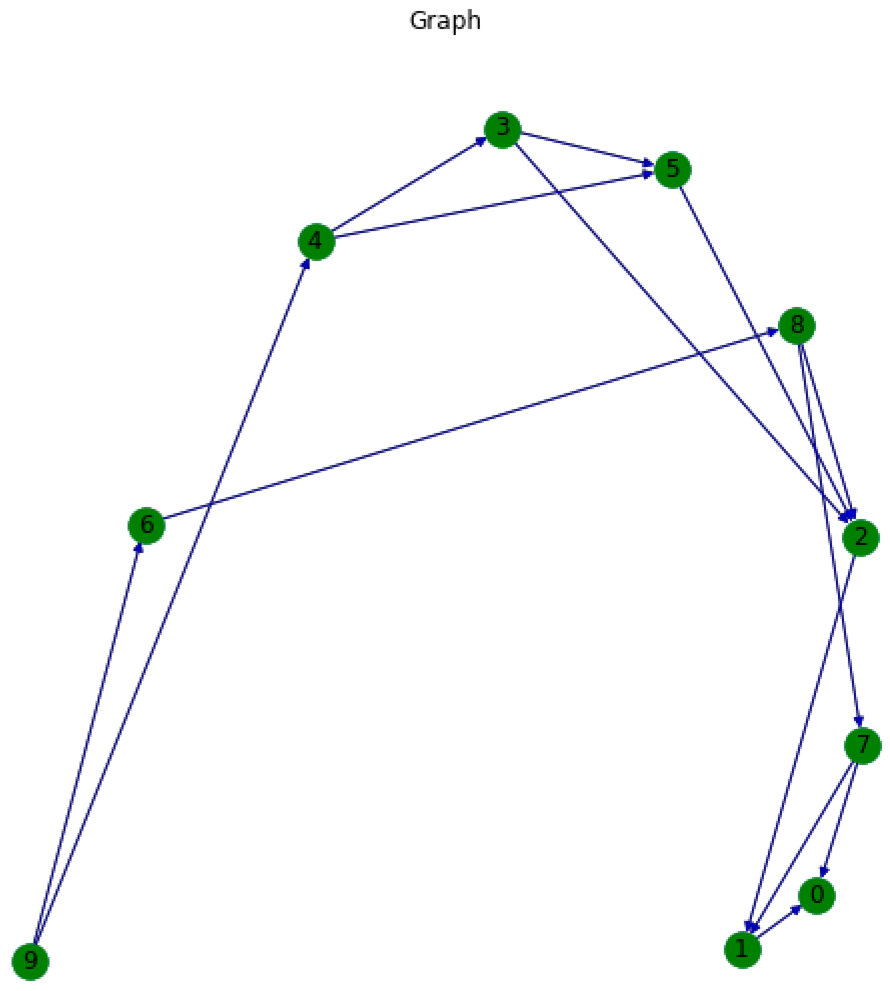
Programowanie sieciowe

In [37]:

```
import networkx as nx
import numpy as np
import matplotlib.pyplot as plt
import netgraph
```

In [86]:

```
weights = [
    (1, 0, 4),
    (7, 0, 8),
    (7, 1, 11),
    (2, 1, 8),
    (8, 2, 2),
    (5, 2, 4),
    (3, 2, 7),
    (4, 3, 9),
    (3, 5, 14),
    (4, 5, 10),
    (6, 8, 6),
    (8, 7, 7),
    (9, 6, 3),
    (9, 4, 2)
]

G = nx.MultiDiGraph()

G.add_weighted_edges_from(weights)

fig = plt.figure(figsize=(7, 7))

pos = nx.spiral_layout(G)

nx.draw(G, pos=pos, with_labels=True)
nx.draw_networkx_edges(G, pos, width=1,alpha=0.5,edge_color='b')
nx.draw_networkx_nodes(G, pos,node_size=300, node_color="g")
plt.title("Graph")
plt.show()
# print(nx.get_edge_attributes(G,'weight'))
```



9 - wierzcholek do ktorego nie wchodzi zaden inny

0 - wierzcholek z ktorego nie wychodzi zaden inny

In [181…

```python
M = nx.to_numpy_array(G, nodelist=range(10))

def changeIndexing(M):
    M[M > 0] = 1  # Tworze macierz binarna
    columns = [i for i in range(len(M))]
    path = []

    while len(M) > 1:
        n = len(M)

        for i in range(n):
            if False not in list(M[:,i] == 0):
                mask = [k for k in range(n) if k != i]
                M = M[np.ix_(mask, mask)]
                path.append(columns.pop(i))
                break

    return path
```

In [186…

```python
newIndexes = changeIndexing(M)
```

In [193…

```python
def findCritical(G, start):
    pass
```

In [197…

```python
start = newIndexes[0]
M = nx.to_numpy_array(G, nodelist=[i for i in range(10)])
```

```
findCritical(M, start)
print(M)
```

```
[[ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 4.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  8.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  7.  0.  0. 14.  0.  0.  0.  0.]
 [ 0.  0.  0.  9.  0. 10.  0.  0.  0.  0.]
 [ 0.  0.  4.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  6.  0.]
 [ 8. 11.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  2.  0.  0.  0.  0.  7.  0.  0.]
 [ 0.  0.  0.  0.  2.  0.  3.  0.  0.  0.]]
```