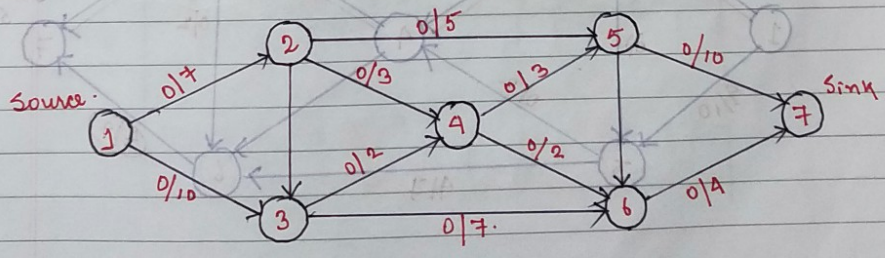
Network Flow

Ford Fulkerson Algorithm:

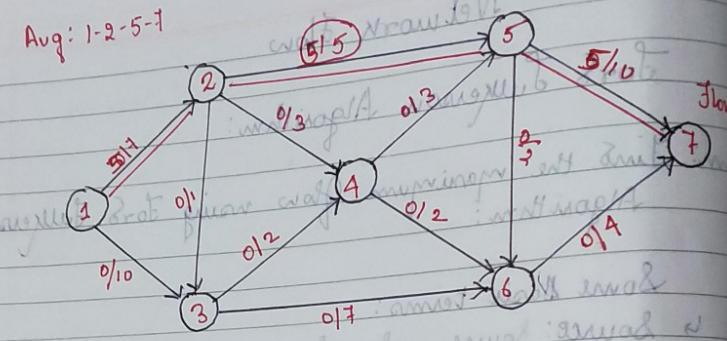
Aim: Find the maximum flow using Ford Fulkerson Algorithm:

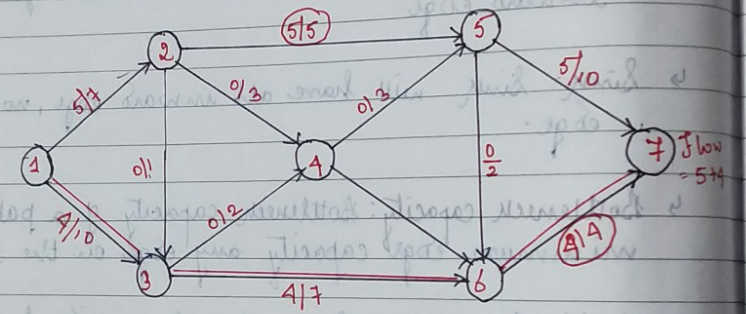
Some basic terms:

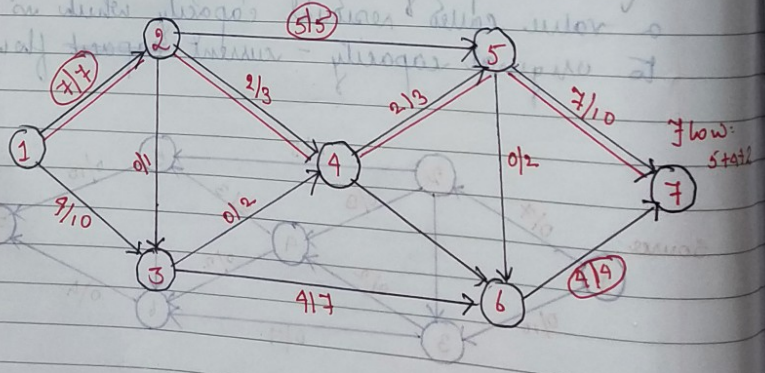
- Source: Source vertex has all outward edge and no inward edge.
- Sink: Sink will have all inward edge, no outward edge.
- Bottleneck capacity: Bottleneck capacity of a path is the minimum edge capacity any edge in the path.
- Residual capacity: Every edge of a residual graph has a value called residual capacity which is equal to original capacity - current ~~capacity~~ flow.

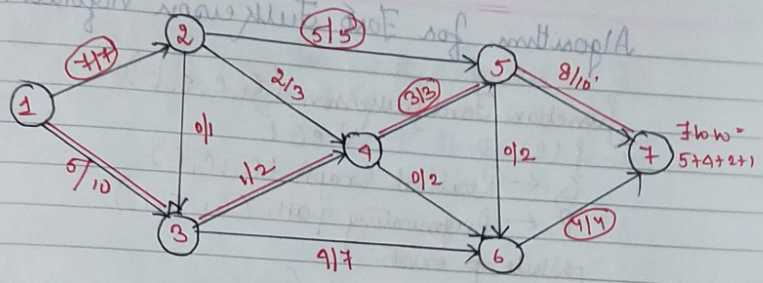


Aug: 1-2-5-7


5/5
5/10
5/7
0/3
0/3
0/1
0/2
0/2
0/10
0/7
0/4
Flow: 5


5/5
5/7
0/3
0/3
5/10
0/1
0/2
4/10
4/7
4/4
Flow = 5+4


5/5
7/7
2/3
2/3
7/10
0/1
0/2
9/10
0/2
4/7
4/4
Flow: 5+4+2



Maximum flow through given network ⇒
$5+4+2+1 = 12$

Pseudocode:

