

Push Relabel Algorithm

Maximum flow in a network graph

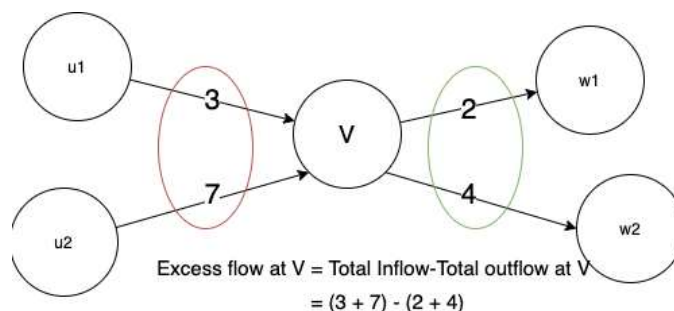
- In a network graph where every edge has a given capacity, maximum flow is defined as the maximum amount of flow that can move from source to sink.
- Maximum flow is calculated keeping in mind two constraints,
 1. For every vertex (except source and sink), incoming flow is equal to outgoing flow.
 2. Flow of an edge shouldn't exceed its capacity.

Conceptual framework

- The idea behind the algorithm is that the edges act as pipes and vertices act as joints and the water flows from a high height to a low height.
- Every node is assigned a **height** variable.
- The source is considered to be at the highest height $|V|$ and sink is at the lowest height 0.

Push operation

- Every vertex is also assigned another variable **excess flow**.
- When a vertex has an excess flow, it **pushes** it to a lower height vertex.
- The amount of flow pushed is equal to the minimum of excess flow of vertex and capacity of connecting edge.
- Excess flow at a vertex V is defined by: Total flow received by V - Total flow going out of V .



Relabel operation

- If the height of a vertex is low and the water is overflowing in that vertex, i.e., the water gets locally trapped, then it's **relabelled**.
- In a relabel operation, the height of the vertex is increased.

Pseudocode

- We initialise the graph with height of source vertex as V .
- The height and excess flow of all the other vertices as 0.
- We keep on performing push and relabel till the excess flow at all the vertices (except source and sink) is zero.

Initialise height of all the vertices as 0

Initialise Height of source = V

Initialise excess flow of all the vertices as 0

For all neighbouring vertices of source node:

Flow and excess flow = capacity of the connecting source-neighbour edge

While there is a vertex with excess flow:

Perform push or relabel

At the end all the vertices should have 0 excess flow

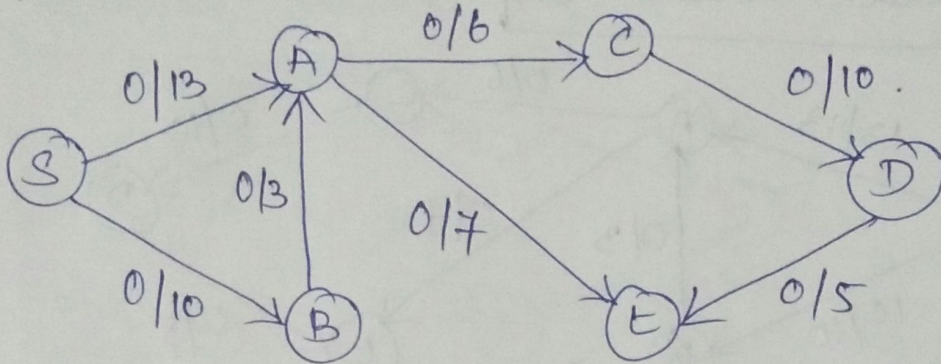
Return maximum flow

Time complexity

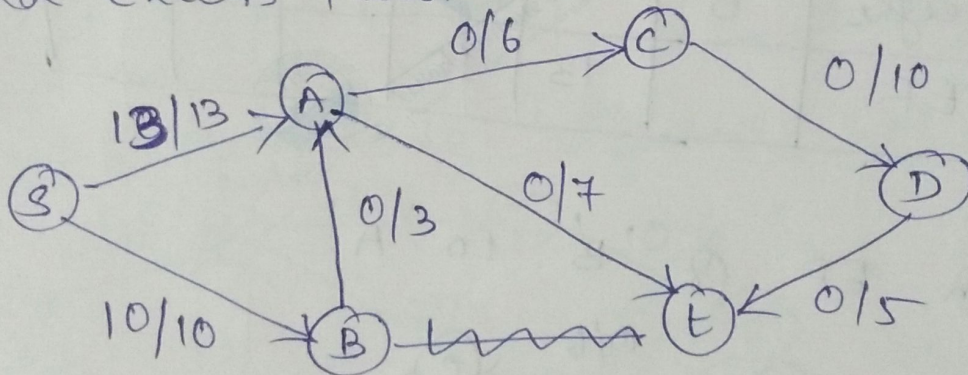
- Push relabel algorithm calculates maximum flow in $O(V^2E)$ time where V is the number of vertices and E is the number of edges in the network.



① Consider the following flow n/w.

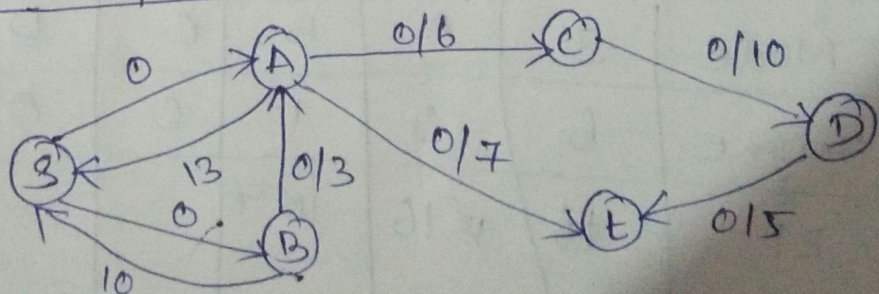


② Initialize the graph by setting heights and excess flow.



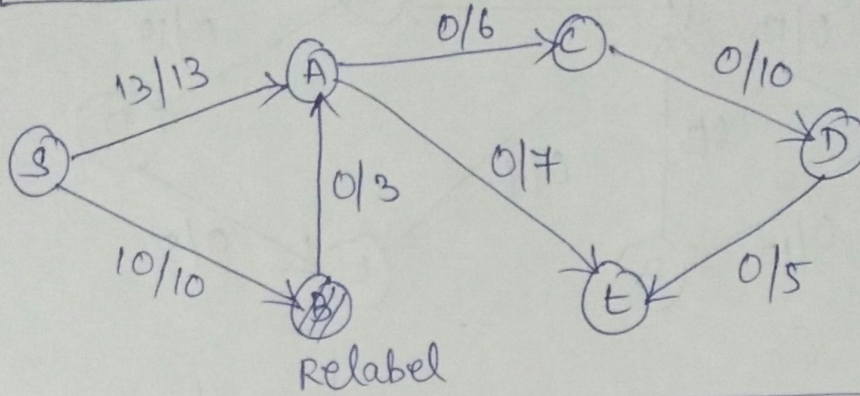
Node	S	A	B	C	D	t
Height	6	0	0	0	0	0
EF		13	10			

Residual graph.



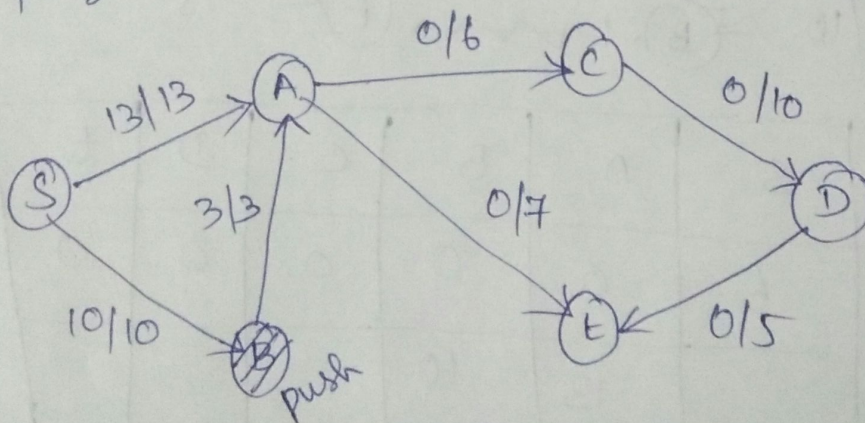
- ③ Consider vertex 'B'. It cannot transfer its excess flow as its neighbour vertex 'A' has same height.

