MAXIMUM FLOW

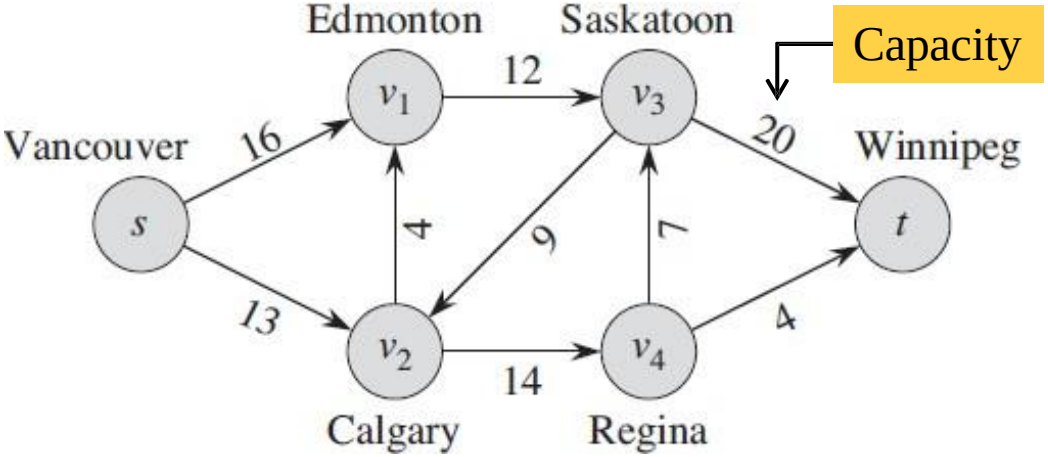
Max-Flow Min-Cut Theorem (Ford Fukerson's Algorithm)

What is Flow Network?

- Flow network is a directed graph $G = (V,E)$ such that each edge has a non-negative capacity $c(u,v) \geq 0$.
- Two distinguished vertices exist in G namely:
  - Source (produces flow and denoted by s): In-degree of this vertex is 0.
  - Sink (consumes flow and denoted by t): Out-degree of this vertex is 0.
- Flow in a network is an integer-valued function f defined on the edges of G satisfying $0 \leq f(u,v) \leq c(u,v)$, for every edge (u,v) in E.

Contd…

- Each edge $(u,v) \in E$ has a non-negative capacity $c(u,v)$.
- If $(u,v) \notin E$ assume $c(u,v)=0$.
- Two special vertices, source s and sink t.
- Every vertex $v \in V$ is on some path from s to t.



$c(s,v_1) = 16$
$c(v_1,s) = 0$
$c(v_2,s) = 0$
$c(v_4,t) = 4$ …

Conditions for Flow Network

- For each edge (u,v) in E, the flow $f(u,v)$ in G is a real valued function $f : V \times V \rightarrow R$ that satisfies the following properties:

- Capacity constraints – The flow along an edge can not exceed its capacity. $\forall (u,v) \in E \; f(u,v) \leq c(u,v)$

- Skew symmetry – The net flow from u to v must be the opposite of the net flow from v to u.

$$\forall (u,v) \in E \; f(u,v) = -f(v,u)$$

- Flow conservation – Unless u is s or t, the net flow to a node is zero. $\forall u \in V : u \neq s \text{ and } u \neq t \Rightarrow \sum_{w \in V} f(u,w) = 0$

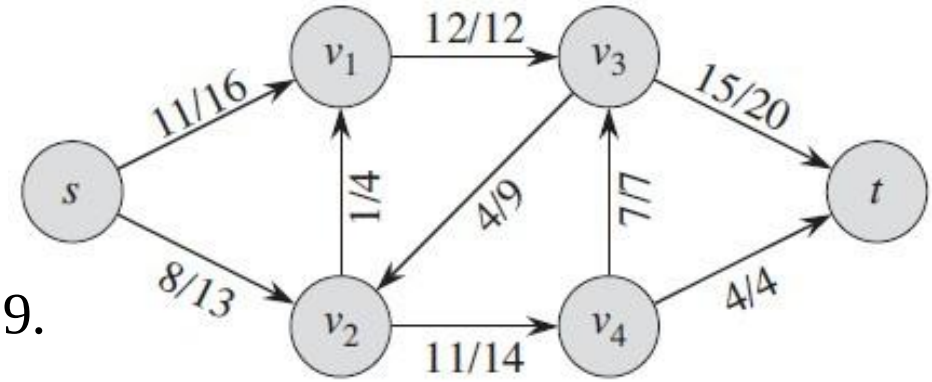
The Value of a Flow

- The value of a flow is given by:

