Data structure - Graph

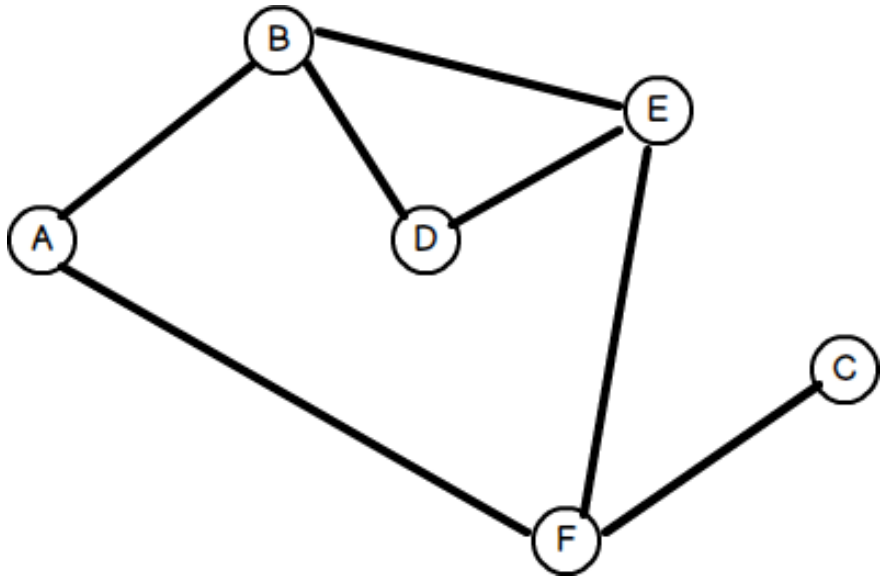
Learning Objectives

In this session, we will learn:

- What are graphs?
- Representation of graphs.
- Implement adjacency list representation of a graph.

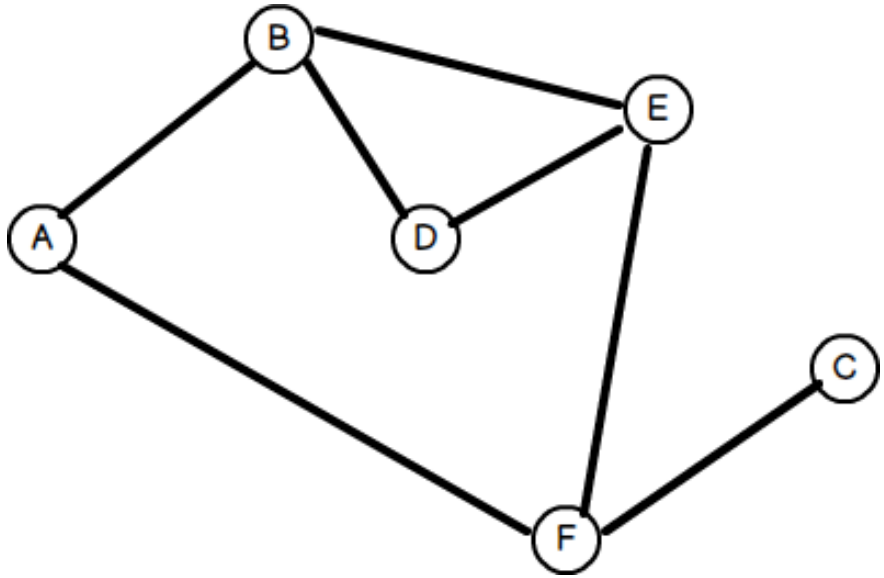
What is a Graph

- A graph can be defined as collection of finite number of vertices and edges.
- Graph is a non linear data structure.



