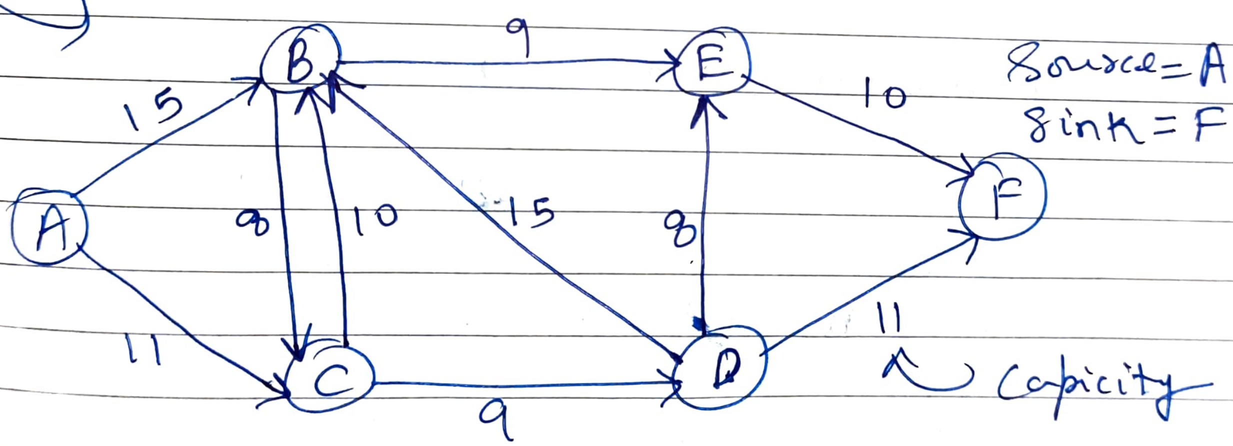
Ford-Fulkerson Algo. for Maximum Flow Problem

- Given a graph which represents a flow network where every edge has a capicity. Also given two vertices source S and sink t in the graph (Network Graph). Flow will be originated from s and will be terminated a sink t.
- Findout the maximum possible flow from s to t with following constraints.

  a) Flow on an edge does not exceed the given capicity of the edge.

  b) In-flow equal to out-flow for every vertex except s & t.

Network



- From A only the flow will come out.
- And to the sink (F), only the flow will go in.

indegree(S) = 0

outdegree(t) = 0

** Suppose edges are pipes with water, where flow represents the volume of water allowed to flow through the pipes.

Terminologies:

* Residual Graph: It's a graph which indicates additional possible flow, If there is such path from source to sink then there is the possibility to add flow.

* Residual capacity => It's the original capacity of the edge minus flow.

* Minimal cut: Also known as bottleneck capacity, which decides max. possible flow from source to sink through an augmented path.

* Augmenting path: Augmenting path can be done in two ways.

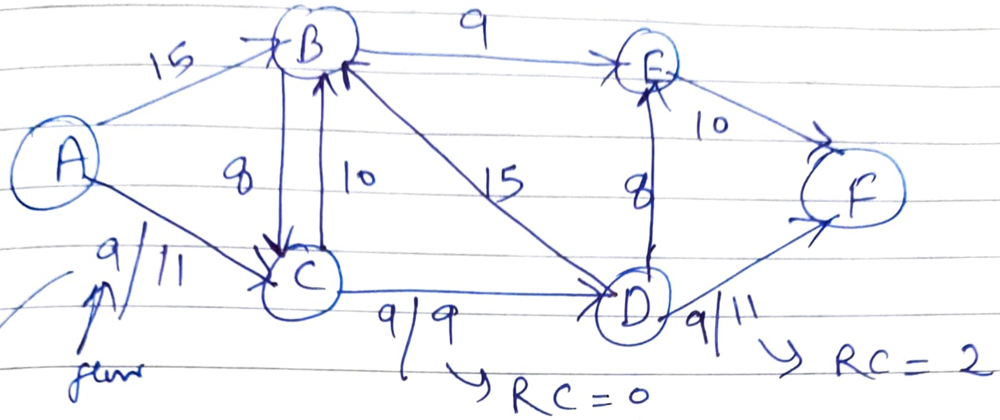
  1) Non-full forward edges

  2) Non-empty backword edges.

* An augmenting path is a path of edges in the residual graph with unused capacity greater than zero from s to t.

Let'us conside a path

- A → C → D → F



- Though above path max flow will be 9 (i.e. minimal cut)

Residual capacity (RC) = 2

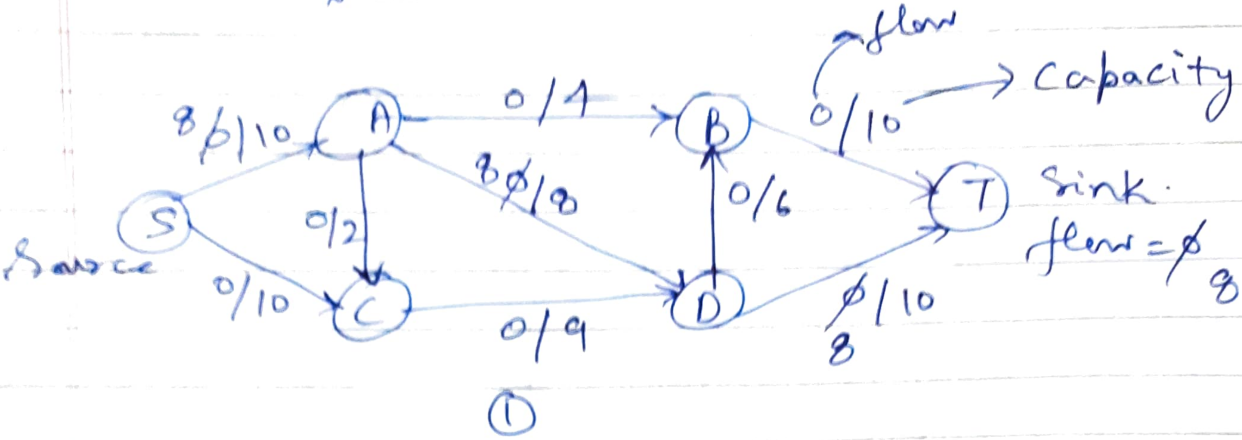
Algo:

Ford-Fulkerson Algorithm:

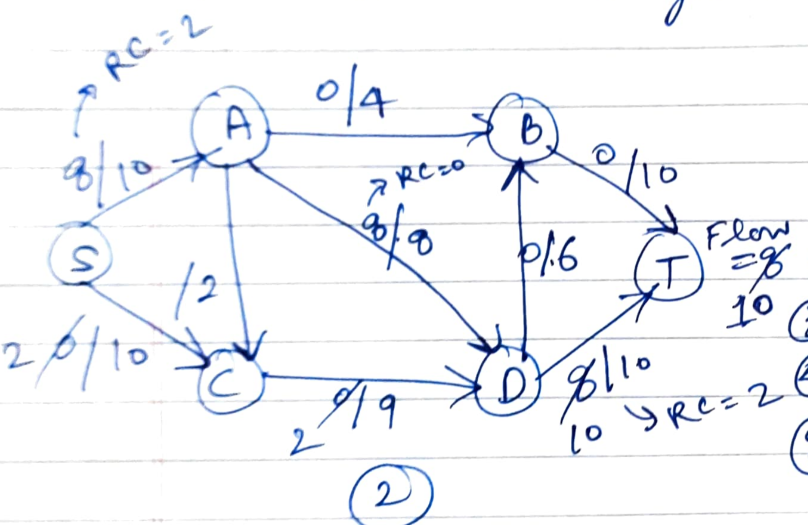
The following is a simple idea of the Algorithm:

1. Start with initial flow as 0
2. While there is an augmenting path from source to sink Add this path flow to flow.
3. Return flow.

