```
from queue import Queue
class Graph:
    def __init__(self, matrix: list) -> None:
        self.matrix = matrix
        self.dfs_list = []
        self.bfs_list = []

    def printGraph(self) -> None:
        for row in self.matrix:
            print(row)

    def dfs(self, source: int, visited: list) -> list:
        visited[source] = True
        self.dfs_list.append(source)
        for node in range(len(self.matrix[source])):
            if not visited[node] and self.matrix[source][node] != 0:
                self.dfs(node, visited)
        return self.dfs_list

    def bfs(self, source: int, visited: list) -> list:
        queue = Queue()
        queue.put(source)
        visited[source] = True
        while not queue.empty():
            curr = queue.get()
            self.bfs_list.append(curr)
            for node in range(len(self.matrix[curr])):
                if not visited[node] and self.matrix[curr][node] != 0:
                    queue.put(node)
                    visited[node] = True
        return self.bfs_list

if __name__ == "__main__":
    matrix = [
```

```
        [0, 1, 1, 0],
        [1, 0, 0, 1],
        [1, 0, 0, 1],
        [0, 1, 1, 0]
    ]

    obj = Graph(matrix)

    print("Adjacency Matrix:")
    obj.printGraph()

    print("\nDFS Traversal starting from node 0:")
    visited = [False] * len(matrix)
    print(obj.dfs(0, visited))

    print("\nBFS Traversal starting from node 0:")
    visited = [False] * len(matrix)
    print(obj.bfs(0, visited))
```

OUTPUT

```
python -u "c:\Users\Lenovo\Desktop\VSC1\ai-final\ass-1.py"
PS C:\Users\Lenovo\Desktop\VSC1> python -u "c:\Users\Lenovo\Desktop\VSC1\ai-final\ass-1.py"
Adjacency Matrix:
[0, 1, 1, 0]
[1, 0, 0, 1]
[1, 0, 0, 1]
[0, 1, 1, 0]

DFS Traversal starting from node 0:
[0, 1, 3, 2]

BFS Traversal starting from node 0:
[0, 1, 2, 3]
```