W3D4_Solution

1. Adjacency Matrix

Vertex order used: **A B C D E F G H I**

Edges from the diagram:
A–B, A–C, A–F
B–F
C–F, C–G
F–H
H–G
D–E, E–I, D–I

Adjacency matrix:

	A	B	C	D	E	F	G	H	I
A	0	1	1	0	0	1	0	0	0
B	1	0	0	0	0	1	0	0	0
C	1	0	0	0	0	1	1	0	0
D	0	0	0	0	1	0	0	0	1
E	0	0	0	1	0	0	0	0	1
F	1	1	1	0	0	0	0	1	0
G	0	0	1	0	0	0	0	1	0
H	0	0	0	0	0	1	1	0	0
I	0	0	0	1	1	0	0	0	0

2. Java Program for Finding Components (DFS)

```
import java.util.*;

public class DFSComponents {

   static void dfs(int u, int[][] a, boolean[] vis, List<Integer> comp) {
      vis[u] = true;
      comp.add(u);

      for (int v = 0; v < a.length; v++) {
         if (a[u][v] == 1 && !vis[v]) {
            dfs(v, a, vis, comp);
         }
      }
   }

   public static void main(String[] args) {

      int[][] a = {
         {0,1,1,0,0,1,0,0,0},
         {1,0,0,0,0,1,0,0,0},
         {1,0,0,0,0,1,1,0,0},
         {0,0,0,0,1,0,0,0,1},
```

```
        {0,0,0,1,0,0,0,0,1},
        {1,1,1,0,0,0,0,1,0},
        {0,0,1,0,0,0,0,1,0},
        {0,0,0,0,0,1,1,0,0},
        {0,0,0,1,1,0,0,0,0}
    };

    boolean[] vis = new boolean[a.length];

    for (int i = 0; i < a.length; i++) {
      if (!vis[i]) {
        List<Integer> comp = new ArrayList<>();
        dfs(i, a, vis, comp);
        System.out.println("Component: " + comp);
      }
    }
  }
}
```

---

3. Java Program for Finding Components (BFS)

```
import java.util.*;

public class BFSComponents {

  static void bfs(int start, int[][] a, boolean[] vis, List<Integer> comp) {
    Queue<Integer> q = new LinkedList<>();
    q.add(start);
    vis[start] = true;

    while (!q.isEmpty()) {
      int u = q.poll();
      comp.add(u);

      for (int v = 0; v < a.length; v++) {
        if (a[u][v] == 1 && !vis[v]) {
          vis[v] = true;
          q.add(v);
        }
      }
    }
  }

  public static void main(String[] args) {

    int[][] a = {
      {0,1,1,0,0,1,0,0,0},
      {1,0,0,0,0,1,0,0,0},
      {1,0,0,0,0,1,1,0,0},
      {0,0,0,0,1,0,0,0,1},
      {0,0,0,1,0,0,0,0,1},
      {1,1,1,0,0,0,0,1,0},
      {0,0,1,0,0,0,0,1,0},
      {0,0,0,0,0,1,1,0,0},
      {0,0,0,1,1,0,0,0,0}
    };

    boolean[] vis = new boolean[a.length];

    for (int i = 0; i < a.length; i++) {
```

```
        if (!vis[i]) {
            List<Integer> comp = new ArrayList<>();
            bfs(i, a, vis, comp);
            System.out.println("Component: " + comp);
        }
    }
  }
}
```