Adjacency Matrix

```
public class AdjacencyGraphMatrixDemo
{
  private int V;//number of Vertices in Graph
  private int E;//number of edges in Graph
  private int[][] adjMatrix;

  public AdjacencyGraphMatrixDemo(int nodes)
  {
    this.V=nodes;
    this.E=0;
    this.adjMatrix=new int[nodes][nodes];
  }

  public void addEdge(int u, int v)
  {
    adjMatrix[u][v]=1;
    adjMatrix[v][u]=1;//because it is an undirected graph
    E++;
  }

  public String toString()
  {
    StringBuilder sb=new StringBuilder();
    sb.append(V+ " vertices, "+E+" edges "+"\n");
    for(int v=0;v<V;v++)
    {
      sb.append(v+": ");
```

```
            for(int w : adjMatrix[v])
            {
            sb.append(w+" ");
            }
            sb.append("\n");
        }

        return sb.toString();
    }

    public static void main(String[] args)
    {
        AdjacencyGraphMatrixDemo g=new AdjacencyGraphMatrixDemo(4);
        g.addEdge(0, 1);
        g.addEdge(1, 2);
        g.addEdge(2, 3);
        g.addEdge(3, 0);

        System.out.println(g);
    }
}
```

Output

```
4 vertices, 4 edges
0: 0 1 0 1
1: 1 0 1 0
2: 0 1 0 1
3: 1 0 1 0
```

Adjacency List

```
import java.util.LinkedList;

public class AdjacencyGraphListDemo

{

    private LinkedList<Integer>[] adj;

    private int V;//number of Vertices

    private int E;//number of Edges

    public AdjacencyGraphListDemo(int nodes)

    {

        this.V=nodes;

        this.E=0;

        this.adj=new LinkedList[nodes];


        for (int v=0; v<V;v++)

        {

            adj[v]=new LinkedList<>();

        }

    }


    public void addEdge(int u,int v)

    {

        this.adj[u].add(v);

        this.adj[v].add(u);

        E++;

    }

    public String toString()

    {

        StringBuilder sb=new StringBuilder();
        sb.append(V+" vertices, "+E+" edges "+"\n");
        for(int v=0;v<V;v++)
        {
            sb.append(v+": ");
            for(int w: adj[v])
            {
                sb.append(w+" ");
            }
            sb.append("\n");
        }
        return sb.toString();
    }
    public static void main(String[] args)
    {
        AdjacencyGraphListDemo g=new AdjacencyGraphListDemo(4);
        g.addEdge(0, 1);
        g.addEdge(1, 2);
        g.addEdge(2, 3);
        g.addEdge(3, 0);
        System.out.println(g);
    }
}
```

Output

```
4 vertices, 4 edges
0: 1 3
1: 0 2
2: 1 3
3: 2 0
```

Breadth/Depth First Search

```
import java.util.LinkedList;

import java.util.Queue;

import java.util.Stack;

public class DepthFirstSearch

{

  private LinkedList<Integer>[] adj;

  private int V;//number of Vertices

  private int E;//number of Edges


  public DepthFirstSearch(int nodes)

  {

    this.V=nodes;

    this.E=0;

    this.adj=new LinkedList[nodes];


    for (int v=0; v<V;v++)

    {

      adj[v]=new LinkedList<>();

    }

  }


  public void addEdge(int u,int v)

  {

    this.adj[u].add(v);

    this.adj[v].add(u);

    E++;

  }
```

```
public String toString()

{

    StringBuilder sb=new StringBuilder();

    sb.append(V+" vertices, "+E+" edges "+"\n");

    for(int v=0;v<V;v++)

    {

        sb.append(v+": ");

        for(int w: adj[v])

        {

            sb.append(w+" ");

        }

        sb.append("\n");

    }

    return sb.toString();

}


public void bfs(int s)

{

    boolean[] visited =new boolean[V];


    Queue<Integer> q=new LinkedList<>();

    visited[s]=true;

    q.offer(s);

    while(!q.isEmpty())

    {

        int u=q.poll();

        System.out.print(u+" ");

        for(int v:adj[u])
        {
            if(!visited[v])
            {
                visited[v]=true;
                q.offer(v);
            }
        }
    }

}

public void dfs(int s)
{
    boolean[] visited=new boolean[V];
    Stack<Integer> stack=new Stack<>();
    stack.push(s);
    while(!stack.isEmpty())
    {
        int u=stack.pop();
        if(!visited[u])
        {
            visited[u]=true;
            System.out.print(u+" ");
            for(int v: adj[u])
            {
                if(!visited[v])
                {
                    stack.push(v);
                }
                }
            }
        }
    }

    public static void main(String[] args)
    {
        DepthFirstSearch g=new DepthFirstSearch(5);
        g.addEdge(0, 1);
        g.addEdge(1, 2);
        g.addEdge(2, 3);
        g.addEdge(3, 0);
        g.addEdge(4, 2);

        System.out.println(g);

        System.out.println("Breadth First Search: ");
        g.bfs(0);

        System.out.println();
        System.out.println("Deep First Search: ");
        g.dfs(0);
    }
}
```

Output

```
5 vertices, 5 edges

0: 1 3

1: 0 2

2: 1 3 4

3: 2 0

4: 2


Breadth First Search:

0 1 3 2 4

Deep First Search:

0 3 2 4 1
```