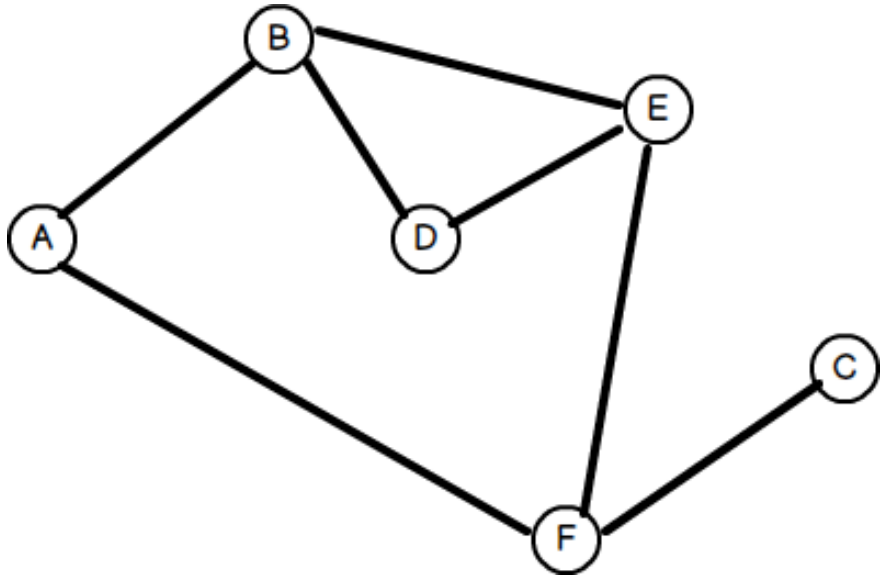
Vertex

- A vertex, a.k.a node in a graph.
- It represents information such as name or identifier of a node in a graph.



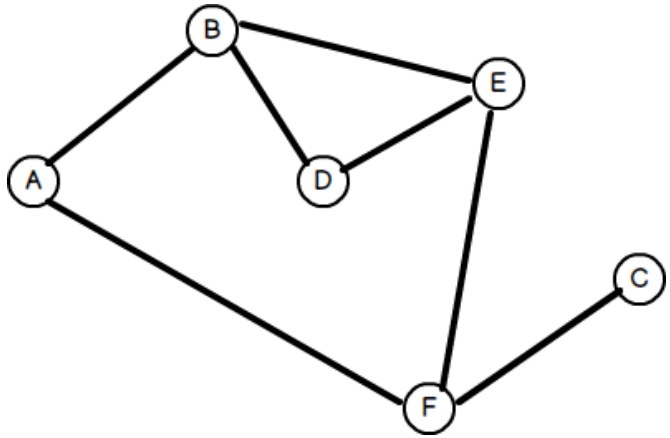
Edge

- An edge is a line connecting two adjacent vertices.



Graph

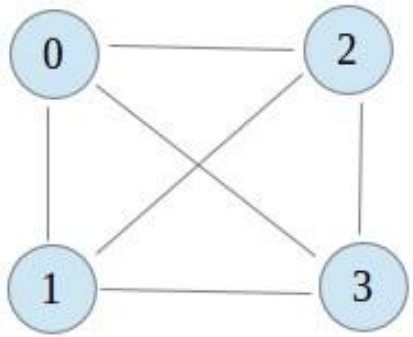
Figure: G



$$G = (V, E)$$
$$V(G) = \{A, B, C, D, E, F\}$$
$$E(G) = \{(A, B), (B, D), (B, E), (A, F), (F, E), (C, F)\}$$

Complete graph

Figure: G



In a given graph G, $\exists(n-1)$ adjacent vertices $\forall G_v$, where n in number of vertices in G.

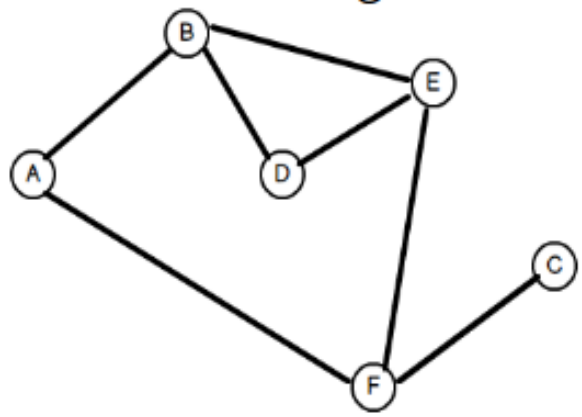
Adjacency Matrix

- G can be represented as square Matrix M of values 0^s and 1^s.
- $M_{ij} = 1, \exists\ E_{(v_i, v_j)}$, else $M_{ij} = 0$.
- $M_i \& M_j$ represents $V(G)$.

M

	A	B	C	D	E	F
A	0	1	0	0	0	1
B	1	0	0	1	1	0
C	0	0	0	0	0	1
D	0	1	0	0	1	0
E	0	1	0	1	0	1
F	1	0	1	0	1	0

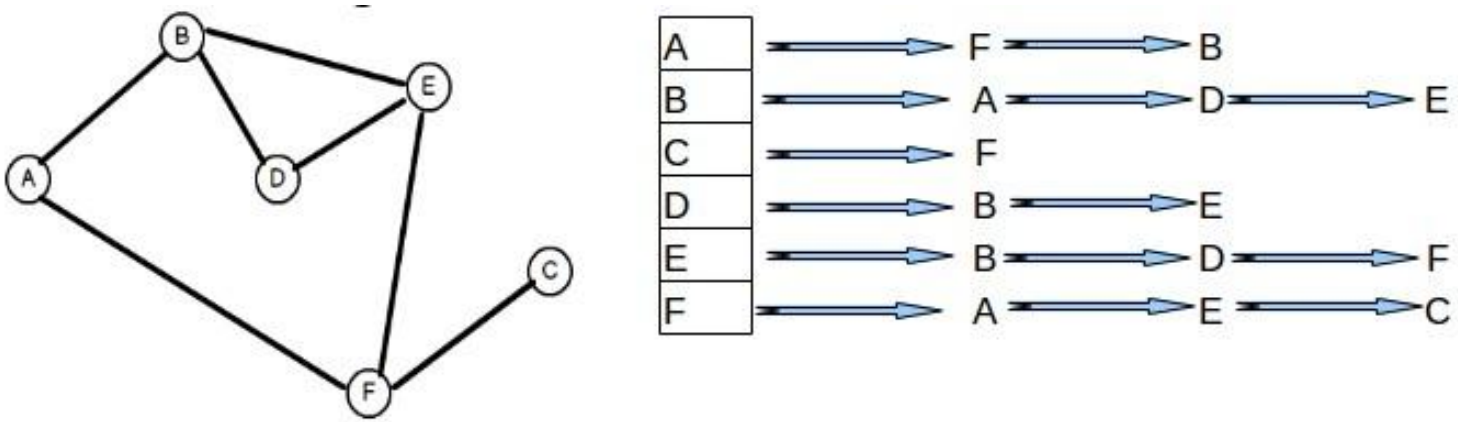
G



Adjacency List

Graph G can be represented as array of lists where A_i is list of adjacent vertices of V_i from $G(V)$.

Figure: Adjacency list representation of a graph.

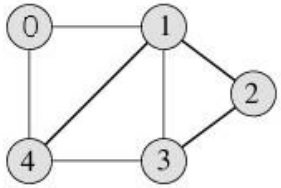


Implementation

Implementing functions

1. *node *newNode(int vertex)*, returns a new node with **vertex** in it.
2. *adjacencyList *createGraph(int n)*, returns an array which represents G with n vertices.
3. *void displayGraph(int n)*, prints G of n vertices.
4. *void addAdjacentVertex(int vertex, int adjVertex)*, adds **adjVertex** as adjacent vertex to **vertex**.

Considering the following graph.



Summary

- Graph definition.
- Graph representation as a data structure.
- Implementing adjacency list for a graph.

Questions