$$| f | = \sum_{v \in V} f(s, v) = \sum_{v \in V} f(v, t)$$

- The flow into the sink node (t) is same as flow going out from the source node (s) and thus the flow is conserved.
- Total amount of flow from source s is equal to total amount of flow into the sink t.

Example



- A flow f in G with value $|f| = 19$.
- Each edge (u, v) is labeled by $f(u,v)/c(u,v)$. Slash notation separates the flow and capacity; it does not indicate division.

$$|f| = \sum_{v \in V} f(s,v) = \sum_{v \in V} f(v,t) = 19$$

- The flow across nodes v_1, v_2 , v_3, and v_4 are also conserved as flow into them = flow out of them.
- v_1: 11 + 1 = 12
- v_2 : 8 + 4 = 1 + 11
- v_3: 12 + 7 = 4 + 15, and
- v_4: 11 = 7 + 4.

Edge	Capacity	Flow	Comment
(s,v_1)	16	11	$11 \leq 16$ Valid
(s,v_2)	13	8	$8 \leq 13$ Valid
(v_2,v_1)	4	1	$1 \leq 4$ Valid
(v_1,v_3)	12	12	$12 \leq 12$ Valid
(v_3,v_2)	9	4	$4 \leq 9$ Valid
(v_2,v_4)	14	11	$11 \leq 14$ Valid
(v_4,v_3)	7	7	$7 \leq 7$ Valid
(v_3,t)	20	15	$15 \leq 20$ Valid
(v_4,t)	4	4	$4 \leq 4$ Valid

The Maximum Flow Problem

- Given:
  - Graph $G(V,E)$,
  - $f(u,v)$ = flow on edge (u,v),
  - $c(u,v)$ = capacity of edge (u,v),
  - s = source node, t = sink node.
- Maximize: $|f|$
- Subject to:

$$\sum_{v \in V} f(u,v) - \sum_{v \in V} f(v,u) = 0, \forall u \neq s,t$$

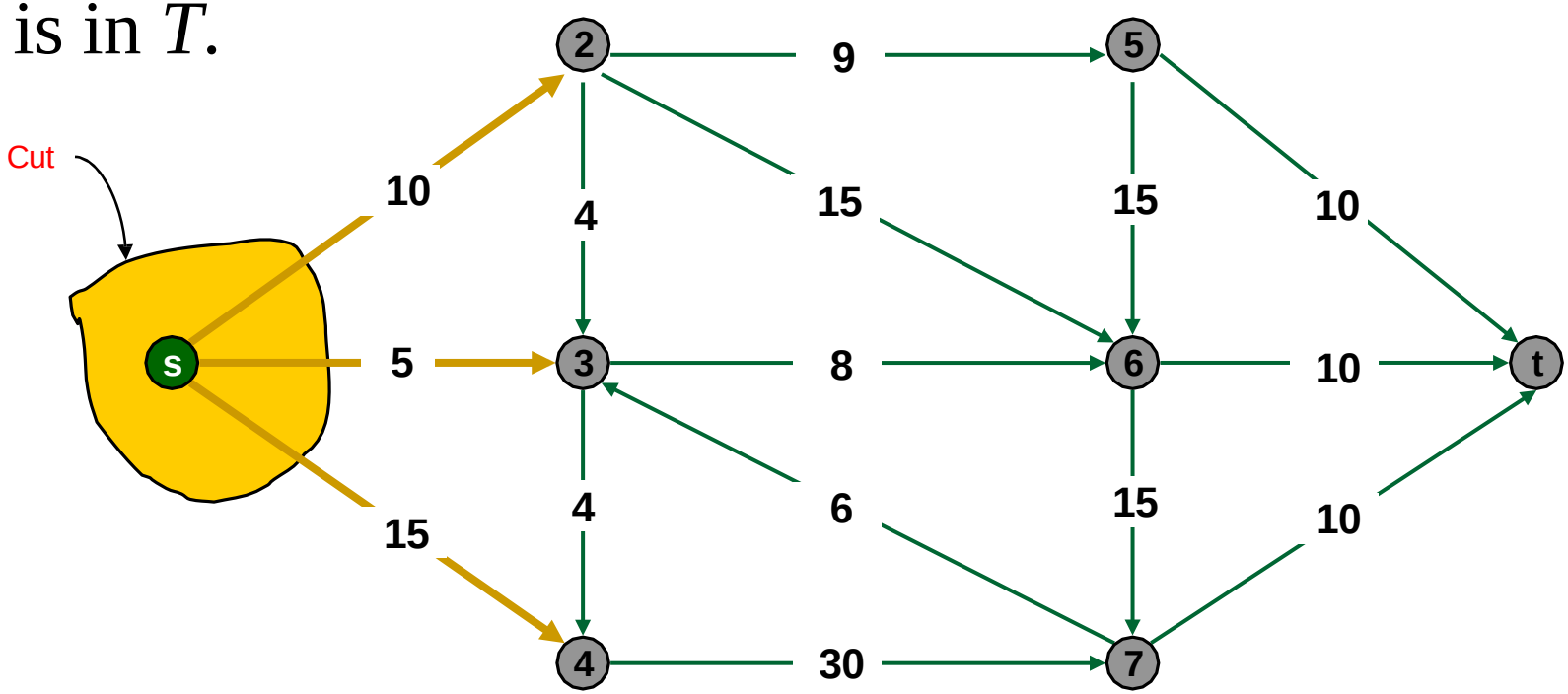
$$\sum_{v \in V} f(s,v) = \sum_{v \in V} f(v,t) = |f|$$

$$0 \leq f(u,v) \leq c(u,v), \forall (u,v) \in E$$

In simple terms maximize the s to t flow, while ensuring that the flow is feasible.

