In [1]:

```
class Graph:
    def __init__(self):
        self.adjacency_list = {}

    def add_edge(self, u, v):
        if u not in self.adjacency_list:
            self.adjacency_list[u] = []
        self.adjacency_list[u].append(v)

    def dfs(self, start, visited=None):
        if visited is None:
            visited = set()
        visited.add(start)
        print(start, end=' ')
        for neighbor in self.adjacency_list.get(start, []):
            if neighbor not in visited:
                self.dfs(neighbor, visited)

    def bfs(self, start):
        visited = set()
        queue = [start]
        visited.add(start)
        while queue:
            node = queue.pop(0)
            print(node, end=' ')
            for neighbor in self.adjacency_list.get(node, []):
                if neighbor not in visited:
                    visited.add(neighbor)
                    queue.append(neighbor)

    def path_exists(self, start, end, method='dfs'):
        visited = set()
        if method == 'dfs':
            return self._dfs_path_exists(start, end, visited)
        elif method == 'bfs':
            return self._bfs_path_exists(start, end)

    def _dfs_path_exists(self, current, end, visited):
        if current == end:
            return True
        visited.add(current)
        for neighbor in self.adjacency_list.get(current, []):
            if neighbor not in visited:
                if self._dfs_path_exists(neighbor, end, visited):
                    return True
        return False

    def _bfs_path_exists(self, start, end):
        visited = set()
        queue = [start]
        visited.add(start)
        while queue:
            node = queue.pop(0)
            if node == end:
                return True
            for neighbor in self.adjacency_list.get(node, []):
                if neighbor not in visited:
                    visited.add(neighbor)
                    queue.append(neighbor)
        return False


graph = Graph()

graph.add_edge(1, 2)
graph.add_edge(1, 3)
graph.add_edge(2, 4)
graph.add_edge(3, 4)
graph.add_edge(4, 5)
graph.add_edge(5, 6)


print("DFS Traversal:")
graph.dfs(1)
print("\n")


print("BFS Traversal:")
graph.bfs(1)
print("\n")


start_node = 1
end_node = 5
print(f"Path exists from {start_node} to {end_node} (DFS):", graph.path_exists(start_nod
print(f"Path exists from {start_node} to {end_node} (BFS):", graph.path_exists(start_nod
```

```
DFS Traversal:
1 2 4 5 6 3

BFS Traversal:
1 2 3 4 5 6

Path exists from 1 to 5 (DFS): True
Path exists from 1 to 5 (BFS): True
```

In [ ]: