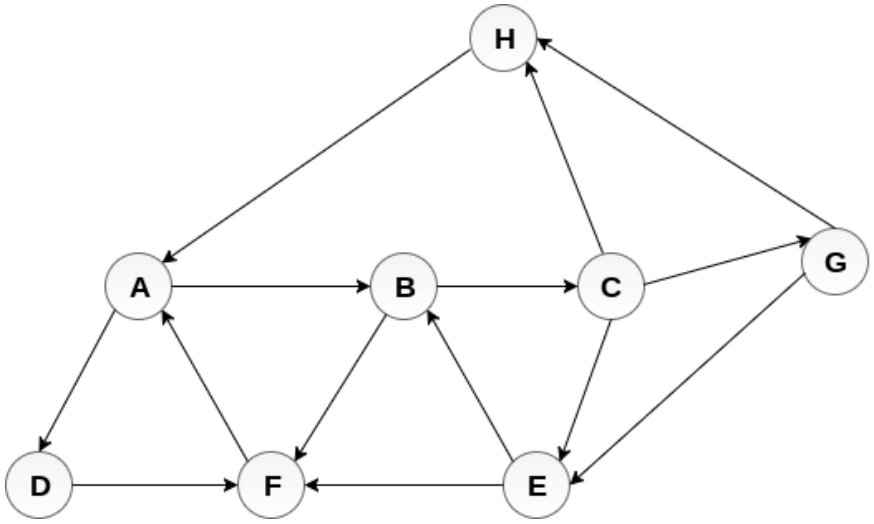
```
import java.io.*;
import java.util.*;


class Graph {

   private HashMap<String, LinkedList<String>> adj;

   private boolean isDirected = true;

   Graph() {
       adj = new HashMap<String, LinkedList<String>>();
   }

   void addEdge(String v, String w) {
       if (!adj.containsKey(v))
           adj.put(v, new LinkedList<String>());

       adj.get(v).add(w);

       if (!isDirected) {
           if (!adj.containsKey(w))
               adj.put(w, new LinkedList<String>());

           adj.get(w).add(v);
       }
   }

   boolean DFS(String v, String d, HashSet<String> visitSet) {
       HashSet<String> visited = visitSet == null ? new HashSet<String>() :
visitSet;
       visited.add(v);
       System.out.print(v + " ");

       if (v.equals(d)) {
           return true;
       }

       Iterator<String> i = adj.get(v).listIterator();
       while (i.hasNext()) {
           String n = i.next();
           if (!visited.contains(n))
               if (DFS(n, d, visited))
                   return true;
       }
```

```
        return false;
    }

    void BFS(String s, String d) {
        HashSet<String> visited = new HashSet<String>();

        LinkedList<String> queue = new LinkedList<String>();

        visited.add(s);
        queue.add(s);

        while (queue.size() != 0) {
            s = queue.poll();
            System.out.print(s+" ");

            if (s.equals(d))
                return;

            Iterator<String> i = adj.get(s).listIterator();
            while (i.hasNext()) {
                String n = i.next();
                if (!visited.contains(n)) {
                    visited.add(n);
                    queue.add(n);
                }
            }
        }
    }

    public static void main(String args[]) {
        Graph g = new Graph();

        g.addEdge("H", "A");
        g.addEdge("A", "D");
        g.addEdge("A", "B");
        g.addEdge("B", "F");
        g.addEdge("B", "C");
        g.addEdge("C", "E");
        g.addEdge("C", "G");
        g.addEdge("C", "H");
        g.addEdge("G", "H");
        g.addEdge("G", "E");
        g.addEdge("E", "F");
        g.addEdge("E", "B");
        g.addEdge("F", "A");
```

```
        g.addEdge("D", "F");

        System.out.println("Following is Depth First Traversal H -> E:");
        g.DFS("H", "E", null);

        System.out.println("\n\nFollowing is Breadth First Traversal H -> E:");
        g.BFS("H", "E");
    }
}
```



Adjacency Lists

A : B, D
B : C, F
C : E, G, H
G : E, H
E : B, F
F : A
D : F
H : A

OUTPUT :-

```
Following is Depth First Traversal H -> E:
H A D F B C E

Following is Breadth First Traversal H -> E:
H A C G D B F E
```