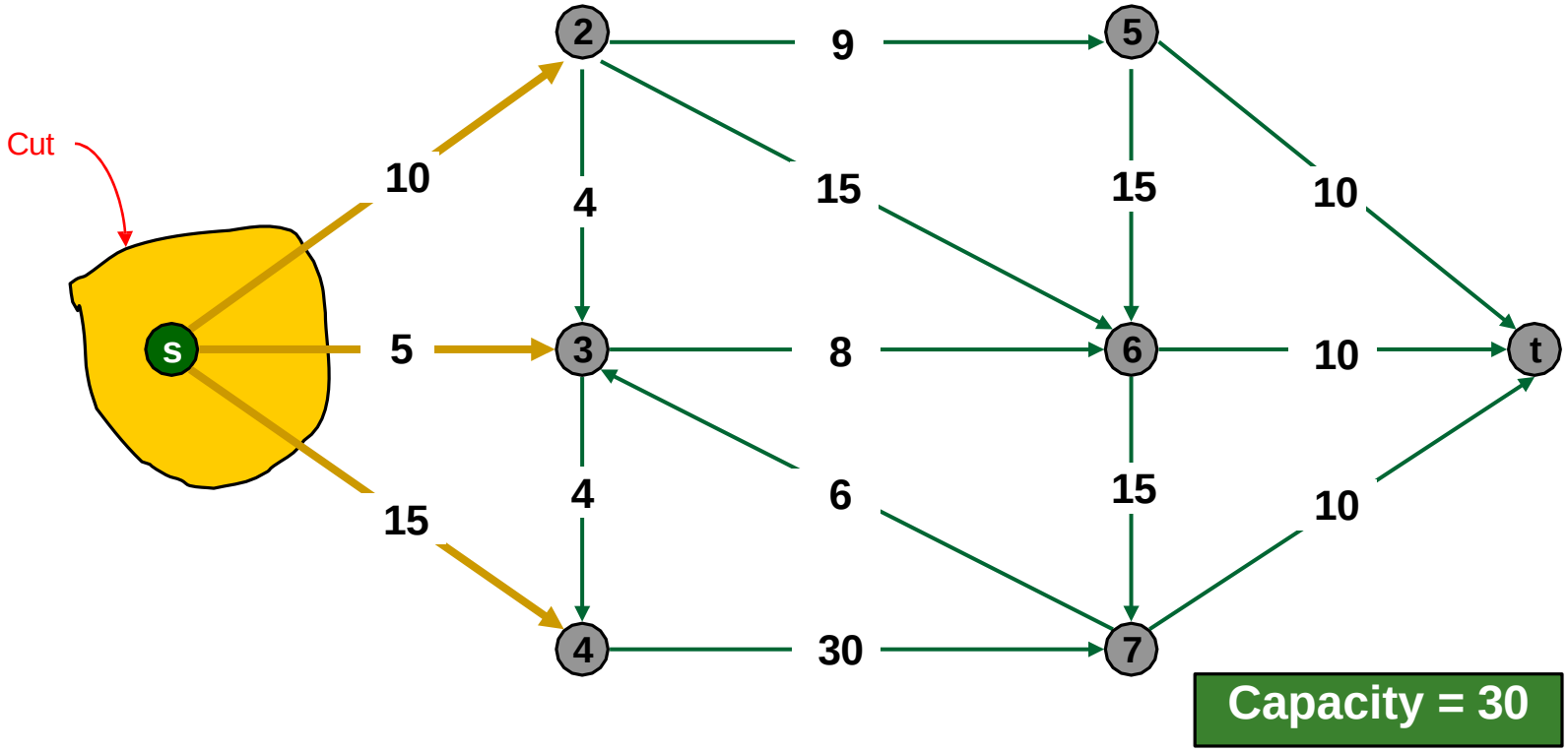
Cuts of Flow Networks

- A Cut in a network is a partition of V into S and T ($T = V - S$) such that s (source) is in S and t (target) is in T.



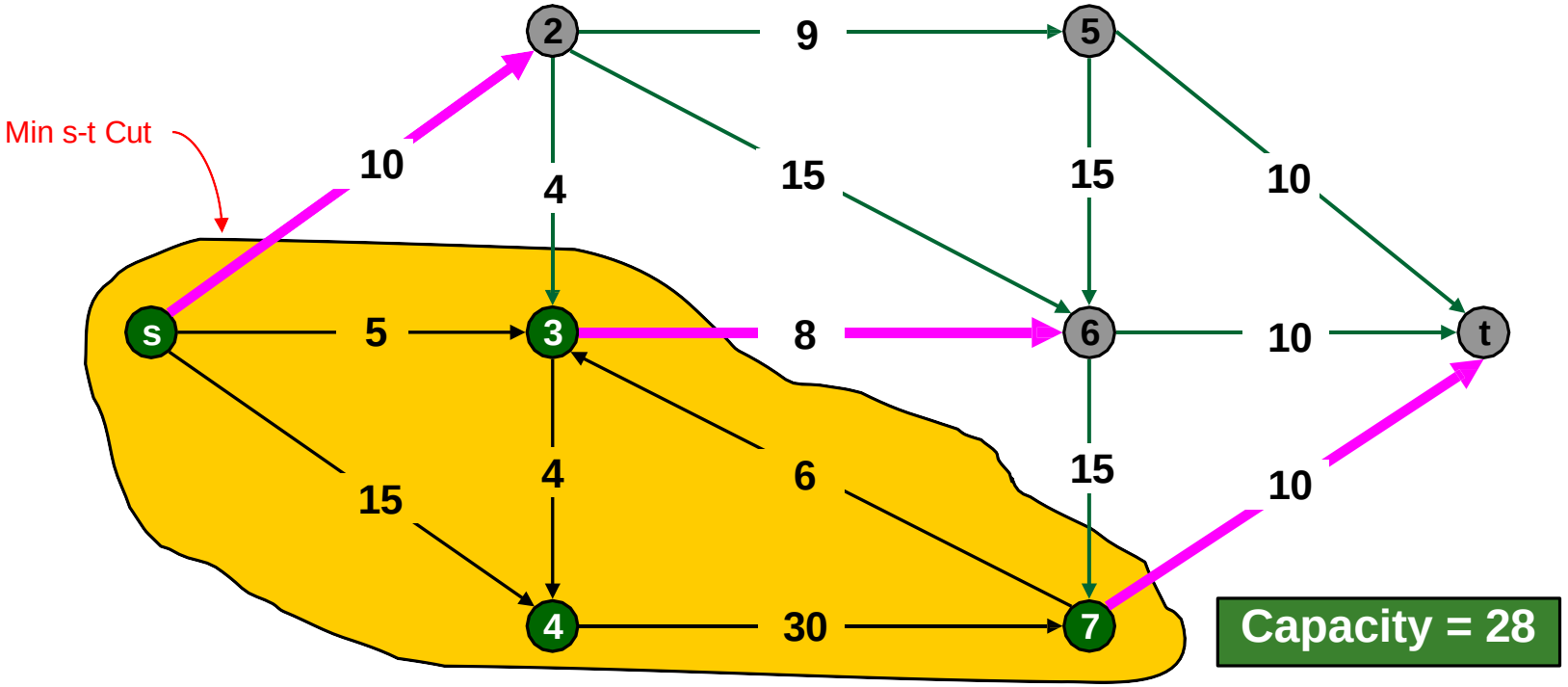
Capacity of Cut (S,T)

