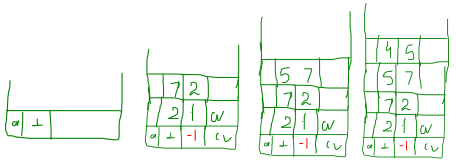
No Color → −1

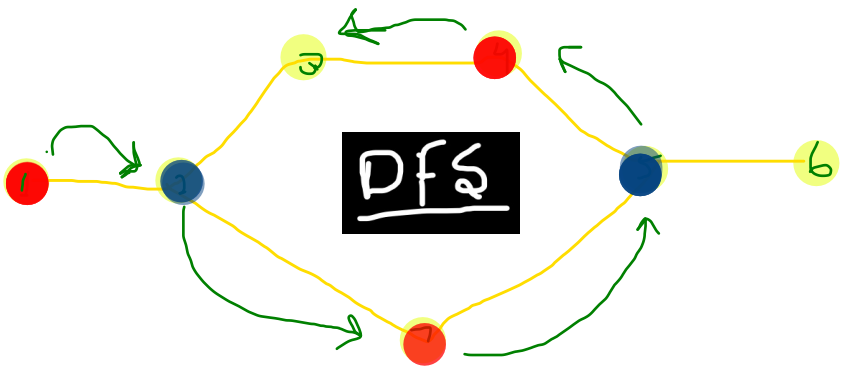
R → 1

B → 2

color vector

1	2		1	2		1
1	2	3	4	5	6	7



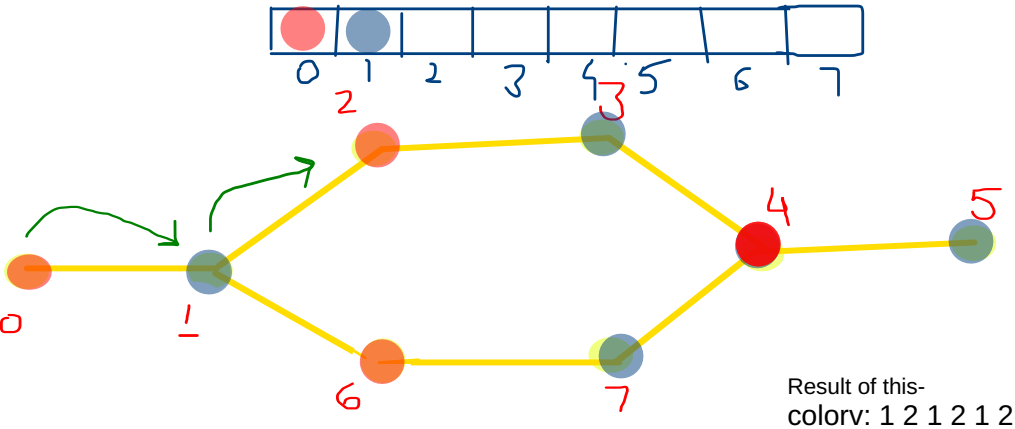


caller

for i = 1 : n

dfs(adj, i, −1, cv)

	3	4	
	4	5	
	5	7	
	7	2	
	2	1	w
α	⊥	-1	cv




```
bool checkBipartite(vector<int> adj[], int V) {
    vector<int> colorv(V, -1);

    for(int i=0; i<V; ++i) {
        if(colorv[i] == -1) {
            colorv[i] = 1;
            if(!colorGraphDFS(adj, i, colorv)) return false;
        }
    }

    for(auto x: colorv) {
        cout << x << ' ';
    }
    cout << endl;

    return true;
}
```

Result of this-

colorv: 1 2 1 2 1 2 1 2

1. graphs that can be colored using exactly 2 colors such that no two adjacent nodes have same colors.

2. Also, it a graph has odd length cycle it's not a bipartite graph else it's a bipartite graph.

```
bool colorGraphDFS(vector<int> adj[], int node, vector<int> &colorv){

    for(auto child: adj[node]) {
        if(colorv[child] == -1) { // if that node is not visited or it's uncolored
            colorv[child] = (colorv[node] == 1 ? 2 : 1); // then color it with alternate color that it's parent has.

            if(!colorGraphDFS(adj, child, colorv)) return false;
        } else if(colorv[child] == colorv[node]) return false;
    }
    return true;
}
```