In [1]:
```
import function_clique_finder as cf
import function_grouping_homomorphics as gh
import function_adjacency as aj
import function_drawing as dr
import function_bron_kerbosch_clique_finder as bc
import function_matrix_to_edge_connection as mc
import numpy as np
```

In [2]:
```
#step 1- define vars
n = 4
k = 2
l = 2
#step 2 - get adj_matrices
adjm_list = aj.all_adj_matrix(n)
adjm_list2 = gh.group_matrices_by_ones(adjm_list)
```

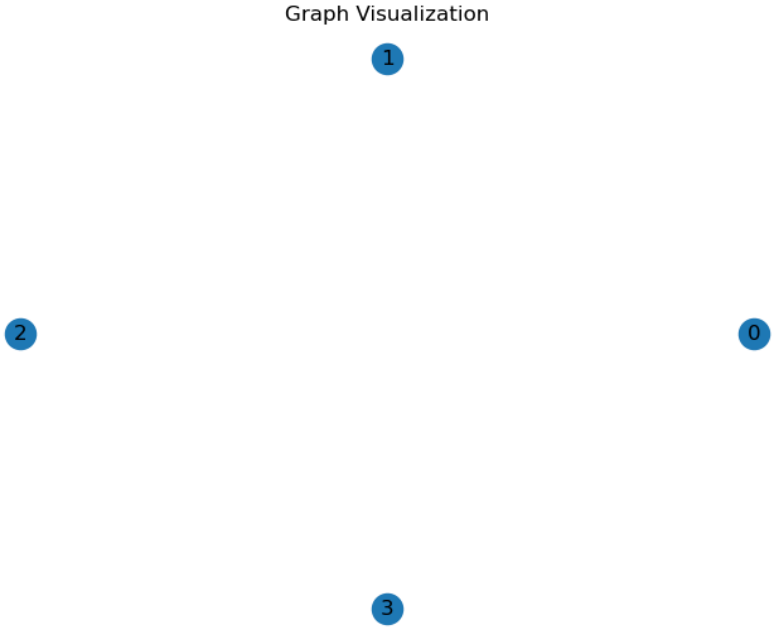
```
We can expect 11 groups of graphs.
```

In [8]:
```
#result2 = cf.find_cliques5(k, l, adjm_list)
```

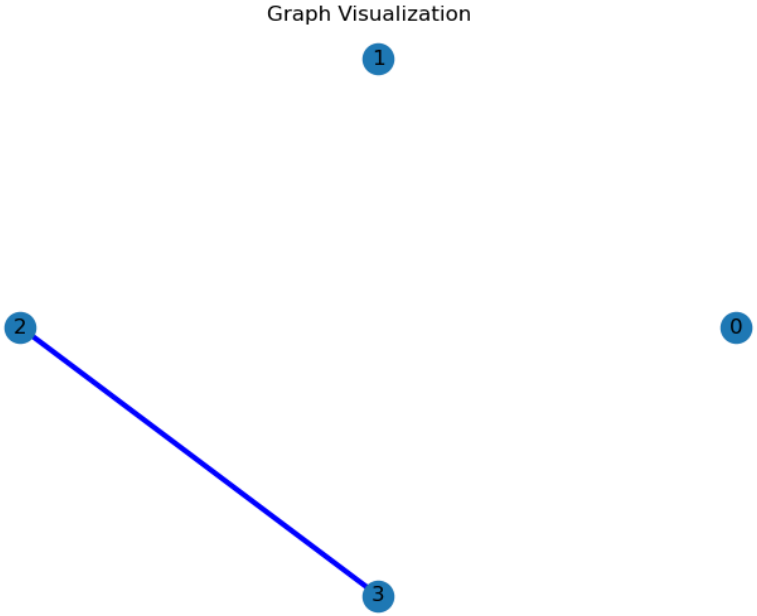
In [4]:
```
b = adjm_list2[0];
```

In [9]:
```
for matrix in b:
    print(np.array(matrix))
    dr.draw_graph_1color(matrix)
    print('here are the connections for graph:')
    dd = mc.adj_mat_dict(matrix)
    print(dd)
    print('here are the cliques in the graph:')
    gg = bc.MaximalCliquesFinder(dd)
    gg.find_cliques()
    gg.print_cliques()

    print()
    print('*****************************************')
```

