BFS & DFS Algorithms

October 24, 2024

[5]:
```python
#BFS
from collections import deque

def bfs_graph(graph, start):
    visited = set()
    queue = deque([start])

    while queue:
        node = queue.popleft()
        if node not in visited:
            visited.add(node)
            print(node, end=' ')

            # Enqueue all unvisited neighbors
            for neighbor in graph[node]:
                if neighbor not in visited:
                    queue.append(neighbor)


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


bfs_graph(graph, 'A')
```

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

[6]:
```python
#DFS
def dfs_graph(graph, start, visited=None):
    if visited is None:
        visited = set()
```

```
    visited.add(start)
    print(start, end=' ')


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


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


print("DFS Traversal (Graph):")
dfs_graph(graph, 'A')
```

```
DFS Traversal (Graph):
A B D E F C
```

[ ]: