In [ ]:

```python
# This file groups adjacency matrices that are
# homomorphic to one another.

import numpy as np
import function_drawing as dr
from collections import defaultdict

#A counting function - to determine the edge sequences
def ones_per_row(matrix): #Another way to find the 'degree' of a vertex
    return [row.count(1) for row in matrix]

#Function takes in a list of matrices---
def group_matrices_by_ones(matrices): #regardless of row order group the
    groups = defaultdict(list)
    llist = []
    keeys = []

    for matrix in matrices:
        ones_count = ones_per_row(matrix)
        ones_count_sorted = tuple(sorted(ones_count))
        groups[ones_count_sorted].append(matrix)

    for key, group in groups.items():
        A = np.array(group)
        ########################################################
        #uncomment this for a pretty view

        # print(f"Group {key}:")
        # print(f"representative matrix")
        # print()
        # print(A[0])

        # #To draw the graph
        # B = dr.draw_graph(A[0])
        # print("=" * 40)

        ########################################################

        llist.append(A[0].tolist())
        keeys.append(key)

    nnn = len(groups)

    #print(f'We can expect {nnn} groups of graphs.')
    #print()


    return llist, keeys
#where the keeys are the edge sequences.
```