppose I run a sandwich shop. I need to hire people to run my shop.
- I need **exactly one person for every day** of the week. The following people apply:
  - Anna can work on Mon, Wed and Fri;
  - Lucy can work on Mon, Tue and Thur;
  - Mark can work on Mon and in weekends;
  - Peter can work on Thur and Fri.
- **How many people do I need?**

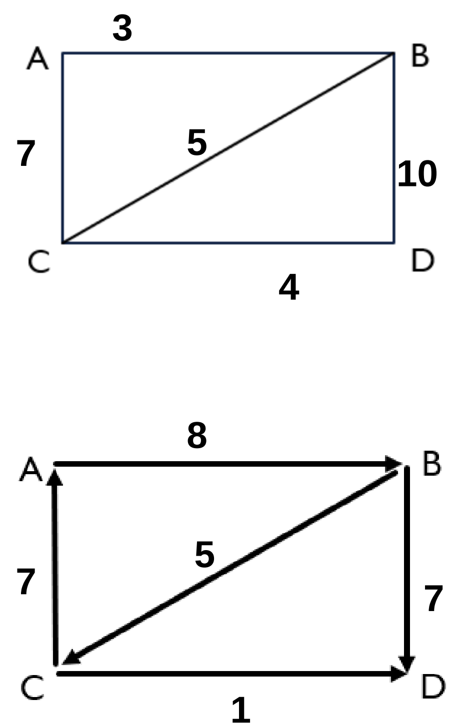
Who should I hire?

Weighted graph

A weighted graph is a graph in which all edges are associated with numerical values (called weights of edges).



Unweighted graphs



Weighted graphs

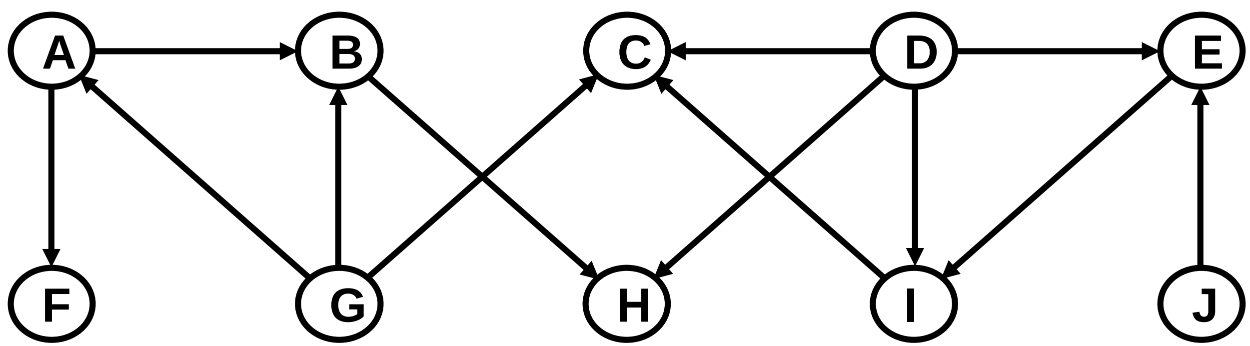
Another scheduling problem

- Quick! I need to get dressed. I need to wear:
  - **my shoes;**
  - **my tie;**
  - **my boxer shorts;**
  - **my shirt;**
  - **my jacket;**
  - **my trousers;**
  - **my belt;**
  - **my watch;**
  - **my glasses;**
  - **my socks.**

- **Constraints, for example:**
  - **Obviously, I can't put on my shoes before my socks.**
- **In what order should I put on all my clothes?**

Topological sort

- A **topological sort** of a graph with vertices ***V***
  - arranges the vertices in a line, such that all the edges point from the left to the right.



