def bfs\_without\_queue(graph, start):

    visited = [False] \* len(graph)

    to\_visit = [start]

    next\_level = []

    while to\_visit:

        for node in to\_visit:

            if not visited[node]:

                print(node, end=" ")

                visited[node] = True

                next\_level.extend(i for i, is\_connected in enumerate(graph[node]) if is\_connected and not visited[i])

        to\_visit = next\_level

        next\_level = []

graph = [

    [0, 1, 1, 0, 0],

    [1, 0, 1, 1, 0],

    [1, 1, 0, 0, 1],

    [0, 1, 0, 0, 1],

    [0, 0, 1, 1, 0]

]

bfs\_without\_queue(graph, 0)

from collections import deque

class Node:

    def \_\_init\_\_(self, value):

        self.value = value

        self.neighbors = []

    def add\_neighbor(self, node):

        self.neighbors.append(node)

def bfs\_with\_queue(start\_node):

    visited = set()

    queue = deque([start\_node])

    while queue:

        current\_node = queue.popleft()

        if current\_node not in visited:

            print(current\_node.value, end=" ")

            visited.add(current\_node)

            for neighbor in current\_node.neighbors:

                if neighbor not in visited:

                    queue.append(neighbor)

a = Node('A')

b = Node('B')

c = Node('C')

d = Node('D')

e = Node('E')

a.add\_neighbor(b)

a.add\_neighbor(c)

b.add\_neighbor(d)

c.add\_neighbor(e)

bfs\_with\_queue(a)