Ford–Fulkerson Algorithm for Maximum Flow Problem

Maximum Flow Problem

- Given a graph which represents a flow network where 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 with following constraints:
- a) Flow on an edge doesn’t exceed the given capacity of the edge.
- b) Incoming flow is equal to outgoing flow for every vertex except s and t.

Ford Fulkerson Algorithm

1) Start with initial flow as 0.

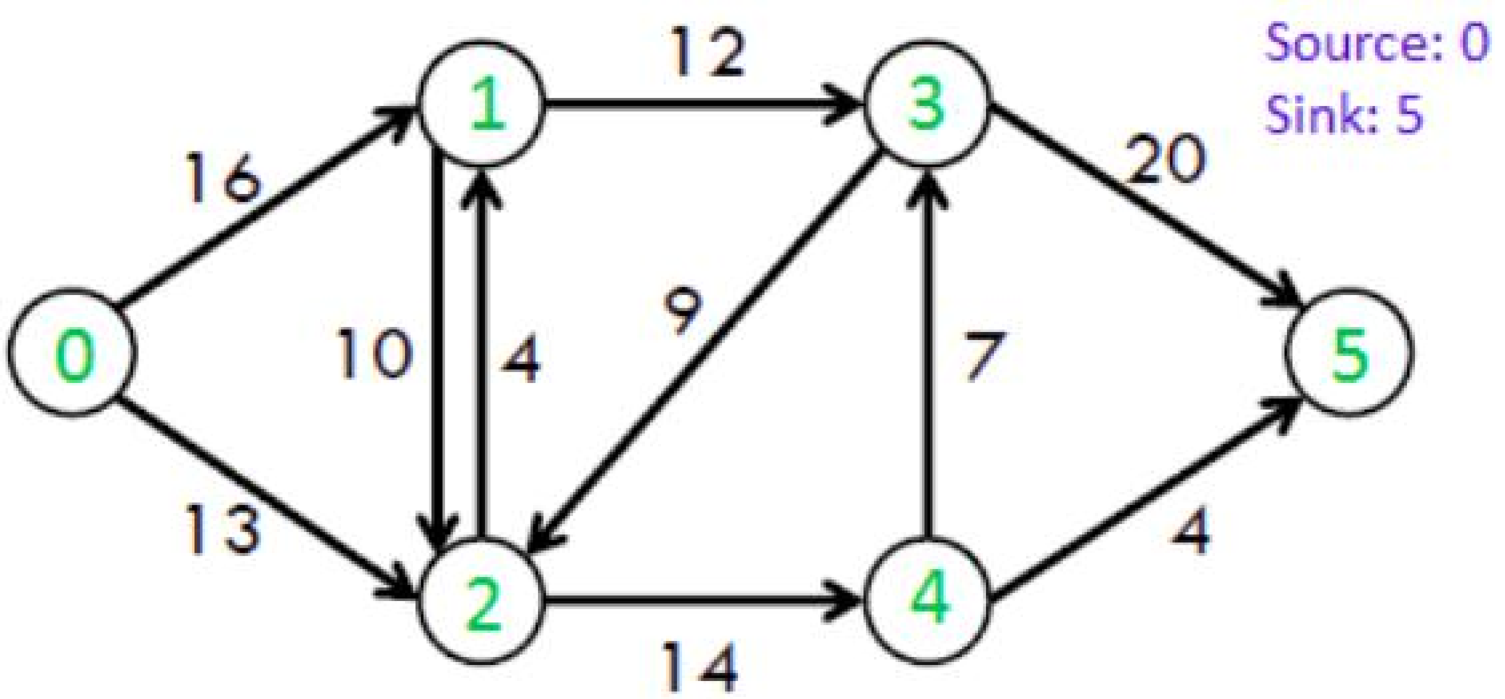
2) While there is a augmenting path from source to sink.

    Add this path–flow to flow.

3) Return flow.

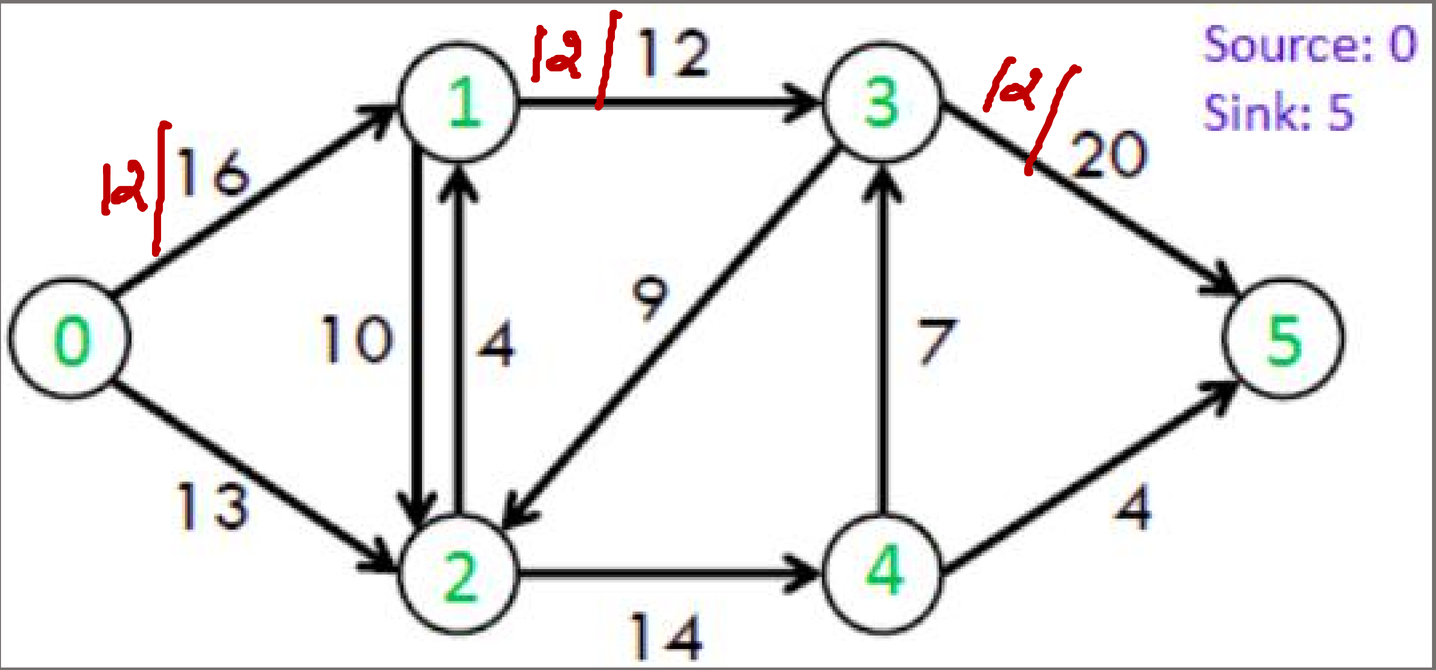
Implementation of Ford Fulkerson Algorithm

