Data Structures

Lecture 15
Graph Basics

Credit: Prantik Paul, Lecturer, BRAC University

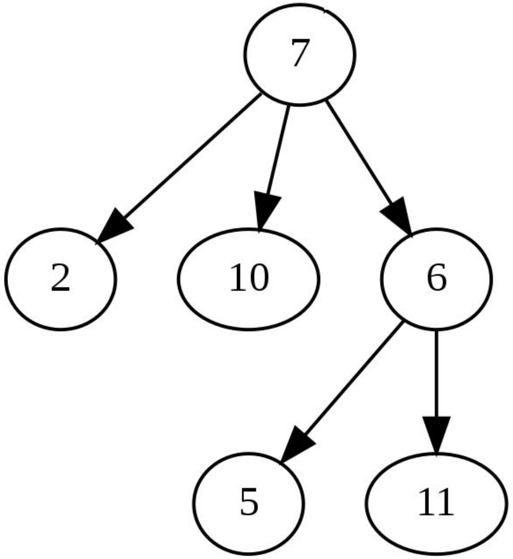
Graph - A superset of Trees

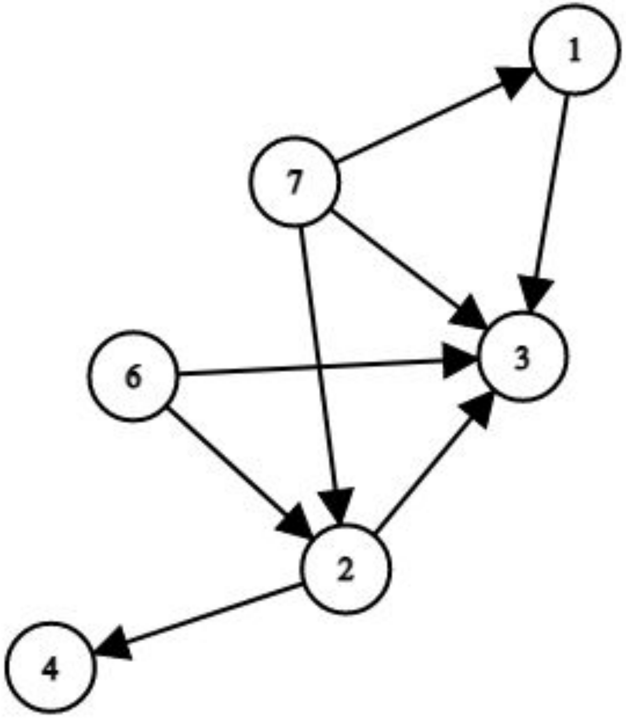
- $G := (V, E)$

- $V :=$ Set of vertices (nodes)

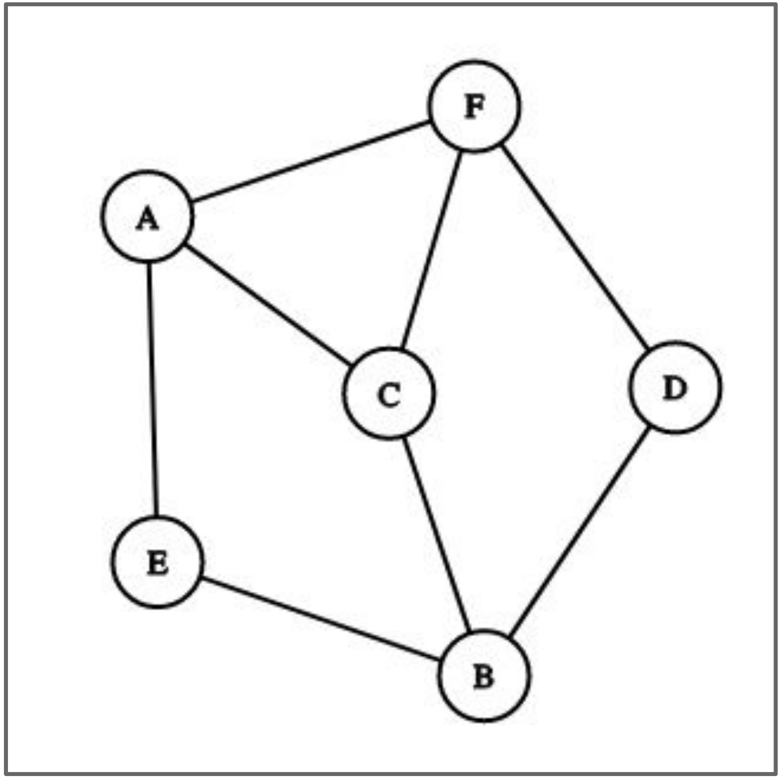
- $E :=$ Set of edges

Graph - A superset of Trees





Types of Graphs

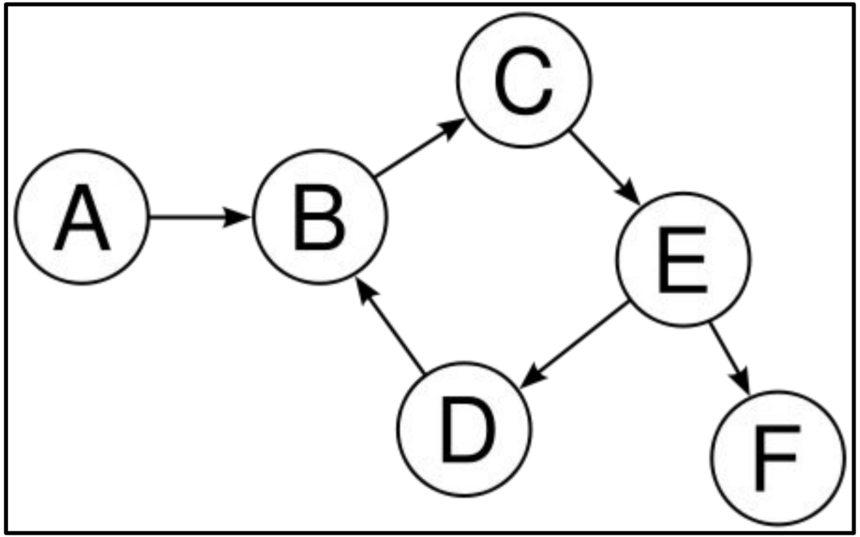


Undirected Graph

V = {A, B, C, D, E, F}

E = { {A,E}, {A,C} , {A,F}, {B,C}, {B,D}, {B,E}, {C,F}, {D,F} }

Types of Graphs



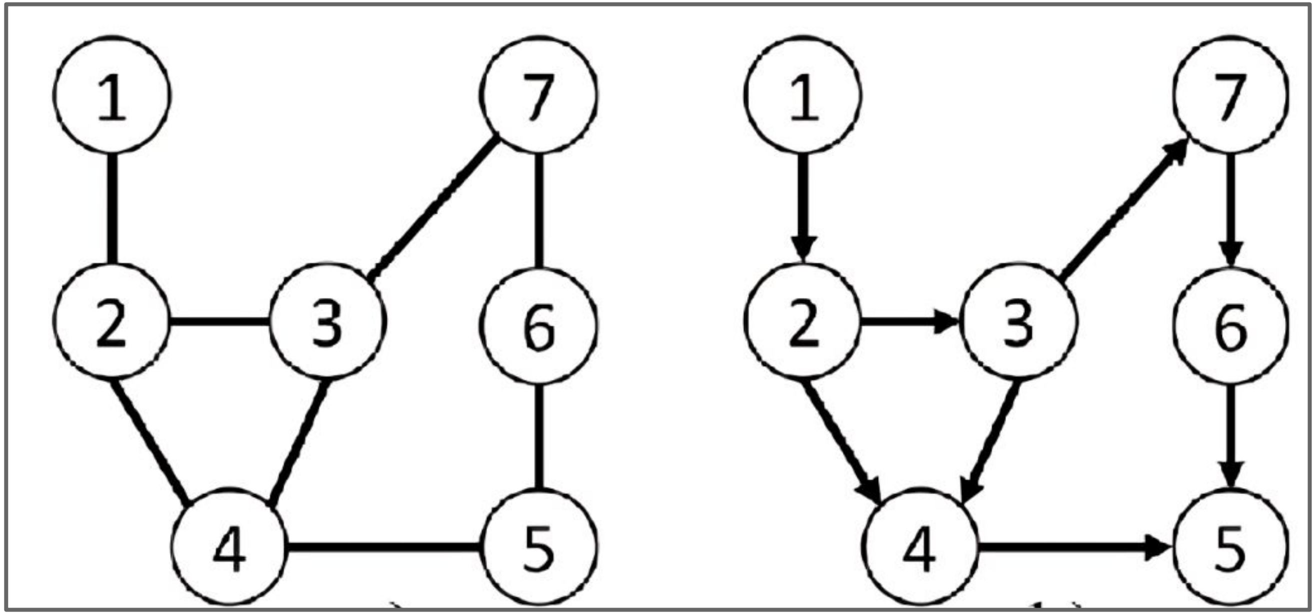
Directed Graph

V = {A, B, C, D, E, F}

E = { {A,B}, {B,C}, {C,E}, {D,B}, {E,D}, {E,F} }

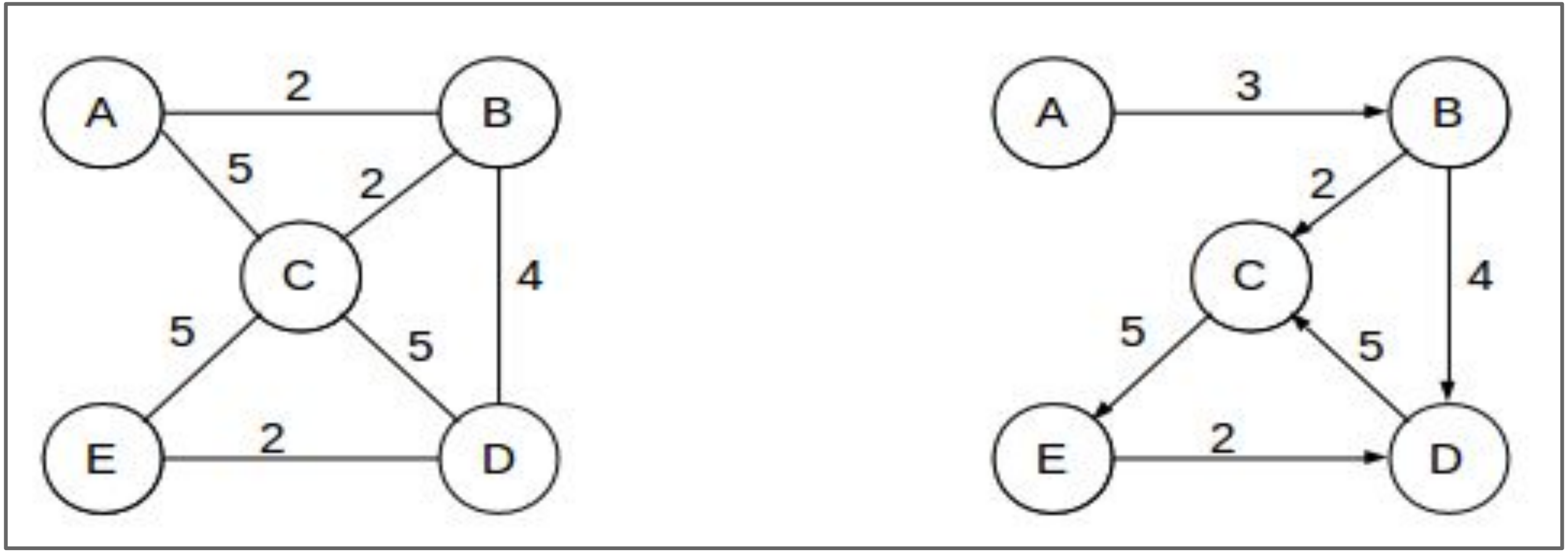
Types of Graphs

Unweighted Graph



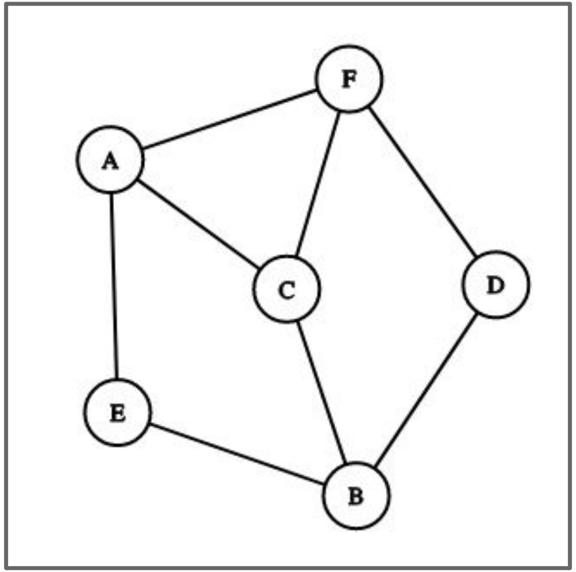
Types of Graphs

Weighted Graph



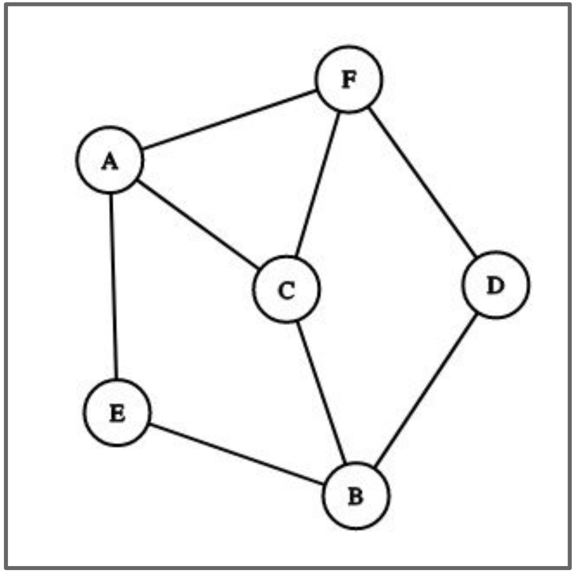
Graph - Basic Terminologies

