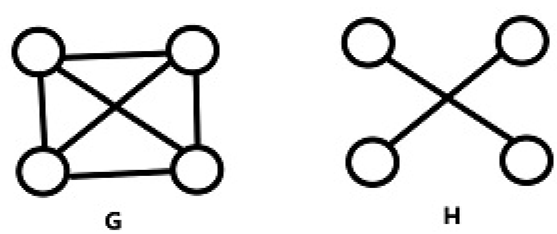
Week 2

Few types of Graphs, Isomorphism, Decomposition, Connectivity and Bipartite Graph*

2.1 Subgraph

Definition 2.1. A **subgraph** of a graph G is a graph H such that $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$.
where $V(H)$,$V(G)$ are vertex set of H and G and $E(H)$,$E(G)$ are Edge set of H and G.

Example:



In the above example, clearly $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$.Hence H is Subgraph of G.

2.2 Independent Set

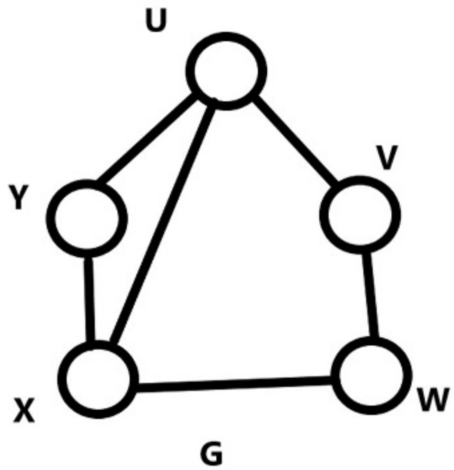
Definition 2.2. Independent Set:A set of pairwise non-adjacent vertices is called as **Independent Set**.It is also called as **stable set**.

*Lecturer: Anand Mishra. Scribe: Kishore Kumar Kalathur Chenchu(M21CS058).

2.3 Clique

Definition 2.3. **Clique:**A set of pairwise adjacent vertices is called as **Clique**.

Example:

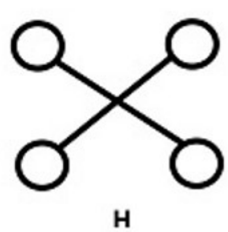


Independent Set: {u,w} is independent set of size 2.Here,we don't have independent set of size 3.
Clique: {u,x,y} is clique of size 3.

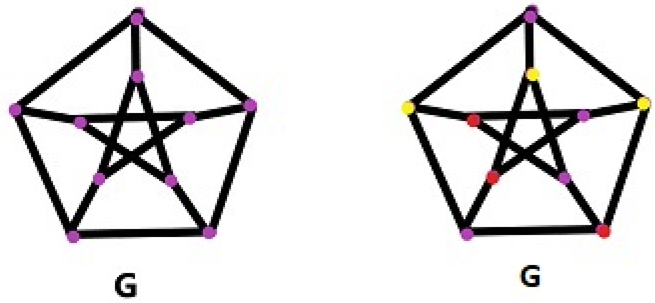
2.4 Chromatic Number

Definition 2.4. The Chromatic number $\chi(G)$ of a graph G, is the minimum number of colors needed to label the vertices so that adjacent vertices receive different color.

Example:



$\chi(H) = 2$

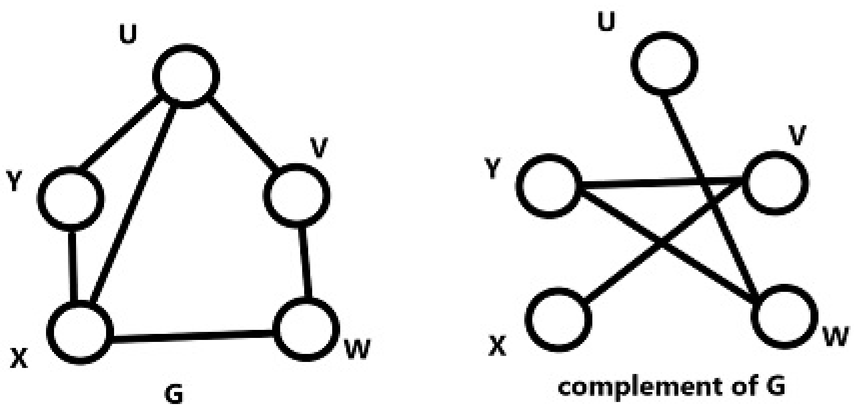


$$\chi(G) = 3$$

2.5 Complement of a Graph

Definition 2.5. The complement $\bar{G}$ of a simple graph G is the simple graph with vertex set $V(G)$ defined by $uv \in E(\bar{G})$ iff $uv \notin E(G)$.