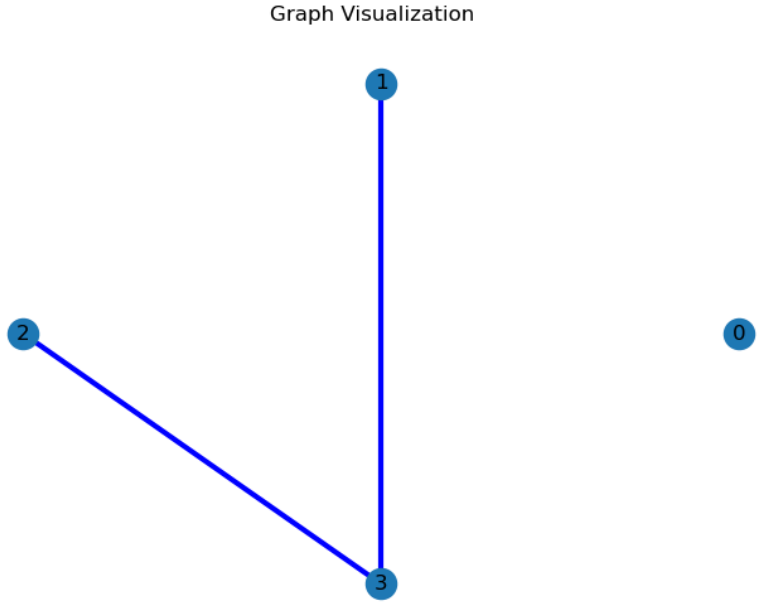
```
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
Graph:
```

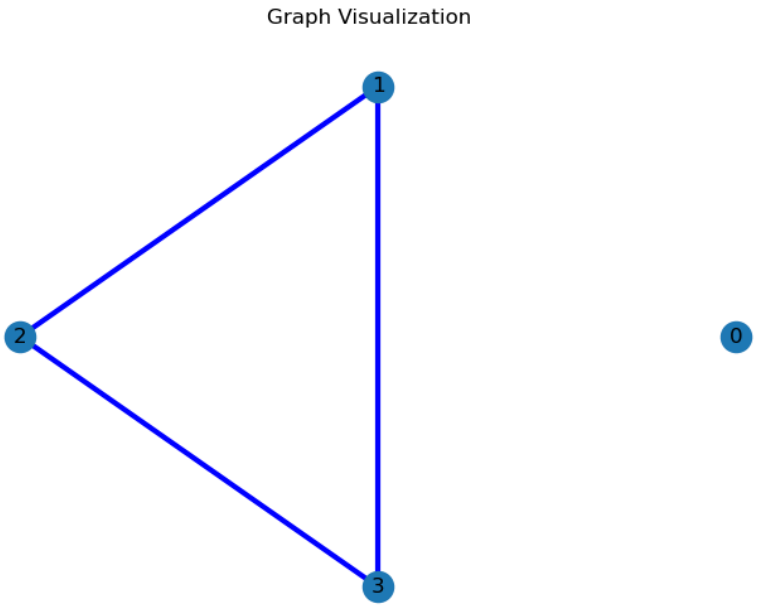

```
<Figure size 400x300 with 0 Axes>
here are the connections for graph:
{0: [], 1: [], 2: [], 3: []}
here are the cliques in the graph:
[0]
[1]
[2]
[3]

*****************************************
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 1.]
 [0. 0. 1. 0.]]
Graph:
```



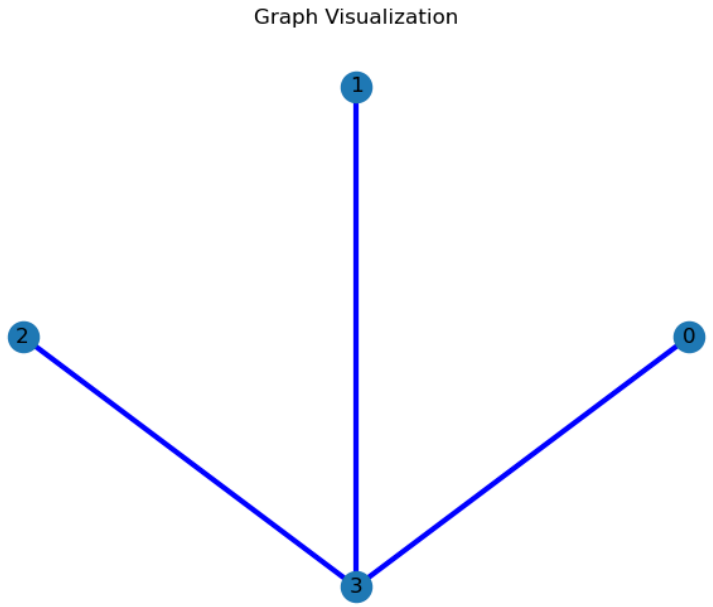

```
<Figure size 400x300 with 0 Axes>
here are the connections for graph:
{0: [], 1: [], 2: [3], 3: [2]}
here are the cliques in the graph:
[0]
[1]
[2, 3]

****************************************
[[0. 0. 0. 0.]
 [0. 0. 0. 1.]
 [0. 0. 0. 1.]
 [0. 1. 1. 0.]]
Graph:
```