Example:

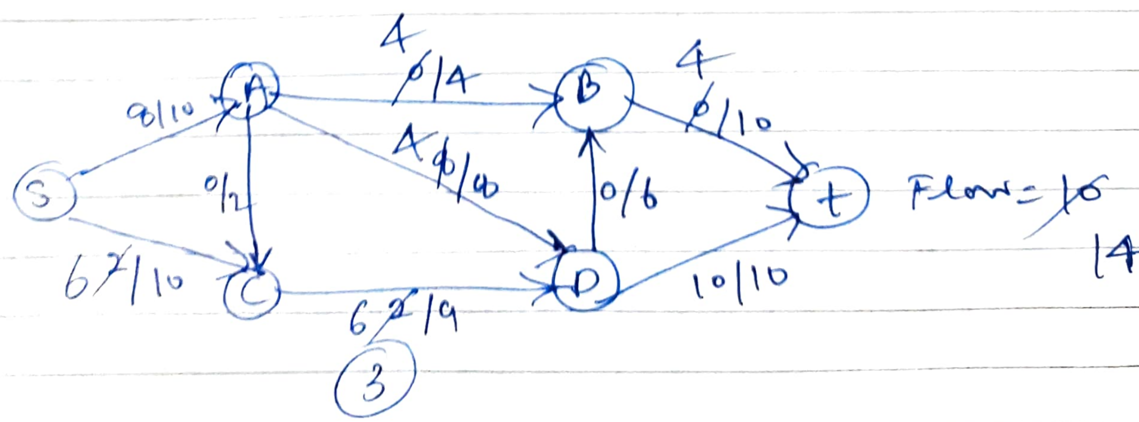


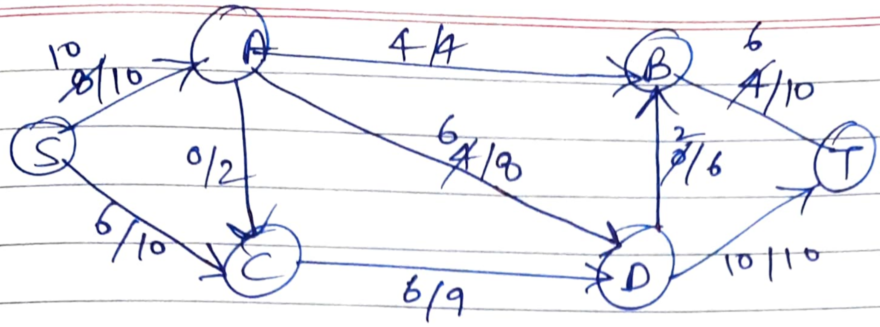
Now select an augmenting path. S→A→D→T



Augmenting path	Bottle Neck capacity
① S→A→D→T	8
② S→C→D→T	2
③ S→C→D→A→B→T	4
④ S→A→D→B→T	2
⑤ S→C→D→B→T	3

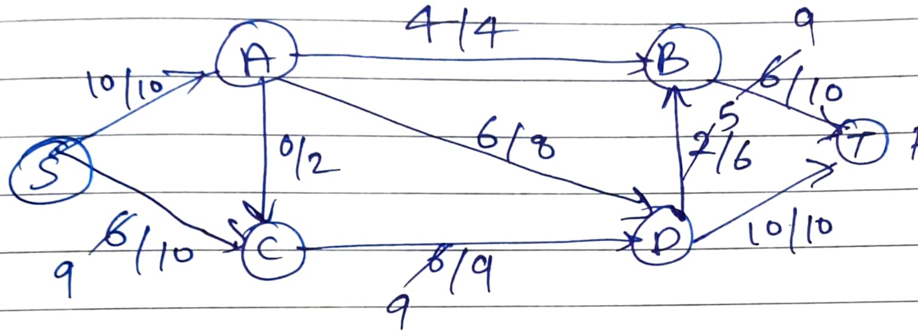
S→A→D→T path is capable to carry another flow value of 2.





Flow = 14 + 2 = 16

(4)



Flow = 19

(5)

is there any other Augmenting path?

Time Complexity =)

We have three steps in Algo.

(i) Find an augmenting path

(ii) Compute the bottleneck Capacity

(iii) Augment each edge and total flow

→ O(E)

E = No. of edges.

→ to calculate all these → O(F)

F = maximum Flow

So Total Complexity of Algo = O(E × F)