$$c(S,T) = \sum_{u \in S, v \in T} c(u,v)$$



Min Cut

- Min s-t cut (also called as a Min Cut) is a cut of minimum capacity

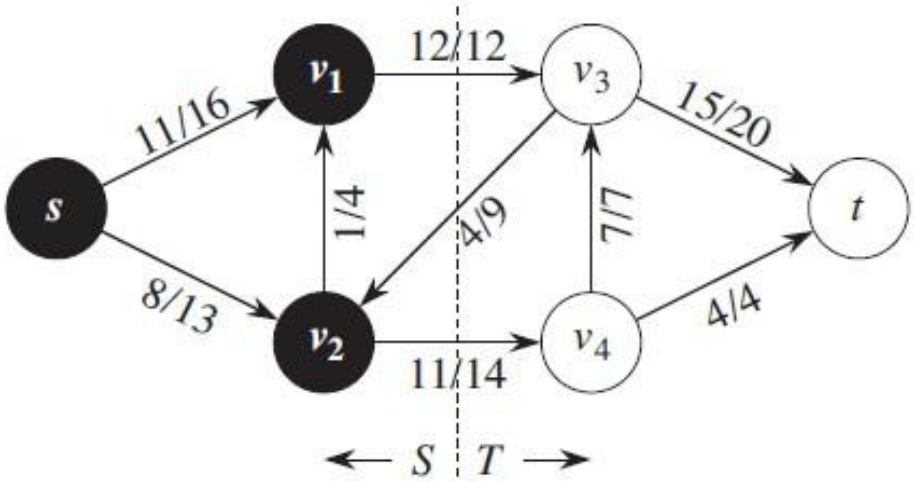


Flow of Min Cut

- Let f be the flow and let (S,T) be a cut. Then $|f| \leq$ capacity(S,T).

$$
\begin{aligned}
|f| &= f(S,T) \\
&= \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{u \in S} \sum_{v \in T} f(v,u) \\
&\leq \sum_{u \in S} \sum_{v \in T} f(u,v) \\
&\leq \sum_{u \in S} \sum_{v \in T} c(u,v) \\
&= c(S,T) .
\end{aligned}
$$

Capacity = $c(v_1,v_3) + c(v_2,v_4)$
$= 12 + 14 = 26.$
Flow = $f(v_1,v_3) + f(v_2,v_4) - f(v_3,v_2)$
$= 12 + 11 – 4 = 19.$