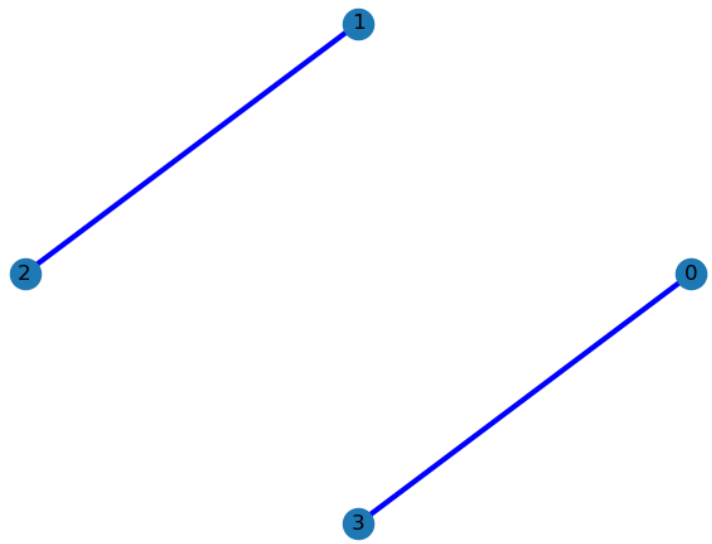

```
<Figure size 400x300 with 0 Axes>
here are the connections for graph:
{0: [], 1: [3], 2: [3], 3: [1, 2]}
here are the cliques in the graph:
[0]
[3, 1]
[3, 2]

****************************************
[[0. 0. 0. 0.]
 [0. 0. 1. 1.]
 [0. 1. 0. 1.]
 [0. 1. 1. 0.]]
Graph:
```




```
<Figure size 400x300 with 0 Axes>
here are the connections for graph:
{0: [], 1: [2, 3], 2: [1, 3], 3: [1, 2]}
here are the cliques in the graph:
[0]
[1, 2, 3]

****************************************
[[0. 0. 0. 1.]
 [0. 0. 0. 1.]
 [0. 0. 0. 1.]
 [1. 1. 1. 0.]]
Graph:
```


```
<Figure size 400x300 with 0 Axes>
here are the connections for graph:
{0: [3], 1: [3], 2: [3], 3: [0, 1, 2]}
here are the cliques in the graph:
[3, 0]
[3, 1]
[3, 2]

****************************************
[[0. 0. 0. 1.]
 [0. 0. 1. 0.]
 [0. 1. 0. 0.]
 [1. 0. 0. 0.]]
Graph:
```




```
<Figure size 400x300 with 0 Axes>
here are the connections for graph:
{0: [3], 1: [2], 2: [1], 3: [0]}
here are the cliques in the graph:
```



```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[9], line 9
      7 print('here are the cliques in the graph:')
      8 gg = bc.MaximalCliquesFinder(dd)
----> 9 gg.find_cliques()
     10 gg.print_cliques()
     12 print()

File ~/Desktop/honours_project/Research-project/function_bron_kerbosch_clique_finder.py:13, in MaximalCliquesFinder.find_cliques(self)
     11 def find_cliques(self):
     12     nodes = list(self.graph.keys())
---> 13     self._extend([], nodes, [])

File ~/Desktop/honours_project/Research-project/function_bron_kerbosch_clique_finder.py:36, in MaximalCliquesFinder._extend(self, compsub, candidates, not_set)
     33 new_not_set = [v for v in not_set if v in self.graph[candidate]]
     35 # Recursive call to extend compsub
---> 36 self._extend(new_compsub, new_candidates, new_not_set)
     38 # Move candidate to not_set
     39 candidates.remove(candidate)

File ~/Desktop/honours_project/Research-project/function_bron_kerbosch_clique_finder.py:23, in MaximalCliquesFinder._extend(self, compsub, candidates, not_set)
     19     return
     21 # Branch and bound: Choose a pivot
     22 #pivot = candidates[0] if candidates else not_set[0] #1st element in candidate chosen
---> 23 pivot = max(candidates, key=lambda node: len(self.graph[node])) #element with highest degree chosen.
     25 # Iterate through candidates not connected to the pivot
     26 for candidate in candidates[:]:

ValueError: max() arg is an empty sequence
```

```
<Figure size 640x480 with 0 Axes>
```

In [ ]: