Advance Algorithm

Experiment 7

Ayush Jain 60004200132 B3

Aim : To implement Ford Fulkerson

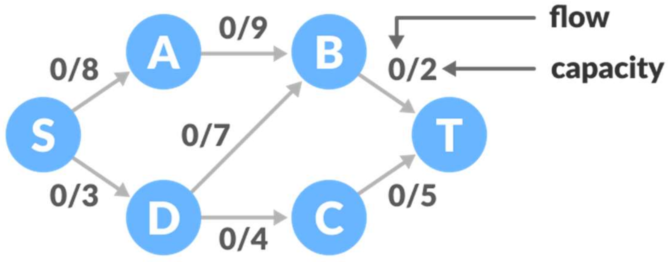
Theory:

Ford-Fulkerson Algorithm

Ford-Fulkerson algorithm is a greedy approach for calculating the maximum possible flow in a network or a graph.

A term, flow network, is used to describe a network of vertices and edges with a source (S) and a sink (T). Each vertex, except S and T, can receive and send an equal amount of stuff through it. S can only send and T can only receive stuff.

We can visualize the understanding of the algorithm using a flow of liquid inside a network of pipes of different capacities. Each pipe has a certain capacity of liquid it can transfer at an instance. For this algorithm, we are going to find how much liquid can be flowed from the source to the sink at an instance using the network.



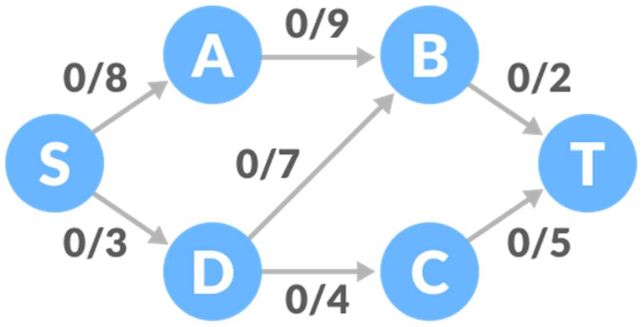
Flow network graph

How Ford-Fulkerson Algorithm works?
The algorithm follows:

1. Initialize the flow in all the edges to 0.
2. While there is an augmenting path between the source and the sink, add this path to the flow.
3. Update the residual graph.
4. We can also consider reverse-path if required because if we do not consider them, we may never find a maximum flow.

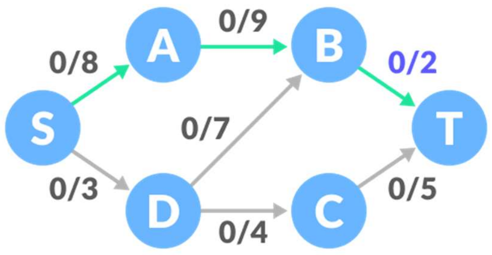
The above concepts can be understood with the example below.

Ford-Fulkerson Example
The flow of all the edges is 0 at the beginning.



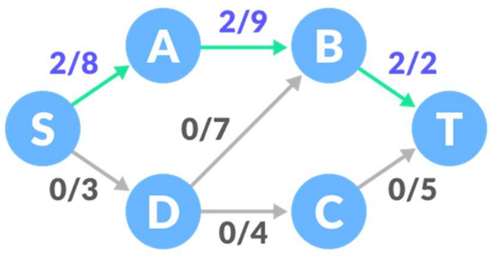
Flow network graph example

1. Select any arbitrary path from S to T. In this step, we have selected path S-A-B-T.

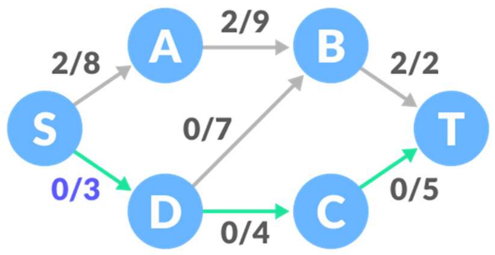


Find a path

The minimum capacity among the three edges is 2 (B-T). Based on this, update the flow/capacity for each path.

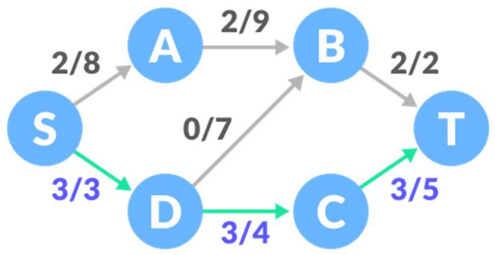


Update the capacities
2. Select another path S-D-C-T. The minimum capacity among these edges is 3 (S-D).



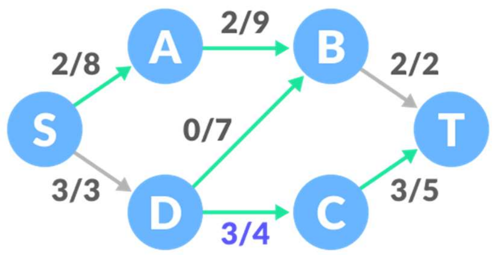
Find next path

Update the capacities according to this



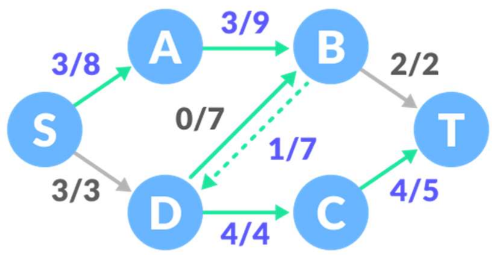
Update the capacities

3. Now, let us consider the reverse-path B-D as well. Selecting path S-A-B-D-C-T. The minimum residual capacity among the edges is 1 (D-C).



Find next path

Updating the capacities.



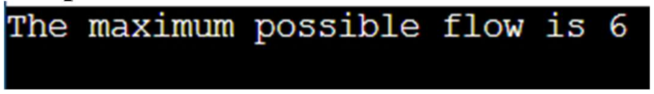
Update the capacities

The capacity for forward and reverse paths are considered separately.
4. Adding all the flows = 2 + 3 + 1 = 6, which is the maximum possible flow on the flow network.

Code:

```python
from collections import defaultdict

class Graph:
    def __init__(self, graph):
        self.graph = graph
        self.num_vertices = len(graph)

    def bfs(self, source, sink, parent):
        visited = [False] * self.num_vertices
        queue = [source]
        visited[source] = True

        while queue:
            u = queue.pop(0)
            for v, capacity in enumerate(self.graph[u]):
                if not visited[v] and capacity > 0:
                    queue.append(v)
                    visited[v] = True
                    parent[v] = u
                    if v == sink:
                        return True

        return False

    def find_max_flow(self, source, sink):
        parent = [-1] * self.num_vertices
        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, 8, 0, 0, 3, 0],
         [0, 0, 9, 0, 0, 0],
         [0, 0, 0, 0, 7, 2],
         [0, 0, 0, 0, 0, 5],
         [0, 0, 7, 4, 0, 0],
         [0, 0, 0, 0, 0, 0]]

g = Graph(graph)

source = 0
sink = 5

print("The maximum possible flow is %d " % g.find_max_flow(source, sink))
```

Output:



```
The maximum possible flow is 6
```

Conclusion: We successfully implemented Ford Fulkerson.