Ford-Fulkerson Method

- FORD-FULKERSON-METHOD(G,s,t)

1. initialize flow f to 0
2. **while** there exists an augmenting path p in the residual network G_f
3. augment flow f along p
4. **return** f

ResidualNetwork

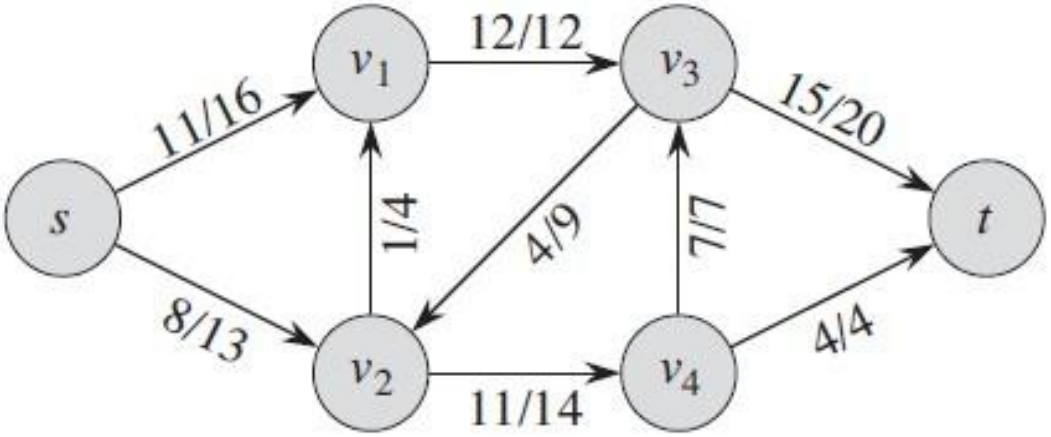
- Given a flow network $G = (V,E)$ and a flow f, the ***residual network*** of G induced by f is $G_f = (V,E_f)$, where

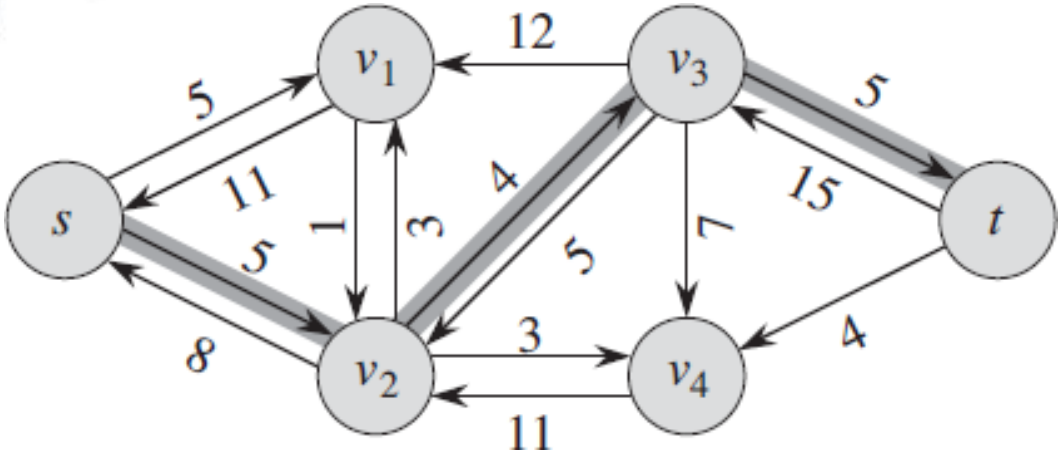
  $E_f = \{(u,v) \in V \times V : c_f(u,v) > 0\}$ with residual capacity c_f

$$c_f(u,v) = \begin{cases} c(u,v) - f(u,v) & \text{if } (u,v) \in E, \\ f(v,u) & \text{if } (v,u) \in E, \\ 0 & \text{otherwise.} \end{cases}$$

