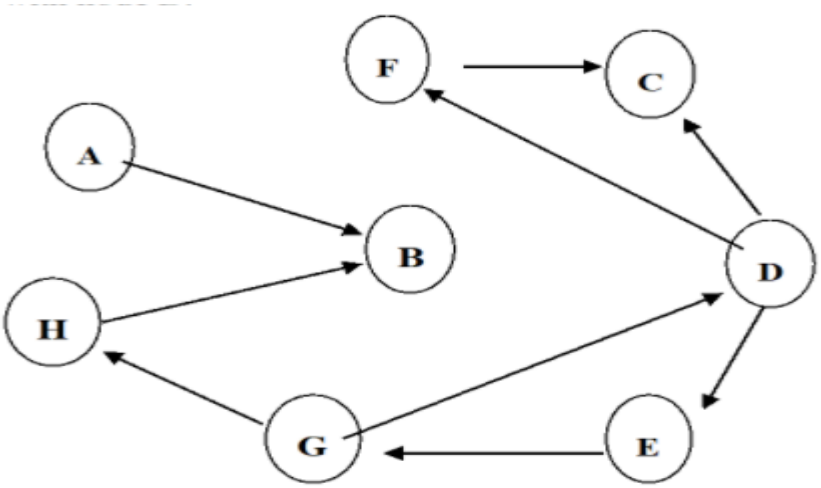
Graphs

1. Write a Program to implement DFS algorithm and print the DFS sequence for below graph start with node D.




CODE-

```
import java.util.*;

class Graph {

 private LinkedList<Character> adjLists[];

 private boolean visited[];

 Graph(int vertices) {

  adjLists = new LinkedList[vertices];

  visited = new boolean[vertices];

  for (int i = 0; i < vertices; i++)

   adjLists[i] =new LinkedList<Character>();

  }

 void addEdge(char src, char dest) {
```

```
  adjLists[src%8].add(dest);
}
void DFS(int vertex) {
  visited[vertex%8] = true;
  System.out.print((char)vertex + " ");
  Iterator<Character> ite =adjLists[vertex%8].listIterator();
  while (ite.hasNext()) {
    int adj = ite.next();
    if (!visited[adj%8])
       DFS(adj);
  }
}
public static void main(String args[]) {
   Graph g = new Graph(8);
   g.addEdge('A' , 'B' );
   g.addEdge('H' , 'B' );
   g.addEdge('G' , 'H' );
   g.addEdge('G' , 'D' );
   g.addEdge('E' , 'G' );
   g.addEdge('D' , 'E' );
   g.addEdge('D' , 'F' );
```

```
    g.addEdge('D' , 'C' );
    g.addEdge('F' , 'C' );
    System.out.println(" Depth First Traversal of the graph");
    g.DFS('D');
  }
}
```

```java
import java.util.*;
class Graph {
  private LinkedList<Character> adjLists[];
  private boolean visited[];
  Graph(int vertices) {
    adjLists = new LinkedList[vertices];
    visited = new boolean[vertices];
    for (int i = 0; i < vertices; i++)
      adjLists[i] =new LinkedList<Character>();
    }
  void addEdge(char src, char dest) {
    adjLists[src%8].add(dest);
  }
  void DFS(int vertex) {
    visited[vertex%8] = true;
    System.out.print((char)vertex + " ");
    Iterator<Character> ite =adjLists[vertex%8].listIterator();
    while (ite.hasNext()) {
      int adj = ite.next();
      if (!visited[adj%8])
         DFS(adj);
    }
  }
  public static void main(String args[]) {
     Graph g = new Graph(8);
     g.addEdge('A' , 'B' );
     g.addEdge('H' , 'B' );
```

```
24    public static void main(String args[]) {
25        Graph g = new Graph(8);
26        g.addEdge('A' , 'B' );
27        g.addEdge('H' , 'B' );
28        g.addEdge('G' , 'H' );
29        g.addEdge('G' , 'D' );
30        g.addEdge('E' , 'G' );
31        g.addEdge('D' , 'E' );
32        g.addEdge('D' , 'F' );
33        g.addEdge('D' , 'C' );
34        g.addEdge('F' , 'C' );
35        System.out.println(" Depth First Traversal of the graph");
36        g.DFS('D');
37    }
38 }
```

