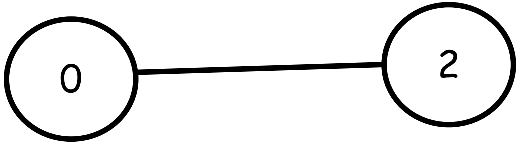
Graphs

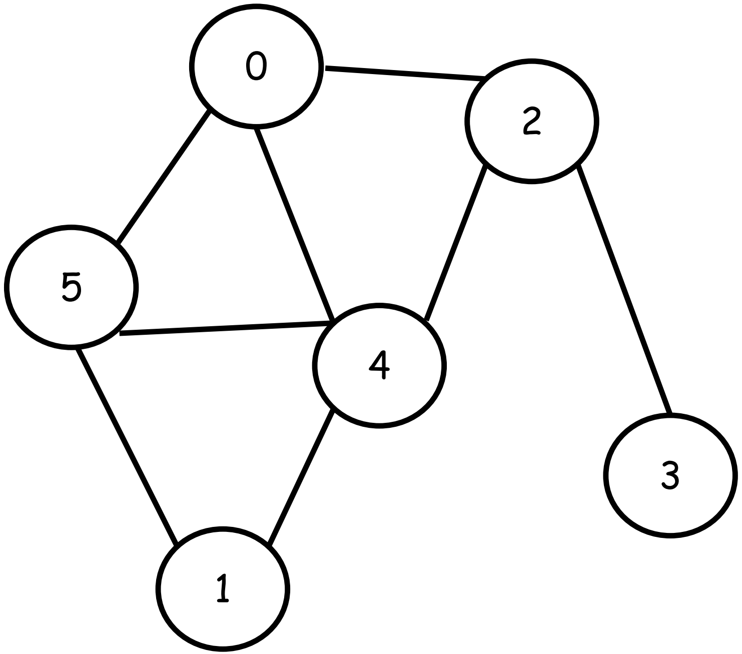
Undirected Graph: Vertices and edges



No Direction

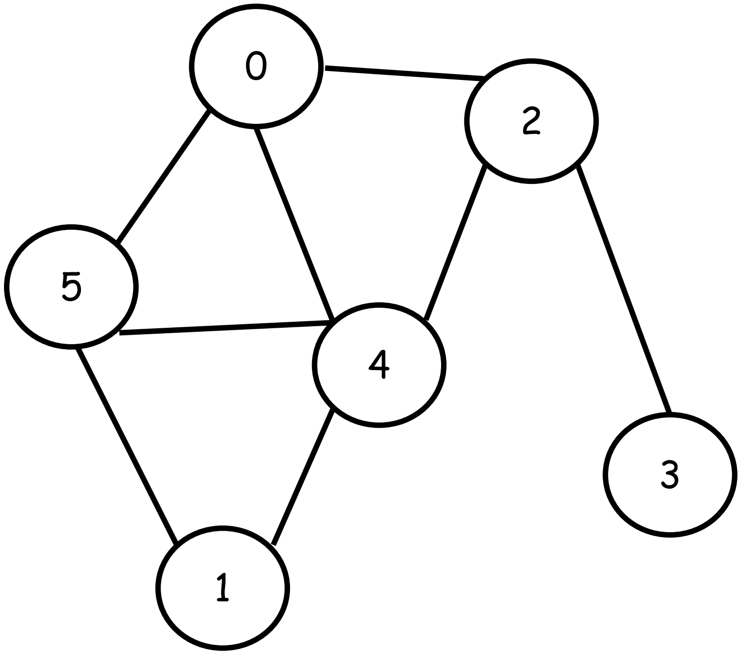
Representation: 02
20

Undirected Graph: Vertices and edges



Vertices = {0, 1, 2, 3, 4, 5}

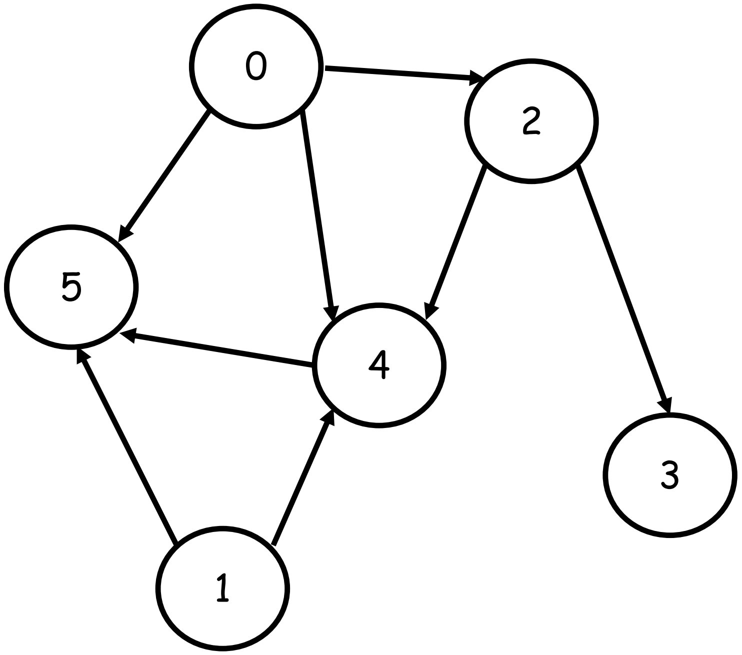
Undirected Graph: Vertices and edges

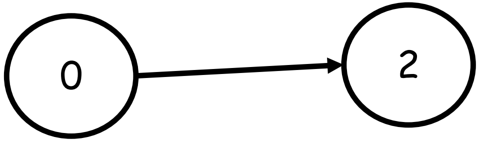


Edges = {02, 05, 04, 15, 14, 24, 23, 45 }

No convention
Which vertex to write first.

Directed Graph: Vertices and edges



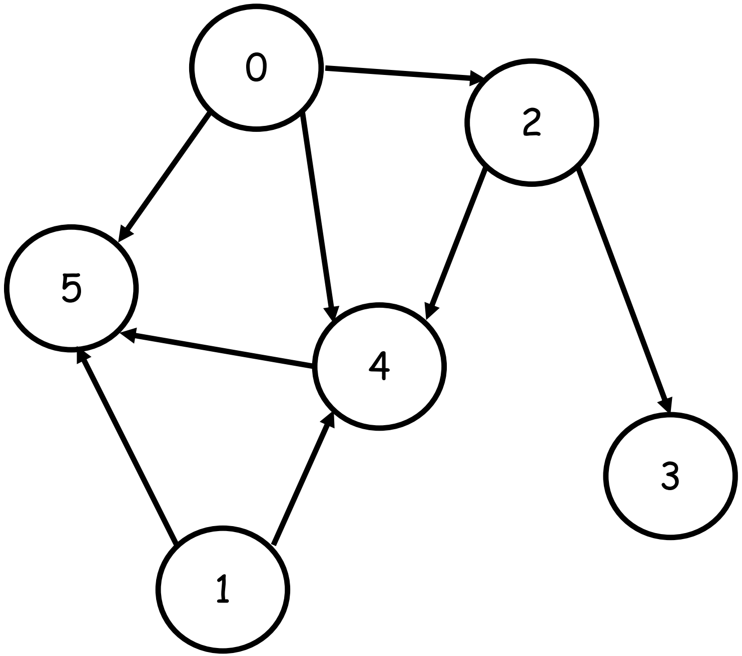


Direction: From Node 0 - Node 2
Node 0: Initial Vertex
Node 2: Terminal Vertex
Direction: Initial Vertex - Terminal Vertex

Representation: 02

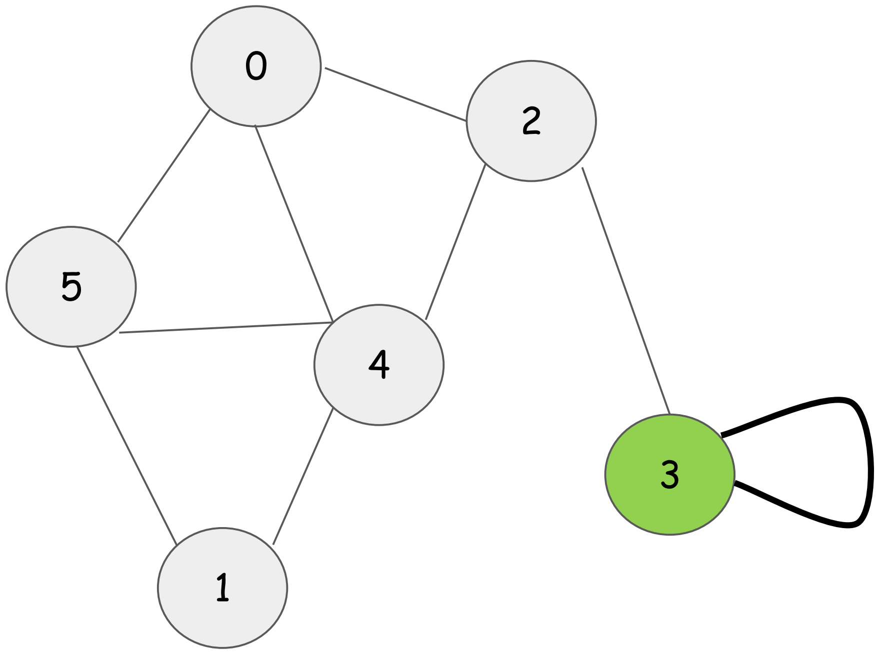
~~20~~

Directed Graph: Vertices and edges

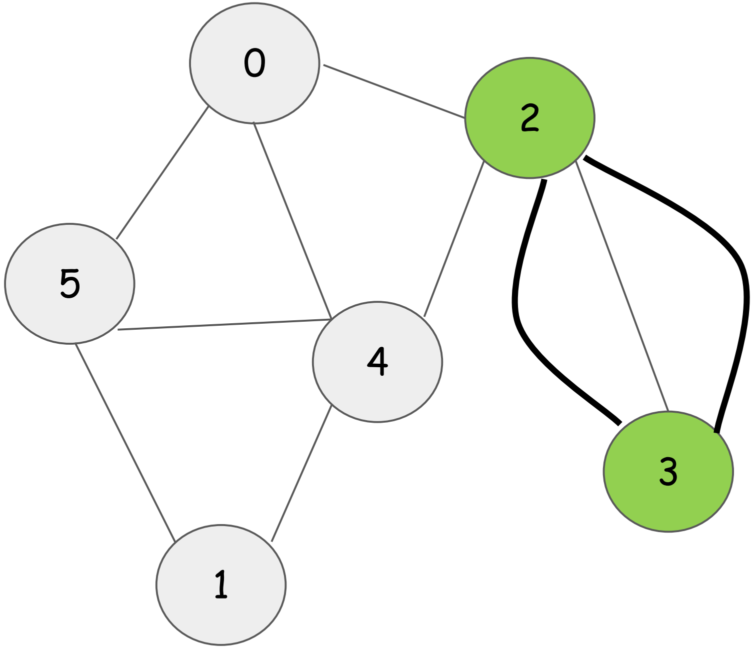


Edges = {05, 04, 02, 24, 45, 15, 14, 23}

Loops in Graphs

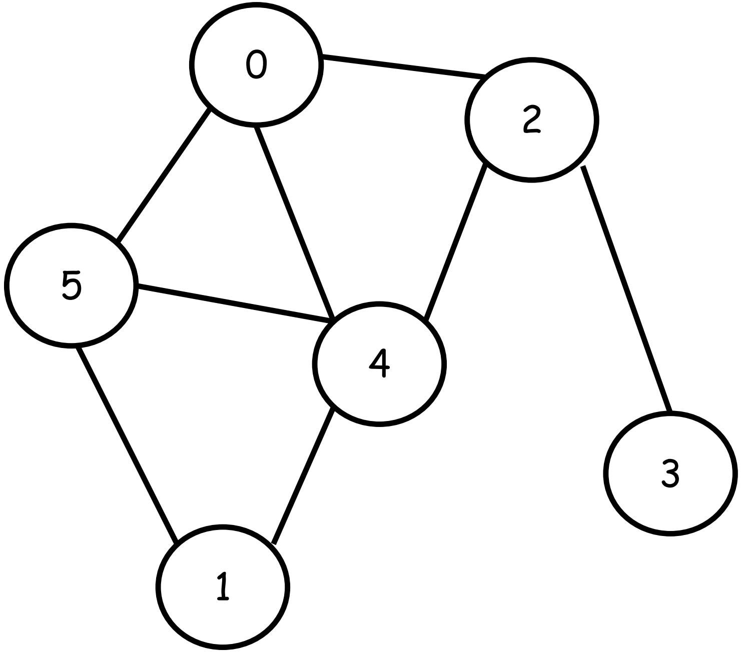


Multiple edges in Graphs



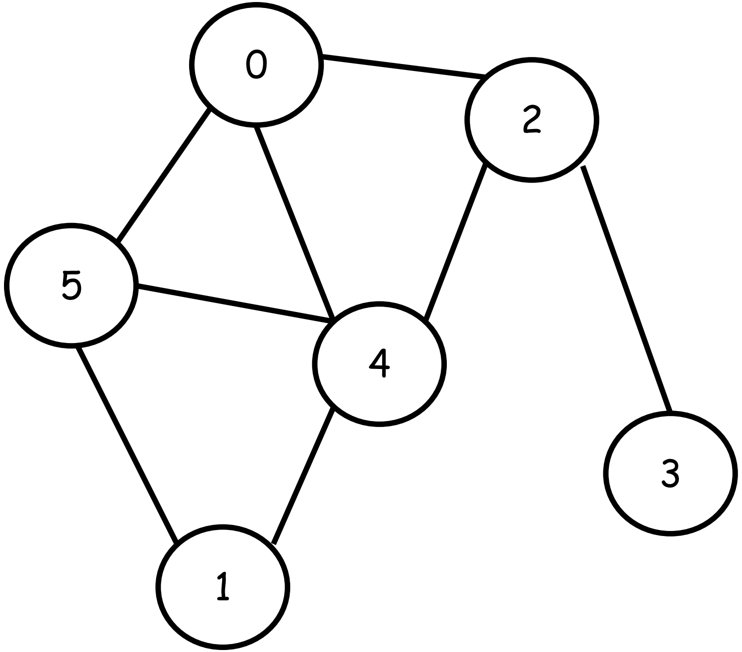
Adjacency Matrix Representation: Undirected Graph