Relabel vertex B



Node	S	A	B	C	D	t
Height	6	0	1	0	0	0
EF		13	10			

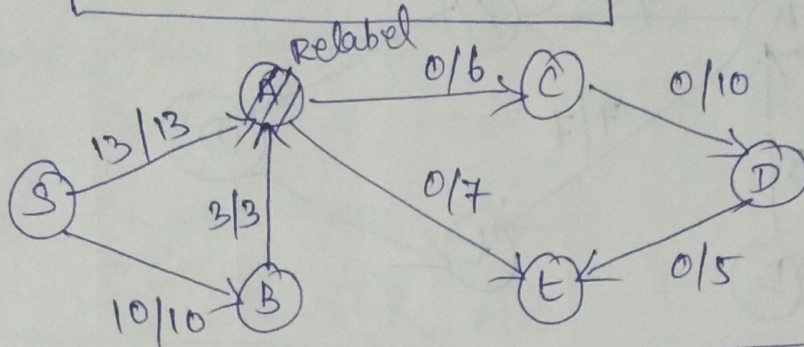
- ④ Push EF of 'B' to 'A'.



Node	S	A	B	C	D	t
Height	6	0	1	0	0	0
EF	13	13 16	10 7			

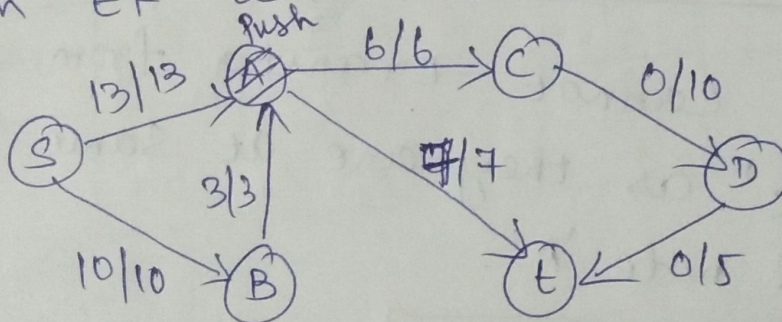
- ⑤ Similarly, consider node 'A'. Relabel its height to 1. so that flow can be transferred to node 'c' and sink 't'.

Relabel vertex A



node	S	A	B	C	D	t
Height	6	1	1	0	0	0
EF		16	7			

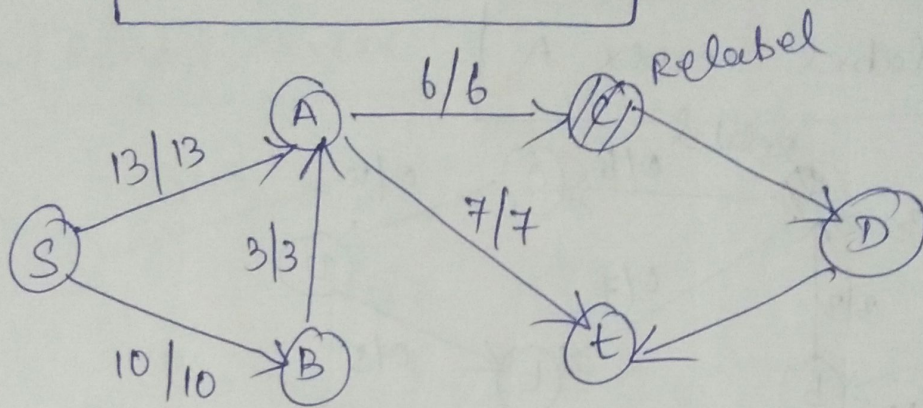
- ⑥ push EF to node 'c' and sink 't'.