Example:



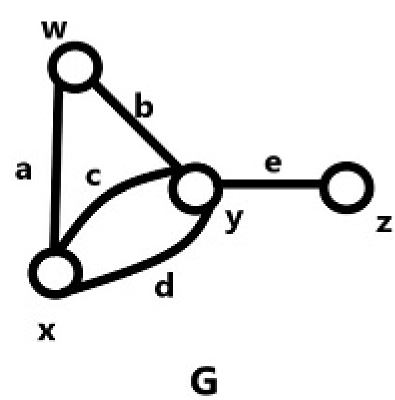
In the above example,Complement of G is nothing but the graph with edges which are not present in the G.

2.6 Representation of Graphs

Few ways to Represent graphs are:
1.Adjacency Matrix Representation
2.Incident Matrix Representation
3.Linked List Representation

1.Adjacency Matrix Representation: An adjacency matrix of a graph G denoted as $A(G)$ is $|V|X|V|$ size square matrix whose $(i-j)^{th}$ element denote number of edges between i^{th} and j^{th} vertices. ($|V|$ = number of vertices)

Example:



$$\mathbf{A(G)} = \begin{array}{c c} & \begin{array}{cccc} w & x & y & z \end{array} \\ \begin{array}{c} w \\ x \\ y \\ z \end{array} & \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 2 & 0 \\ 1 & 2 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

Here, $\mathbf{A(G)}$ is the Adjacency matrix.

For Example if we consider Face book graph as the adjacency matrix representation then each node represent a person,then if we take one row and sum up all the values of that particular row then we obtain the number of friends of a particular person.(In general we get the degree of that node.)

Advantages of Adjacency Matrix Representation:

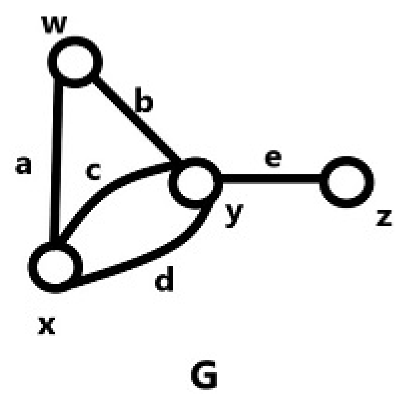
1.If we want to perform operations like adding/deletion or check whether the vertices are adjacent or not very frequently, then it is recommended to use adjacency matrix since we can perform these operations in constant time.
2.If the graph contains edges in order of V^2, then it is better to use adjacency matrix as compared to adjacency list.

Disadvantages of Adjacency Matrix Representation:

1.Traversing the graph using BFS/DFS requires $O(V^2)$ time in case of Adjacency Matrix.

2.Incident Matrix Representation: An incident matrix of a graph G denoted as $M(G)$ is $|V|X|V|$ size square matrix whose $(i-j)^{th}$ element=1 if i^{th} vertex is an endpoint of j^{th} edge. ($|V|$ = number of vertices,$|E|$ = number of edges)

Example:

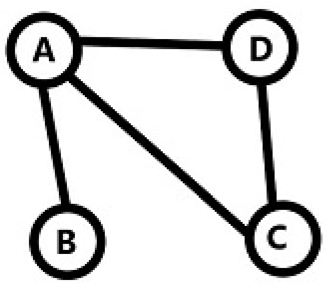


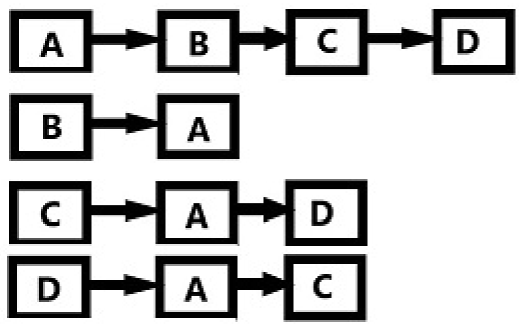
$$\mathbf{M(G)} = \begin{array}{c} \\ w \\ x \\ y \\ z \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

Here, **M(G)** is the Incident matrix.