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



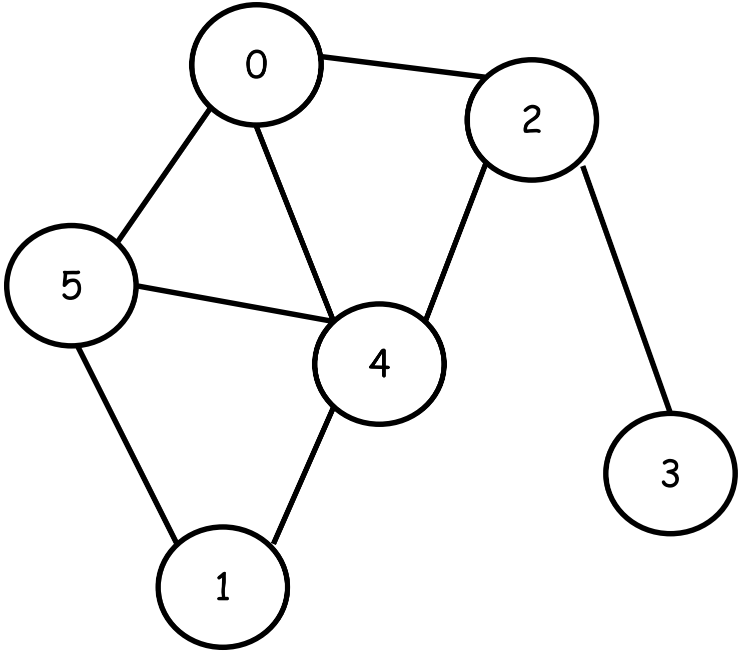
Adjacency Matrix Representation: Undirected Graph

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



Adjacency Matrix Representation: Undirected Graph

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



Adjacency Matrix Representation: Undirected Graph

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

A 2D array:
a[Number of vertices]
[Number of vertices]

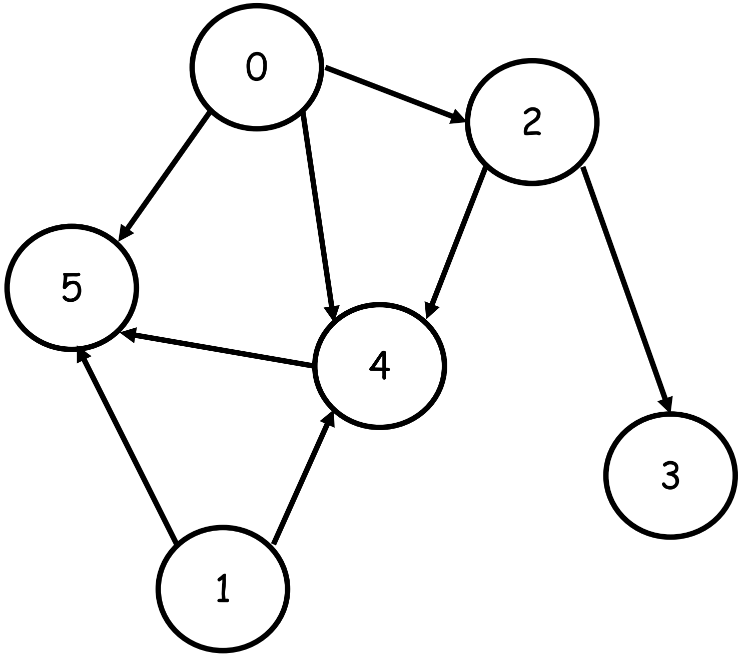
a[i][j] = a[j][i]

Adjacency Matrix Representation: Directed Graph

Terminal Vertex

Initial Vertex

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

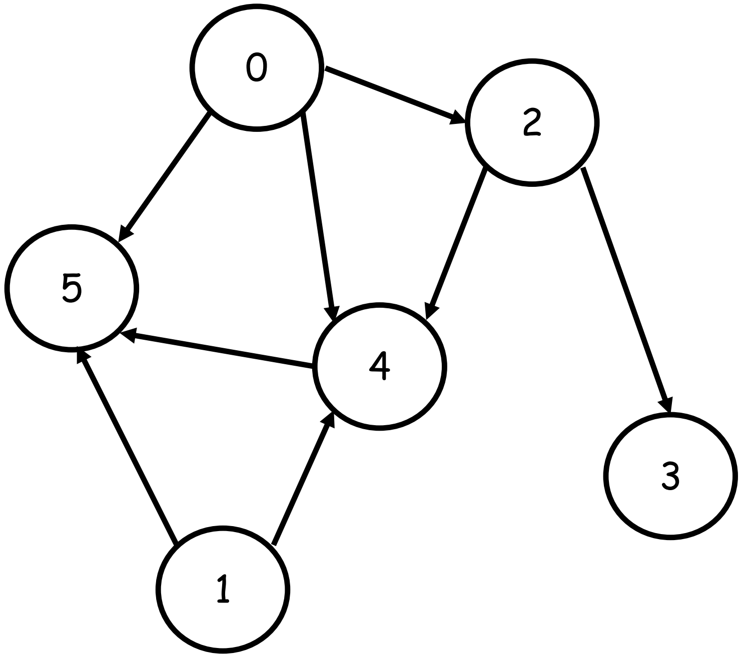


Adjacency Matrix Representation: Directed Graph

Terminal Vertex

Initial Vertex

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

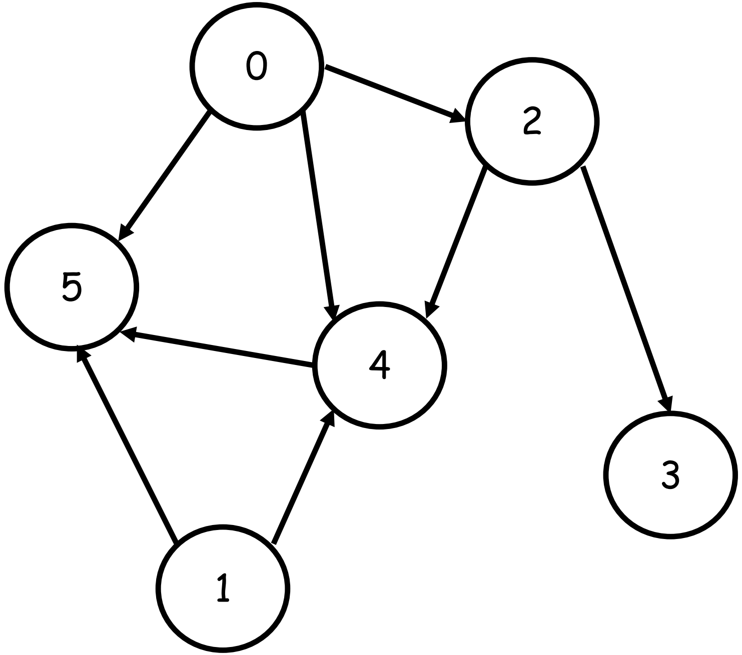


Adjacency Matrix Representation: Directed Graph

Terminal Vertex

Initial Vertex

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

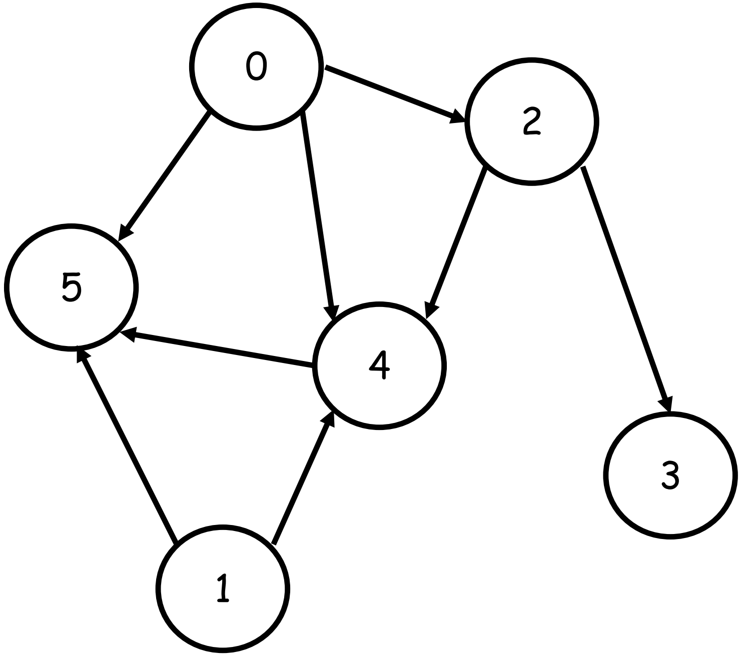


Adjacency Matrix Representation: Directed Graph

Terminal Vertex

Initial Vertex

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



Adjacency Matrix Representation: Directed Graph

Terminal Vertex

Initial Vertex

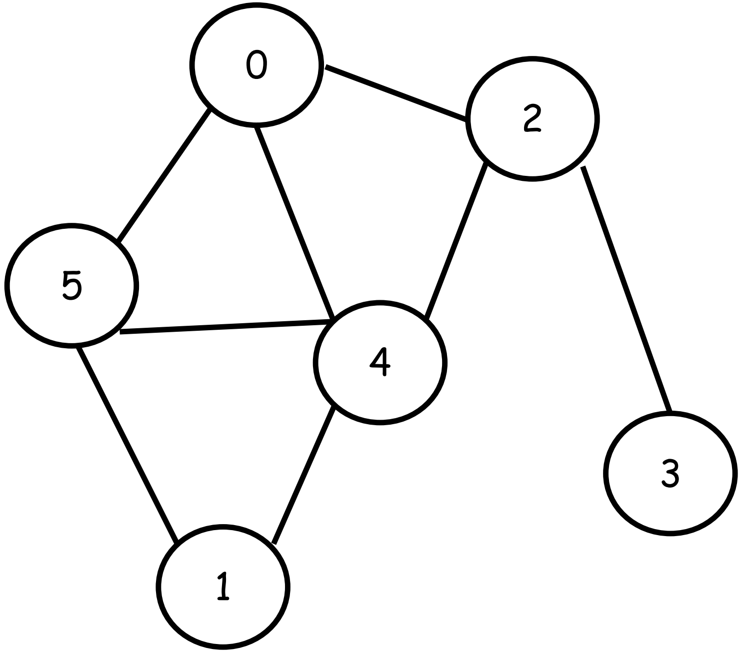
	0	1	2	3	4	5
0	0	0	1	0	1	1
1	0	0	0	0	1	1
2	0	0	0	1	1	0
3	0	0	0	0	0	0
4	0	0	0	0	0	1
5	0	0	0	0	0	0

A 2D array:
a[Number of vertices]
[Number of vertices]

a[i][j] = a[j][i]/
a[i][j] = a[j][i]
No Restriction

Degree of a vertex: Undirected Graph

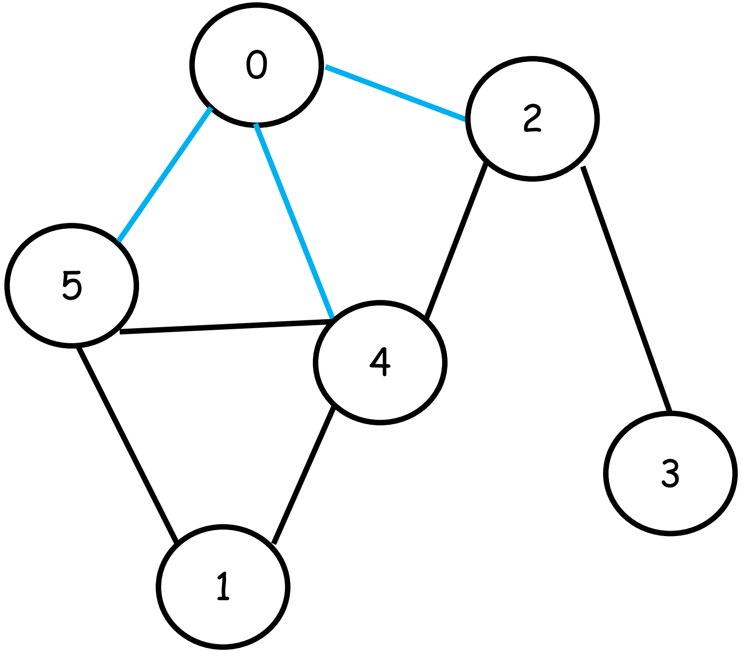
- Number of edges incident with it
- In case of Loop add +2



