Name: Prerna Sunil Jadhav
Sap ID: 60004220127
Batch: C22
Course: Advance Algorithm lab

EXP 6

AIM: Max Flow Network - Ford fulkerson

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 a source vertex to a sink vertex in a directed weighted graph, subject to capacity constraints on the edges.
The algorithm works by iteratively finding an augmenting path, which is a path from the source to the sink in the residual graph, i.e., the graph obtained by subtracting the current flow from the capacity of each edge.
The algorithm then increases the flow along this path by maximum possible amount, which is the minimum capacity of the edges along the path.

Algorithm:

1. Start with initial flow as 0.
2. While there exists an augmenting path from the source to sink:

- find an augmenting path using any path-finding algorithm, such as breadth-first search or depth-first search
- Determine the amount of flow that can be sent along the augmenting path, which is the minimum residual capacity along the edge of the path.
- Increase the flow along the augmenting path by the determined amount.

3. Return the maximum flow.

CONCLUSION: ~~The~~ we studied and implemented ~~an~~ ford-fulkerson algorithm for maximum flow problem.

Name:	Prerna Sunil Jadhav
Sap Id:	60004220127
Class:	T. Y. B. Tech (Computer Engineering)
Course:	Advance Algorithm Laboratory
Course Code:	DJ19CEL602
Experiment No.:	06

AIM: Implement Ford Fulkerson Method (Max Flow Network).

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 C:\Users\Jadhav\Documents\BTech\Docs\6th Sem\AA\Code> & C:/msys64/mingw64/bin/python.exe "c:/Users/Jadhav/Documents
/BTech/Docs/6th Sem/AA/Code/MaxFlowNetwork_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 C:\Users\Jadhav\Documents\BTech\Docs\6th Sem\AA\Code>
```