Node	S	A	B	C	D	t
Height	6	1	1	0	0	0
EF		16 6	7	6		

- ⑦ Now consider Node 'c'. Relabel it. So that it can allow EF to node 'D'

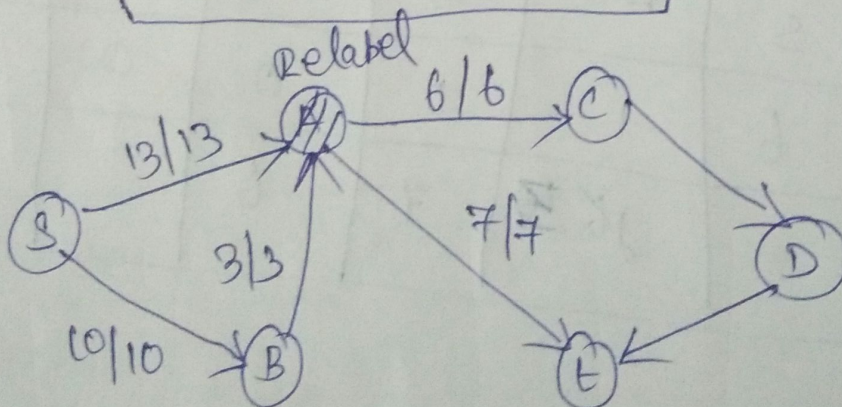
Relabel Node c



Node	s	A	B	c	D	t
Height	6	1	1	0	0	0
EF		3	7	6		

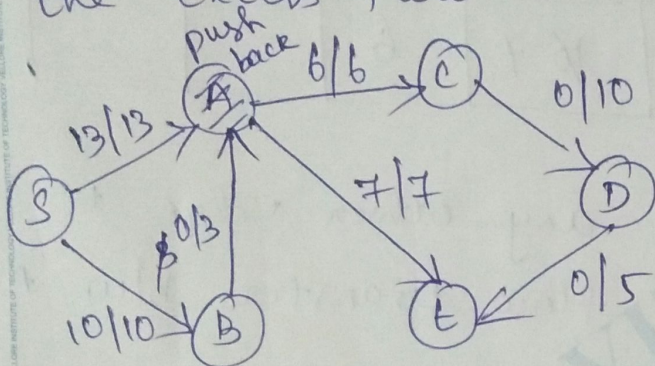
- ⑧ The Flow cannot transfer from node 'A' to node 'c' as they are at same height. So, relabel node 'A'.

Relabel Node 'A'



Node	s	A	B	C	D	t
Height	6	2	1	1	0	0
EF		3	7	6		

- ⑨ Since the height of node 'A' is ~~high~~ higher than height of node 'B', it can push back the excess flow to 'B'.



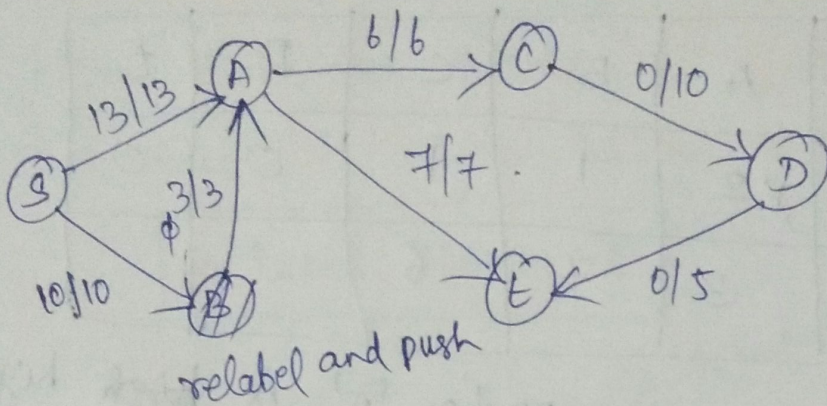
Node	s	A	B	C	D	t
Height	6	2	1	1	0	0
EF		3	10	6		

- ⑩ Since 'B' has no other edge to transfer the flow, relabel it again and transfer the extra flow back to 'A'

Relabel 'B'

and

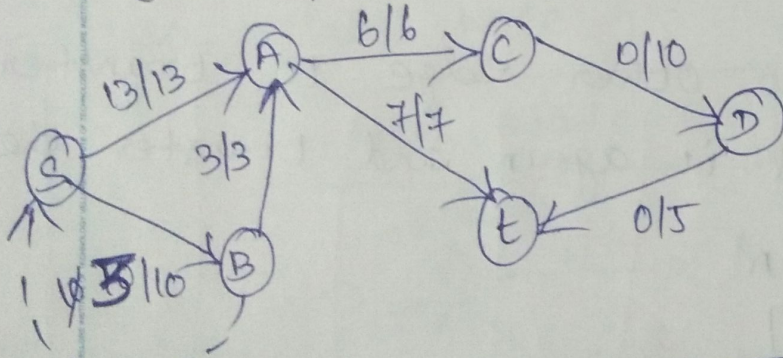
Note: As 'B' received back flow from 'A' again transfer back to 'A', height will be $ht(A) + 1$



Node	S	A	B	C	D	t
Height	6	2	3	1	0	0
EF		3	7	6		

(ii) 'A' also don't have any other edge to transfer flow. so, flow transfers b/w 'A' and 'B' in to and fro manner.

This to and fro operation b/w 'A' and 'B' continues till height of either 'A' or 'B' is greater than source. Then, we push back the extra flow to source node.



push back.

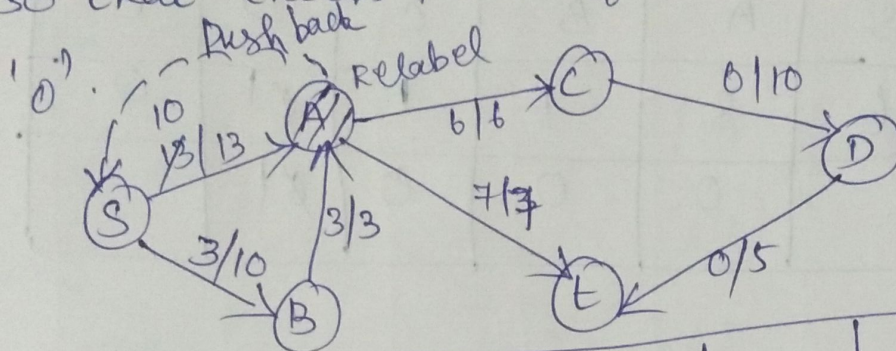
Node	S	A	B	C	D	t
Height	6	2	3	1	0	0
EF		3	0	6		

To & Fro operation

Node	A	B
Ht	4	3
EF	0	10
Ht	4	5
EF	3	7
Ht	6	5
EF	0	10
Ht	6	7
EF	3	7

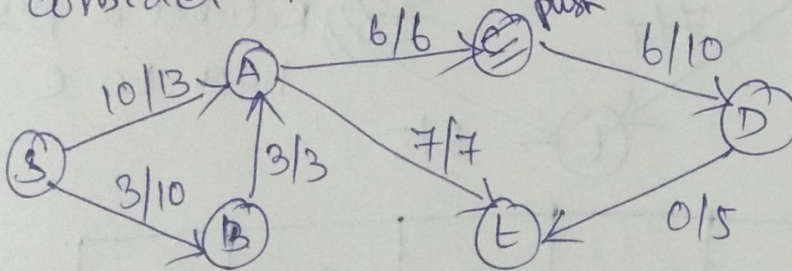


- ⑫ Similarly, node 'A' is relabelled and excess flow is pushed to source node. so that excess flow of node 'A' also becomes '0'.



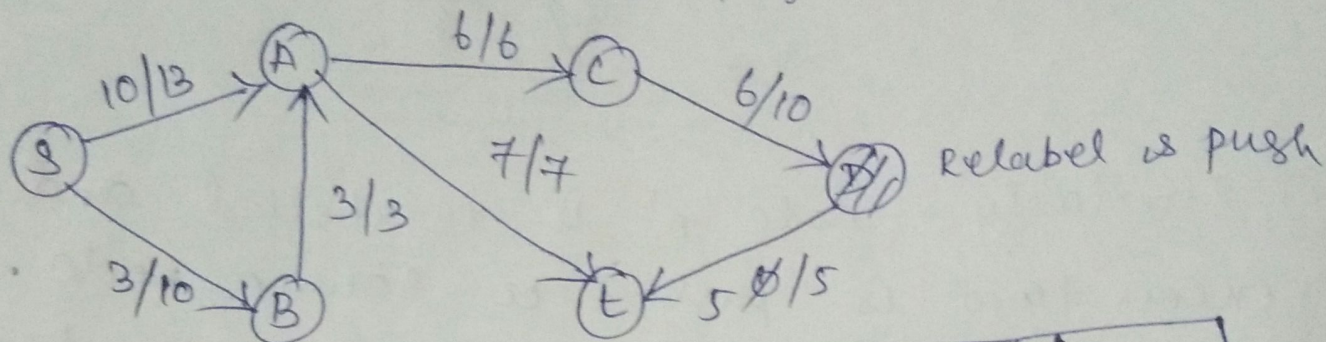
Node	S	A	B	C	D	t
Height	6	8	7	1	0	0
EF		0	0	6		