3.Linked List Representation: It is also called as adjacency list representation.An adjacency list represents a graph as an array of linked lists. The index of the array represents a vertex and each element in its linked list represents the other vertices that form an edge with the vertex.

Example:





In the above figure, Here,we take nodes as one set and we maintain a links(edges associated with that node).This representation needs $O(V + E)$ space.

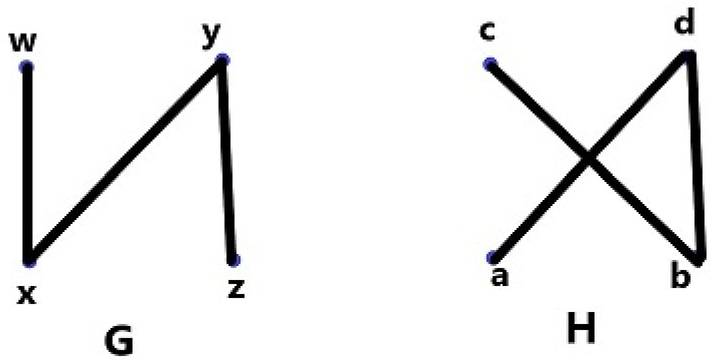
Advantages of Adjacency list Representation:
1.The Adjacency list allows us to compactly represent a sparse graph.
2.Traversing the graph using BFS/DFS requires $O(V + E)$ time in case of Adjacency List.
3.The Adjacency list also allows us to easily find all the links that are directly connected to a particular vertex.

Disadvantages of Adjacency list Representation:
1.Checking if an edge belongs to the graph requires $O(V)$ or $O(E)$ time.

2.7 Isomorphism

Definition 2.6. An Isomorphism from a simple graph G to a simple graph H is a bijection $f : V(G) \to V(H)$ *suchthat* $uv \in E(G)$ iff $f(u)f(v) \in E(H)$.

Example:



Here, in the above example G and H are isomorphic as

$$\begin{aligned} f(w) &= c \\ f(x) &= b \\ f(y) &= d \\ f(z) &= a \end{aligned}$$

Theorem 2.7. *The isomorphic relation is an equivalence relation on the set of simple graphs.*

Proof. In order to prove equivalence relation it is enough to prove the following 3 facts:
a)Set of graphs follows Reflexive Relation
b)Set of graphs follows Symmetric Relation
c)Set of graphs follows Transitive Relation

Let G_1 , G_2 and G_3 be three Graphs with corresponding Vertices and edges as $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$ and $G_3 = (V_3, E_3)$

a)Reflexive Relation: $\forall$G,G$_1$ $\cong$G$_1$
Here Isomorphism says that we need function $f : V_1 \to V_1$ which is Identity function. Hence,Clearly we can say that set of graphs follows Reflexive Relation.

b)Symmetric Relation: To prove symmetric relation it is enough to prove that
If $G_1 \cong G_2$ then $G_2 \cong G_1$.

Given $G_1 \cong G_2$,
$\Rightarrow \exists$ a bijective function f such that f:$V_1 \rightarrow V_2$
$\Rightarrow \exists$ a bijective function f^{-1} such that f^{-1} : $V_2 \rightarrow V_1$
$\Rightarrow \exists G_2 \cong G_1$
Hence,Clearly we can say that set of graphs follows Symmetric Relation.

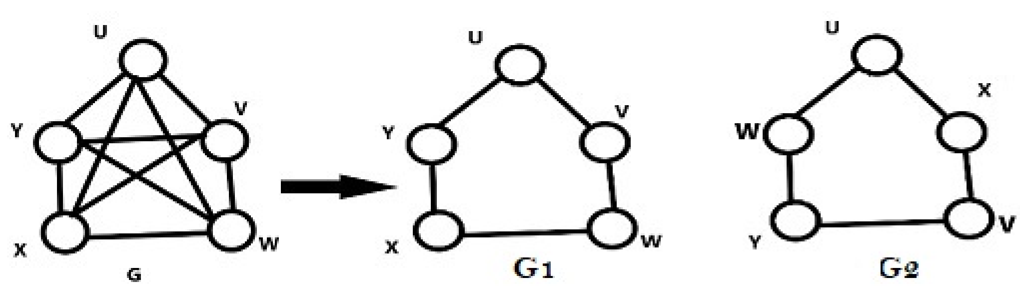
c)Transitive Relation: To prove Transitive relation it is enough to prove that
If $G_1 \cong G_2$ and $G_2 \cong G_3$ then $G_1 \cong G_3$.
Given $G_1 \cong G_2$ and $G_2 \cong G_3$
$\Rightarrow \exists$ bijective functions f, g such that f:$V_1 \rightarrow V_2$ and g: $V_2 \rightarrow V_3$
$\Rightarrow \exists$ a bijective function $g o f$ such that $g o f$:$V_1 \rightarrow V_3$
$\Rightarrow \exists G_1 \cong G_3$
Hence,Clearly we can say that set of graphs follows Transitive Relation.
Hence,we can conclude that isomorphic relation is an equivalence relation on the set of simple graphs.

□

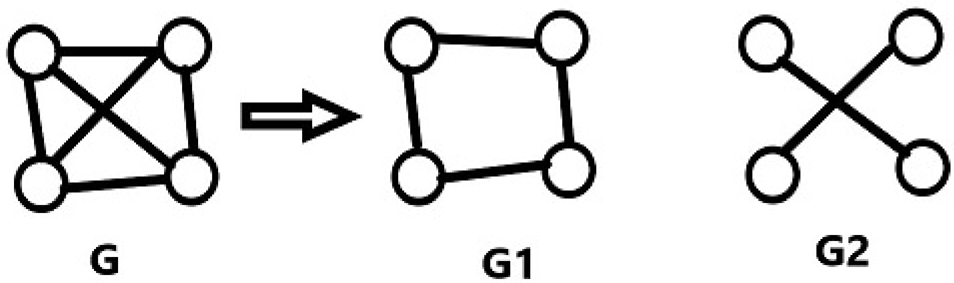
2.8 Decomposition of a Graph

Definition 2.8. Decomposition of a graph is a list of subgraphs such that each edge appears in exactly once in the subgraphs.

Examples of decomposition:
1. K_5 can be decomposed into two C_5 's



2. k_4 cannot be decomposed into two C_4 's



3. k_n an be decomposed into any n-vertex graph and its complement.
4. k_n an be decomposed into $k_{1,n-1}$ and k_{n-1} .

2.9 General notations followed to represent type of graphs

$P_n \rightarrow$ Path graph with n-vertices.
$\mathrm{C}_n \rightarrow$ Cycle graph with n-vertices.
$\mathrm{K}_n \rightarrow$ Complete graph with n-vertices.
$\mathrm{K}_{m,n} \rightarrow$ Bipartite graph between two independent sets.
$\mathrm{N}_n \rightarrow$ Null graph with n-vertices.
$\mathrm{W}_n \rightarrow$ Wheel graph with n-vertices.
$\mathrm{Q}_n \rightarrow$ Hypercube graph with n-vertices.

2.10 Frequent concepts used in Graph connectivity

Few Frequent concepts generally used in Graph connectivity are:
1.Path
2.Walk
3.Trail
4.Circuit
5.Cycle

Definition 2.9. A **Path** in a graph is a sequence of edges where each edge incident to the next without repeating vertices.

Definition 2.10. A **Walk** in a graph is a sequence of edges such that each edge (except the first one) starts with a vertex where the previous edges ended.

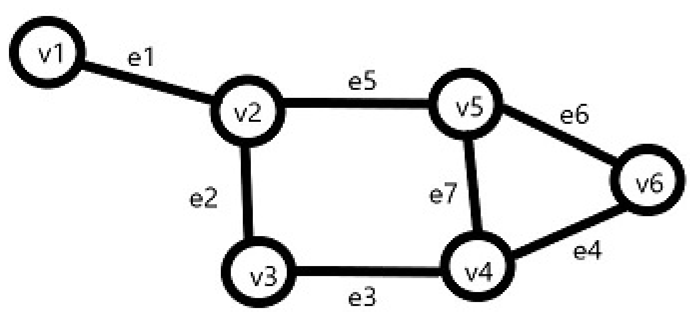
- A walk is said to be an **open walk** if the starting and ending vertices are different.
- A walk is said to be a **closed walk** if the starting and ending vertices are identical.
- The length of a walk is number of edges in it.
- A path is a walk where all edges are distinct.

