Date ______ Page ______

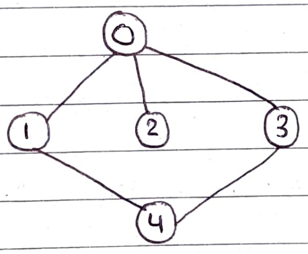
Day - 203

Graph - 7

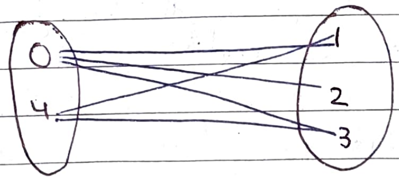
* Bipartite Graph:

⇒ It is a graph in which the vertices can be divided into two distinct sets such that no 2 vertices within the same set are adjacent.

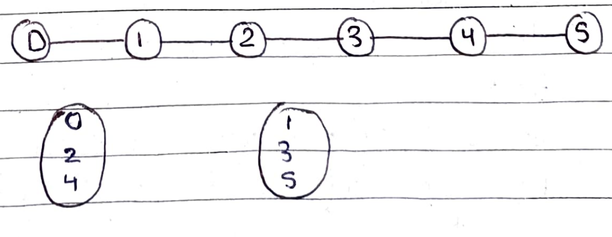
Ex:

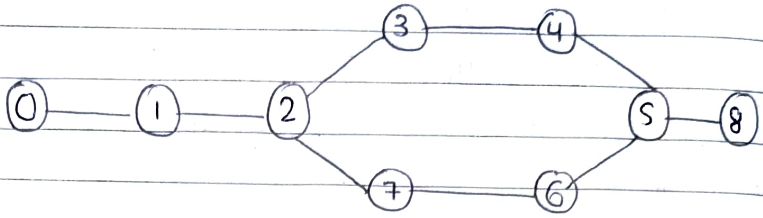


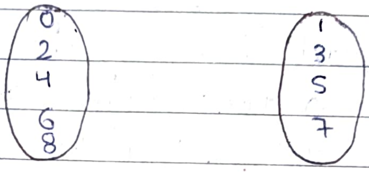
⇒

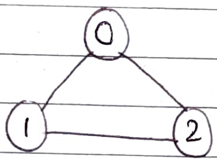


⇒



⇒ 

⇒ 

⇒ 

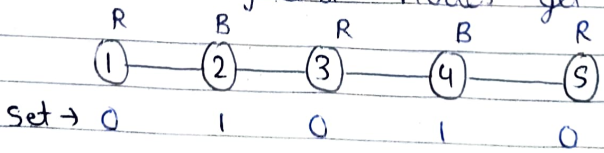
⇒ This is not a biparite graph.

⇒ So, now, how to find any graph that is biparite or not?

⇒ For this, we have a algo that is 2- coloring algo.

⇒ Here, we will use two colors — Red and Blue.

⇒ we will color nodes in such a way that no two adjacent nodes get same color.



Date ____________ Page ________

⇒ Also, if the length of the cycle in the graph is of even length then the graph is biparite otherwise not.

⇒ So, we have to traverse the graph.

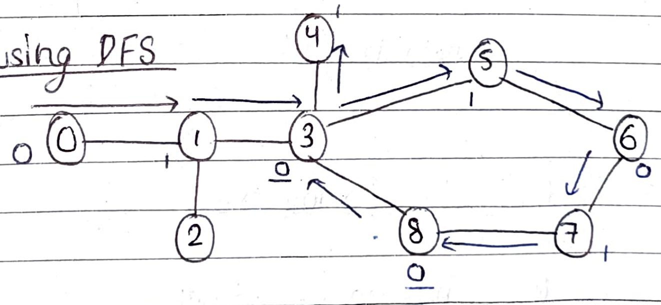
⇒ First, we will use BFS.

⇒ we will take a colour array & init it with -1.

⇒ Also, only those neighbours will go in the queue that ~~didn~~ don't get the colour yet.

⇒ And if any neighbour get the same color, return 1 i.e. the graph is biparite.

By using DFS



⇒ Got same colour of 3 & 8 nodes.

⇒ So, this is not a biparite graph.

⇒ Also, if a node traverse all their neighbours then it should return 1 if biparite ~~and~~ or 0 if not.

Date                    Page

Code

```
bool   isBip( int node, vector<int> adj[],
                  vector <int> &color){
       for(int j=0; j< adj[node].size(); j++){
              if (color[adj[node][j]] == -1){
                  color [adj[node][j]] =
                        ((color[node] +1) % 2;
              if(!isBip( adj [node][j] , adj, color))
                     return 0;
              } else{
                  if( color[node] == color[
                        adj[node][j]] )
                     return 0;
              }
       }
       return 1;
}
```

⇒ Real Life Examples:

⇒ Recommendation System.

⇒ Stable Marriage System.