- ⑬ Consider node 'C' push its EF to node 'D'.



Node	S	A	B	C	D	t
Height	6	8	7	1	0	0
EF		0	0	0	6	

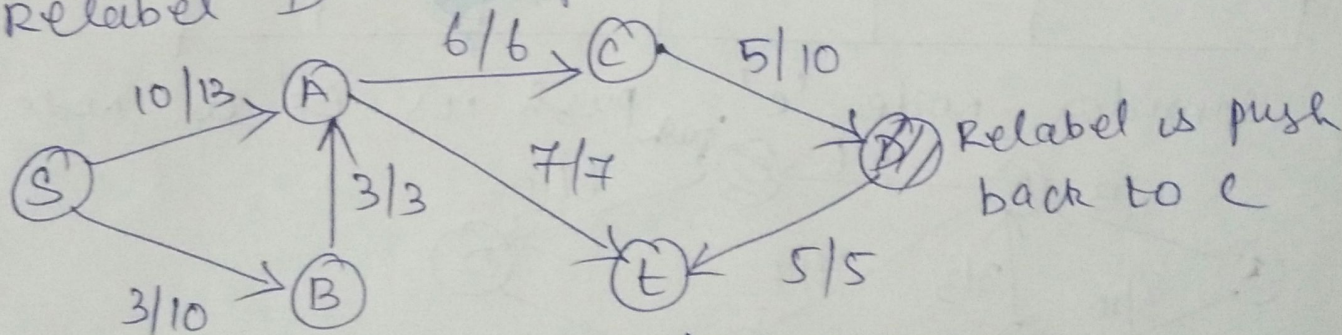
14) Relabel node 'D' and push EF to node 't'



Node	S	A	B	C	D	t
Height	6	8	7	1	0	0
EF		0	0	0	1	

15) 'D' don't have any other edge to push EF. So it pushes back EF to 'C'.

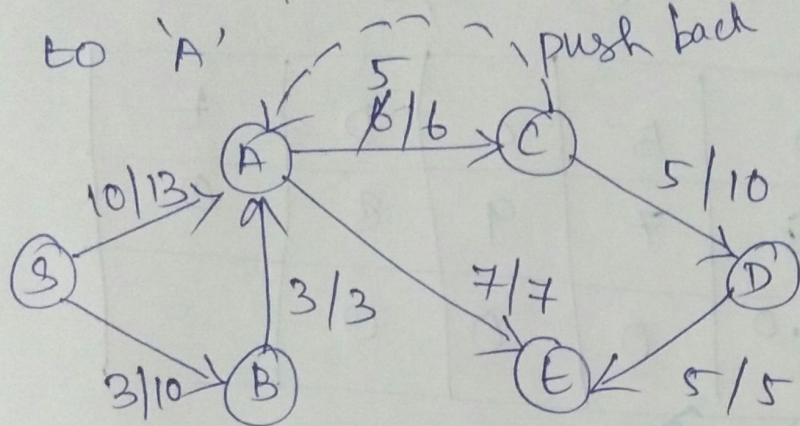
Relabel D and push EF to C.



Node	S	A	B	C	D	t
Height	6	8	7	1	0 2	0
EF		0	0	0 1	0	

16) 'c' again gets relabelled and flow pushed back to 'D'. This to a fro operation will continue until height of 'c' is $>$ than height of 'A'.

