In [1]:
```python
import itertools
import math

def tsp_brute_force(points):
    min_path = None
    min_dist = float('inf')
    for path in itertools.permutations(points):
        distance = 0
        for i in range(len(path)-1):
            distance += math.dist(path[i], path[i+1])
        if distance < min_dist:
            min_dist = distance
            min_path = path
    return min_path, min_dist
```

In [2]:
```python
points = [(0,0), (1,5), (2,3), (5,1)]
min_path, min_dist = tsp_brute_force(points)

print("Shortest path:", min_path)
print("Shortest distance:", min_dist)
```

```
Shortest path: ((0, 0), (1, 5), (2, 3), (5, 1))
Shortest distance: 10.940638766556564
```

In [10]:
```python
from collections import deque
graph = {
    'A': ['B', 'C'],
    'B': ['D', 'E'],
    'C': ['F'],
    'D': [],
    'E': ['F'],
    'F': []
}
def bfs(start, graph):
    visited = {node: False for node in graph}
    queue = deque([start])
    visited[start] = True

    while queue:
        node = queue.popleft()
        print(node, end=" ")

        for neighbor in graph[node]:
            if not visited[neighbor]:
                visited[neighbor] = True
                queue.append(neighbor)

def dfs(start, graph, visited=None):
    if visited is None:
        visited = {node: False for node in graph}
    visited[start] = True
    print(start, end=" ")

    for neighbor in graph[start]:
        if not visited[neighbor]:
            dfs(neighbor, graph, visited)

print("BFS traversal: ", end="")
bfs('A', graph)
print("\nDFS traversal: ", end="")
dfs('A', graph)
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
BFS traversal: A B C D E F
DFS traversal: A B D E F C
```

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js