DFS

In [1]:
```
def dfs(graph, start):
    visited = set()
    stack = [start]
    while stack:
        vertex = stack.pop()
        if vertex not in visited:
            print(vertex, end=" ")
            visited.add(vertex)
            for neighbor in reversed(graph[vertex]):
                if neighbor not in visited:
                    stack.append(neighbor)

graph = {
    'A': ['B', 'C'],
    'B': ['D', 'E'],
    'C': ['F'],
    'D': [],
    'E': ['F'],
    'F': []
}

dfs(graph, 'A')
```

```
A B D E F C
```

BFS WITHOUT QUEUE

In [2]:
```
def bfs_without_queue(graph, start):
    visited = set()
    container = [start]
    while container:
        vertex = container[0]
        container = container[1:]
        if vertex not in visited:
            print(vertex, end=" ")
            visited.add(vertex)

            for neighbor in graph[vertex]:
                if neighbor not in visited:
                    container.append(neighbor)

graph = {
    'A': ['B', 'C'],
    'B': ['D', 'E'],
    'C': ['F'],
    'D': [],
    'E': ['F'],
    'F': []
}
```

```
bfs_without_queue(graph, 'A')
```

```
A B C D E F
```

BFS WITH QUEUE

In [3]:

```
def bfs(graph, start):
    visited = set()
    queue = [start]
    while queue:
        vertex = queue.pop(0)
        if vertex not in visited:
            print(vertex, end=" ")
            visited.add(vertex)

            for neighbor in graph[vertex]:
                if neighbor not in visited:
                    queue.append(neighbor)


graph = {
    'A': ['B', 'C'],
    'B': ['D', 'E'],
    'C': ['F'],
    'D': [],
    'E': ['F'],
    'F': []
}

bfs(graph, 'A')
```

```
A B C D E F
```