- Edmonds–Karp Algorithm
- idea of Edmonds–Karp is to use BFS in Ford Fulkerson implementation as BFS always picks a path with minimum number of edges
- uses adjacency matrix representation though where BFS takes $O(V^2)$ time
- time complexity of the implementation will be $O(EV^3)$



Original Matrix

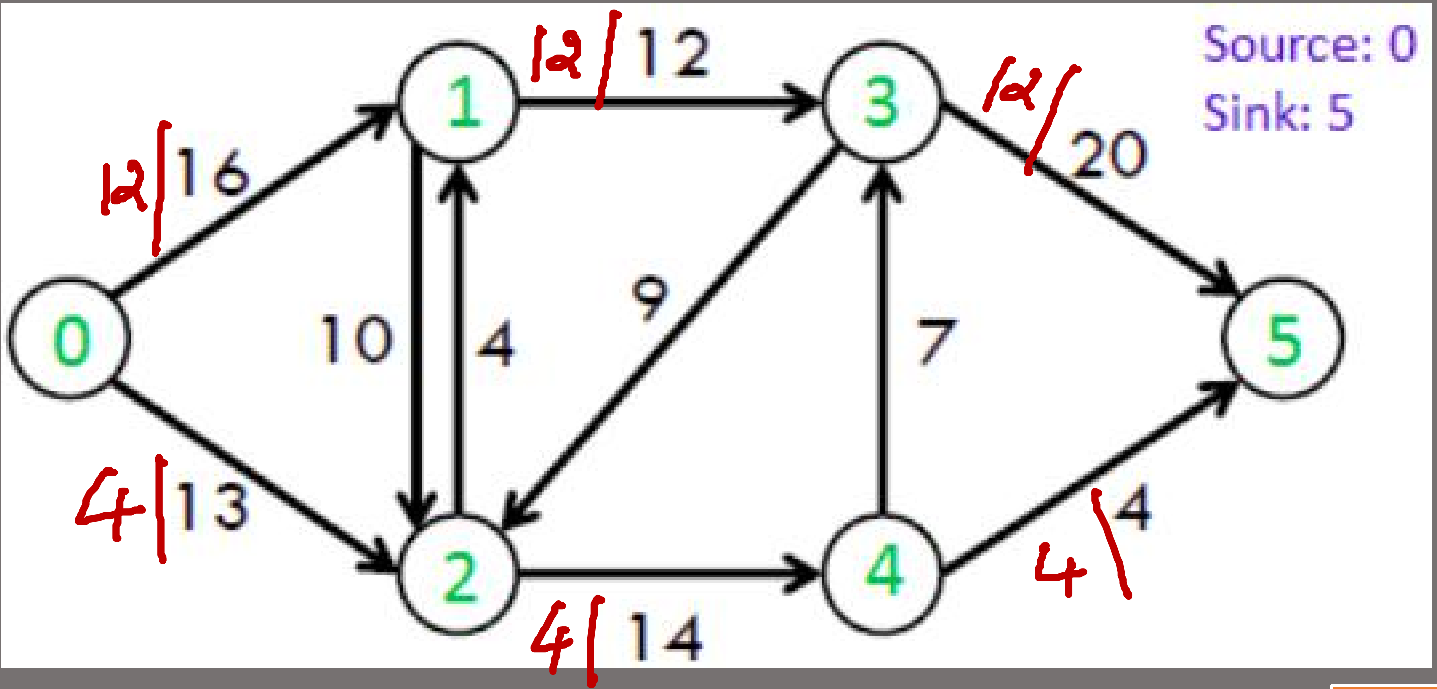
	0	1	2	3	4	5
0	0	16	13	0	0	0
1	0	0	10	12	0	0
2	0	4	0	0	14	0
3	0	0	9	0	0	20
4	0	0	0	7	0	4
5	0	0	0	0	0	0



Residual Matrix

Path	Bottleneck
0 – 1 – 3 – 5	12

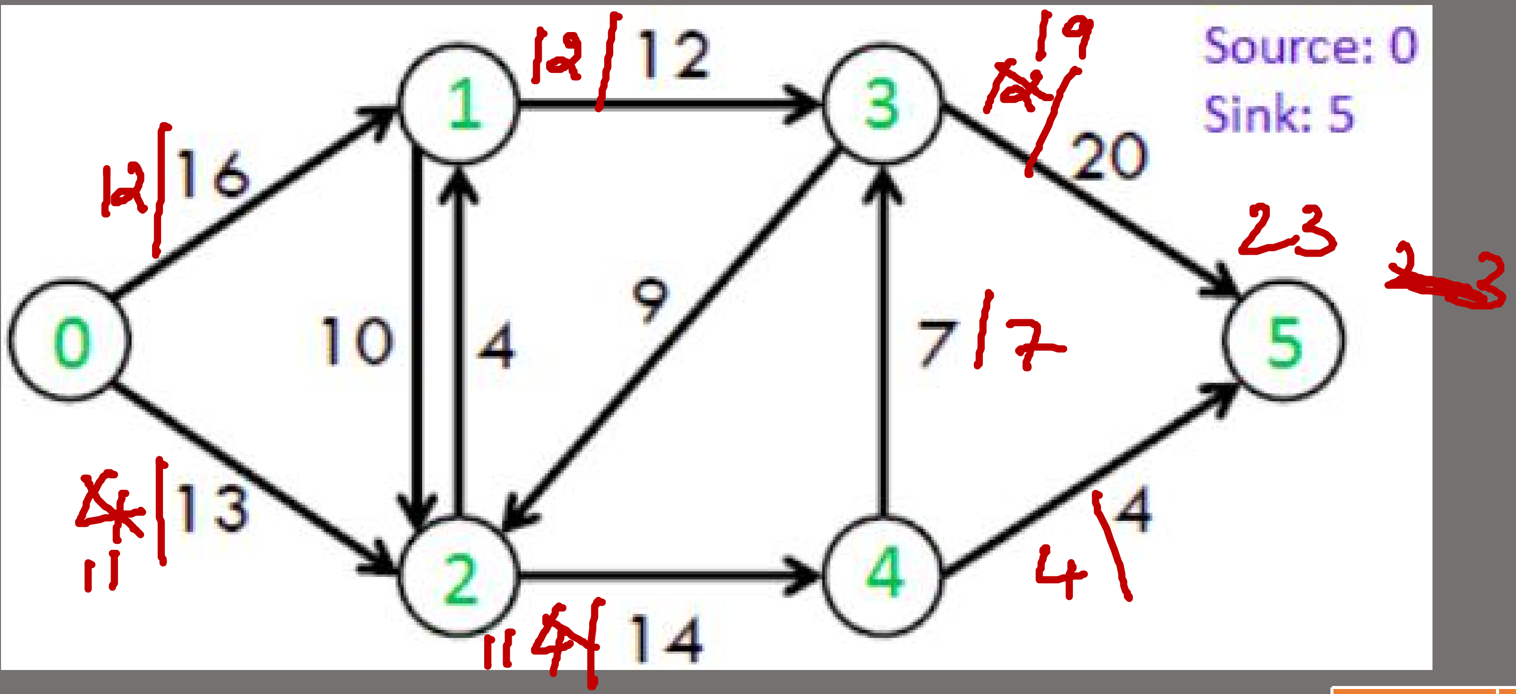
	0	1	2	3	4	5
0	0	4	13	0	0	0
1	12	0	10	0	0	0
2	0	4	0	0	14	0
3	0	12	9	0	0	8
4	0	0	0	7	0	4
5	0	0	0	12	0	0