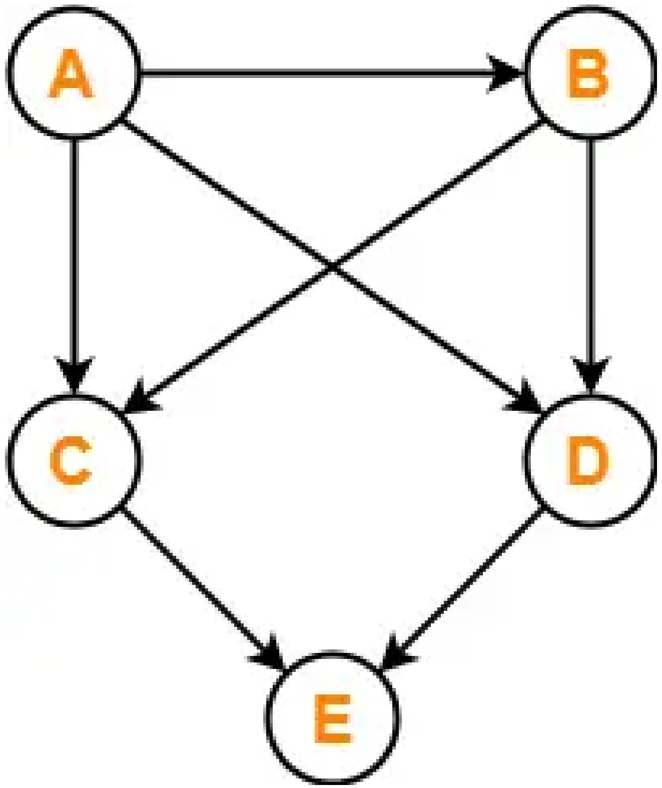
D	G	A	B	F	H	J	E	I	C

Procedure topological sort

  - draw the graph with all the edges coloured black;
  - start with all the nodes that have **no** **black incoming edges**;
- **repeat** until you have no nodes left:
  - colour the **edges going out** from these nodes blue;
  - look for new nodes with **no** incoming black edges.

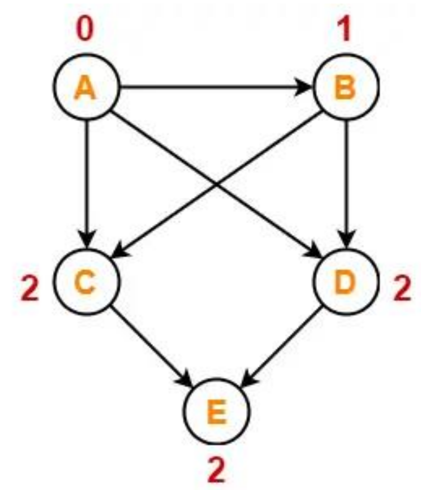
Topological Sorting

Find the number of different topological orderings possible for the given graph-



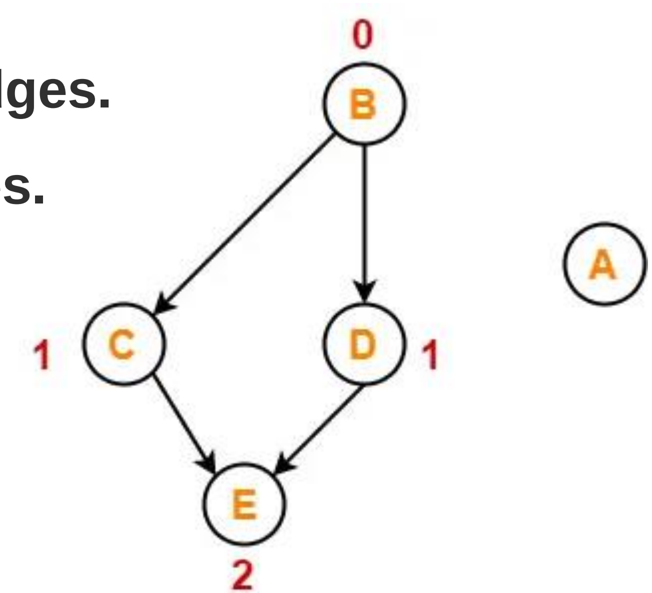
Step 1:

- Write in-degree of each vertex in given graph.



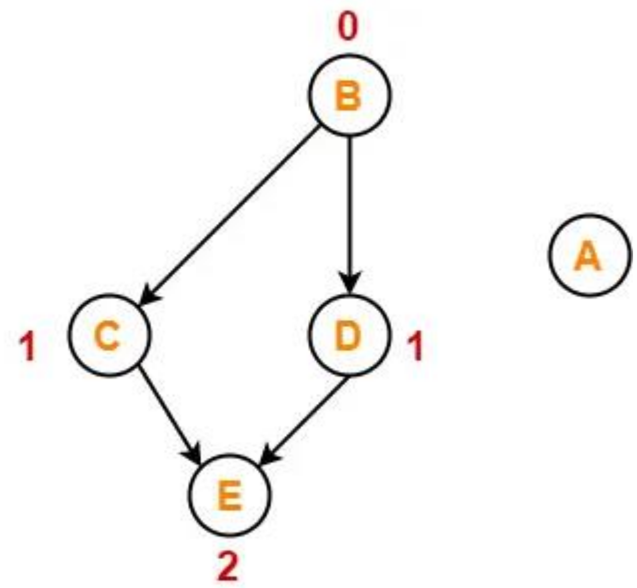
Step-2:

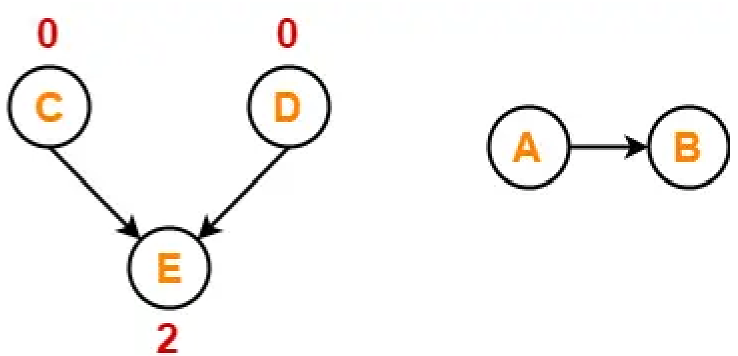
- Vertex-A has the least in-degree.
- So, remove vertex-A and its associated edges.
- Now, update the in-degree of other vertices.



Step-03:

- Vertex-B has the least in-degree.
- So, remove vertex-B and its associated edges.
- Now, update the in-degree of other vertices.



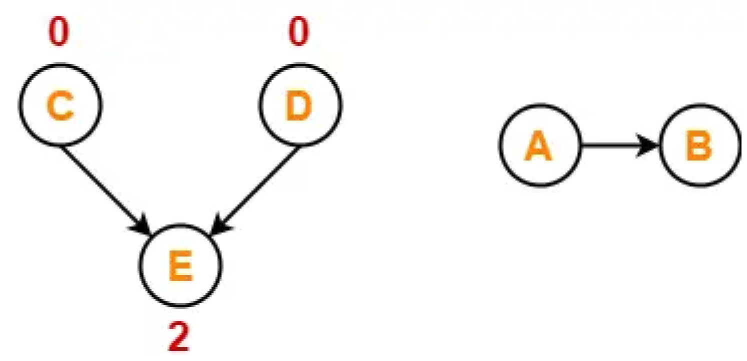


Step-04:

There are two vertices with the least in-degree. So, following 2 cases are possible-

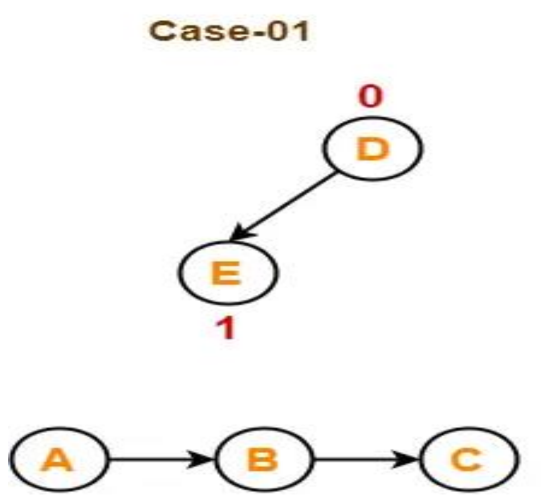
In case-01,

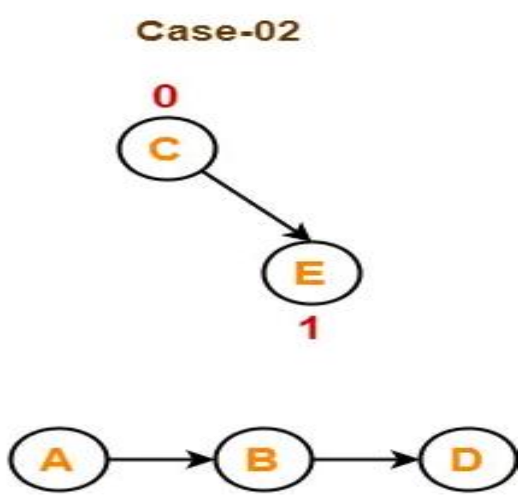
- **Remove vertex-C and its associated edges.**
- **Then, update the in-degree of other vertices.**