Edges : Basic unit connecting vertices



Graph - Basic Terminologies

Adjacent Vertices: Vertices sharing an edge

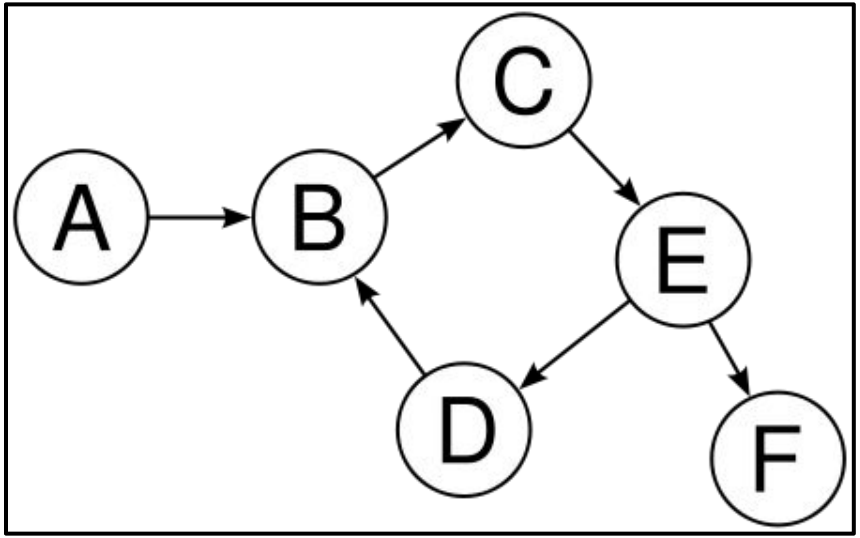


Adj(A) = {E, C, F}

Adj(C) = {A, B, F}

Graph - Basic Terminologies

Adjacent Vertices: Vertices sharing an edge



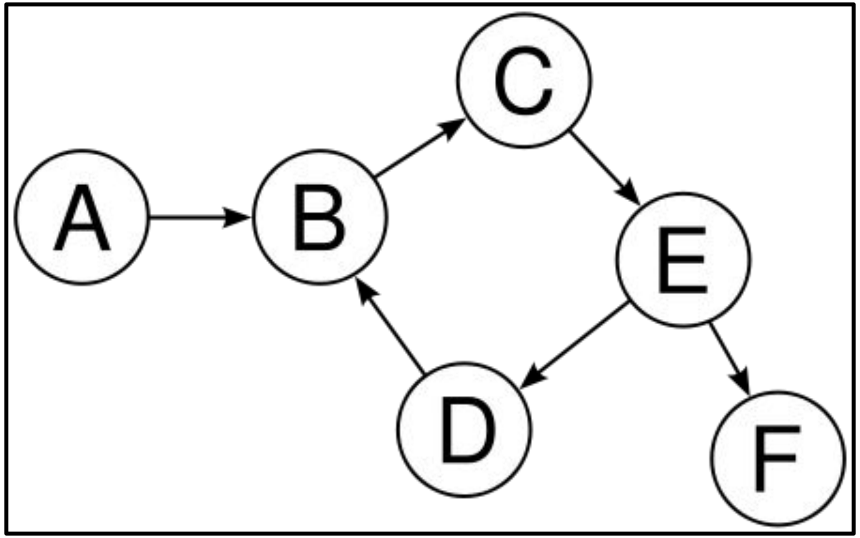
Adj(A) = {B}

Adj(B) = {C}

Adj(E) = {D, F}

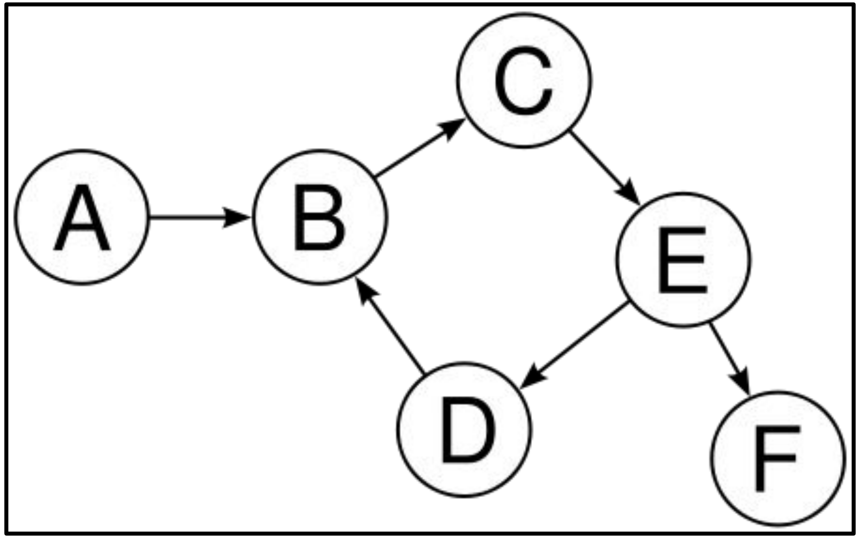
Graph - Basic Terminologies

Outgoing Edges: directed edges starting from a vertex



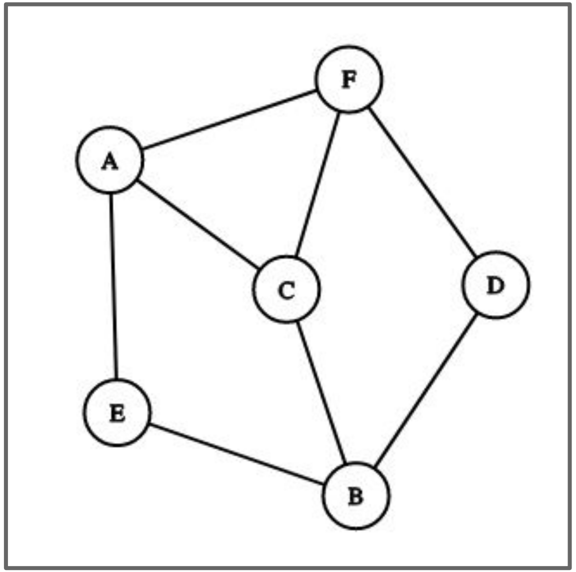
Graph - Basic Terminologies

Incoming Edges: directed edges reaching a vertex



Graph - Basic Terminologies