Residual Matrix

Path	Bottleneck
0 – 1 – 3 – 5	12
0 – 2 – 4 – 5	4

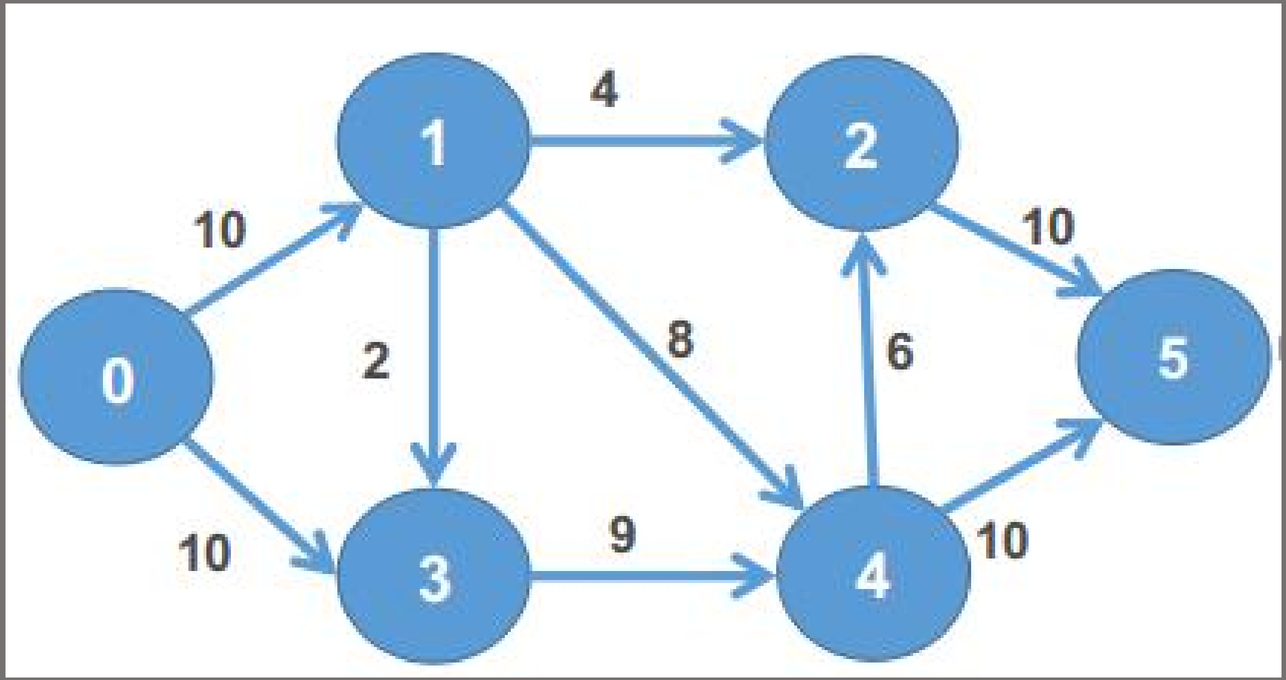
	0	1	2	3	4	5
0	0	4	9	0	0	0
1	12	0	10	0	0	0
2	4	4	0	0	10	0
3	0	12	9	0	0	8
4	0	0	4	7	0	0
5	0	0	0	12	4	0