Then 'c' pushes back EF to 'A'



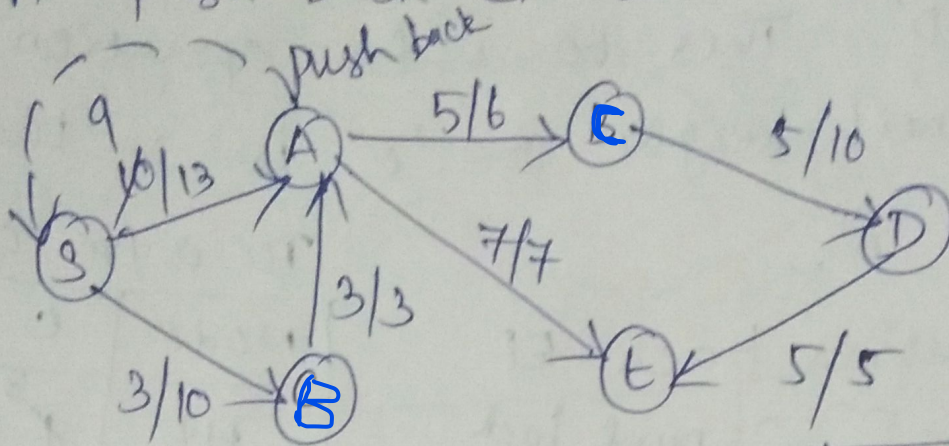
node	S	A	B	C	D	t
Height	6	8	7	9	8	0
EF		$\emptyset$	0	10	0	

To a fro operation

Node	C	D
Ht	3	2
EF	10	$\emptyset$
Ht	3	4
EF	$\emptyset$	10
Ht	5	4
EF	10	$\emptyset$
Ht	5	6
EF	$\emptyset$	10
Ht	7	6
EF	10	$\emptyset$
Ht	7	8
EF	$\emptyset$	10
Ht	9	8
EF	1	0

17

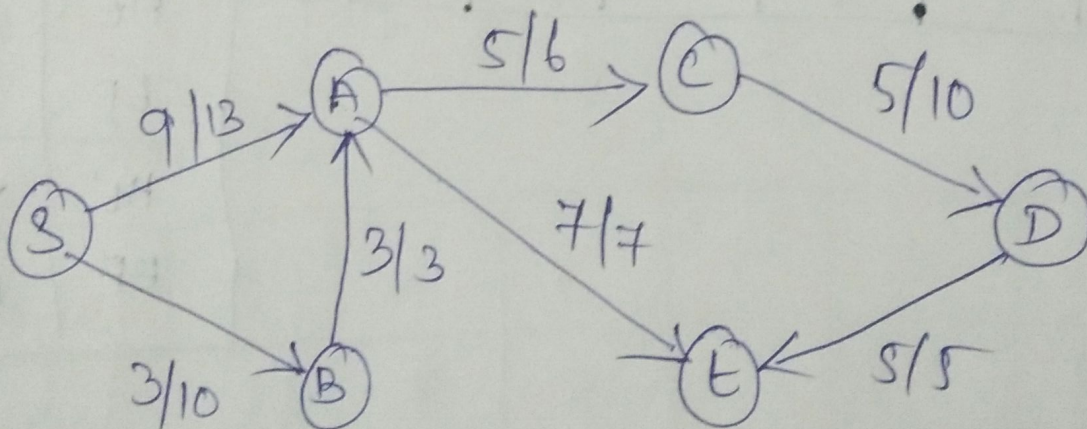
'A' push back EF to '3'.



Node	S	A	B	C	D	t
Height	6	8	7	9	8	0
EF		0	0	0	0	

Now, the excess flow at all nodes become zero.

Hence, algorithm terminates.



Computing Maximum Flow:

Maximum flow can be calculated as
the sum of outgoing flow at source
(or) sum of incoming flow at sink.

$$\begin{aligned}\therefore \text{Maximum Flow} &= 9+3 \text{ (outgoing flow at source)} \\ &\text{(or)} \\ &= 7+5 \text{ (Incoming flow at sink)}\end{aligned}$$

$$\begin{aligned}\text{outgoing flow at source} &= \text{Incoming flow at sink.} \\ 9+3 &= 7+5 = 12. &= \text{maximum flow.}\end{aligned}$$

$$\text{Maximum Flow} = 12.$$