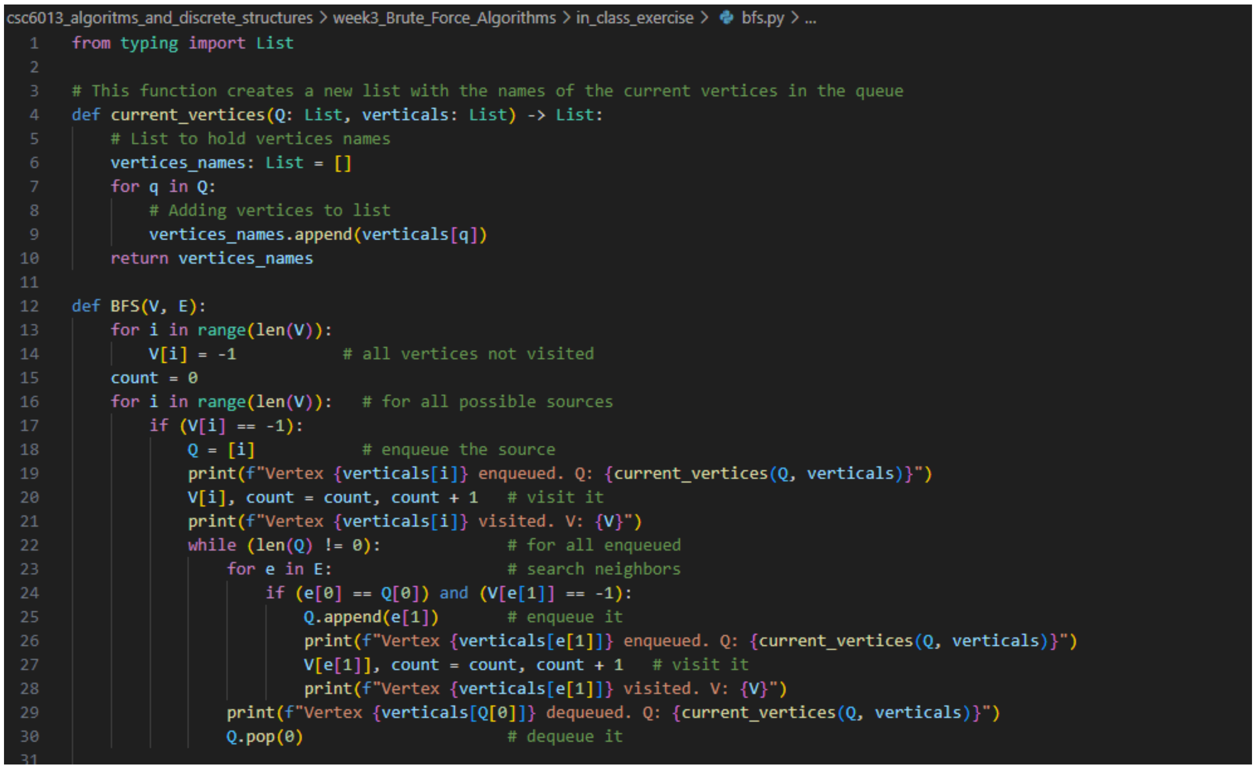
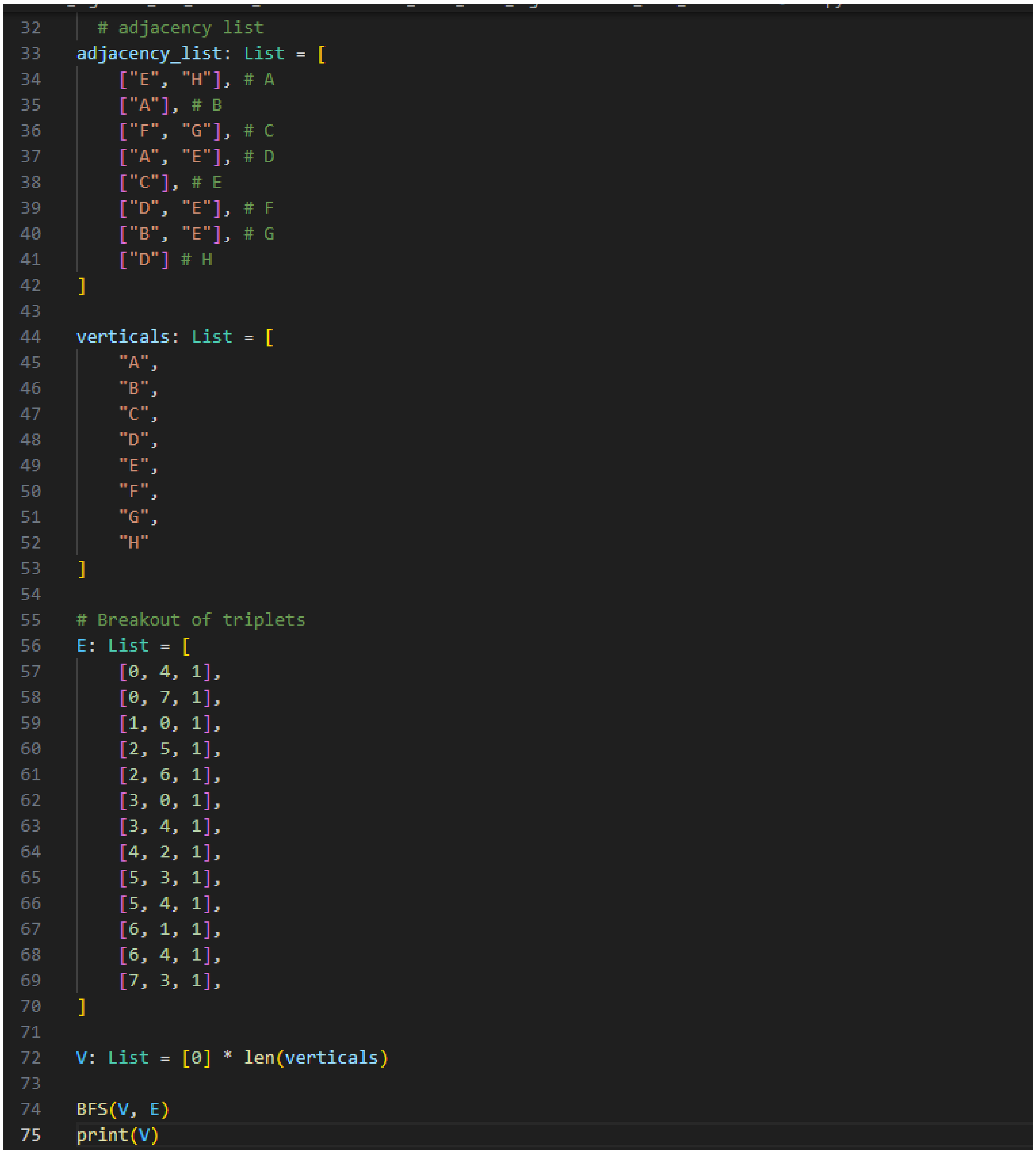

csc6013_algoritms_and_discrete_structures > week3_Brute_Force_Algorithms > in_class_exercise > bfs.py > ...

```python
from typing import List

# This function creates a new list with the names of the current vertices in the queue
def current_vertices(Q: List, verticals: List) -> List:
    # List to hold vertices names
    vertices_names: List = []
    for q in Q:
        # Adding vertices to list
        vertices_names.append(verticals[q])
    return vertices_names

def BFS(V, E):
    for i in range(len(V)):
        V[i] = -1           # all vertices not visited
    count = 0
    for i in range(len(V)):   # for all possible sources
        if (V[i] == -1):
            Q = [i]           # enqueue the source
            print(f"Vertex {verticals[i]} enqueued. Q: {current_vertices(Q, verticals)}")
            V[i], count = count, count + 1   # visit it
            print(f"Vertex {verticals[i]} visited. V: {V}")
            while (len(Q) != 0):             # for all enqueued
                for e in E:                  # search neighbors
                    if (e[0] == Q[0]) and (V[e[1]] == -1):
                        Q.append(e[1])       # enqueue it
                        print(f"Vertex {verticals[e[1]]} enqueued. Q: {current_vertices(Q, verticals)}")
                        V[e[1]], count = count, count + 1   # visit it
                        print(f"Vertex {verticals[e[1]]} visited. V: {V}")
                print(f"Vertex {verticals[Q[0]]} dequeued. Q: {current_vertices(Q, verticals)}")
                Q.pop(0)                     # dequeue it
```



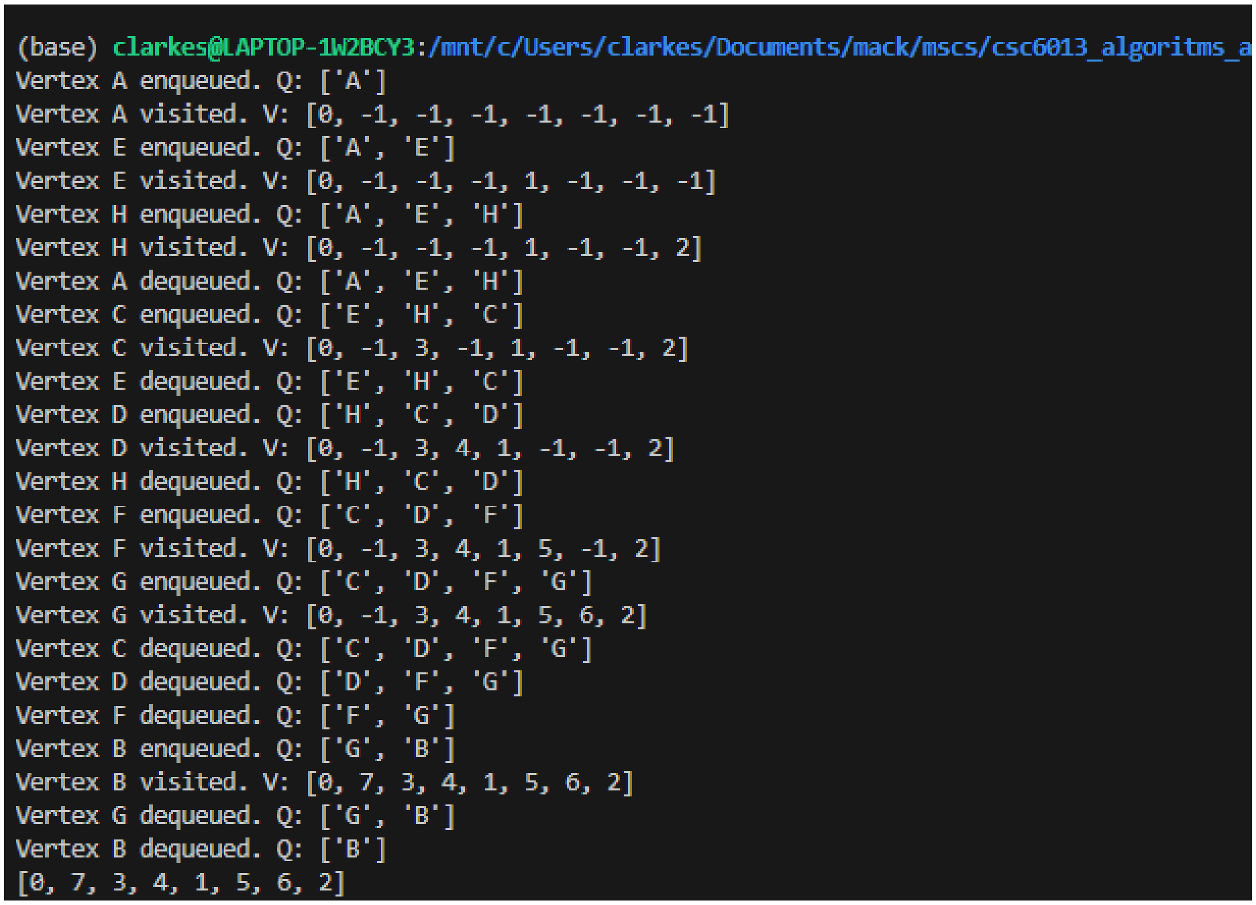
```python
  # adjacency list
adjacency_list: List = [
    ["E", "H"], # A
    ["A"], # B
    ["F", "G"], # C
    ["A", "E"], # D
    ["C"], # E
    ["D", "E"], # F
    ["B", "E"], # G
    ["D"] # H
]

verticals: List = [
    "A",
    "B",
    "C",
    "D",
    "E",
    "F",
    "G",
    "H"
]

# Breakout of triplets
E: List = [
    [0, 4, 1],
    [0, 7, 1],
    [1, 0, 1],
    [2, 5, 1],
    [2, 6, 1],
    [3, 0, 1],
    [3, 4, 1],
    [4, 2, 1],
    [5, 3, 1],
    [5, 4, 1],
    [6, 1, 1],
    [6, 4, 1],
    [7, 3, 1],
]

V: List = [0] * len(verticals)

BFS(V, E)
print(V)
```



```
(base) clarkes@LAPTOP-1W2BCY3:/mnt/c/Users/clarkes/Documents/mack/mscs/csc6013_algoritms_a
Vertex A enqueued. Q: ['A']
Vertex A visited. V: [0, -1, -1, -1, -1, -1, -1, -1]
Vertex E enqueued. Q: ['A', 'E']
Vertex E visited. V: [0, -1, -1, -1, 1, -1, -1, -1]
Vertex H enqueued. Q: ['A', 'E', 'H']
Vertex H visited. V: [0, -1, -1, -1, 1, -1, -1, 2]
Vertex A dequeued. Q: ['A', 'E', 'H']
Vertex C enqueued. Q: ['E', 'H', 'C']
Vertex C visited. V: [0, -1, 3, -1, 1, -1, -1, 2]
Vertex E dequeued. Q: ['E', 'H', 'C']
Vertex D enqueued. Q: ['H', 'C', 'D']
Vertex D visited. V: [0, -1, 3, 4, 1, -1, -1, 2]
Vertex H dequeued. Q: ['H', 'C', 'D']
Vertex F enqueued. Q: ['C', 'D', 'F']
Vertex F visited. V: [0, -1, 3, 4, 1, 5, -1, 2]
Vertex G enqueued. Q: ['C', 'D', 'F', 'G']
Vertex G visited. V: [0, -1, 3, 4, 1, 5, 6, 2]
Vertex C dequeued. Q: ['C', 'D', 'F', 'G']
Vertex D dequeued. Q: ['D', 'F', 'G']
Vertex F dequeued. Q: ['F', 'G']
Vertex B enqueued. Q: ['G', 'B']
Vertex B visited. V: [0, 7, 3, 4, 1, 5, 6, 2]
Vertex G dequeued. Q: ['G', 'B']
Vertex B dequeued. Q: ['B']
[0, 7, 3, 4, 1, 5, 6, 2]
```