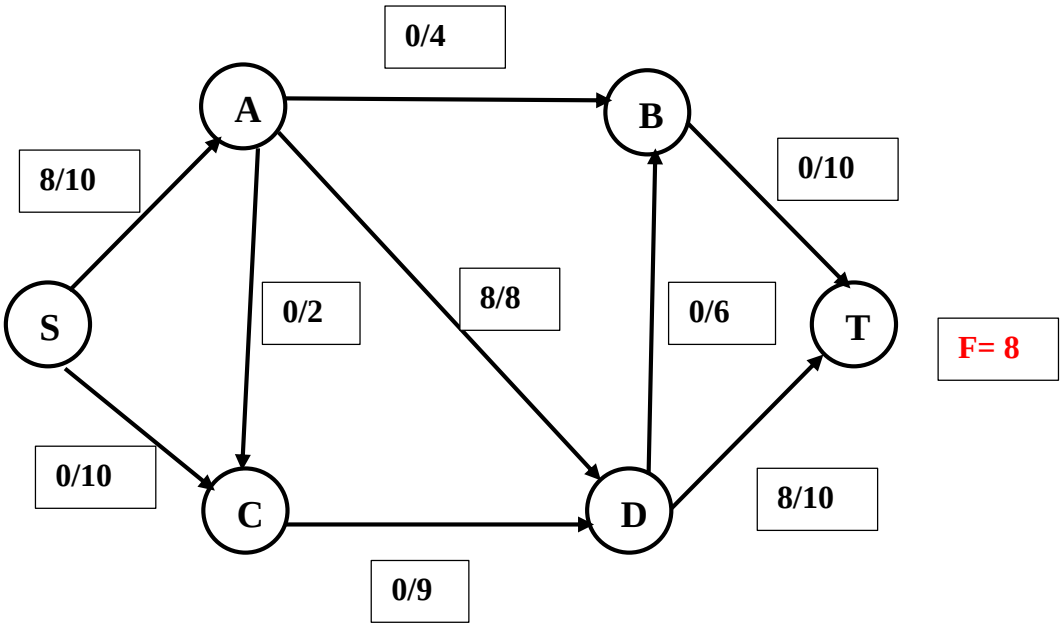
# MAX FLOW MIN CUT PROBLEM

Swarup Kr Ghosh

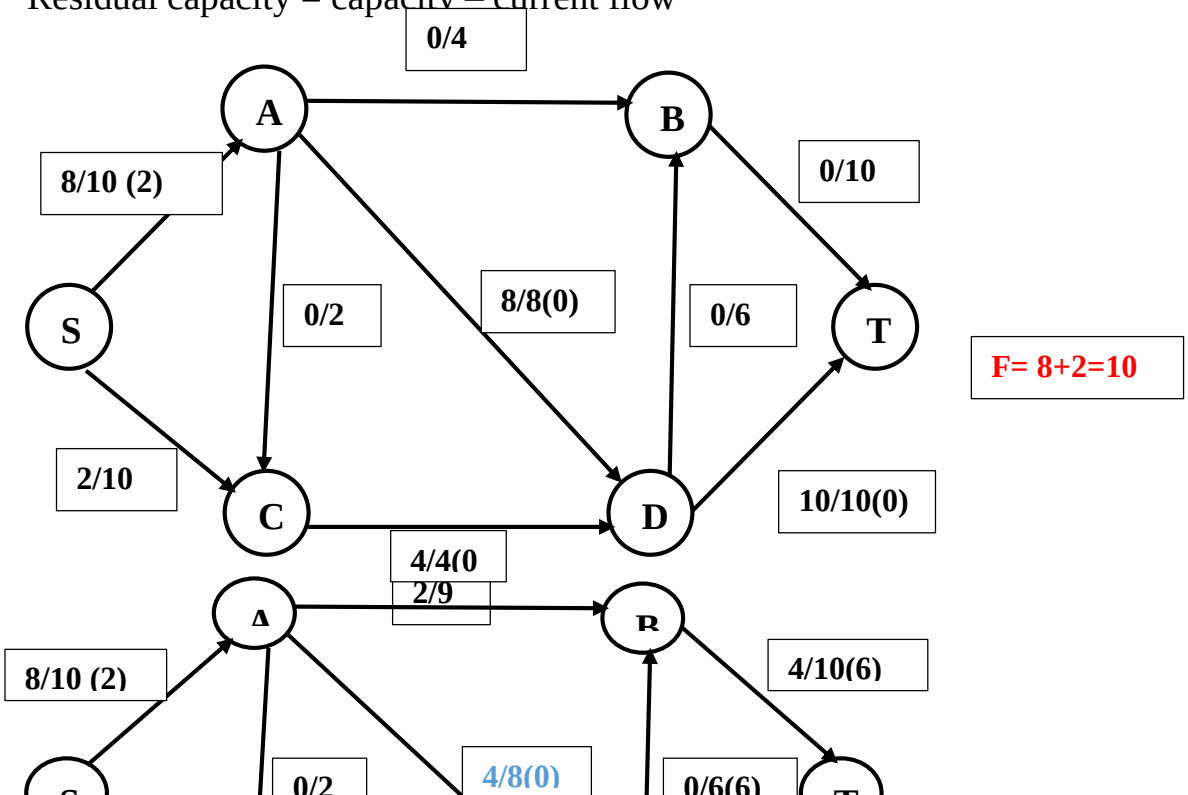
## Ford-Fulkerson Algorithm:

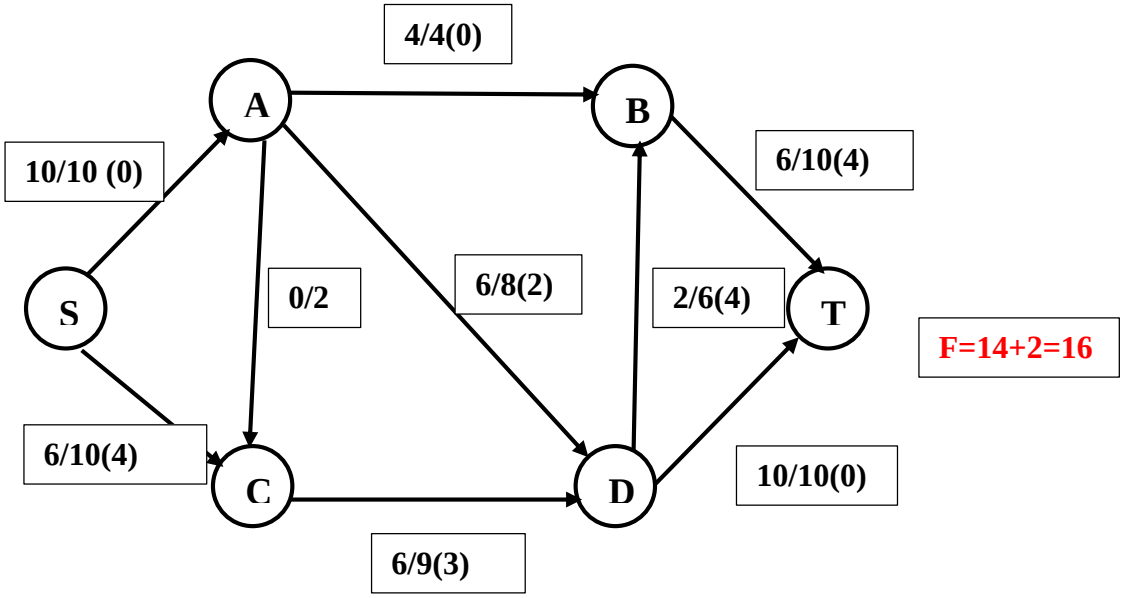
Flow/Capacity---- Source (S) and Sink (T) Initially F (flow) =0

