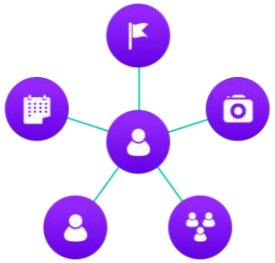
TODO BEFORE

To Read Before #ToBeReady

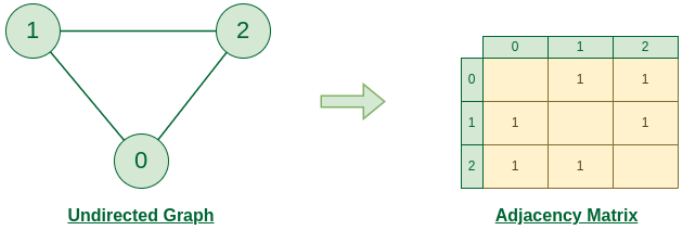
https://github.com/aish21/Algorithms-and-Data-Structures

Great doc covering All ADTS

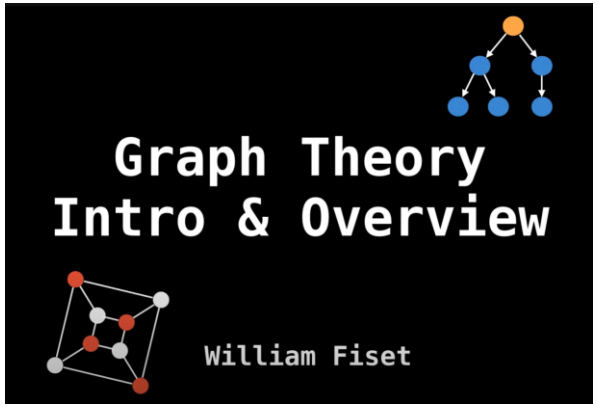
✓ Graph Data Structure



✓ Graph and its representations

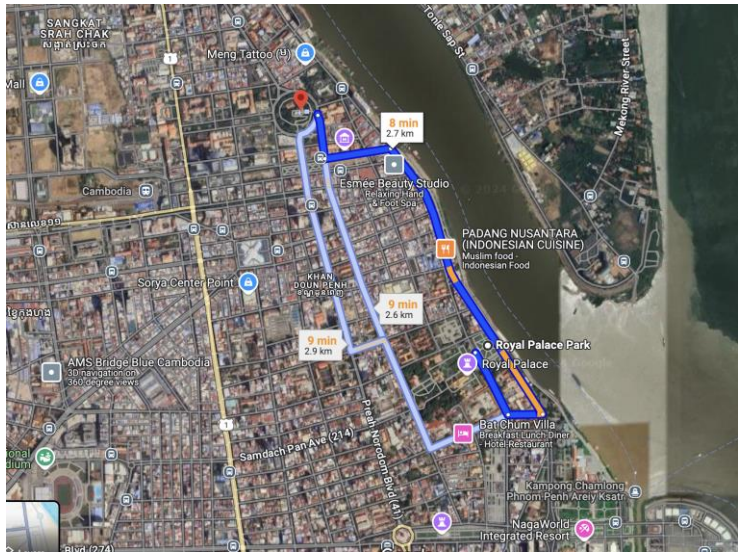


✓ Graph Theory Introduction



ADVANCED ALGORITHM

W10-S1 – Graph



CADT IDT

Objectives for today

- **Understand** the concept of Graph
- **Explore** Graph Terminology
- **Identify** the type of Graph
- **Know** how to represent the Graph

EXPLAIN

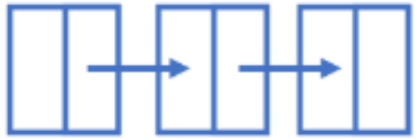
Abstract Data Structures

Linear

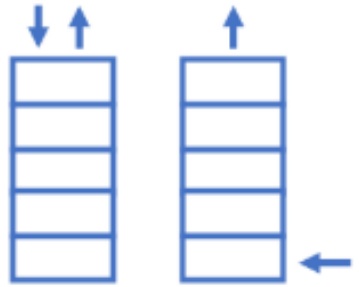
Data elements are arranged **sequentially**