↳ Set total flow = 0

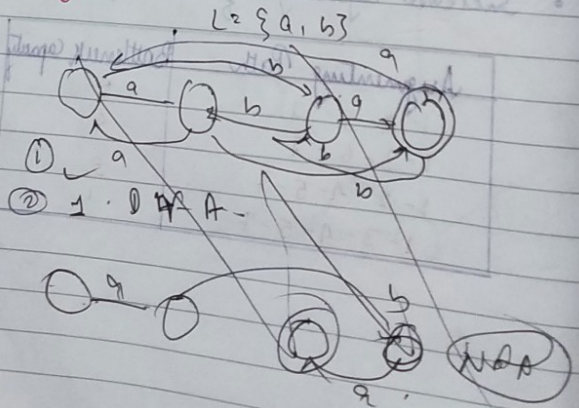
↳ Repeat until there is no path from s to t:
- Run DFS from s to find flow path to ensure reach t.
- Let f be the min. capacity value on path.
- Add f to flow total.

↳ For each edge $u \to v$ on path:
- Decrease capacity of the edge $c(u \to v)$ by f.
- Increase capacity of the edge $c(v \to u)$ by f.

Augmenting Path	Bottleneck capacity
1-2-5-7	5
1-3-6-7	4
1-2-4-5-7	2
1-3-4-5-7	1

Algorithm for Ford Fulkerson Algorithm.

```
function FordFulkerson (G, C, s, t):
    f(e) : 0                    ∀ e ∈ E
    G_f ← Residual Graph (G, C, f)
    P ← Augmenting path (G_f, s, t)
    while p exists:
        b ← min wc in G_f
        for each e ∈ p:
            if e ∈ E:
                f(e) ← f(e) + b
            else:
                f(a) ← f(a) - b
            end if
        end for
        G_f ← Residual Graph (G, C, f):
        P ← Augmenting path (G_f, s, t)
    end while
    return f
end function:
```



Maximum flow Min cut theorem:

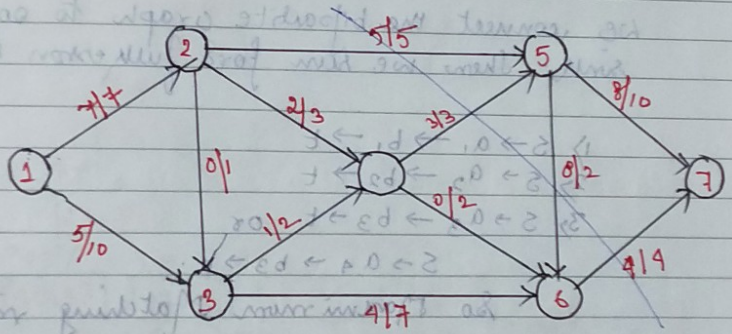
Maximum flow min cut theorem states:

Max Flow = Min cut.

In Previous Ford Fulkerson example:

Flow: 12

So we need to find a cut such that it separates source and sink and the min cut value is 12.

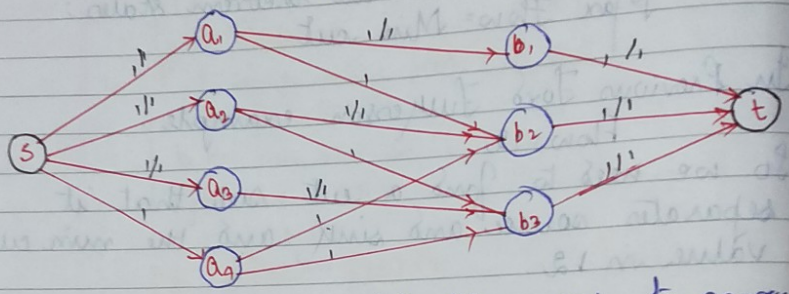


Here flow was 12.

Also from cut = 5 + 3 + 4 = 12

Note: We only consider those edges which are from left to right (source to sink) in the cut rest are ignored.

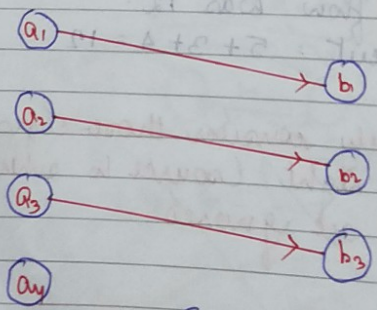
Bipartite Matching: Given a Bipartite ~~Mat~~ Graph we need to find Bipartite Matching.



We connect the bipartite graph to source & sink. Then we run ford fulkerson Algorithm

1) $s \to a_1 \to b_1 \to t$
2) $s \to a_2 \to b_2 \to t$
3) $s \to a_3 \to b_3 \to t$ or
   $s \to a_4 \to b_3 \to t$.

So Maximum Matching is 3.



$\{(a_1, b_1), (a_2, b_2), (a_3, b_3)\}$