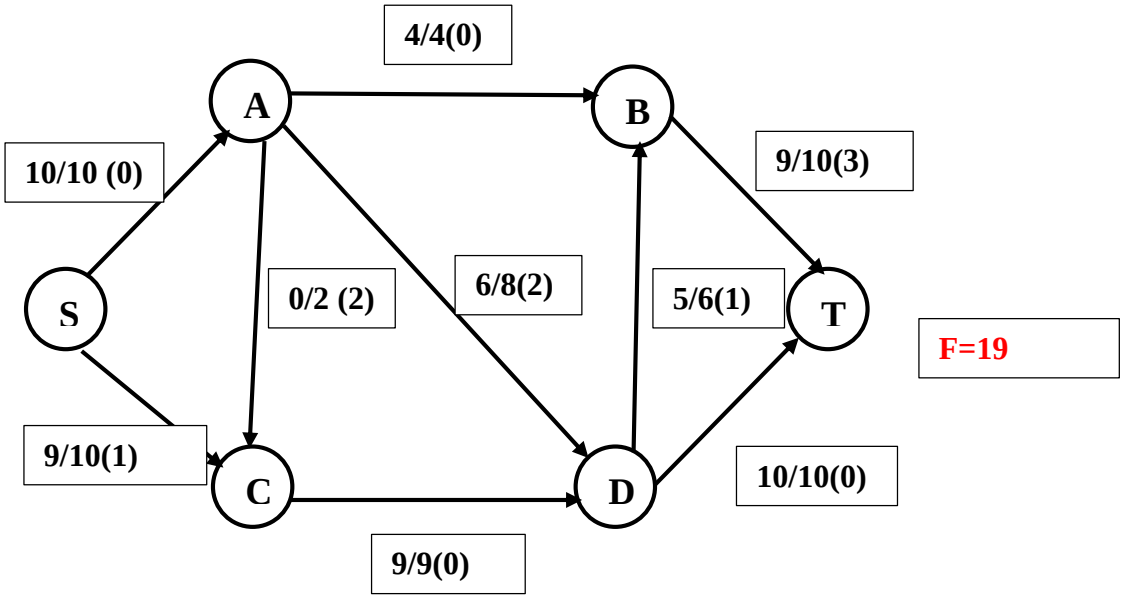



| Augmenting path | Bottle neck capacity |
|---|---|
| 1. S→A→D→T | MIN(10,8,10)= 8 |
| 2. S→ C→D→T | MIN (10, 9, 2) = 2 |
| 3. S→C→D→A→B→T | MIN(8, 7, 8, 4,10)= 4 |
| 4. S→A→D→B→T | MIN(2, 4, 6, 6) = 2 |
| 5. S→C→D→B→T | MIN (4,3,4,4)=3 |

Residual capacity = capacity – current flow




4/4(0)
A
B
10/10 (0)
6/10(4)
S
0/2
6/8(2)
2/6(4)
T
F=14+2=16
6/10(4)
C
D
10/10(0)
6/9(3)


4/4(0)
A
B
10/10 (0)
9/10(3)
S
0/2 (2)
6/8(2)
5/6(1)
T
F=19
9/10(1)
C
D
10/10(0)
9/9(0)

**Problem:**

For a given graph which represents a flow network at which every edge has a capacity. Also given two vertices source (S) and sink (T) in the graph. Find the maximum possible flow from S to T subject to the following constraints:

a. Flow on an edge does not exceed the given capacity of the edge.
b. In-flow is equal to Out-flow for every vertex except S and T.

Terminology:

1. Residual graph: It is a graph which indicates additional possible flow. If there is such path from source to sink then there is a possibility to add flow.
2. Residual capacity: It is the original capacity of the edge minus flow.
3. Minimal cut: It is also known as bottle neck capacity which decide maximum possible flow from source to sink through the augmenting path.
4. Augmenting path: It is defined in two ways:
    a. Non-full forward edges (Residual capacity >0)
    b. Non-empty backward edges (Residual capacity =0)

**Algorithm: Ford-Fulkerson for Max-Flow Min-Cut**

**Procedure: FordFulkerson()**

*Step 1:* Start with an initial flow as 0 (FLOW← 0 )
*Step 2:* While there is an augmenting path from S to T, add two path FLOW to FLOW
*Step 3:* return (FLOW)
End procedure

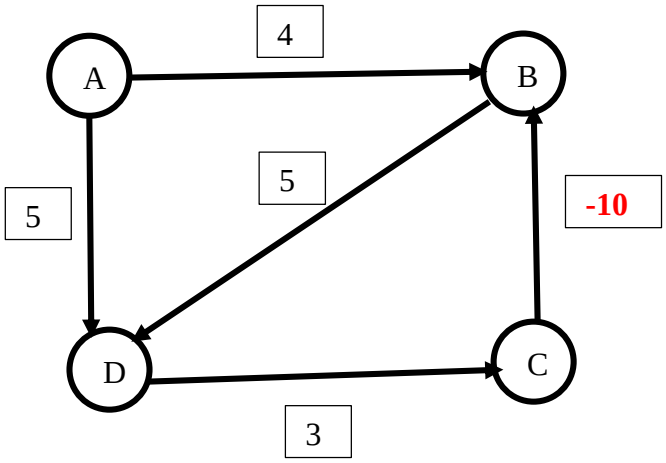
Pseudocode: FordFulkerson

```
Set FLOW = 0
Repeat until there is no path from S to T:
        Run Depth First Search (DFS) from source vertex S to find a flow path
        to end vertex T
        Let f be the minimum capacity value on the path
        Add f to FLOW
        For each edge u → v on the path:
                Decrease capacity of the edge c(u → v) by f
```

Increase capacity of the edge c(v → u) by f

Time Complexity: O (Max_Flow*E)

**Drawback:**




| Iteration | A | B | C | D |
|---|---|---|---|---|
| 0 | 0 | INF | INF | INF |
| 1 | 0 | 4 | INF | 5 |

| | | | | |
|---|---|---|---|---|
| 2 | 0 | -2 | 8 | 5 |
| 3 | 0 | -2 | 8 | 3 |
| | 0 | -4 | 6 | 3 |
| | 0 | | | |

**B-D-C: total weigh= -2**