OUTPUT-

```
Output
java -cp /tmp/qeGsBUTpVs Graph
Depth First Traversal of the graph
D E G H B F C
```

2. Write a Program to implement BFS Algorithm and print the BFS sequence start with node A.

CODE-

```
import java.util.LinkedList;

import java.util.Queue;
```

```
public class Graph {
  private int vertex;
  private Queue <Character> que;
  private LinkedList<Character> adj[];
  Graph(int v) {
    vertex = v;
    adj = new LinkedList [vertex];
    for (int i = 0; i < v; i++) {
      adj[i] = new LinkedList<>();
    }
    que = new LinkedList<Character>();
  }
  void insertEdge(char v, char w) {
     adj[v%8].add(w);
  }
  void BFS(char n) {
    boolean nodes[] = new boolean[vertex];
    char a = 0;
    nodes[n%8] = true;
    que.add(n);
    while (que.size() != 0) {
```

```
    n = que.poll();
    System.out.print((char)(n) + " ");
    for (int i = 0; i < adj[n%8].size(); i++) {
      a = adj[n%8].get(i);
      if (!nodes[a%8]) {
        nodes[a%8] = true;
        que.add(a);
      }
    }
  }
}
public static void main(String args[]) {
   Graph Graph = new Graph(8);
   Graph.insertEdge('A' , 'B' );
   Graph.insertEdge('H' , 'B' );
   Graph.insertEdge('G' , 'H' );
   Graph.insertEdge('G' , 'D' );
   Graph.insertEdge('E' , 'G' );
   Graph.insertEdge('D' , 'E' );
   Graph.insertEdge('D' , 'C' );
   Graph.insertEdge('D' , 'F' );
```

Graph.insertEdge('F' , 'C' );

System.out.println("Breadth First Traversal for the Graph from node A is:");

Graph.BFS('A');

}

}

```
import java.util.LinkedList;
import java.util.Queue;
public class Graph {
  private int vertex;
  private Queue <Character> que;
  private LinkedList<Character> adj[];
  Graph(int v) {
    vertex = v;
    adj = new LinkedList [vertex];
    for (int i = 0; i < v; i++) {
      adj[i] = new LinkedList<>();
    }
    que = new LinkedList<Character>();
  }
  void insertEdge(char v, char w) {
     adj[v%8].add(w);
  }
  void BFS(char n) {
    boolean nodes[] = new boolean[vertex];
    char a = 0;
    nodes[n%8] = true;
    que.add(n);
    while (que.size() != 0) {
      n = que.poll();
      System.out.print((char)(n) + " ");
      for (int i = 0; i < adj[n%8].size(); i++) {
        a = adj[n%8].get(i);
```

```
24        n = que.poll();
25        System.out.print((char)(n) + " ");
26        for (int i = 0; i < adj[n%8].size(); i++) {
27          a = adj[n%8].get(i);
28          if (!nodes[a%8]) {
29            nodes[a%8] = true;
30            que.add(a);
31          }
32        }
33      }
34    }
35    public static void main(String args[]) {
36       Graph Graph = new Graph(8);
37       Graph.insertEdge('A' , 'B' );
38       Graph.insertEdge('H' , 'B' );
39       Graph.insertEdge('G' , 'H' );
40       Graph.insertEdge('G' , 'D' );
41       Graph.insertEdge('E' , 'G' );
42       Graph.insertEdge('D' , 'E' );
43       Graph.insertEdge('D' , 'C' );
44       Graph.insertEdge('D' , 'F' );
45       Graph.insertEdge('F' , 'C' );
46       System.out.println("Breadth First Traversal for the Graph from node A is:"
             );
47       Graph.BFS('A');
48    }
49  }
```

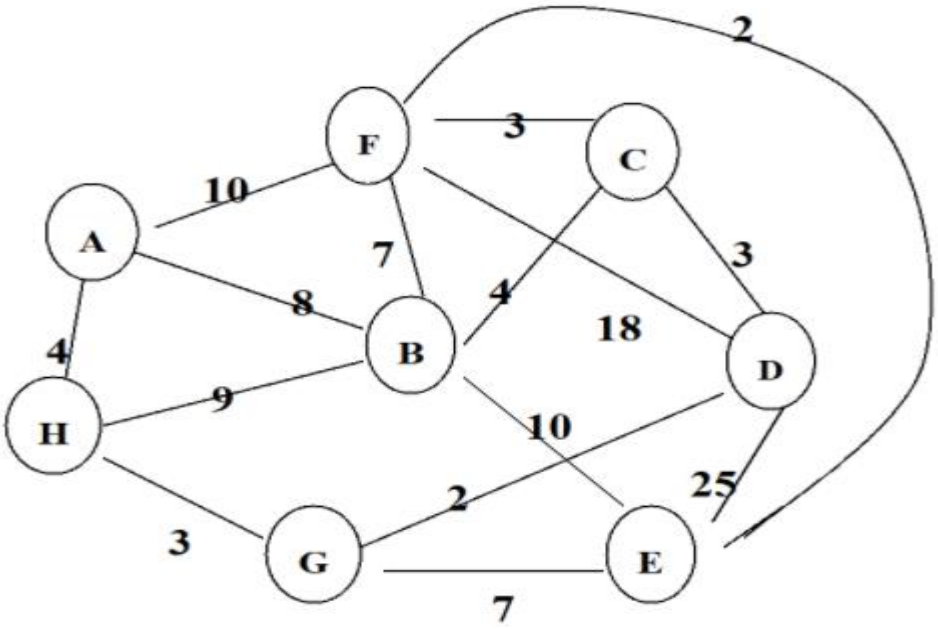
OUTPUT-

```
Output

java -cp /tmp/qeGsBUTpVs Graph
Breadth First Traversal for the Graph from node A is:
A B
```

Challenging:

3. Write a Program to Implement Prim's Algorithm and find the minimum spanning tree for the given graph.




CODE-

```
import java.io.*;

import java.lang.*;

import java.util.*;


class MST {


   // Number of vertices in the graph

   private static final int V = 8;
```

```
                String arr[] = new String[]
{"A","B","C","D","E","F","G","H"};



   int minKey(int key[], Boolean mstSet[])
   {
      // Initialize minimum value
      int min = Integer.MAX_VALUE, min_index = -1;

      for (int v = 0; v < V; v++)
         if (mstSet[v] == false && key[v] < min) {
            min = key[v];
            min_index = v;
         }

      return min_index;
   }

   // A utility function to print the constructed MST

   void printMST(int parent[], int graph[][])
```

```
{
    System.out.println("Edge \tWeight");
    for (int i = 1; i < V; i++)
        System.out.println(arr[parent[i]] + " - " + arr[i] + "\t"
                           + graph[i][parent[i]]);
}

// Function to construct and print MST for a graph
// represented using adjacency matrix representation
void primMST(int graph[][])
{
    int parent[] = new int[V];
    int key[] = new int[V];

    // To represent set of vertices included in MST
    Boolean mstSet[] = new Boolean[V];

    // Initialize all keys as INFINITE
    for (int i = 0; i < V; i++) {
        key[i] = Integer.MAX_VALUE;
        mstSet[i] = false;
```

```
}



key[0] = 0;
parent[0] = -1;
for (int count = 0; count < V - 1; count++) {

   int u = minKey(key, mstSet);

   // Add the picked vertex to the MST Set
   mstSet[u] = true;



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



      if (graph[u][v] != 0 && mstSet[v] == false
         && graph[u][v] < key[v]) {
         parent[v] = u;
         key[v] = graph[u][v];
      }
```

```
        }

        // print the constructed MST
        printMST(parent, graph);
    }

    public static void main(String[] args)
    {
        MST t = new MST();
        int graph[][] = new int[][] { { 0, 8, 0, 0, 0, 10, 0, 4 }, //A
                                      { 8, 0, 4, 0, 10, 7, 0, 9 }, //B
                                      { 0, 4, 0, 3, 0, 3, 0, 0 }, //C
                                      { 0, 0, 3, 0, 25, 0, 2, 0 }, //D
                                                                   { 0, 10, 0,
25, 0, 2, 7, 0 }, //E
                                      { 10, 7, 3, 0, 2, 0, 0, 0 }, //F
                                      { 0, 0, 0, 2, 7, 0, 0, 3 }, //G
                                      { 4, 9, 0, 0, 0, 0, 3, 0 }, //H
                                    };

        // Print the solution
```

```
        t.primMST(graph);
    }
}
```