Degree of a vertex: Undirected Graph

- Number of edges incident with it
- In case of Loop add +2

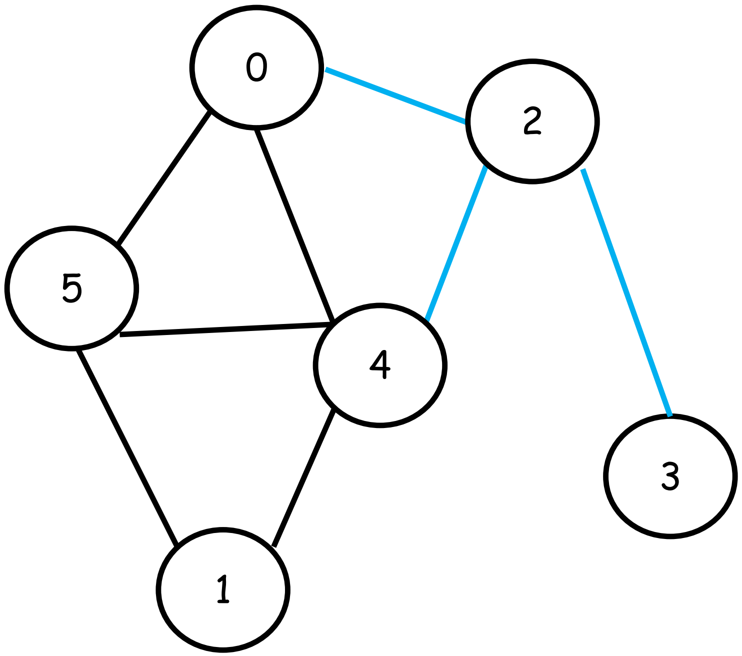
Vertex	Degree
0	3
1	
2	
3	
4	
5	



Degree of a vertex: Undirected Graph

- Number of edges incident with it
- In case of Loop add +2

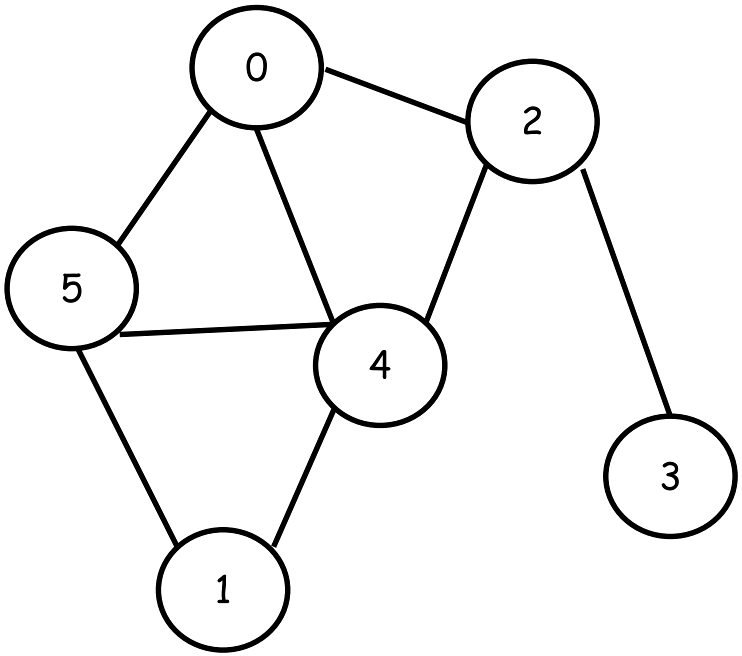
Vertex	Degree
0	3
1	2
2	
3	
4	
5	



Degree of a vertex: Undirected Graph

- Number of edges incident with it
- In case of Loop add +2

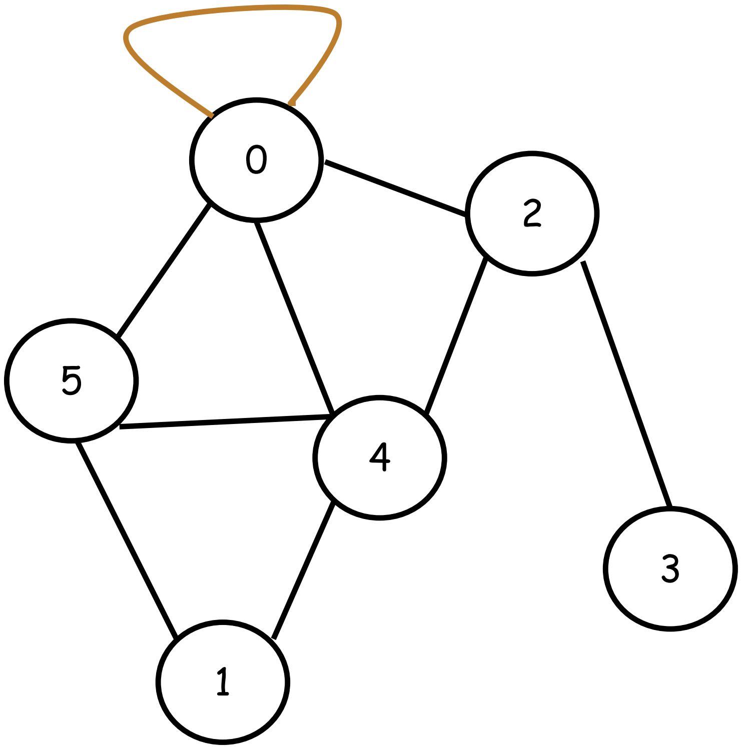
Vertex	Degree
0	3
1	2
2	3
3	1
4	4
5	3



Degree of a vertex: Undirected Graph

- Number of edges incident with it
- In case of Loop add +2

Vertex	Degree
0	5 (3 + 2)
1	2
2	3
3	1
4	4
5	3



Degree of a vertex: Undirected Graph

Vertex	Degree
0	3
1	2
2	3
3	1
4	4
5	3

$\Rightarrow \sum (1's)$

$\Rightarrow \sum (1's)$

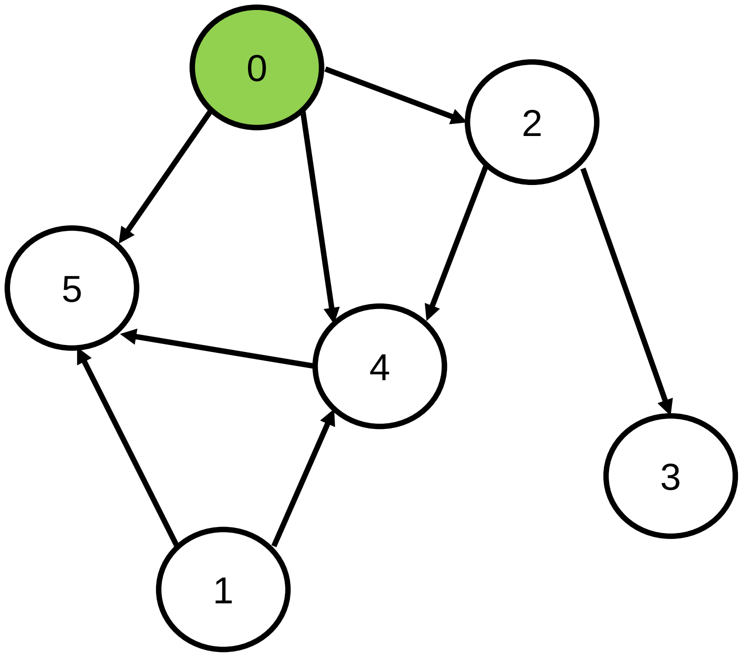
	0	1	2	3	4	5
0	0	0	1	0	1	1
1	0	0	0	0	1	1
2	1	0	0	1	1	0
3	0	0	1	0	0	0
4	1	1	1	0	0	1
5	1	1	0	0	1	0

Degree of a vertex: Directed Graph

- In Degree : No of INCOMING edges
- Out Degree : No of OUTGOING edges

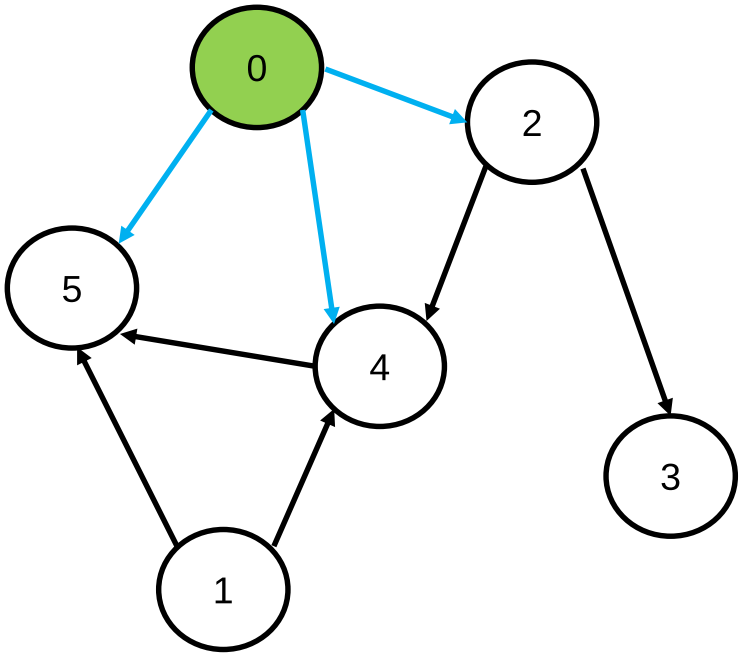
Degree of a vertex: Directed Graph

Vertex	In Degree	Out Degree
0	0	
1		
2		
3		
4		
5		



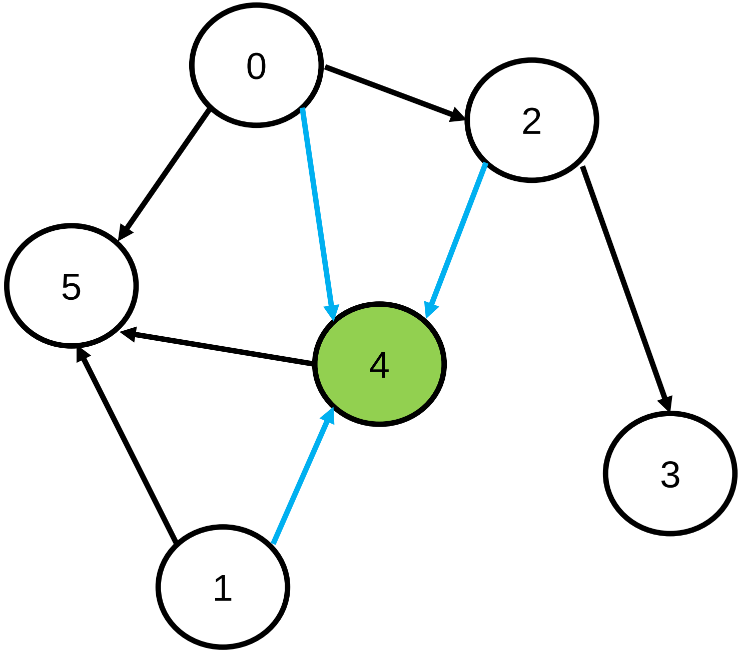
Degree of a vertex: Directed Graph

Vertex	In Degree	Out Degree
0	0	3
1		
2		
3		
4		
5		



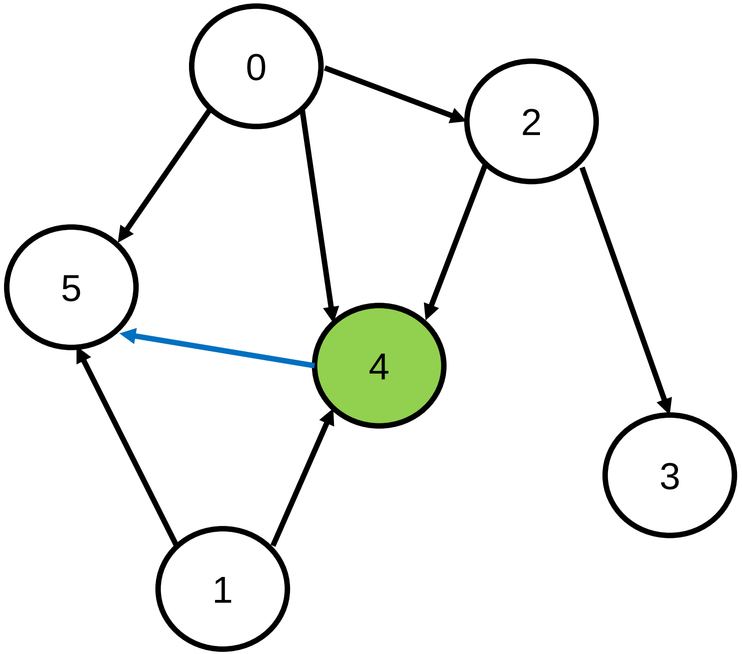
Degree of a vertex: Directed Graph

Vertex	In Degree	Out Degree
0	0	3
1		
2		
3		
4	3	
5		



Degree of a vertex: Directed Graph

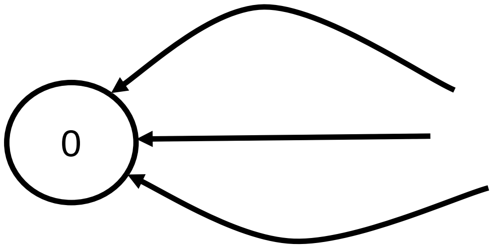
Vertex	In Degree	Out Degree
0	0	3
1		
2		
3		
4	3	1
5		



Degree of a vertex: Directed Graph

Vertex	In Degree	Out Degree
0	0	3

In degree = number of incoming edges
In degree of vertex 0 = number of incoming edges to 0



So, for counting in degrees of vertex 0
0 must be a **TERMINAL VERTEX**

Degree of a vertex: Directed Graph

Vertex	In Degree	Out Degree
0	0	3

$\sum$ (1's)

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

Degree of a vertex: Directed Graph

Vertex	In Degree	Out Degree
0	0	3

$\sum$ (1's)

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

Bipartite Graph

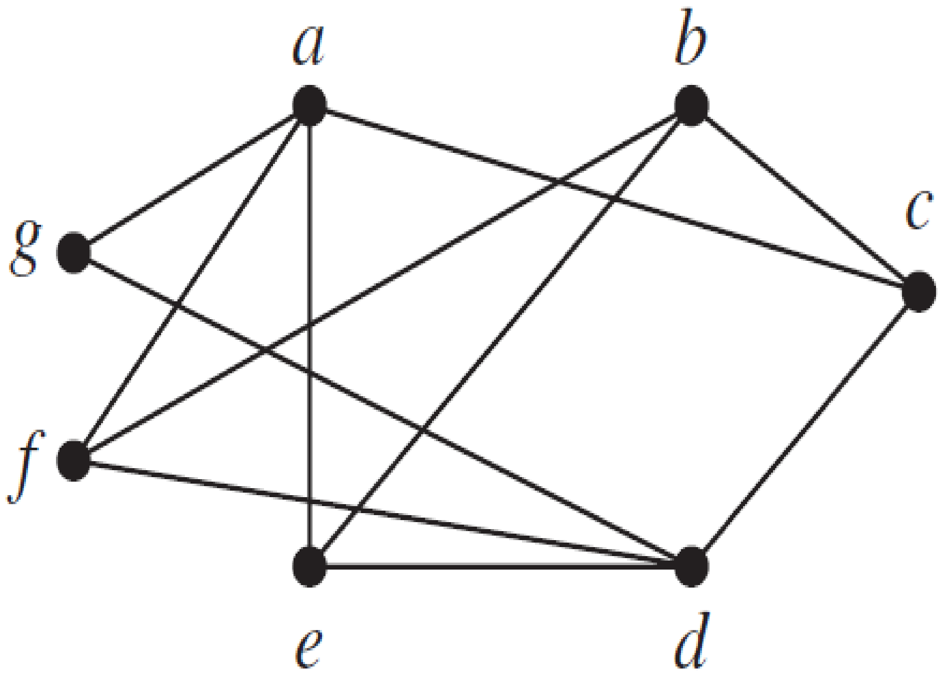
if its vertex set V can be partitioned into two disjoint sets V1 and V2 such that

no edge in G connects either two vertices in V1 or two vertices in V2

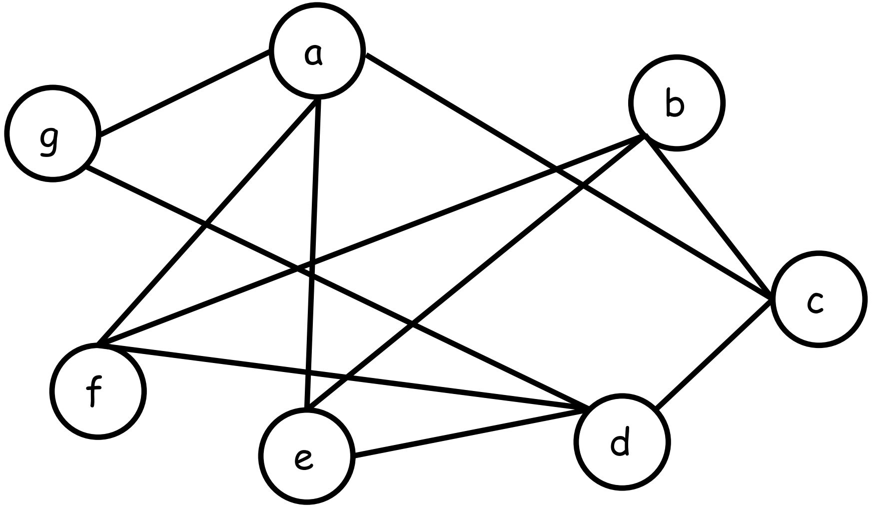
Bipartite Graph

V1 = {a, b, d}

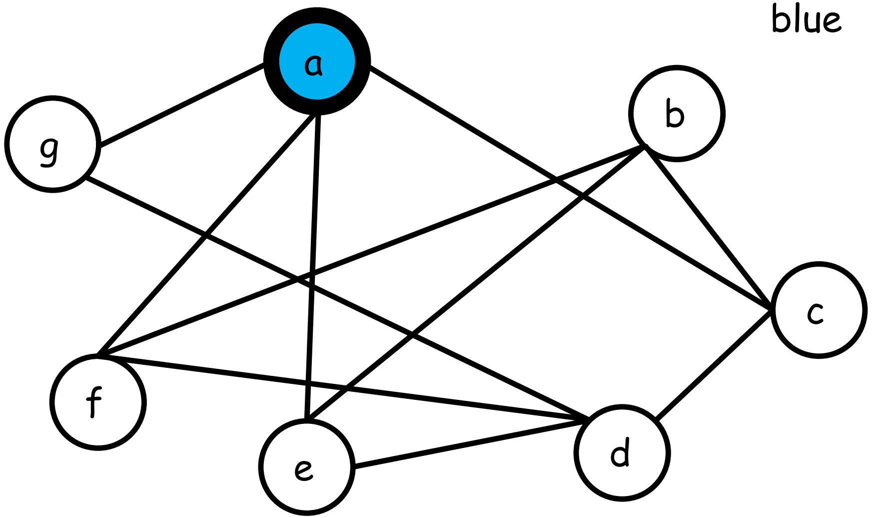
V2 = {c, e, f, g}



How to decide if a graph is bipartite or not

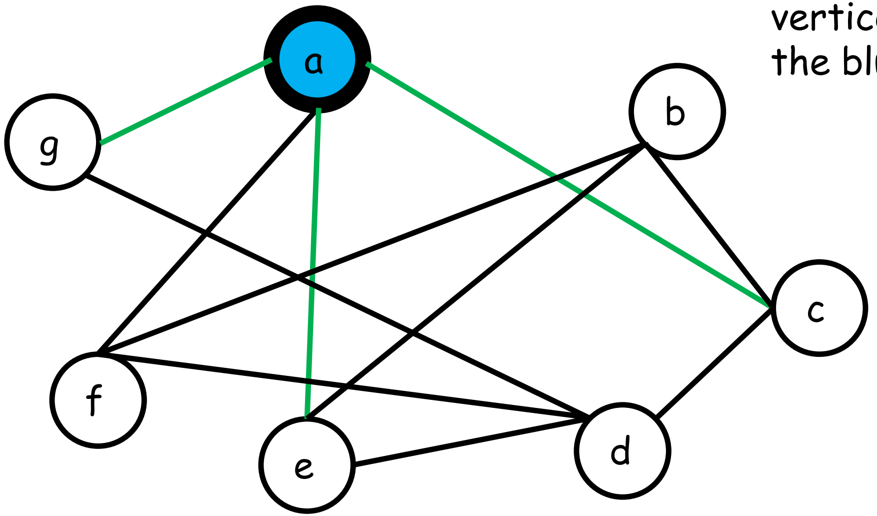


How to decide if a graph is bipartite or not



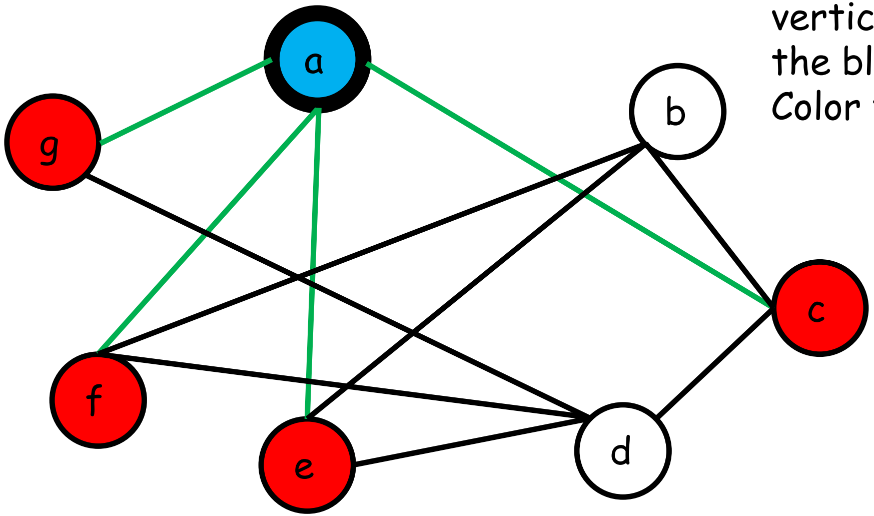
1. Color any of the vertices blue

How to decide if a graph is bipartite or not



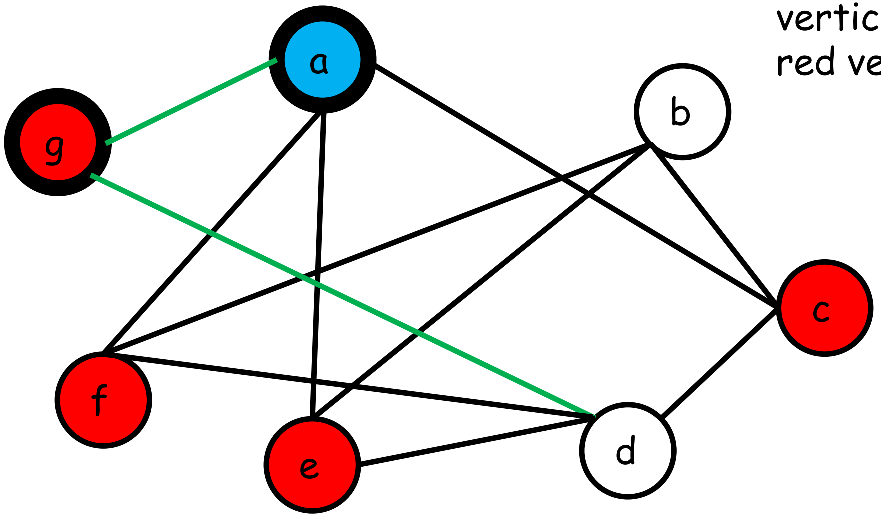
2. Identify all uncolored vertices that are adjacent to the blue vertex.

How to decide if a graph is bipartite or not



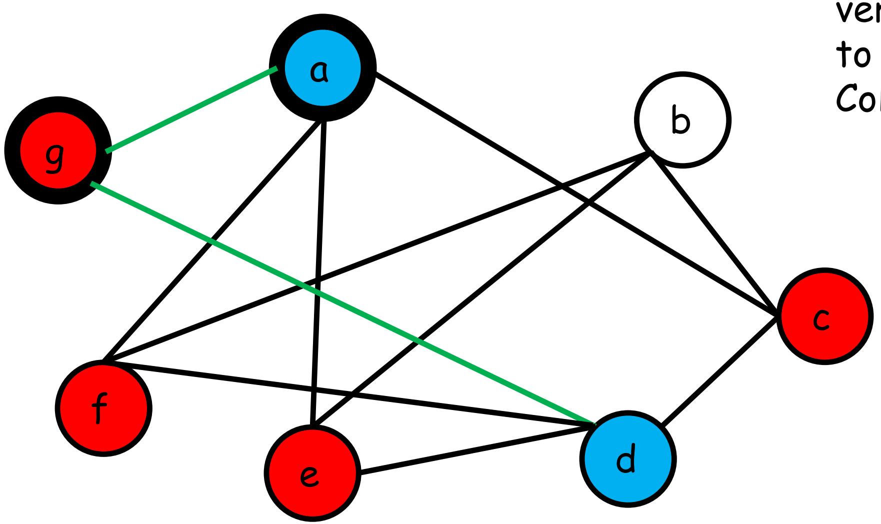
2. Identify all uncolored vertices that are adjacent to the blue vertex.
Color them red

How to decide if a graph is bipartite or not



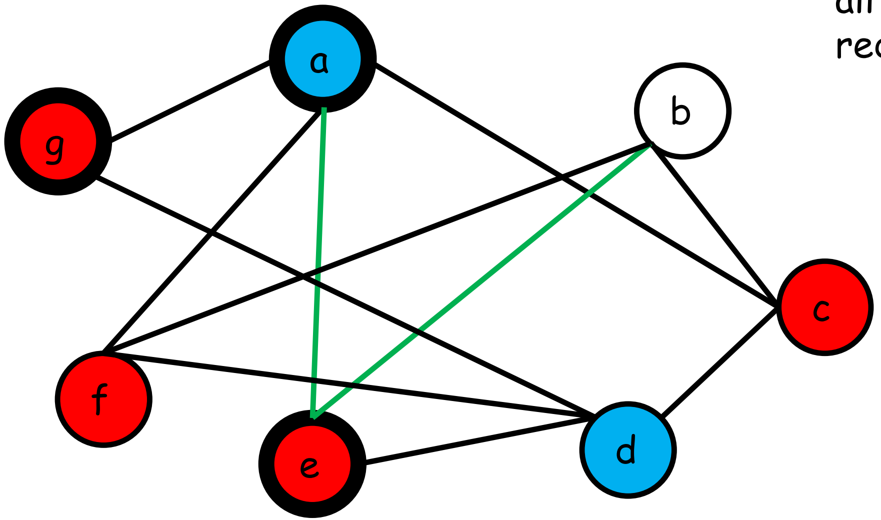
3. Identify all uncolored vertices that are adjacent to a red vertex.