Definition 2.11. An open walk with out repeated edges is called as a **Trail**.

Definition 2.12. A closed Trial is called as a **Circuit**.

Definition 2.13. Traversing a graph such that we do not repeat a vertex nor we repeat a edge but the starting and ending vertex must be same is called as a **Cycle**.

Example:



Here in the above example,

Path: From Vertex V_1 $\rightarrow$V$_6$ is (V_1,V_2,V_3,V_4,V_6) is one possible Path.

Walk:(V_1,V_2,V_3,V_4,V_6,V_5,V_4) is a Walk and also (V_1,V_2,V_3,V_4,V_5,V_6,V_4,V_5,V_2,V_1) is a Walk.Here, (V_1,V_2,V_3,V_4,V_6,V_5,V_4) is an Open Walk and (V_1,V_2,V_3,V_4,V_5,V_6,V_4,V_5,V_2,V_1) is a Closed Walk.

Trail:(V_1,V_2,V_3,V_4,V_6,V_5,V_4) is a Trail.

Circuit:(V_4,V_6,V_5,V_2,V_3,V_4) is a Circuit.

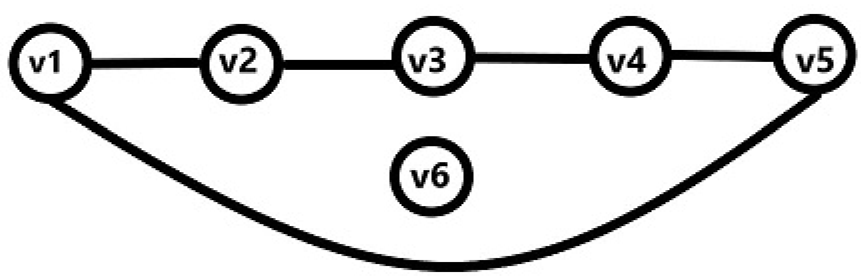
Cycle:(V_2,V_5,V_4,V_3,V_2) is a Cycle, (V_2,V_5,V_6,V_4,V_3,V_2) is also Cycle, (V_1,V_2,V_5,V_4,V_3,V_2) is not a Cycle.

2.11 Connected and disconnected graph

Definition 2.14. A graph is said to be connected if there is a path between any two pair of vertices in that graph.Otherwise,it is called as disconnected graph.

2.12 Connected components

Definition 2.15. Connected component of Graph G is maximal connected subgraphs of G.(in other words, those connected subgraphs which are not contained in larger connected subgraphs of G.)



Here we have two connected components and $V1$,$V2$,$V3$,$V4$,$V5$,$V1$ is maximally connected component.

Theorem 2.16. *Every graph with n vertices and k edges has at least $(n-k)$ components.*

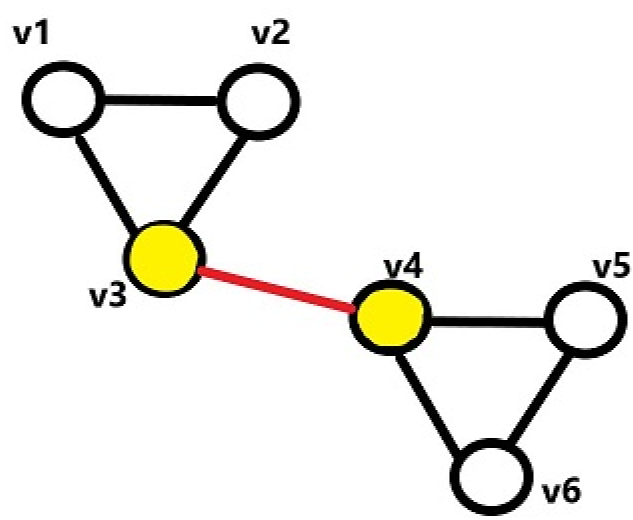
Proof. With k-edges maximum number of nodes that can be connected such that number of connected components = 1 is $(k+1)$.
Hence,Remaining number of nodes $= n-(k+1)$.
Hence,number of connected components $= n-(k+1)$.
Total number of connected components $= n-(k+1)+1 = (n-k)$.
So,Minimum components $= (n-k)$ which is the minimal.

Hence, we can conclude that every graph with n vertices and k edges has at least $(n-k)$ components.

□

2.13 Cut-edge and Cut-Vertex

Definition 2.17. A **cut-edge** or **cut-vertex** of a graph is an edge or vertex whose deletion increases the number of components.



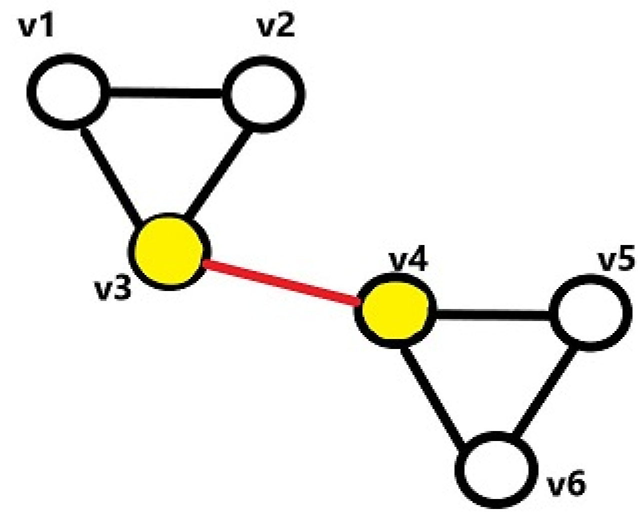
Here,

Cut-edge: $(V_3\ ,\ V_4)$
Cut-Vertices: V_3 and V_4

Theorem 2.18. *An edge is a cut-edge if and only if it belongs to no cycle.*

Proof. Here we need to prove two things:
1) Let an edge is a cut-edge then it does not belong to the cycle.
2) Let an edge does not belong to a cycle then it is a cut-edge.

The above two statements can be easily proven by taking example :



In the above figure, we can clearly state that (V_3 , V_4) is a cut-edge as it doesn't belong to the cycle V_1,V_2,V_3,V_1 and V_4,V_5,V_6,V_4.
Hence, we can say that if an edge is a cut-edge then it does not belong to the cycle.
In the above figure, we can clearly see that (V_3 , V_4) doesn't belong to the cycle V_1,V_2,V_3,V_1 and V_4,V_5,V_6,V_4 and it is a cut-edge.
Hence, we can say that if an edge does not belong to a cycle then it is a cut-edge.

Hence, we can conclude that An edge is a cut-edge if and only if it belongs to no cycle. □

2.14 Bipartite Graph

Definition 2.19. A bipartite graph is a graph whose vertices can be divided into two disjoint and independent sets U and V such that every edge connects a vertex in U to one in V. In a bipartite graph, we have two sets of vertices U and V (known as bi partitions) and each edge is incident on one vertex in U and one vertex in V.

Example: C_6 is a Bipartite Graph.

Theorem 2.20. *A graph is bipartite iff it has no odd cycle.*

Proof. Here we need to prove two things:
1) If a graph is Bipartite then it has no odd cycle.
2) If a graph has no odd Cycle then it is Bipartite.

Let G be a bipartite graph with two independent sets as A and B.In order to have a cycle in a Bipartite graph we need to traverse from A to B to A or B to A to B one or more times. Therefore, we can say that Bipartite graph cannot have odd cycle as we need to come back to place where we have started. Therefore, a Bipartite graph cannot have odd cycle.
So,if a bipartite graph G has no odd cycle then it means that G doesn't have a cycle or it contains even cycle.
When a Graph contains even cycle then we place first vertex in set A and next adjacent vertex in set B like that.It forms a bipartite graph.When Graph doesn't have a cycle obviously graph G can be Bipartite.

Hence, we can conclude that graph is bipartite iff it has no odd cycle. □

Examples of Bipartite Graph :

1.A simple Path is a Bipartite Graph as alternate vertices are placed in two independent sets.
2.C_n is a Bipartite Graph iff n is even.It is the realization from the above proof.

2.15 Questions discussed in Class(Slido.com)

Q1) What will be the chromatic number for an empty graph having n vertices?

A)0

B)1

C)2

D)n

Q2) What will be the chromatic number for a tree having more than 1 vertex?

A)0

B)1

C)2

D)varies with type of tree

Q3) The chromatic number of star graph with 3 vertices is greater than that of a tree with same number of vertices.

A)True

B)False

Answers for above Questions:

Q1) B
Q2) C
Q3) B