Residual Matrix

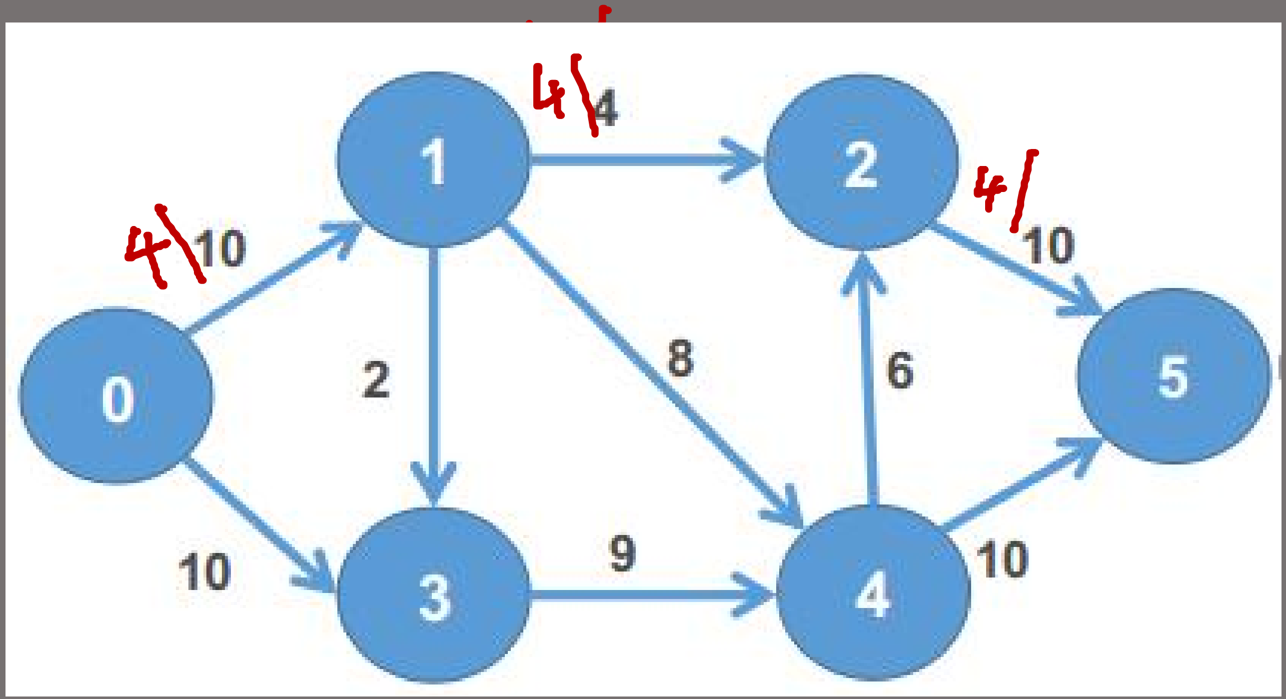
Path	Bottleneck
0 – 1 – 3 – 5	12
0 – 2 – 4 – 5	4
0 – 2 – 4 – 3 – 5	7

	0	1	2	3	4	5
0	0	4	2	0	0	0
1	12	0	10	0	0	0
2	11	4	0	0	3	0
3	0	12	9	0	7	1
4	0	0	11	0	0	0
5	0	0	0	19	4	0



Original Matrix

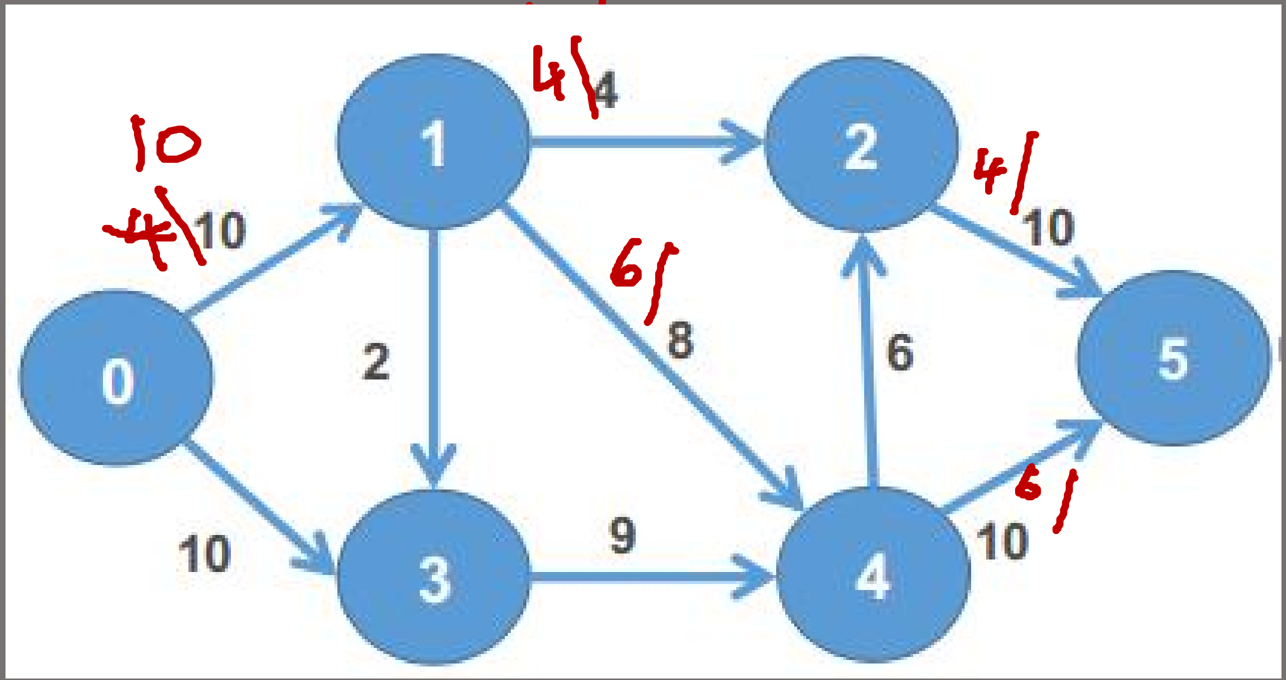
	0	1	2	3	4	5
0	0	10	0	10	0	0
1	0	0	4	2	8	0
2	0	0	0	0	0	10
3	0	0	0	0	9	0
4	0	0	6	0	0	10
5	0	0	0	0	0	0