How to decide if a graph is bipartite or not



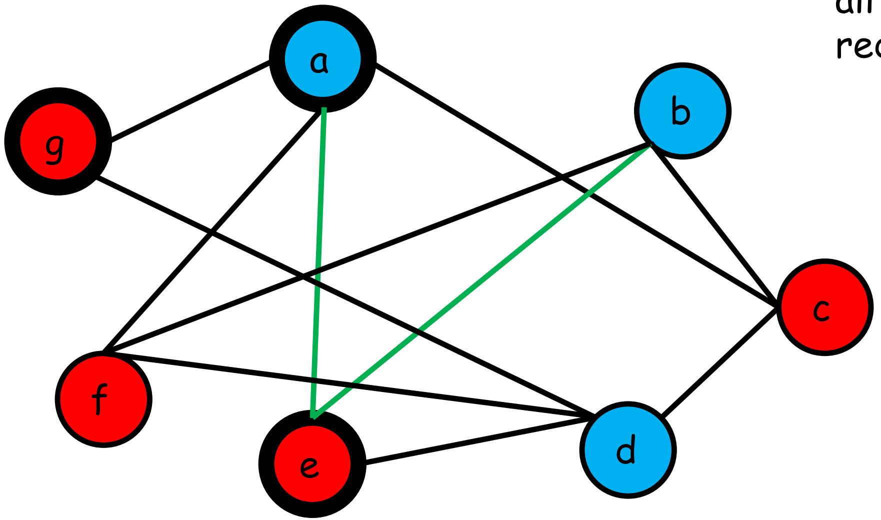
3. Identify all uncolored vertices that are adjacent to a red vertex.
Color them blue.

How to decide if a graph is bipartite or not



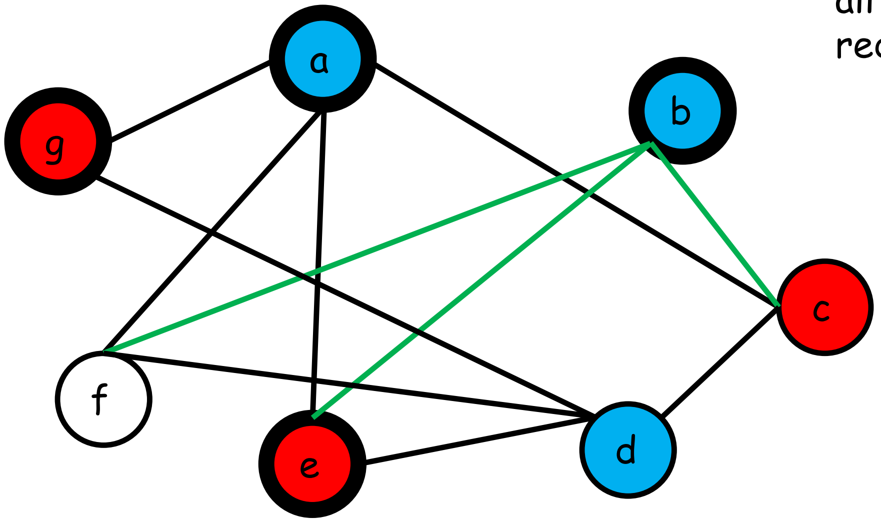
Repeat steps 2 and 3 until all the vertices are colored red or blue.

How to decide if a graph is bipartite or not



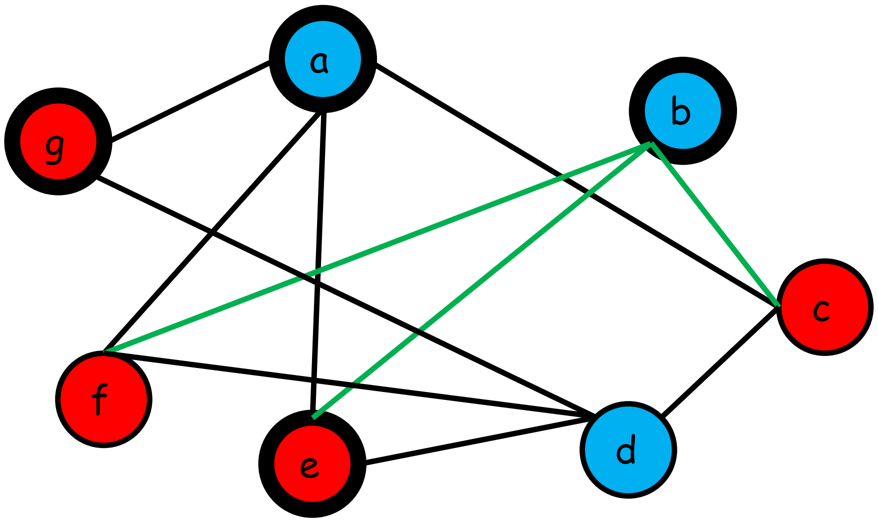
Repeat steps 2 and 3 until all the vertices are colored red or blue.

How to decide if a graph is bipartite or not



Repeat steps 2 and 3 until all the vertices are colored red or blue.

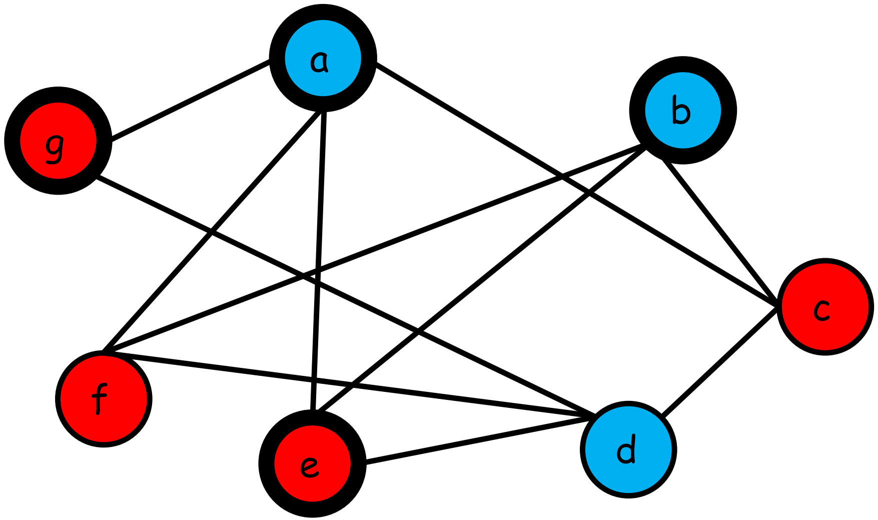
How to decide if a graph is bipartite or not



If there are any two vertices adjacent of the same color, then your graph is not bipartite, otherwise it is bipartite

∴Bipartite graph

How to decide if a graph is bipartite or not



Disjoint sets
V1 = {a, b, d}
V2 = {c, e, f, g}