```
# Jesse Reid CSC6013 E#3

def BFS(V, E):
    for i in range(len(V)):
        V[i] = -1  # All vertices not visited
    count = 0

    # Convert index to letter
    index_to_letter = ["A", "B", "C", "D", "E", "F", "G", "H"]
    letter_to_index = {}
    for i in range(len(index_to_letter)):
        letter_to_index[index_to_letter[i]] = i

    for i in range(len(V)):   # for all possible sources
        if V[i] == -1:
            Q = [i]  # Enqueue the source
            V[i], count = count, count + 1  # Visit it

            # Each time a vertex A is visited print: "Vertex A visited" and the current array V
            if index_to_letter[i] == "A":
                print(f"Vertex {index_to_letter[i]} visited", V)

            while len(Q) != 0:  # For all enqueued
                current = Q[0]

                # Each time a vertex C is dequeued print: "Vertex C dequeued" and the current
queue Q
                if index_to_letter[current] == "C":
                    print(f"Vertex {index_to_letter[current]} dequeued", [index_to_letter[idx] for idx in
Q])

                for e in E:  # Search neighbors
                    if e[0] == index_to_letter[current] and V[letter_to_index[e[1]]] == -1:  # Directed
graph condition
                        Q.append(letter_to_index[e[1]])  # Enqueue it

                        # Each time a vertex B is enqueued print: "Vertex B enqueued" and the current
queue Q
                        if e[1] == "B":
                            print(f"Vertex {e[1]} enqueued", [index_to_letter[idx] for idx in Q])

                        V[letter_to_index[e[1]]], count = count, count + 1  # Visit it

                        # Each time a vertex A is visited print: "Vertex A visited" and the current array V
                        if index_to_letter[i] == "A":
                            print(f"Vertex {index_to_letter[i]} visited", V)
                Q.pop(0)  # Dequeue it

# an array with all verticies holding information if they are visited or not
V = [-1] * 8
# an adjacency list of edges with triplets
E = [
    ["A","E",1], ["A","H",1],
    ["E","C",1],
```

```
    ["H","D",1],
    ["C","F",1], ["C","G",1],
    ["D","A",1], ["D","E",1],
    ["F","D",1], ["F","E",1],
    ["G","B",1], ["G","E",1],
    ["B","A",1]
    ]

BFS(V, E)
print(V)


Python 3.12.0 (v3.12.0:0fb18b02c8, Oct  2 2023, 09:45:56) [Clang 13.0.0 (clang-1300.0.29.30)]
on darwin
Type "help", "copyright", "credits" or "license()" for more information.

===== RESTART: /Users/jreid/Documents/JLR_dev_code/merrimack/CSC6013/E3.py =====
Vertex A visited [0, -1, -1, -1, -1, -1, -1, -1]
Vertex A visited [0, -1, -1, -1, 1, -1, -1, -1]
Vertex A visited [0, -1, -1, -1, 1, -1, -1, 2]
Vertex A visited [0, -1, 3, -1, 1, -1, -1, 2]
Vertex A visited [0, -1, 3, 4, 1, -1, -1, 2]
Vertex C dequeued ['C', 'D']
Vertex A visited [0, -1, 3, 4, 1, 5, -1, 2]
Vertex A visited [0, -1, 3, 4, 1, 5, 6, 2]
Vertex B enqueued ['G', 'B']
Vertex A visited [0, 7, 3, 4, 1, 5, 6, 2]
[0, 7, 3, 4, 1, 5, 6, 2]
```