Array

Linked List



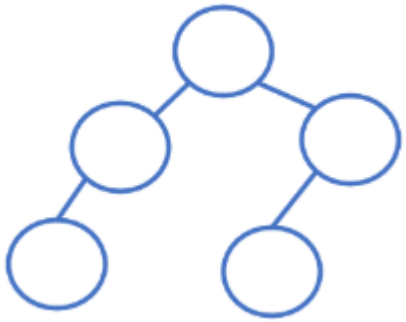
Stack & Queue



Non-Linear

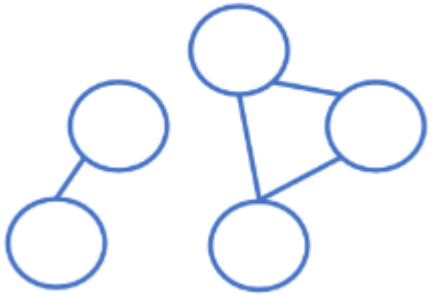
Data elements are **not** arranged sequentially

Tree



Every node can have
1 parent only

Graph

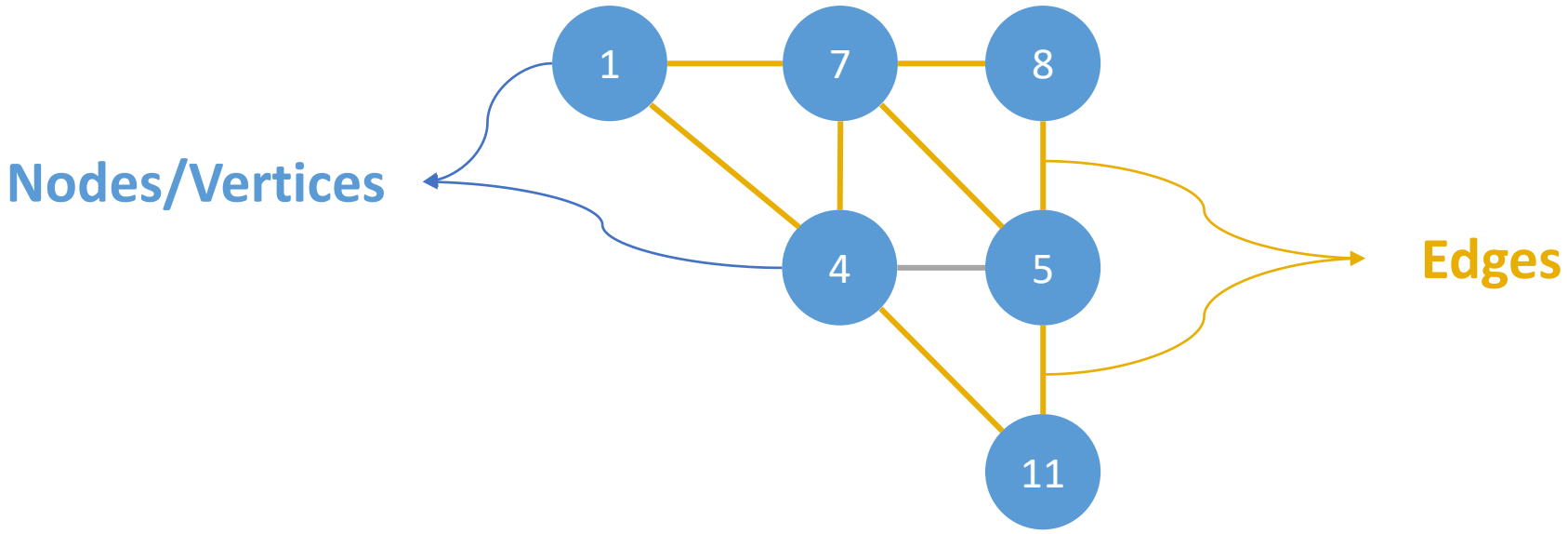


There is not rules
for the connection of
nodes

EXPLAIN

Graph **Definition**

A **graph** is a non-linear data structure consisting of two main components—**vertices** and **edges**—used to represent relationships or connections between objects.



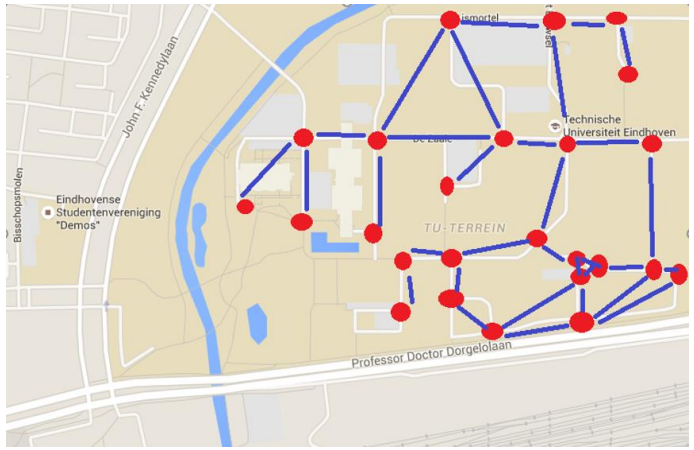
EXPLAIN

Why **Graph** ?

