```
Print("************************BFS*****************")

Def breadth_first_search(graph_structure, starting_node):
  Explored = set()
  Exploration_queue = deque([starting_node])

  While exploration_queue:
    Current_node = exploration_queue.popleft()
    If current_node not in explored:
      Print(current_node)
      If current_node == "G":
        Print("Search stopped!")
        Exit(1)
      Explored.add(current_node)
      Exploration_queue.extend(neighbor for neighbor in graph_structure[current_node] if neighbor
not in explored)

Class Network:
  Def __init__(self):
    Self.adjacency_list = {}

  Def add_vertex(self, vertex):
    If vertex not in self.adjacency_list:
      Self.adjacency_list[vertex] = []

  Def add_connection(self, vertex1, vertex2):
    If vertex1 in self.adjacency_list and vertex2 in self.adjacency_list:
      Self.adjacency_list[vertex1].append(vertex2)
      Self.adjacency_list[vertex2].append(vertex1)  # For undirected graph
    Def show_graph(self):
        For vertex in self.adjacency_list:
            Print(f'{vertex}: {self.adjacency_list[vertex]}')
Network = Network()
Network.add_vertex('A')
Network.add_vertex('B')
Network.add_vertex('C')
Network.add_vertex('D')
Network.add_vertex('E')
Network.add_vertex('F')
Network.add_vertex('G')
Network.add_vertex('H')
Network.add_vertex('I')
Network.add_vertex('J')
Network.add_vertex('K')
Network.add_vertex('L')
Network.add_vertex('M')
Network.add_vertex('N')

Network.add_connection('A', 'B')
Network.add_connection('A', 'F')
Network.add_connection('A', 'D')
Network.add_connection('A', 'E')
Network.add_connection('B', 'K')
Network.add_connection('B', 'J')
Network.add_connection('K', 'M')
Network.add_connection('K', 'N')
Network.add_connection('D', 'G')
Network.add_connection(‘E’, ‘C’)

Network.add_connection(‘E’, ‘H’)

Network.add_connection(‘E’, ‘I’)

Network.add_connection(‘I’, ‘L’)


Network.show_graph()


Print(“Breadth First Search:”)

Breadth_first_search(network.adjacency_list, “A”)




From collections import deque

Class Network:

  Def __init__(self):

    Self.network_map = {}


  Def add_vertex(self, vertex):

    If vertex not in self.network_map:

      Self.network_map[vertex] = []


  Def add_connection(self, vertex1, vertex2):

    If vertex1 in self.network_map and vertex2 in self.network_map:

      Self.network_map[vertex1].append(vertex2)

      Self.network_map[vertex2].append(vertex1)  # For undirected graph


  Def show_network(self):

    For vertex in self.network_map:

      Print(f’{vertex}: {self.network_map[vertex]}’)
```

```
# Create an instance of Network
Net = Network()
Net.add_vertex(0)
Net.add_vertex(1)
Net.add_vertex(2)
Net.add_vertex(3)
Net.add_vertex(4)

# Add connections between vertices
Net.add_connection(0, 1)
Net.add_connection(0, 4)
Net.add_connection(4, 1)
Net.add_connection(4, 3)
Net.add_connection(2, 1)
Net.add_connection(2, 3)
Net.add_connection(1, 3)

# Display the network
Net.show_network()
```