In case-02,

- **Remove vertex-D and its associated edges.**
- **Then, update the in-degree of other vertices.**



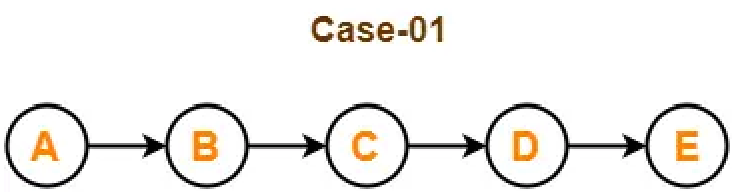


Step-05:

Now, the above two cases are continued separately in the similar manner.

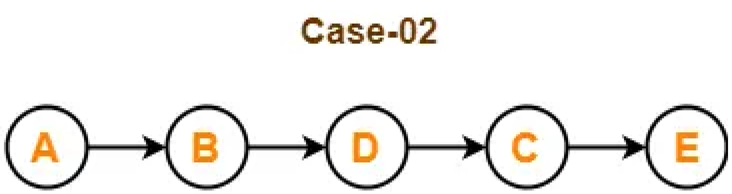
In case-01,

- **Remove vertex-D since it has the least in-degree.**
- **Then, remove the remaining vertex-E.**



In case-02,

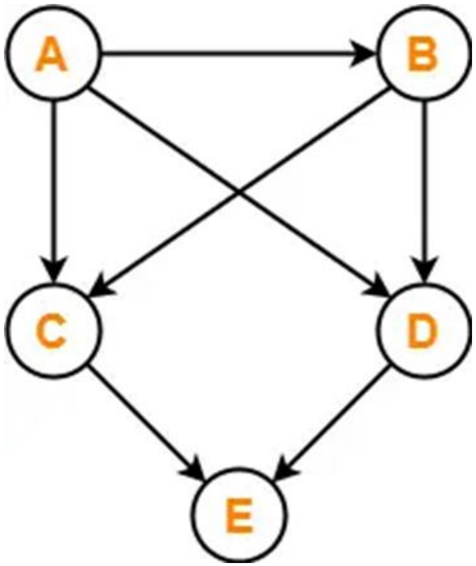
- **Remove vertex-C since it has the least in-degree.**
- **Then, remove the remaining vertex-E.**



Conclusion-

For the given graph, following **2** different topological orderings are possible-

- **A B C D E**
- **A B D C E**



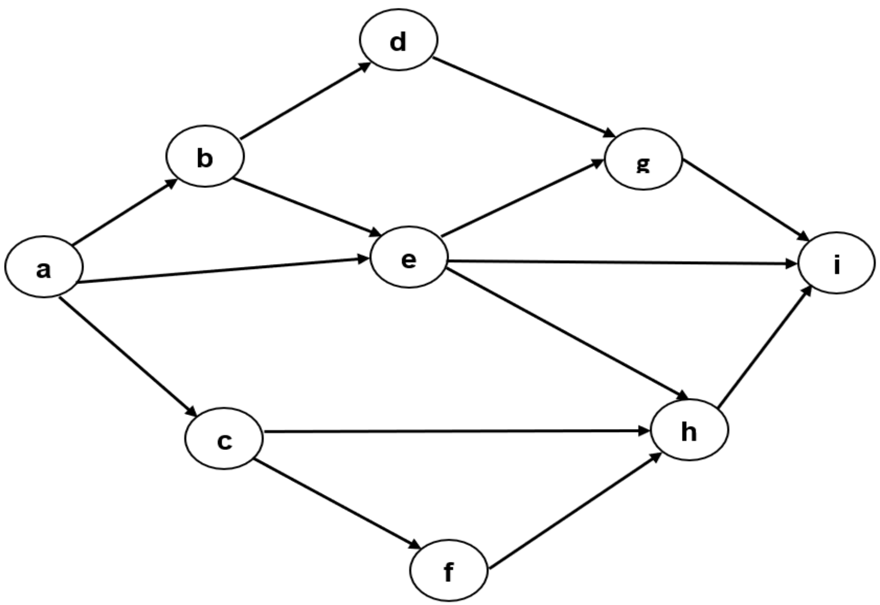
Exercise 9.1

- Draw a graph and use topological sort to give an order in which I should put on all my clothes.
- Quick! I need to get dressed. I need to wear:

  - **my shoes;**
  - **my tie;**
  - **my boxer shorts;**
  - **my shirt;**
  - **my jacket;**
  - **my trousers;**
  - **my belt;**
  - **my watch;**
  - **my glasses;**
  - **my socks.**

Exercise 9.2

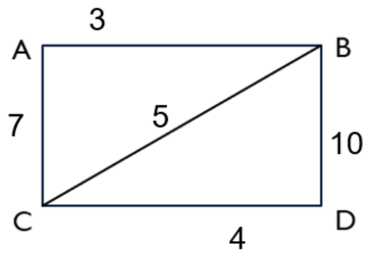
- Topological sort is not unique.
- List at least two orders in which the nodes of the following graph are topologically sorted.



Shortest path

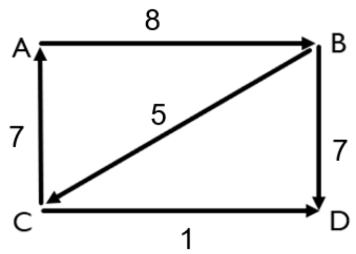
In a weighted graph, the sum of weights of edges in a path is called cost.

We are interested in finding the shortest path (and minimum cost) between any two vertices in a weighted graph.



Paths between A and D	Costs
A-B-D	3+10=13
A-C-D	7+4=11
A-B-C-D	3+5+4=12
A-C-B-D	7+5+10=22

Shortest path between A and D is A-C-D, minimum cost is 11.



Paths between A and D	Costs
A-B-D	8+7=15
A-B-C-D	8+5+1=14

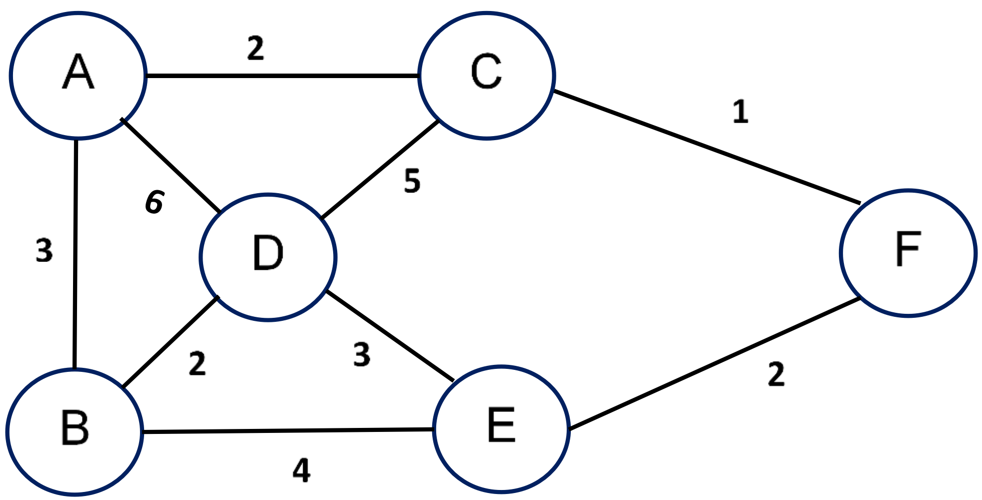
Shortest path between A and D is A-B-C-D, minimum cost is 14.

Dijkstra Algorithm

- Dijkstra Algorithm is a very famous greedy algorithm.
- It is used for solving the single source shortest path problem.
- It computes the shortest path from one particular source node to all other remaining nodes of the graph.

