Breadth First Search Implementation

Pseudocode:

```
BFS(G, s)
 1  for each vertex u □ V [G] - {s}
 2       do color[u] ← WHITE
 3          d[u] ← ∞
 4          π[u] ← NIL
 5  color[s] ← GRAY
 6  d[s] ← 0
 7  π[s] ← NIL
 8  Q ← Ø
 9  ENQUEUE(Q, s)
10  while Q ≠ Ø
11       do u ← DEQUEUE(Q)
12          for each v □ Adj[u]
13              do if color[v] = WHITE
14                    then color[v] ← GRAY
15                         d[v] ← d[u] + 1
16                         π[v] ← u
17                         ENQUEUE(Q, v)
18          color[u] ← BLACK
```

1. Apply BFS on a graph that you will take as input as a matrix. Use Java as the language for implementation.

   The template of the code is given below –

```
import java.util.Scanner;
public class BFS{
   public static void main(String [] args){
      Scanner sc = new Scanner(System.in);
      int[][] graph = takeInputGraph(sc);
      System.out.println("Give input of the source node");
      int s = sc.nextInt();
      bfs(graph,s);
   }

   public static int[][] takeInputGraph(Scanner sc){
      System.out.println("Input the number of nodes in the graph");
      int node = sc.nextInt();
      System.out.println("Input the number of edges in the graph");
      int edge = sc.nextInt();
      int[][] mat = new int[node][node];
      for(int c=0; c<edge; c++){
         System.out.println("Enter the first node of the "+(c+1)+"th edge");
         int node1 = sc.nextInt();
         System.out.println("Enter the second node of the "+(c+1)+"th edge");
         int node2 = sc.nextInt();
         mat[node1][node2] = 1;
         mat[node2][node1] = 1;
      }
      return mat;
   }

   public static void bfs(int[][] g, int s){

   }
}
```