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

class MST {

    // Number of vertices in the graph
    private static final int V = 8;

        String arr[] = new String[] {"A","B","C","D","E","F","G","H"};


    int minKey(int key[], Boolean mstSet[])
    {
        // Initialize minimum value
        int min = Integer.MAX_VALUE, min_index = -1;

        for (int v = 0; v < V; v++)
            if (mstSet[v] == false && key[v] < min) {
                min = key[v];
                min_index = v;
            }

        return min_index;
    }
```

```
26
27      // A utility function to print the constructed MST
28
29      void printMST(int parent[], int graph[][])
30      {
31          System.out.println("Edge \tWeight");
32          for (int i = 1; i < V; i++)
33              System.out.println(arr[parent[i]] + " - " + arr[i] + "\t"
34                                 + graph[i][parent[i]]);
35      }
36
37      // Function to construct and print MST for a graph
38      // represented using adjacency matrix representation
39      void primMST(int graph[][])
40      {
41          int parent[] = new int[V];
42          int key[] = new int[V];
43
44          // To represent set of vertices included in MST
45          Boolean mstSet[] = new Boolean[V];
46
47          // Initialize all keys as INFINITE
48          for (int i = 0; i < V; i++) {
49              key[i] = Integer.MAX_VALUE;
50              mstSet[i] = false;
51          }
52
```

```
54          key[0] = 0;
55          parent[0] = -1;
56          for (int count = 0; count < V - 1; count++) {
57
58              int u = minKey(key, mstSet);
59
60              // Add the picked vertex to the MST Set
61              mstSet[u] = true;
62
63
64              for (int v = 0; v < V; v++)
65
66
67                  if (graph[u][v] != 0 && mstSet[v] == false
68                      && graph[u][v] < key[v]) {
69                      parent[v] = u;
70                      key[v] = graph[u][v];
71                  }
72          }
73
74          // print the constructed MST
75          printMST(parent, graph);
76      }
77
```

```
        // print the constructed MST
        printMST(parent, graph);
    }

    public static void main(String[] args)
    {
        MST t = new MST();
        int graph[][] = new int[][] { { 0, 8, 0, 0, 0, 10, 0, 4 }, //A
                                      { 8, 0, 4, 0, 10, 7, 0, 9 }, //B
                                      { 0, 4, 0, 3, 0, 3, 0, 0 }, //C
                                      { 0, 0, 3, 0, 25, 0, 2, 0 }, //D
                                      { 0, 10, 0, 25, 0, 2, 7, 0 }, //E
                                      { 10, 7, 3, 0, 2, 0, 0, 0 }, //F
                                      { 0, 0, 0, 2, 7, 0, 0, 3 }, //G
                                      { 4, 9, 0, 0, 0, 0, 3, 0 }, //H
                                    };

        // Print the solution
        t.primMST(graph);
    }
}
```

OUTPUT-

```
Output

java -cp /tmp/nUW0F7yUYZ MST
Edge    Weight
C - B   4
D - C   3
G - D   2
F - E   2
C - F   3
H - G   3
A - H   4
```

----------X----------

Thank you!