A graph can store information that naturally forms a relationship or connection between entities.

Social Network of Friend

Google Maps and GPS Navigation

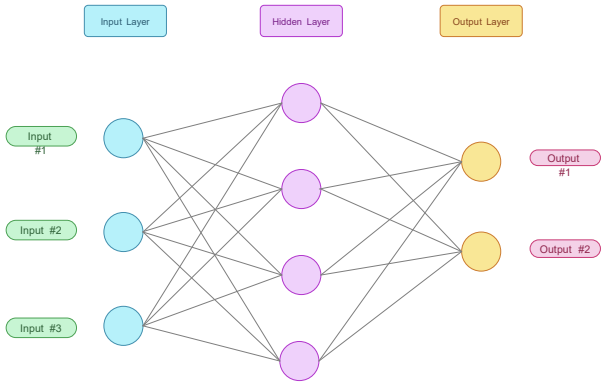


EXPLAIN

Why **Graph** ?

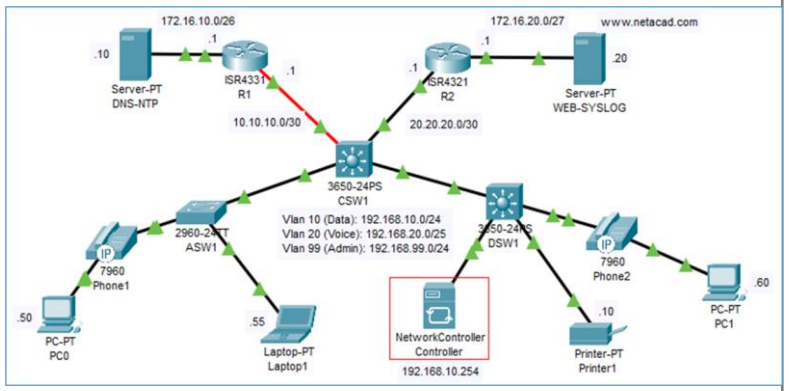
A graph can store information that naturally forms a relationship or connection between entities.

Neural Network



Flight Routes

Network Management

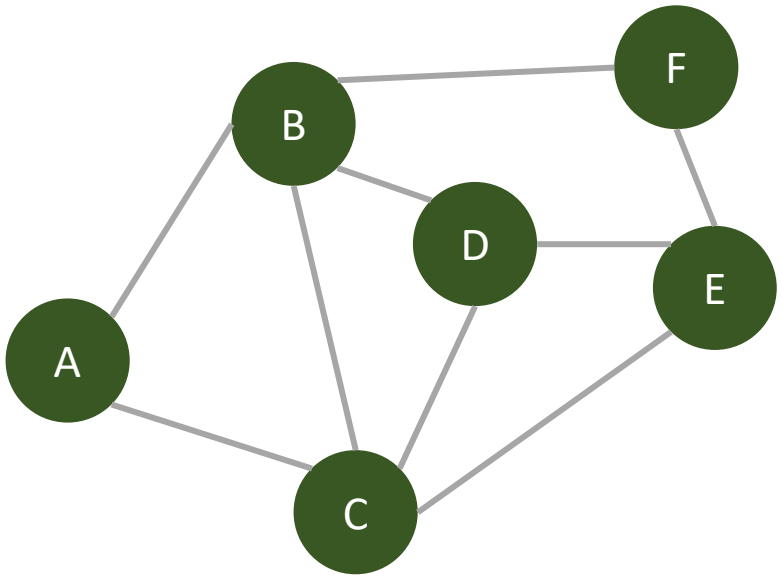


EXPLAIN

Type of Graph

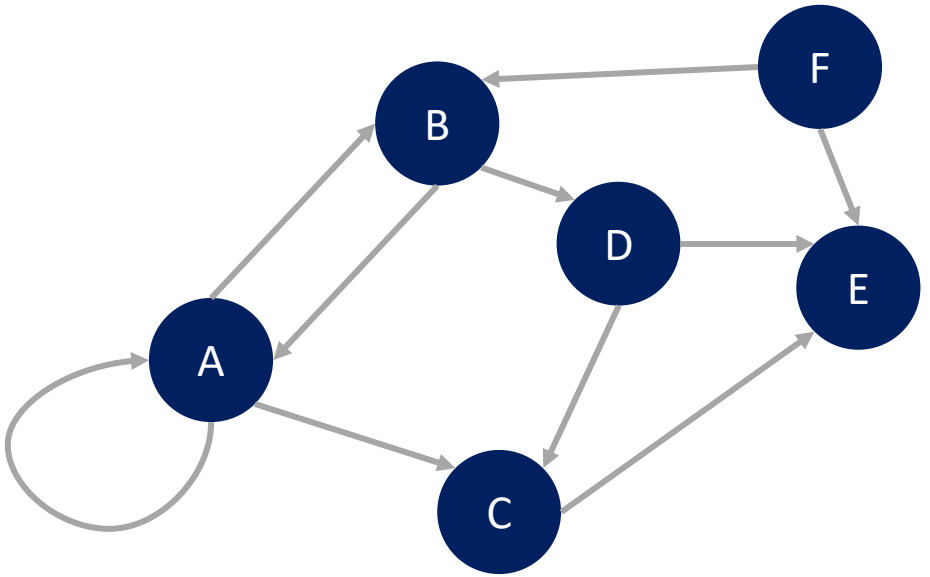
An **Undirected graph** is a graph in which edges have no orientation.

(arrowless connections)



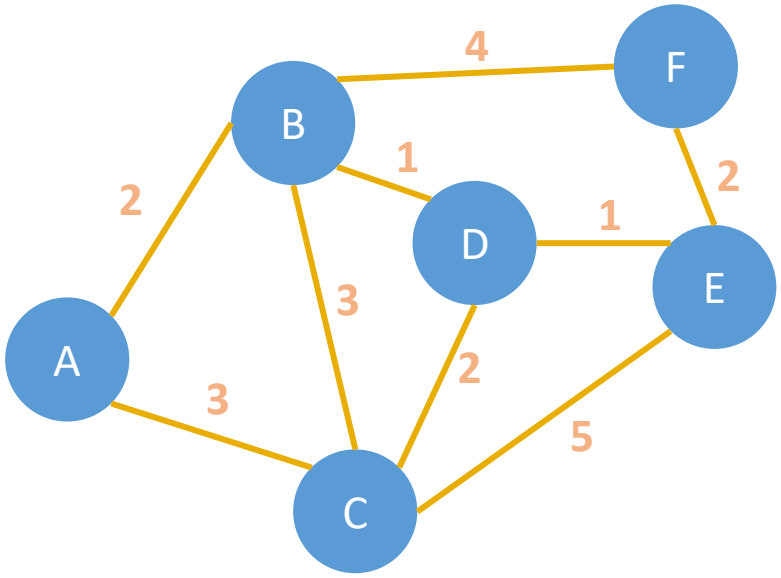
An **Directed graph (Digraph)** is a graph in which edges have orientation.

edges with arrows connect one vertex to another



EXPLAIN

Graph **Terminology**



- Graph can have an edge that contains a certain **weight** to represent a value such as **cost**, **distance**, **quantity**,etc. This is called a **weighted graph**.
  - Example:

  Edge(A, B) = **2**

  Edge(C, B) = **3**

- The **degree of a vertex** is the number of edge that are connected to it.
  - Example:

  deg(C)= **4**

  deg(D)= **3**

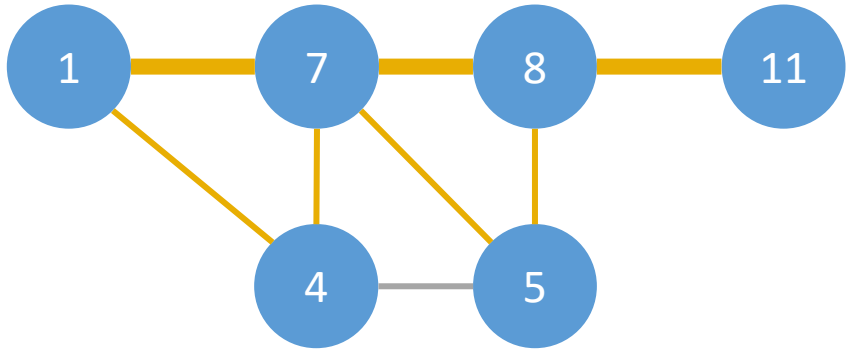
REMARK:

- A **weighted graph** (or known as a network) is a graph in which a number (the weight) is assigned to each edge.
- The weights might represent the costs, lengths or capacities, depending on the problem we are dealing with.
- Such graphs arise in many contexts, for example in **shortest path problems** such as the traveling salesman problem.

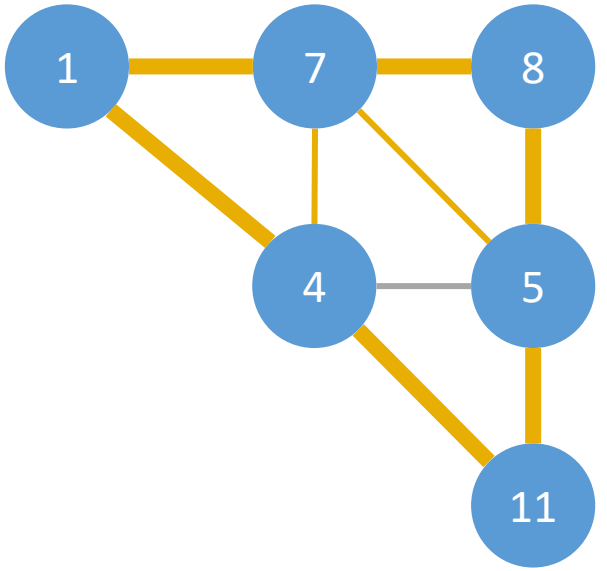
EXPLAIN

Graph **Terminology**

A **path** is a sequence of vertices connected by edges with no repeated edges.



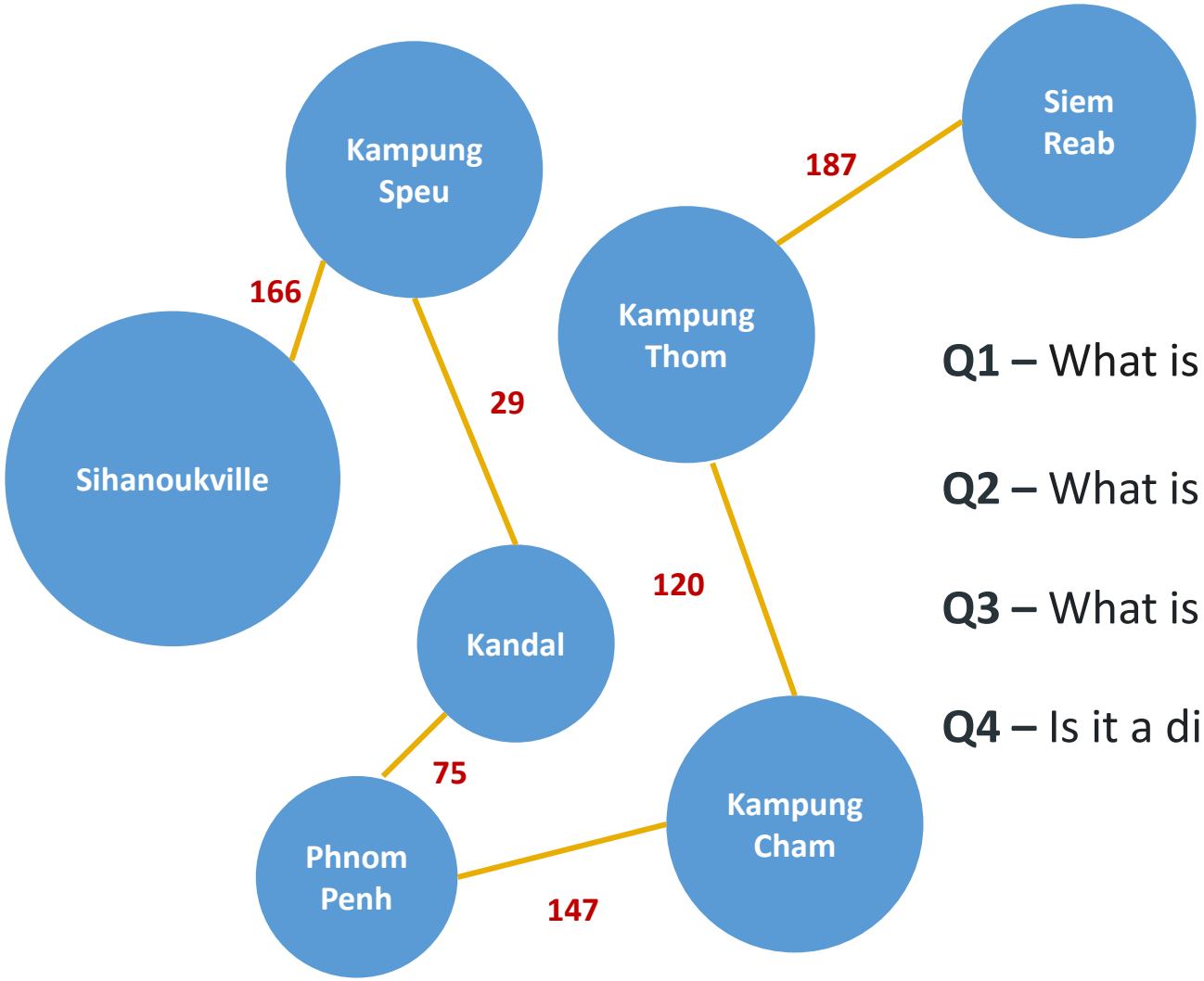
Cycle is a **path** where the start and end vertex are the same.



PRACTICE

Graph

Look at this tree and answer the questions



Q1 – What is the total number of vertices and edges in the graph?

Q2 – What is the weight of **Edge**(**Phnom Penh, Kandal**)?

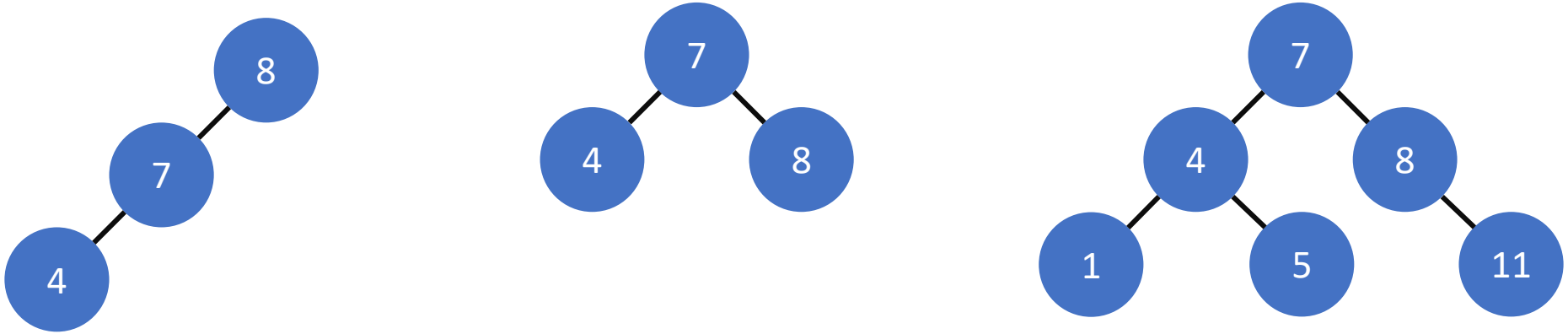
Q3 – What is the **degree of a vertex Kandal?**

Q4 – Is it a directed graph or undirected graph?

EXPLAIN

Special Graph

A **Tree** is an **Undirected graph** with no cycles.
It's a connected graph with **N** vertices and **N-1** edges



EXPLAIN

Graph **Representation**

A graph ADT can be represented by:

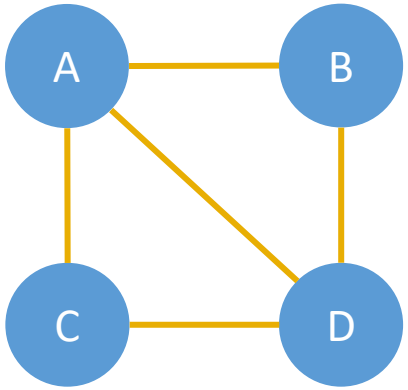
- An adjacency matrix (using 2D array)
- An adjacency list (using linkedlist)
- Edge list (triplet)

EXPLAIN

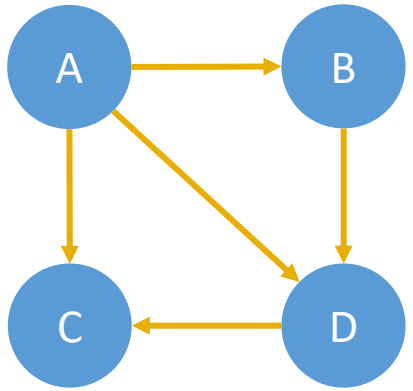
Graph **with an adjacency matrix**

An **Undirected graph** & **Directed graph** can be represented by an adjacency matrix (AM):

- If there is an edge between vertex i and vertex j, **AM[i][j] = 1**
- If there is no edge between vertex i and vertex j, **AM[i][j] = 0**



	A	B	C	D
A	0	1	1	1
B	1	0	0	1
C	1	0	0	1
D	1	1	1	0



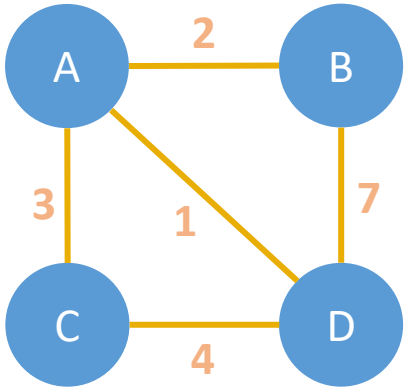
	A	B	C	D
A	0	1	1	1
B	0	0	0	1
C	0	0	0	0
D	0	0	1	0

EXPLAIN

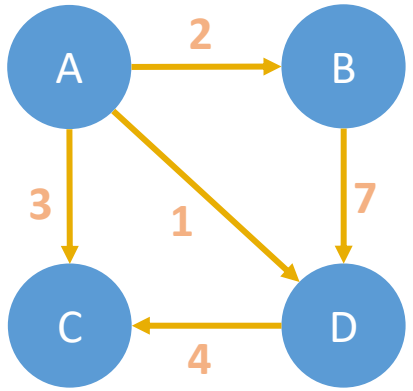
Graph **with an adjacency matrix**

Similarly, the weighted graph can be represented by an adjacency matrix (AM):

- If there is an edge between vertex i and vertex j, **AM[i][j]= w**
- If there is no edge between vertex i and vertex j, **AM[i][j]= 0**



	A	B	C	D
A	0	2	3	1
B	2	0	0	7
C	3	0	0	4
D	1	7	4	0



	A	B	C	D
A	0	2	3	1
B	0	0	0	7
C	0	0	0	0
D	0	0	4	0

EXPLAIN

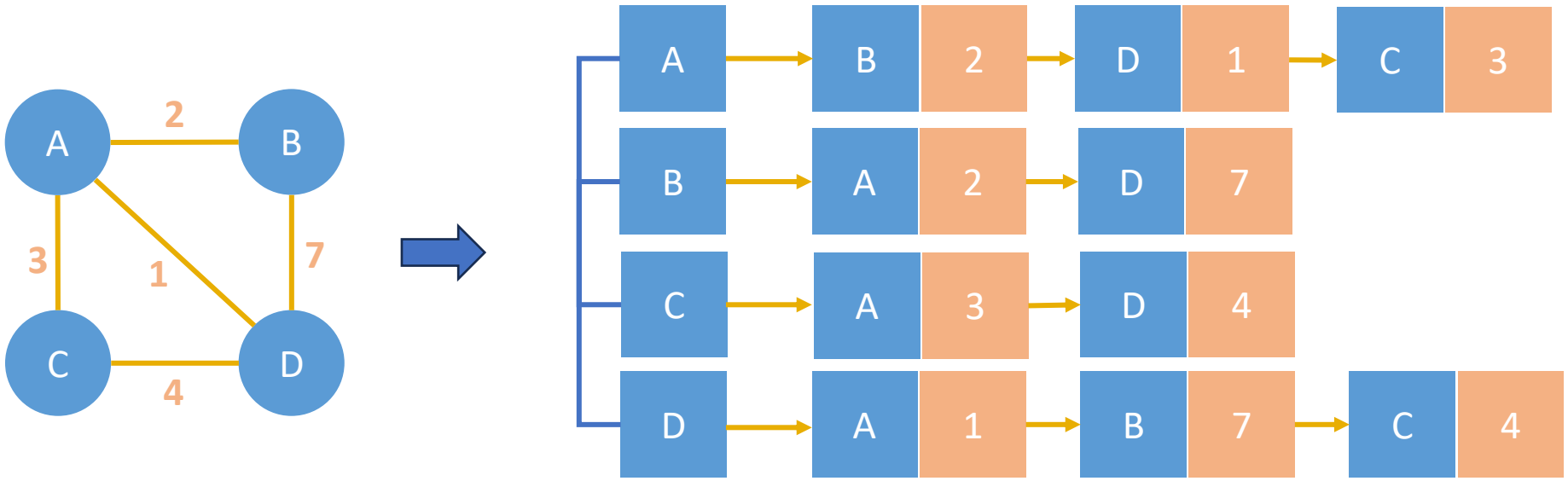
Graph **with an adjacency matrix**

Pros	Cons
Space efficiency for represent dense graph	Requires **O(V^2)** space
Edge weight lookup is O(1)	Iterating over all edges takes **O(V^2)** time
Simplest graph representaion	