Degree : Total number of edges connected to a vertex



deg(A) = 3

deg(D) = 2

deg(B) = 3

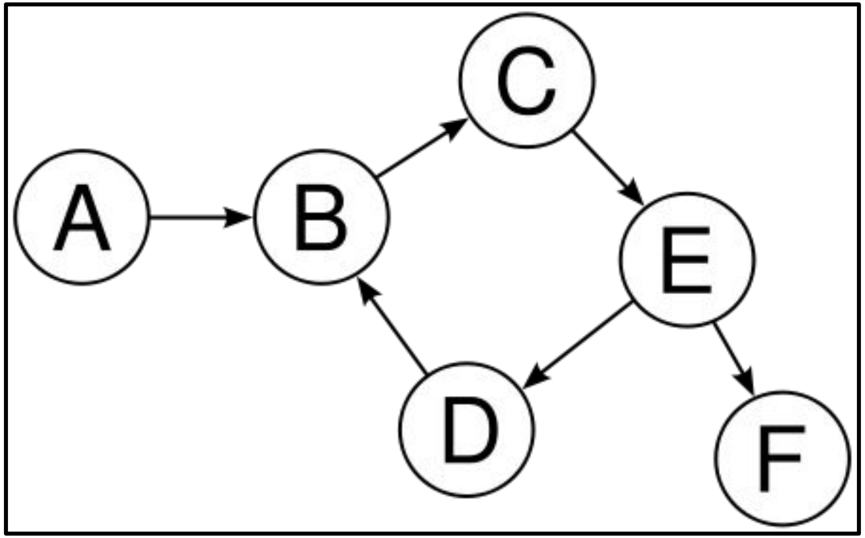
deg(E) = 2

deg(C) = 3

deg(F) = 3

Graph - Basic Terminologies

In Degree : total number of incoming edges to a vertex



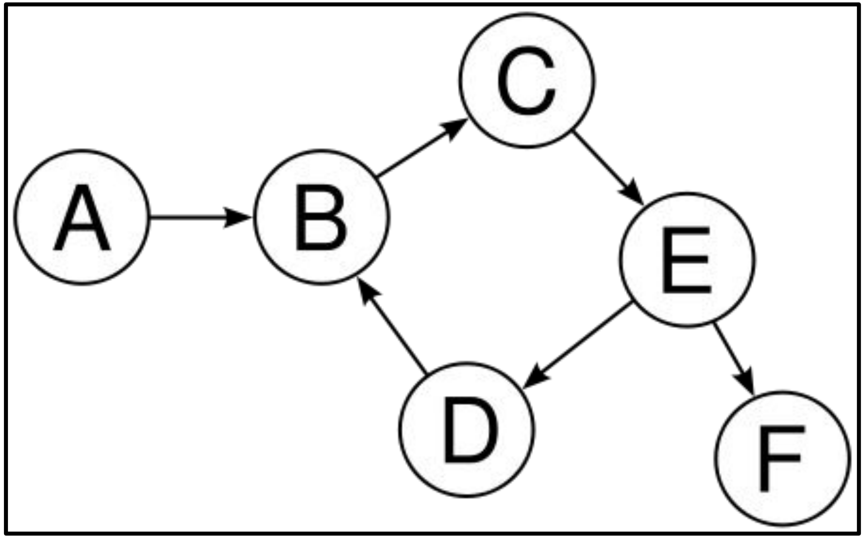
Indeg(A) = 0

Indeg(B) = 2

Indeg(C) = 1

Graph - Basic Terminologies

Out Degree: total number of outgoing edges from a vertex



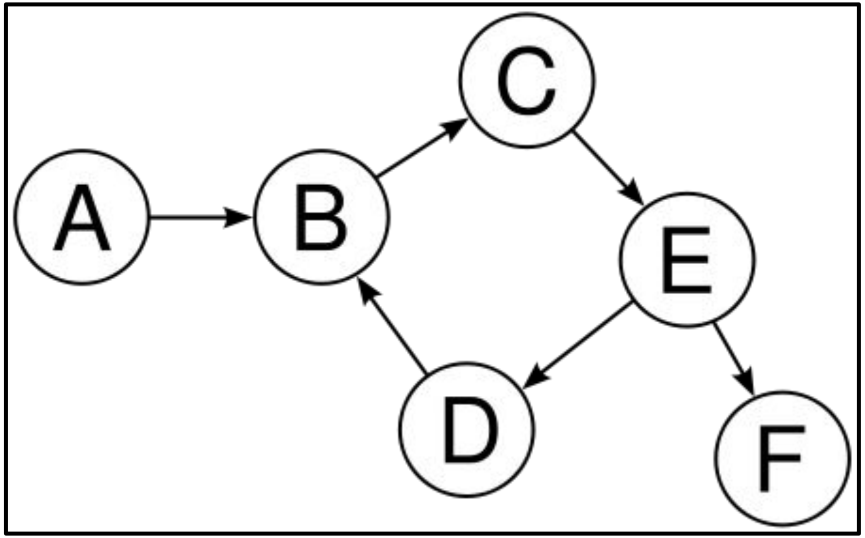
Outdeg(A) = 1

Outdeg(E) = 2

Outdeg(F) = 0

Graph - Basic Terminologies

Degree: Indegree + Outdegree



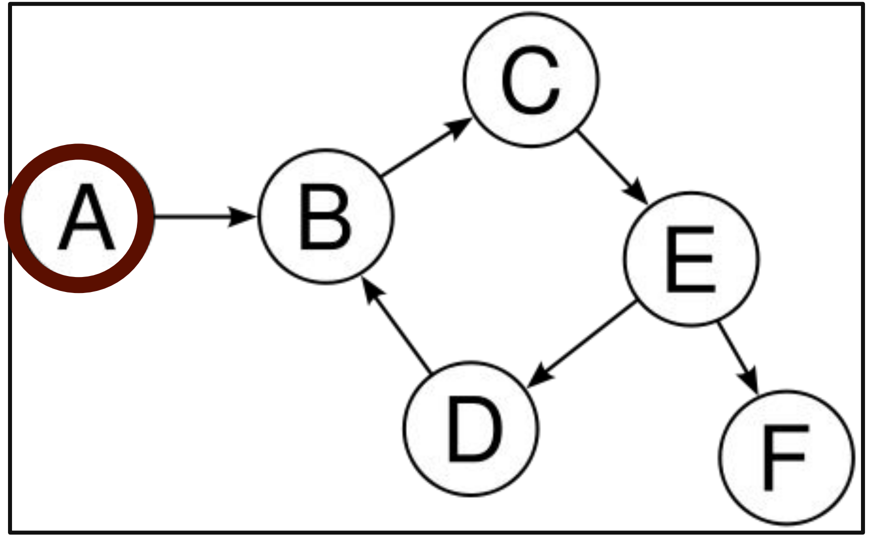
deg(A) = 1

deg(E) = 3

deg(F) = 1

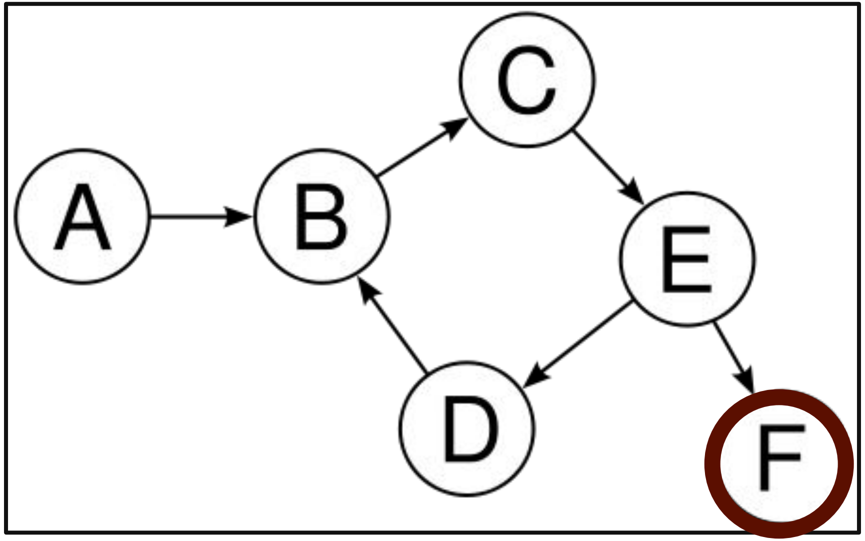
Graph - Basic Terminologies

Source Vertex : A vertex with in-degree zero



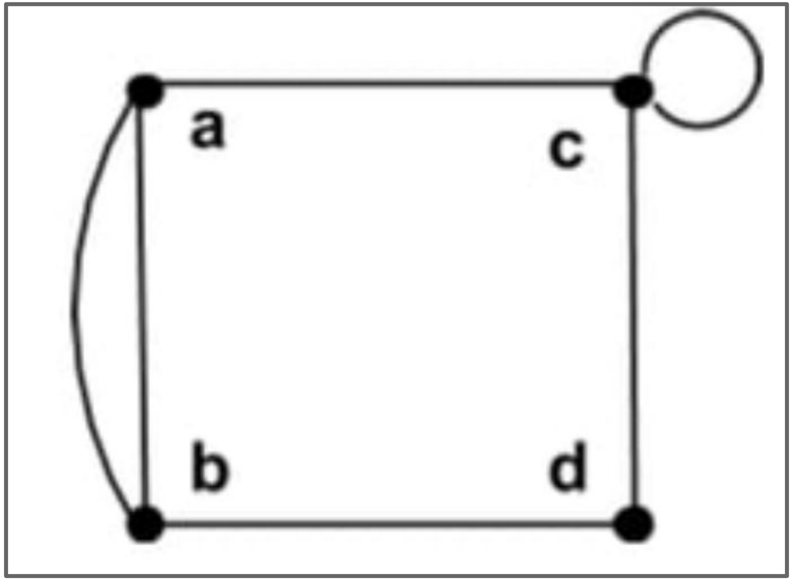
Graph - Basic Terminologies

Sink Vertex : A vertex with out-degree zero



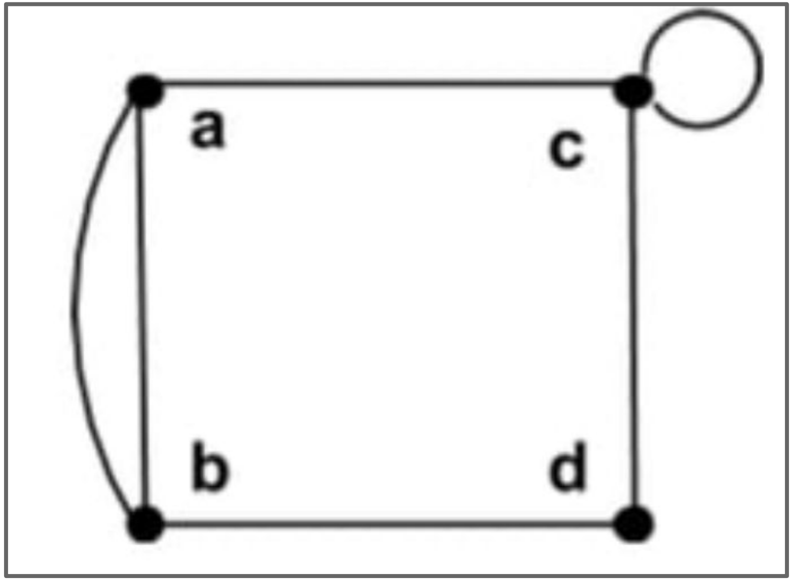
Graph - Basic Terminologies

Parallel Edges : Multiple edges between the same pair of vertices



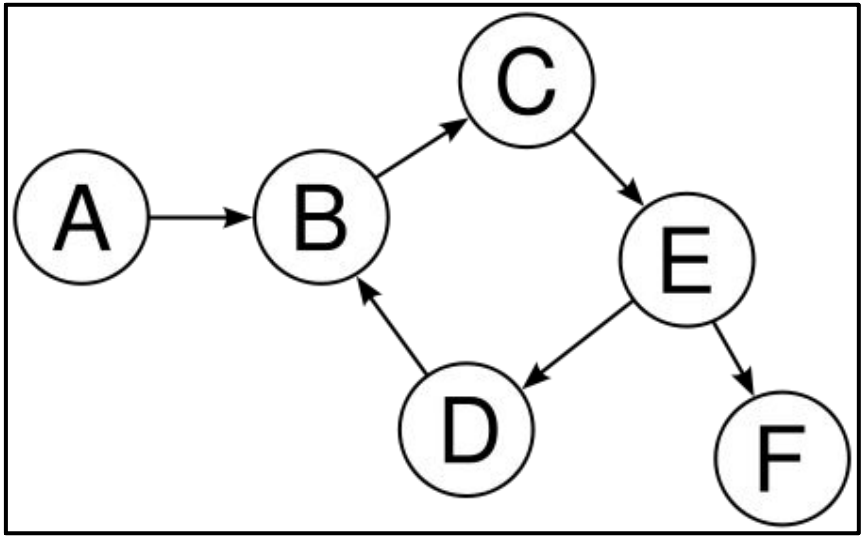
Graph - Basic Terminologies

Self Loop : Edge between a vertex and itself



Graph - Basic Terminologies

Path : sequence of vertices where each adjacent pair is connected by an edge



Path(A, F)

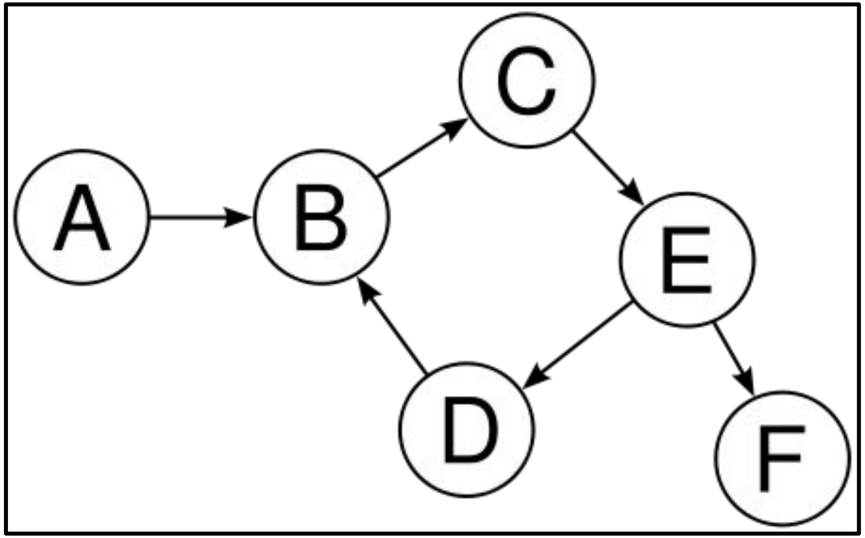