Residual Matrix

Path	Bottleneck
0 – 1 – 2 – 5	4

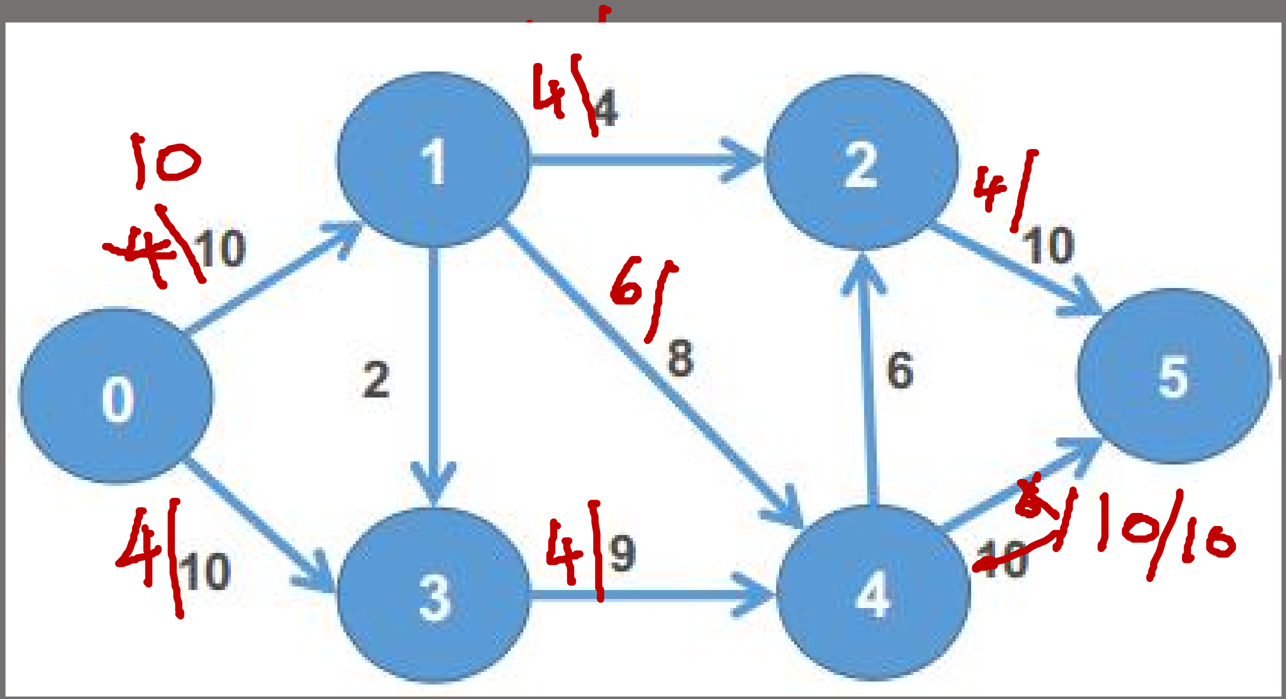
	0	1	2	3	4	5
0	0	6	0	10	0	0
1	4	0	0	2	8	0
2	0	4	0	0	0	6
3	0	0	0	0	9	0
4	0	0	6	0	0	10
5	0	0	4	0	0	0



Residual Matrix

Path	Bottleneck
0 – 1 – 2 – 5	4
0 – 1 – 4 – 5	6

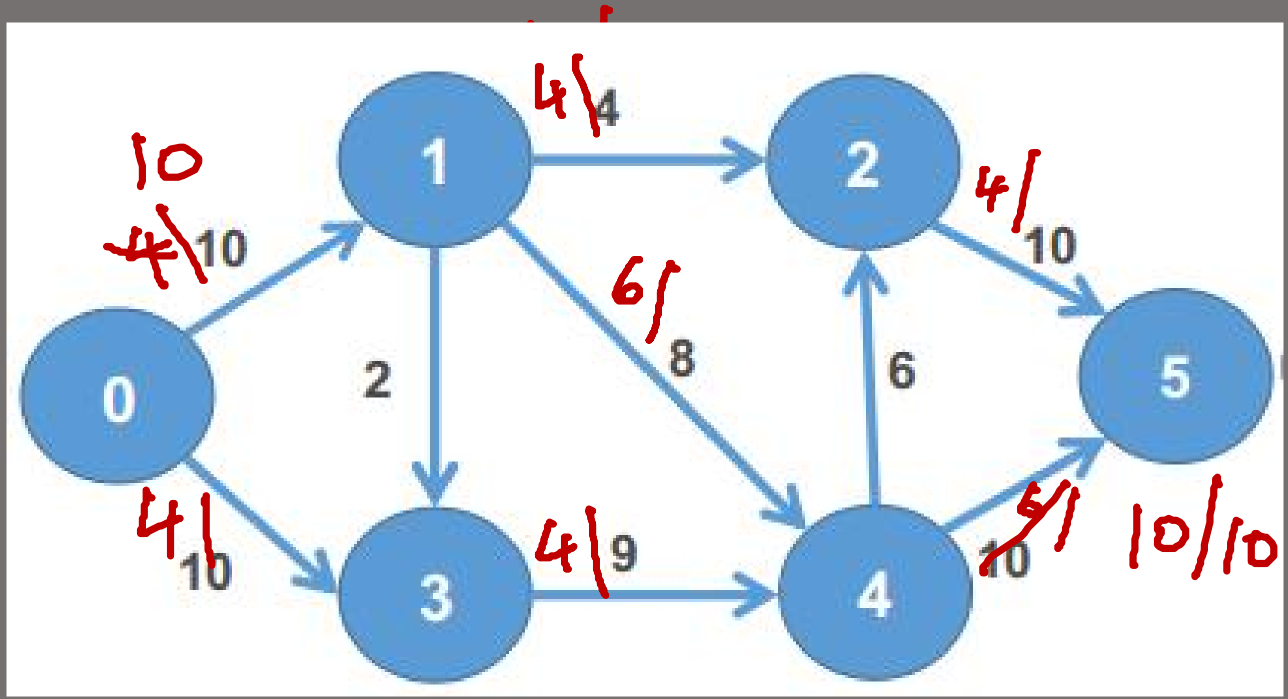
	0	1	2	3	4	5
0	0	0	0	10	0	0
1	10	0	0	2	2	0
2	0	4	0	0	0	6
3	0	0	0	0	9	0
4	0	6	6	0	0	4
5	0	0	4	0	6	0



Residual Matrix

Path	Bottleneck
0 – 1 – 2 – 5	4
0 – 1 – 4 – 5	6
0 – 3 – 4 – 5	4

	0	1	2	3	4	5
0	0	0	0	6	0	0
1	10	0	0	2	2	0
2	0	4	0	0	0	6
3	4	0	0	0	5	0
4	0	6	6	4	0	0
5	0	0	4	0	10	0



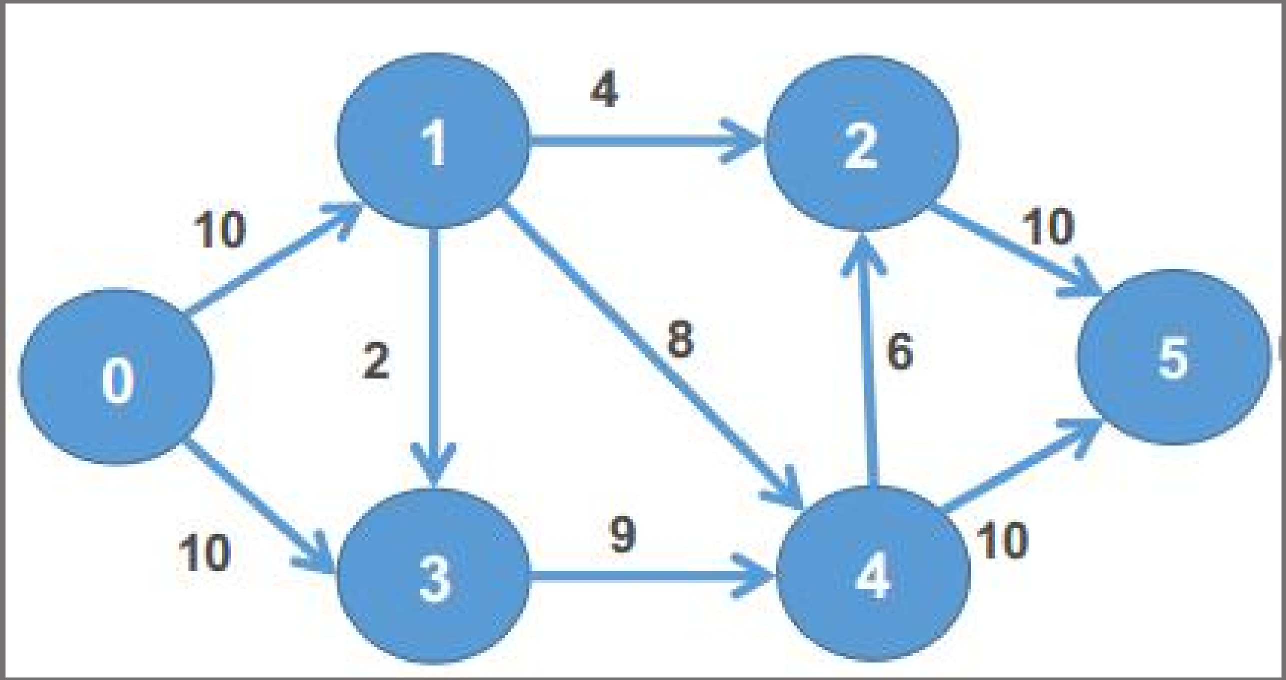
Residual Matrix

Path	Bottleneck
0 – 1 – 2 – 5	4
0 – 1 – 4 – 5	6
0 – 3 – 4 – 5	4
0 – 3 – 4 – 2 – 5	5

	0	1	2	3	4	5
0	0	0	0	1	0	0
1	10	0	0	2	2	0
2	0	4	0	0	5	1
3	9	0	0	0	0	0
4	0	6	1	9	0	0
5	0	0	9	0	10	0

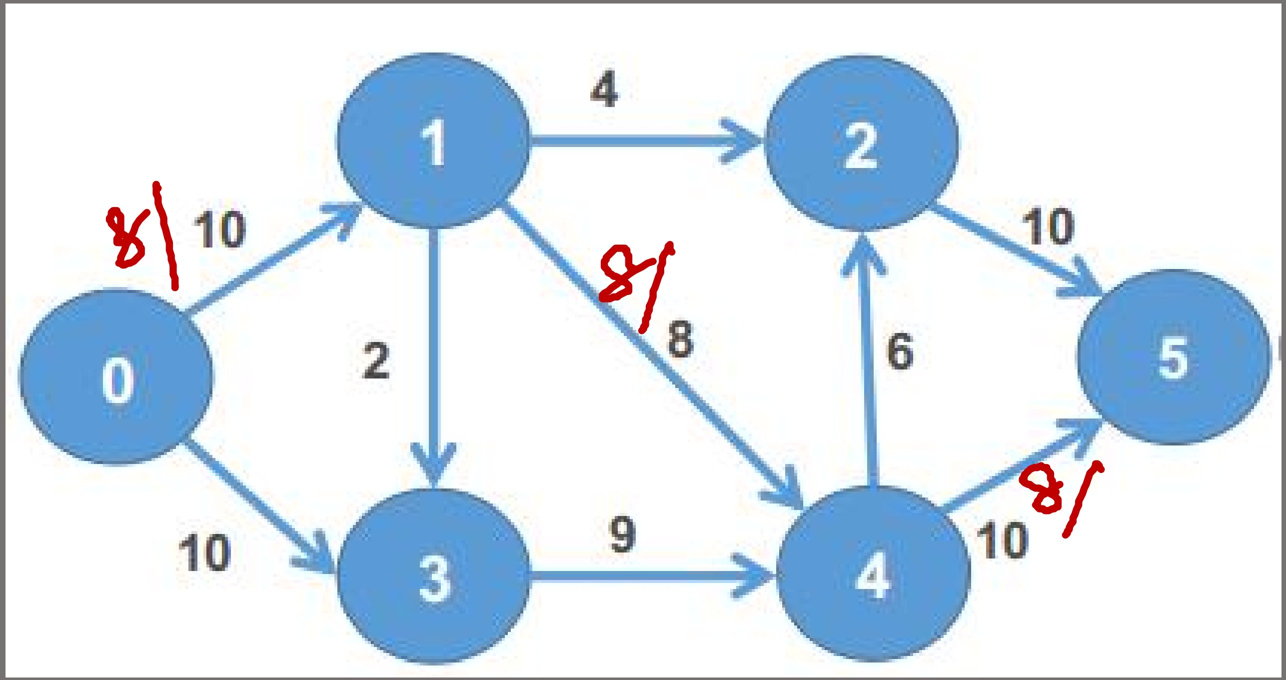
Theoretical Solution using of Ford Fulkerson Algorithm

- Augumenting path can be chosen randomly
  - through non–full forward edges or
  - through non–zero bacward edges



