Experiment 8

Kreena Shah
60004210243
C'32

Aim : To understand & implement Ford Fulkerson Method (Graph Based Algorithm)

Theory :
The Ford Fulkerson algorithm is a widely used algorithm to solve the maximum flow problem in a flow network.
The maximum flow problem involves determining the maximum amount of flow that can be sent from source to sink in a directed weighted graph, subject to capacity constraints on edges.

Algorithm

Start with initial flow as 0
While there exists an augmenting path from source to sink
- Find an augmenting path using any path finding algorithm
- Determine the amount of flow that can be sent along the augmenting path which is the minimum residual capacity along edges of path

~~Return~~
- Increase the flow along the augmenting path by the determined amount

Return maximum flow

Code :

```
from collections import defaultdict

class Graph:

    def __init__(self, graph):
        self.graph = graph
        self. ROW = len(graph)

    def BFS(self, s, t, parent):
        visited = [False]*(self.ROW)
        queue = []
        queue.append(s)
        visited[s] = True
        while queue:
```

```
            u = queue.pop(0)
            for ind, val in enumerate(self.graph[u]):
                if visited[ind] == False and val > 0:
                    queue.append(ind)
                    visited[ind] = True
                    parent[ind] = u
                    if ind == t:
                        return True
        return False

    def FordFulkerson(self, source, sink):
        parent = [-1]*(self.ROW)
        max_flow = 0

        while self.BFS(source, sink, parent) :
            path_flow = float("Inf")
            s = sink
            while(s != source):
                path_flow = min (path_flow, self.graph[parent[s]][s])
                s = parent[s]

            max_flow += path_flow

            v = sink
            while(v != source):
                u = parent[v]
                self.graph[u][v] -= path_flow
                self.graph[v][u] += path_flow
                v = parent[v]
        return max_flow

graph = [[0, 16, 13, 0, 0, 0],
        [0, 0, 10, 12, 0, 0],
        [0, 4, 0, 0, 14, 0],
        [0, 0, 9, 0, 0, 20],
        [0, 0, 0, 7, 0, 4],
        [0, 0, 0, 0, 0, 0]]

g = Graph(graph)
source = 0; sink = 5
print ("The maximum possible flow is %d " % g.FordFulkerson(source, sink))
```

Output :

```
PS C:\Users\Admin\OneDrive\Desktop\DJ\SEM6 Pracs\AA> py .\ford-fulkerson.py
The maximum possible flow is 23
```

Conclusion : During my experimentation, I encountered several challenges.

One significant challenge was dealing with graphs containing negative edge ~~waits~~ weights, which can cause the algorithm to enter an infinite loop.
To overcome this, I implemented a cycle detection mechanism & ensured that the algorithm terminates even in presence of negative cycles.
Another challenge was determining the optimal choice of augmenting paths to maximize flow efficiency.
To address this, I implemented different path selection strategies such as BFS or DFS.
Through iterative refinement & experimentation, I successfully addressed these challenges & obtained satisfactory results.