EXPLAIN

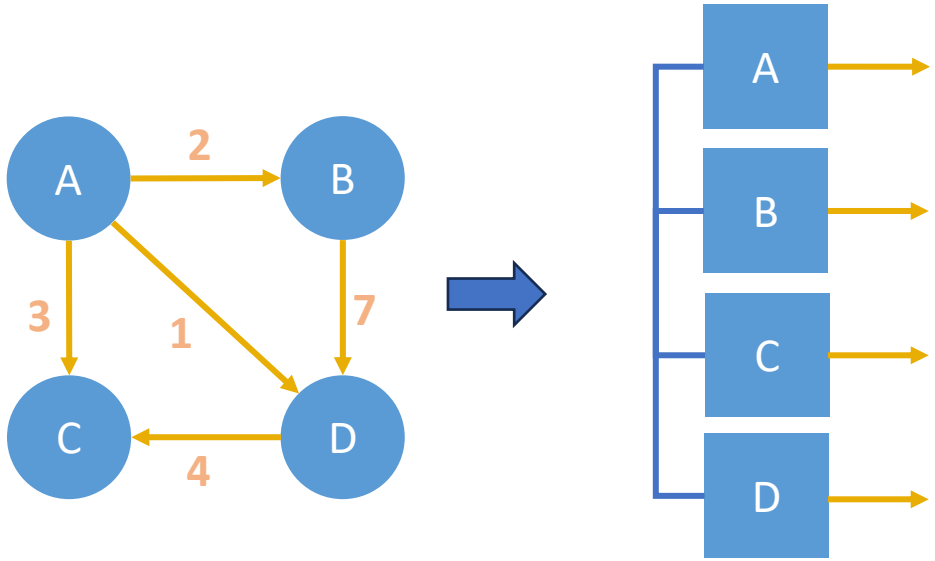
Graph **with an adjacency list**

An **adjacency list** is a way to represent a **graph** as a map from vertices to lists of edges.



PRACTICE

Represent a directed graph with **adjacency list**



EXPLAIN

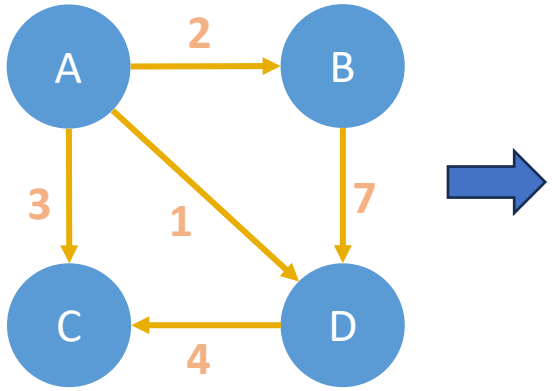
Graph **with an adjacency list**

Pros	Cons
Space efficiency sparse graph	Less space efficient for dense graph
Iterating over all edge is efficient	Edge weight lookup is **O(E)**
	Slightly more complex graph representation

EXPLAIN

Graph **with edge list**

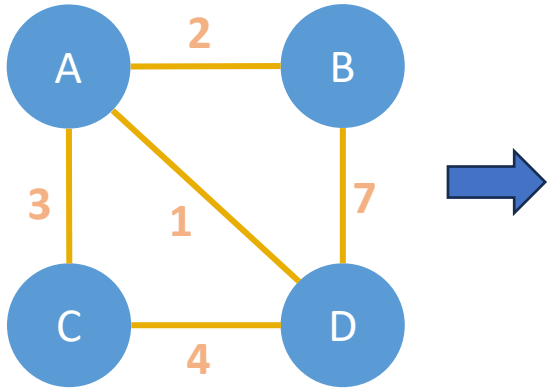
An **edge list** is a way to represent a graph simply as an unordered list of edges.
A triplet (u, v, w) is used to present each connection,




```
[(A,B,2), (A,C,3),
(A,D,1), (B,D,7),
(D,C,4)]
```

PRACTICE

Represent an undirected graph with **edge list**




```
[(),
 (),
 ()…
]
```

EXPLAIN

Graph **with edge list**

Pros	Cons
Space efficiency sparse graph	Less space efficient for denser graph
Iterating over all edge is efficient	Edge weight lookup is **O(E)**
Very simple structure	

3-2-1 Challenge

- ✓ List three things you **learned** today.
- ✓ List two **questions** you still have.
- ✓ List one aspect of the lesson or topic you **enjoyed**.