Name: Jigar Siddhpura **SAPID:** 60004200155

DIV: C/C2 **Branch:** Computer Engineering

AA - Experiment 6 - Ford Fulkerson Algorithm

60004210155
Jigar Siddhpura
C22

AA - Experiment 6 - Ford Fulkerson

Aim: To implement Ford Fulkerson algorithm.

Theory: It is a method used to find maximum flow in the network. Imagine a network as a system of pipes connecting a src, where water flows into a sink, where water flows out. Each pipe has a capacity, which represents how much water it can carry. The goal is to determine maximum amount of water that can flow from the src to the sink without surpassing the capacity of any pipe.

To achieve this, ford-fulkerson algo starts with an initial flow of zero & iteratively inc. it by finding augmented path. An augmented path is a route from src to sink where additional flow can be added without violating any pipe's capacity. This is like finding an alternate routes for water to flow, through the network, optimizing its path to maximize the flow.

The algo continues to find the augmenting paths & incrementally inc. the flow, until it can no longer find any path. At that point, the flow achieved is the maximum possible within the network constraints. It is a fundamental tool in network flow optimization, widely used in various applications such as transportation, communication & resource allocation.

Conclusion: Coding this algo presents challenges such as efficiently finding augmented paths in large network, handling cycles & multiple paths btw nodes & ensuring compatibility with diff. data structures & input formats. To overcome these hurdles while performing fold fulkerson experiment, we used techniques like DFS for path finding method of residual graph techniques was used to address issues related to cycles & termination condition.

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 not visited[ind] and val > 0:
                    queue.append(ind)
                    visited[ind] = True
                    parent[ind] = u
        return visited[t], parent

    def ford_fulkerson(self, source, sink):
        max_flow = 0
        parent = [-1] * self.ROW
        while True:
            found_path, parent = self.bfs(source, sink, parent)
            if not found_path:
                break
            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

            # Print the augmented path and its minimum value
            path = [sink]
            v = sink
            while v != source:
                u = parent[v]
                path.insert(0, u)
                v = u
            print("Augmented path: ", " -> ".join(str(x) for x in path), " Minimum flow: ", path_flow)

            v = sink
            while v != source:
                u = parent[v]
                self.graph[u][v] -= path_flow
                self.graph[v][u] += path_flow
```

```
            v = u

        return max_flow

graph = [ [0, 2, 3, 0, 0],
        [0, 0, 0, 0, 3],
        [0, 1, 0, 1, 0],
        [0,0,0,0,3],
        [0, 0, 0, 0, 0]]

g = Graph(graph)
source = 0
sink = 4
print("Max Flow: %d " % g.ford_fulkerson(source, sink))
```

OUTPUT :

```
PS D:\SEM-6\AA\EXPERIMENTS> python -u "d:\SEM-6\AA\EXPERIMENTS\fordfulkerson.py"
Augmented path:  0 -> 1 -> 4  Minimum flow:  2
Augmented path:  0 -> 2 -> 1 -> 4  Minimum flow:  1
Augmented path:  0 -> 2 -> 3 -> 4  Minimum flow:  1
Max Flow: 4
PS D:\SEM-6\AA\EXPERIMENTS>
```