A → B → C → E → F

Path(D, A)

No path

Graph - Basic Terminologies

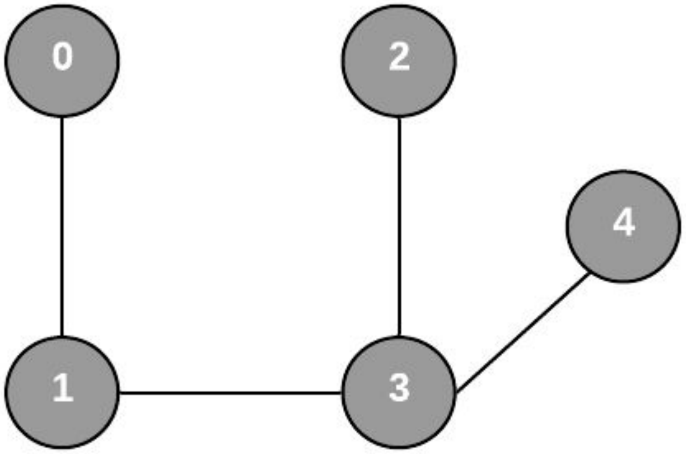
Cycle : closed path where the first and last vertices are the same



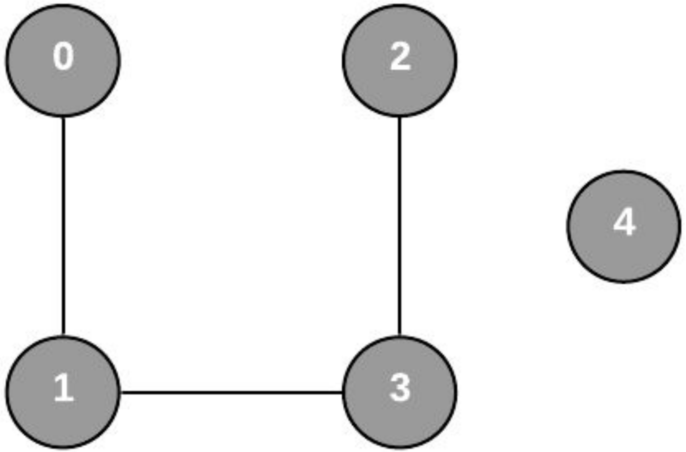
B → C → E → D → B

Graph - Basic Terminologies

Connected Graph: Has a path between every pair of vertices



Connected Graph



Disconnected Graph

Graph - Basic Terminologies

Connected Graph: Has a path between every pair of vertices

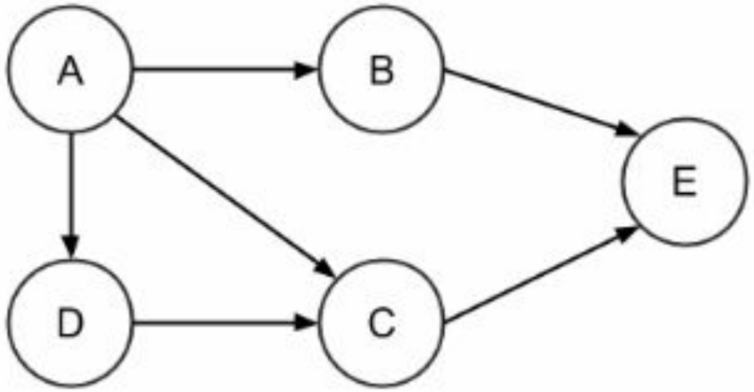


Figure: Disconnected Graph

Graph - Basic Terminologies

Sparse Graph: number of edges is considerably less than the maximum number of edges

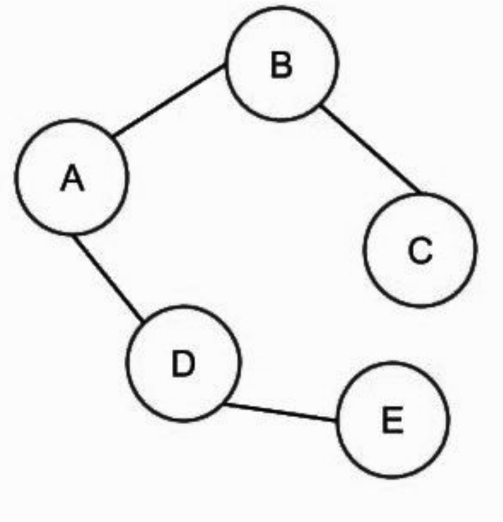


Figure: Sparse Graph

Graph - Basic Terminologies

Dense Graph: number of edges is close to the maximum number of edges

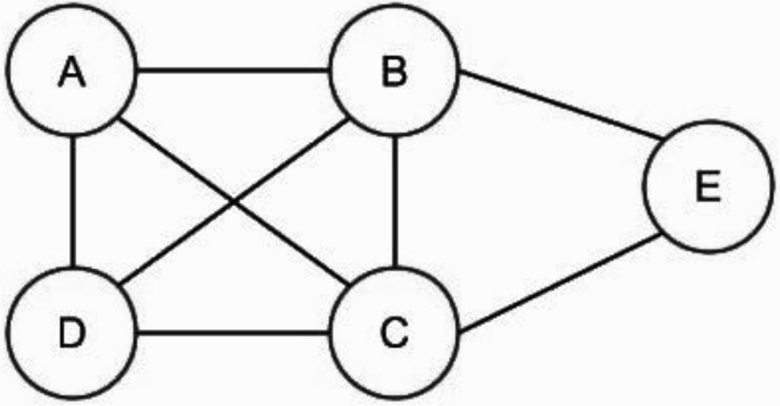
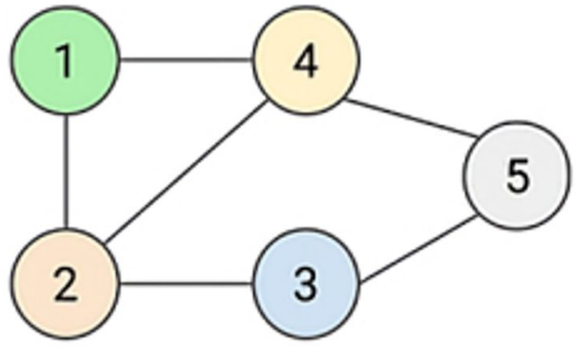
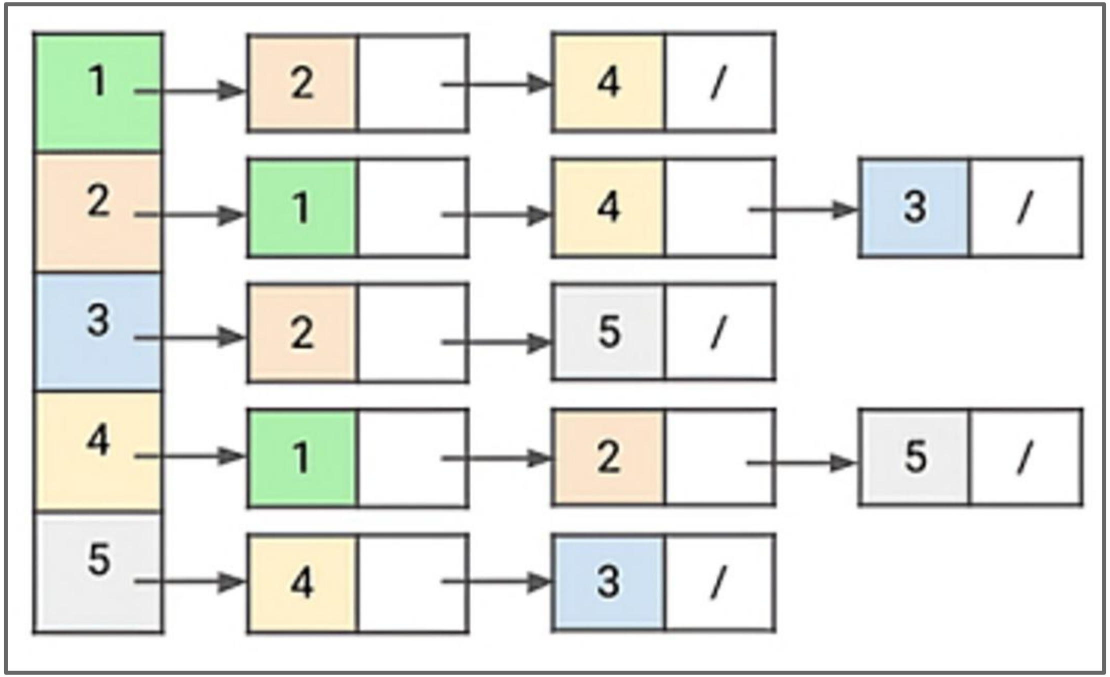


Figure: Dense Graph

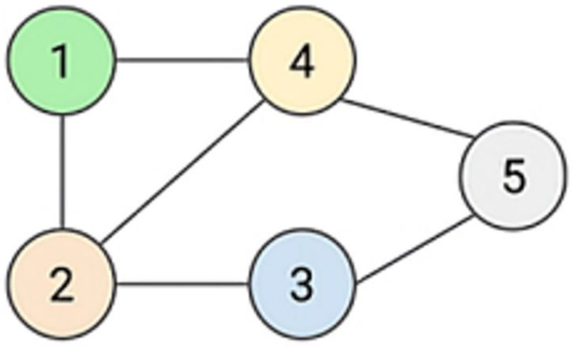
Graph - Representation - Adjacency List



Undirected Graph



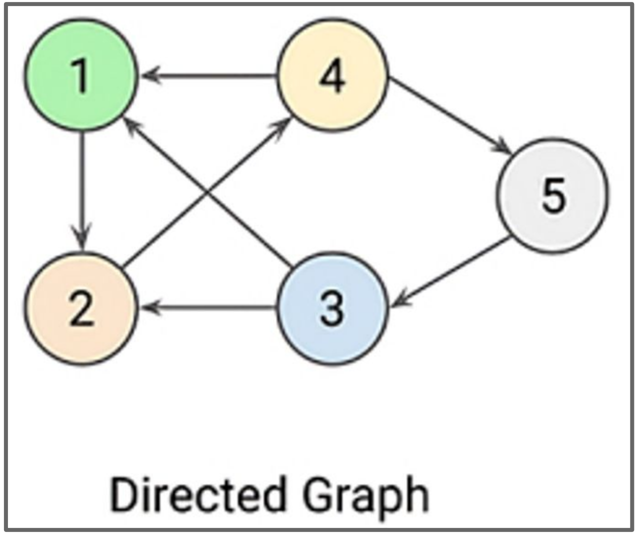
Graph - Representation - Adjacency Matrix

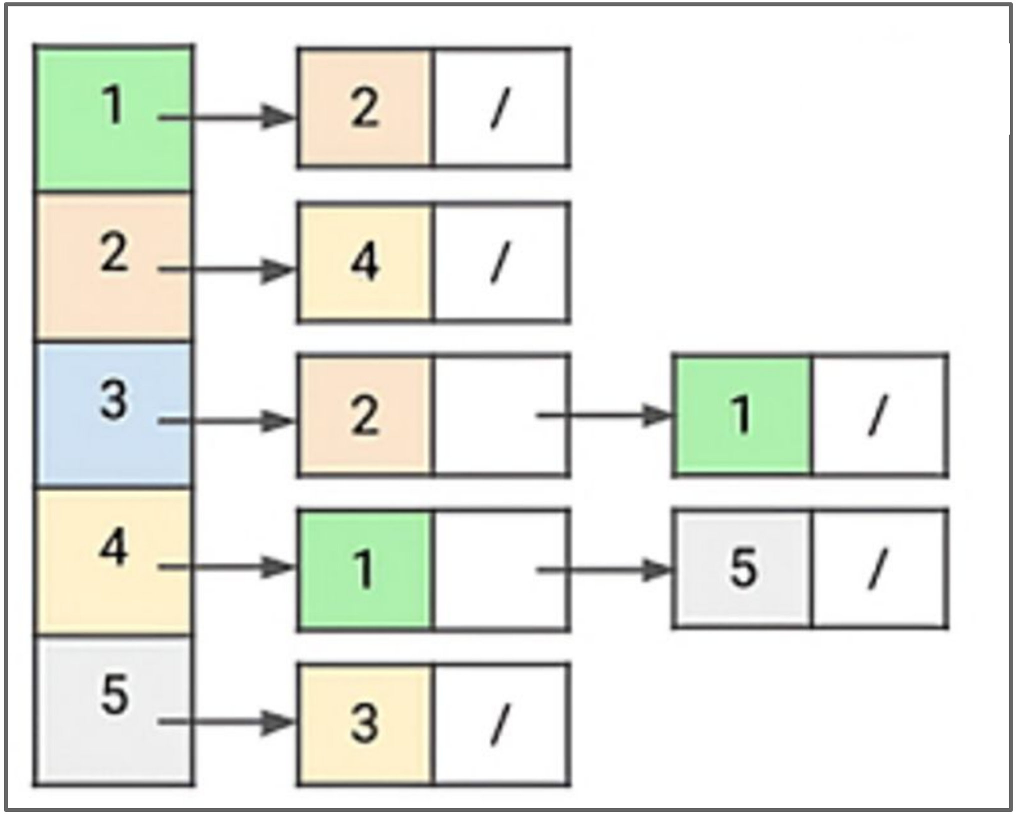


Undirected Graph

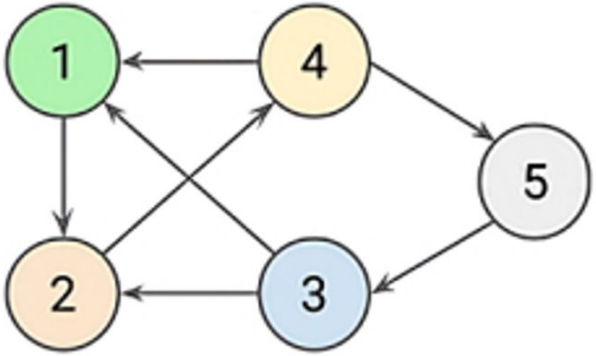
	1	2	3	4	5
1	0	1	0	1	0
2	1	0	1	1	0
3	0	1	0	0	1
4	1	1	0	0	1
5	0	0	1	1	0

Graph - Representation - Adjacency List





Graph - Representation - Adjacency Matrix



Directed Graph

	1	2	3	4	5
1	0	1	0	0	0
2	0	0	0	1	0
3	1	1	0	0	0
4	`1	0	0	0	1
5	0	0	1	0	0

Graph - Representation

What to do for Weighted Graph's Adjacency List Representation?

Store Tuples like <Vertice, Weight>
Instead of only Vertice

Graph - Representation

What to do for Weighted Graph's Adjacency List Representation?

Create an Edge Class with all the necessary attributes and use them as the nodes of the linked list.

Graph - Representation

What to do for Weighted Graph's Adjacency List Representation?

Create an Edge Class with all the necessary attributes and use them as the nodes of the linked list.

```
Class Edge {
        int ep1
        int ep2
        Int weight
}
```

Graph - Representation

What to do for Weighted Graph's Adjacency Matrix Representation?

Store Weight in the Matrix Entry Instead of 0 or 1