Dijkstra Algorithm

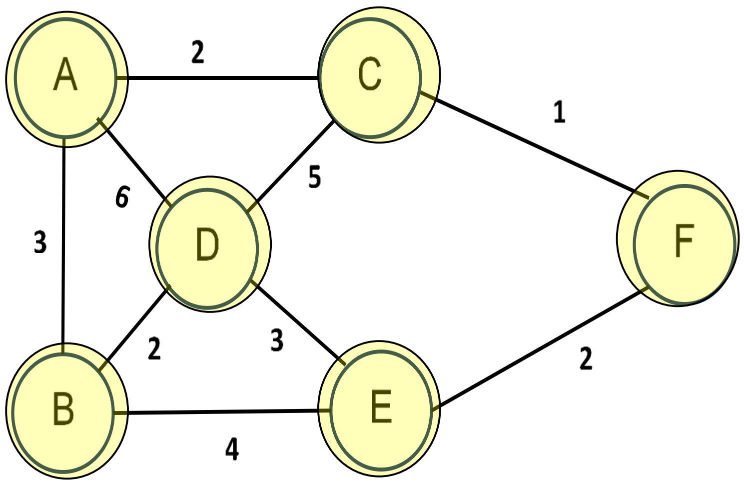
- Tracing Dijkstra's algorithm to find shortest path starting at vertex A.



Dijkstra Algorithm

- Tracing Dijkstra's algorithm to find shortest path starting at vertex A.

	A	B	C	D	E	F
A	$\boxed{0}$	∞	∞	∞	∞	∞
C		3	$\boxed{2}$	6	∞	∞
F		3		6	∞	$\boxed{3}$
B		$\boxed{3}$		6	5	
E				5	$\boxed{5}$	
D				$\boxed{5}$		

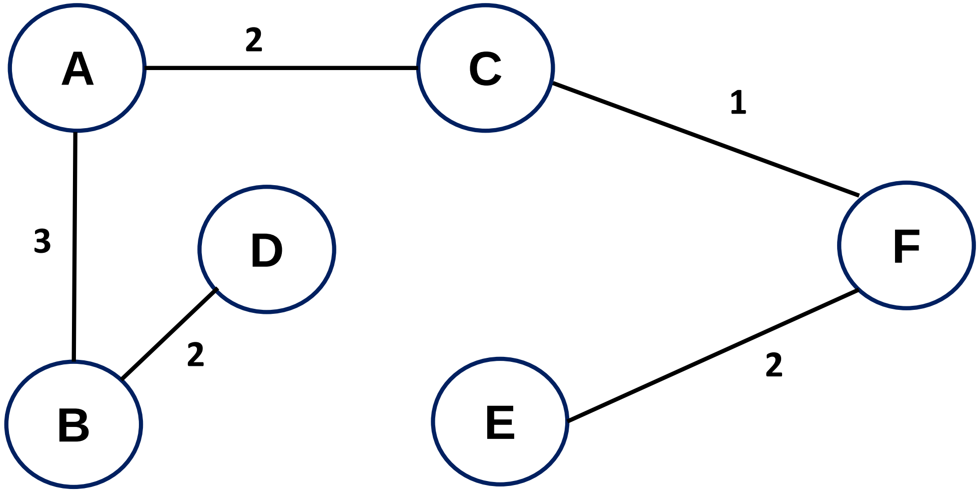


Note :

- We can choose either vertex B or F in third row .
- We can choose either vertex D or E in fifth row.

Dijkstra Algorithm

- The resulting vertex-weighted shortest path graph is



Applications of Graphs :

- Representing networks and routes in communication, transportation and travel applications
- Routes in GPS
- Interconnections in social networks and other network-based applications
- Mapping applications
- Ecommerce applications to present user preferences
- Resource utilization and availability in an organization
- Document link map of a website to display connectivity between pages through hyperlinks
- Robotic motion and neural networks

Review Quiz

Q1: What is the maximum number of edges in a bipartite graph having 10 vertices?

a) 24

b) 21

c) 25

d) 16

Q2 : Which of the following statements about topological sorting is true?

a) Topological sorting can be applied to any graph.

b) Topological sorting is used for finding the shortest path in a graph.

c) Topological sorting is only applicable to Directed Acyclic Graphs (DAGs).

d) Topological sorting is used to detect cycles in a graph.