Theortically Original Matrix

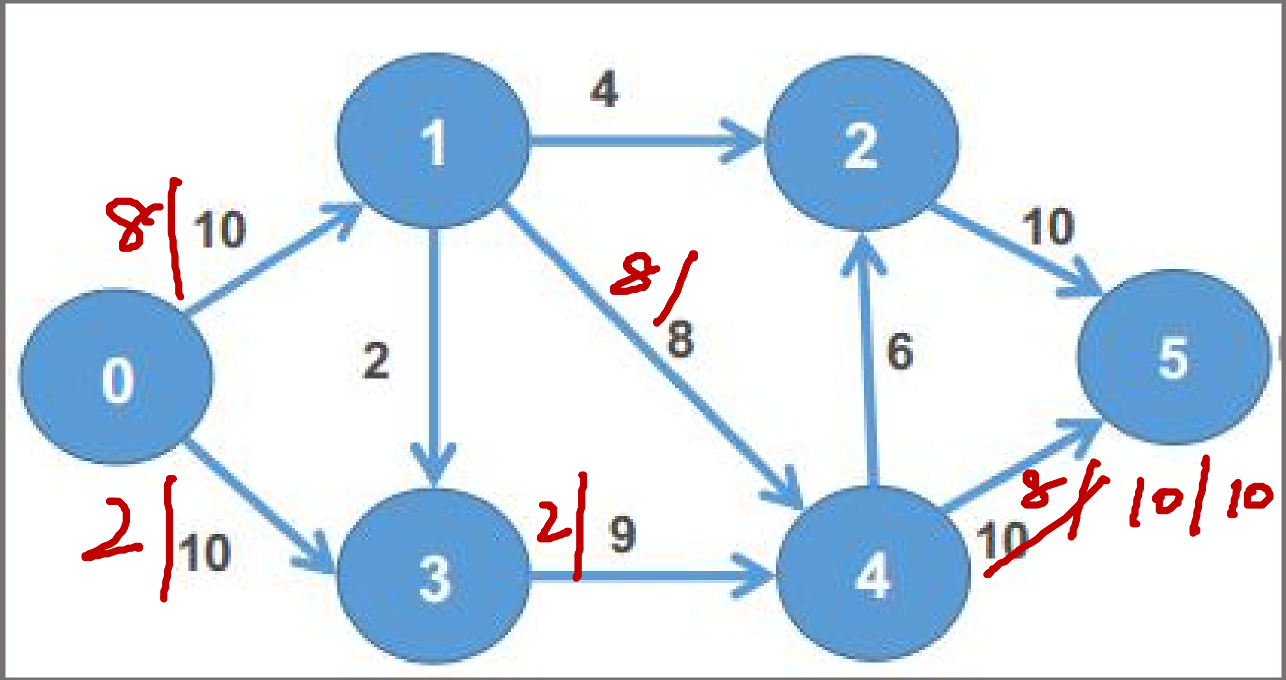
	0	1	2	3	4	5
0	0	10	0	10	0	0
1	0	0	4	2	8	0
2	0	0	0	0	0	10
3	0	0	0	0	9	0
4	0	0	6	0	0	10
5	0	0	0	0	0	0



Theortically Residual Matrix

Path	Bottleneck
0 – 1 – 4 – 5	8

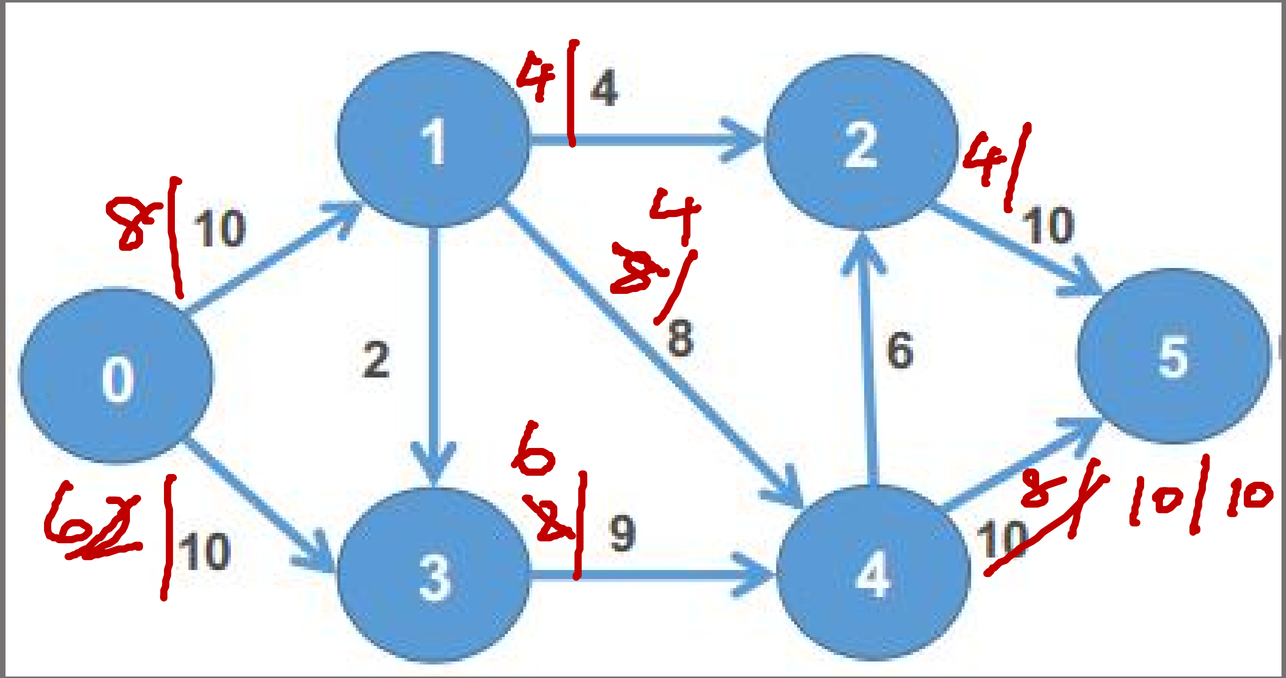
	0	1	2	3	4	5
0	0	2	0	10	0	0
1	8	0	4	2	0	0
2	0	0	0	0	0	10
3	0	0	0	0	9	0
4	0	8	6	0	0	2
5	0	0	0	0	8	0



Theortically Residual Matrix

Path	Bottleneck
0 – 1 – 4 – 5	8
0 – 3 – 4 – 5	2

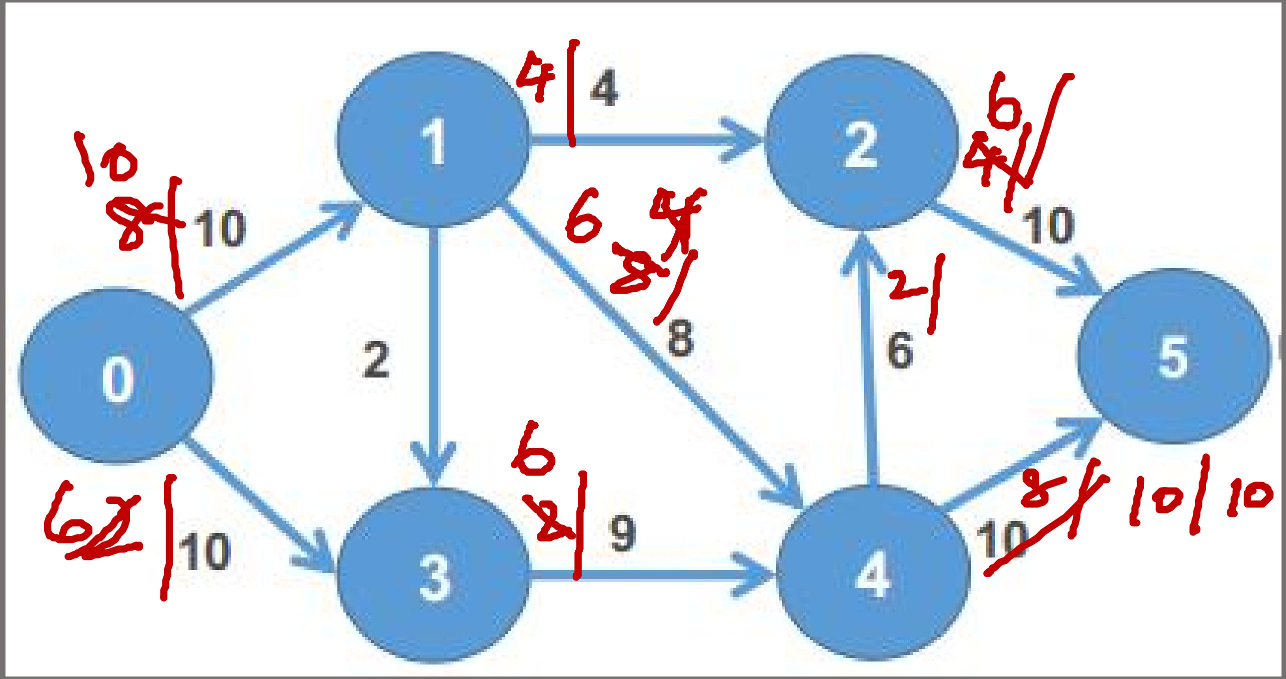
	0	1	2	3	4	5
0	0	2	0	8	0	0
1	8	0	4	2	0	0
2	0	0	0	0	0	10
3	2	0	0	0	7	0
4	0	8	6	2	0	0
5	0	0	0	0	10	0



Theortically Residual Matrix

Path	Bottleneck
0 – 1 – 4 – 5	8
0 – 3 – 4 – 5	2
0 – 3 – 4 – 1 – 2 – 5	4

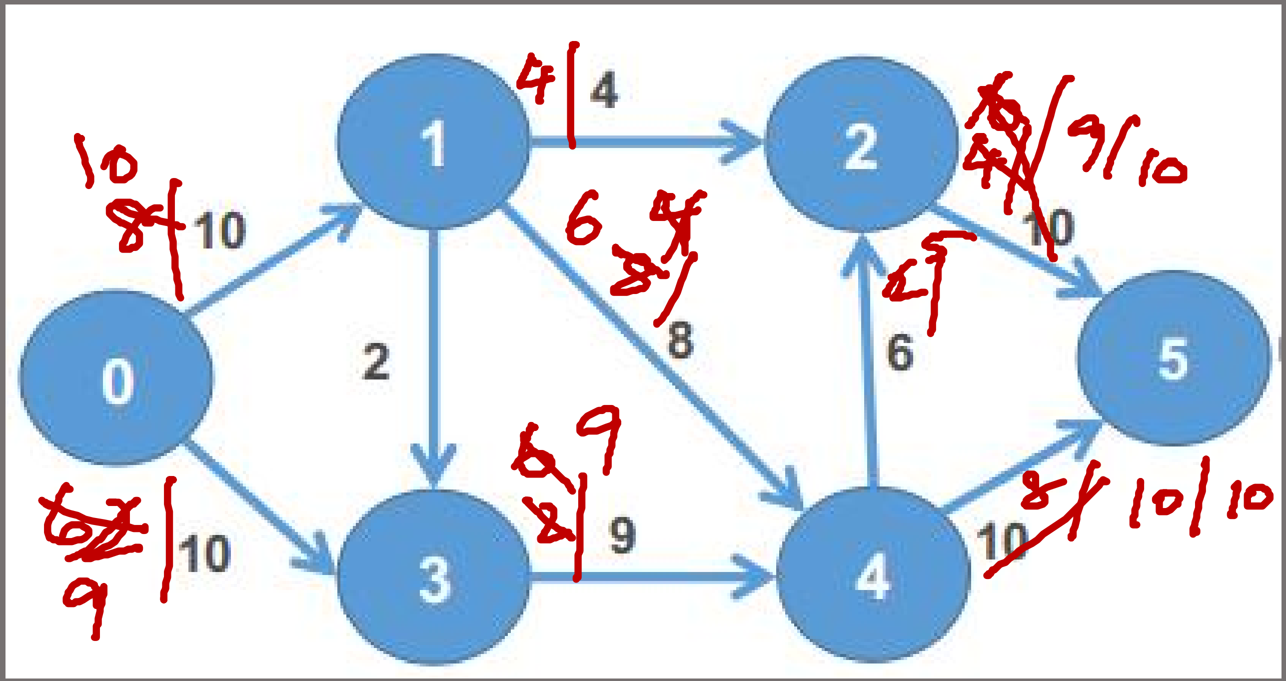
	0	1	2	3	4	5
0	0	2	0	4	0	0
1	8	0	0	2	4	0
2	0	4	0	0	0	6
3	6	0	0	0	3	0
4	0	4	6	6	0	0
5	0	0	4	0	10	0



Theortically Residual Matrix

Path	Bottleneck
0 – 1 – 4 – 5	8
0 – 3 – 4 – 5	2
0 – 3 – 4 – 1 – 2 – 5	4
0 – 1 – 4 – 2 – 5	2

	0	1	2	3	4	5
0	0	0	0	4	0	0
1	10	0	0	2	2	0
2	0	4	0	0	2	4
3	6	0	0	0	3	0
4	0	6	4	6	0	0
5	0	0	6	0	10	0



Theortically Residual Matrix

Path	Bottleneck
0 – 1 – 4 – 5	8
0 – 3 – 4 – 5	2
0 – 3 – 4 – 1 – 2 – 5	4
0 – 1 – 4 – 2 – 5	2
0 – 3 – 4 – 2 – 5	3

	0	1	2	3	4	5
0	0	0	0	1	0	0
1	10	0	0	2	2	0
2	0	4	0	0	5	1
3	9	0	0	0	0	0
4	0	6	1	9	0	0
5	0	0	9	0	10	0

Complexity of Ford Fulkerson Algorithm

- $O(E * M)$ where E is the number of edges
- M – Maximum flow through the network
- Worst case unit value is added to flow in each iteration