Contd…

$$c_f(u,v) = \begin{cases} c(u,v) - f(u,v) & \text{if } (u,v) \in E, \\ f(v,u) & \text{if } (v,u) \in E, \\ 0 & \text{otherwise.} \end{cases}$$





Augmenting Path

- An augmenting path p is a simple path from s to t on a residual network G_f.
- The maximum capacity by which we can increase the flow on each edge in an augmenting path p is known as the residual capacity of p, given by

$$c_f(p) = \min\{c_f(u,v) : (u,v) \text{ is on } p\}$$

Max-flow min-cut theorem

- If f is a flow in a flow network $G = (V,E)$ with source s and sink t, then the following conditions are equivalent:

1. f is a maximum flow in G.
2. The residual network G_f contains no augmenting paths.
3. $|f| = c(S,T)$ for some cut (S,T) of G.

Note:

If $|f| = c(S,T)$, then $c(S,T)$ is the required min-cut.

Basic Ford-Fulkerson Algorithm

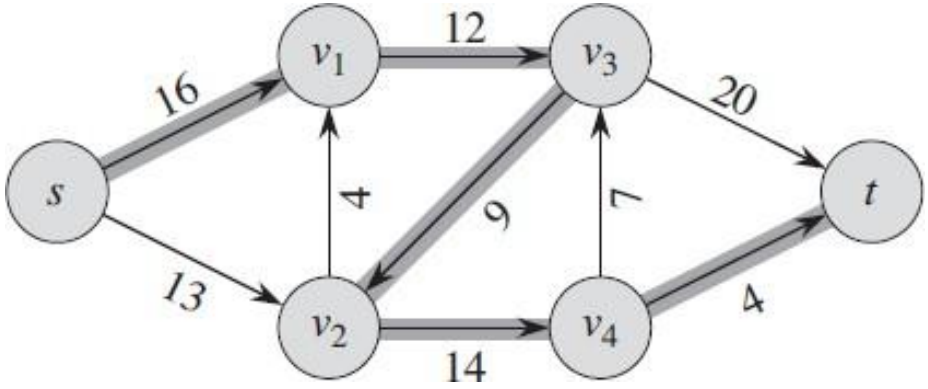
```
FORD-FULKERSON(G, s, t)
1  for each edge (u, v) ∈ G.E
2      (u, v).f = 0
3  while there exists a path p from s to t in the residual network G_f
4      c_f(p) = min{c_f(u, v) : (u, v) is in p}
5      for each edge (u, v) in p
6          if (u, v) ∈ E
7              (u, v).f = (u, v).f + c_f(p)
8          else (v, u).f = (v, u).f − c_f(p)
```

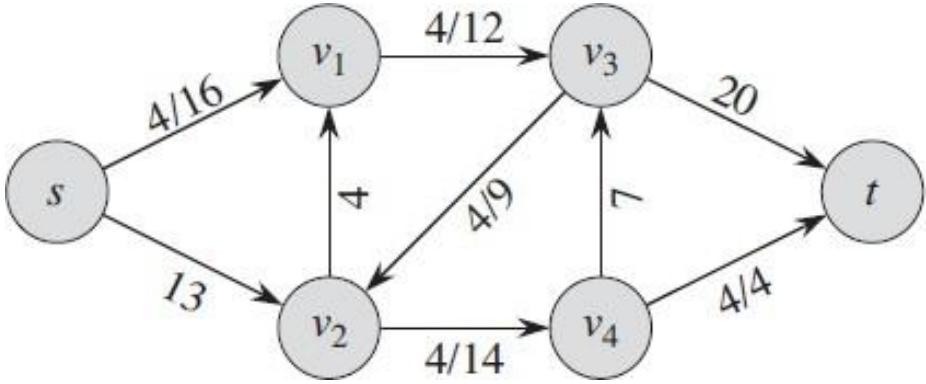
- Line 7. Add residual capacity to the flow over an existing edge (*u*,*v*) in *E*.
- Line 8. Subtract residual capacity from the flow over an existing edge (*v*,*u*) in *E*.

Example

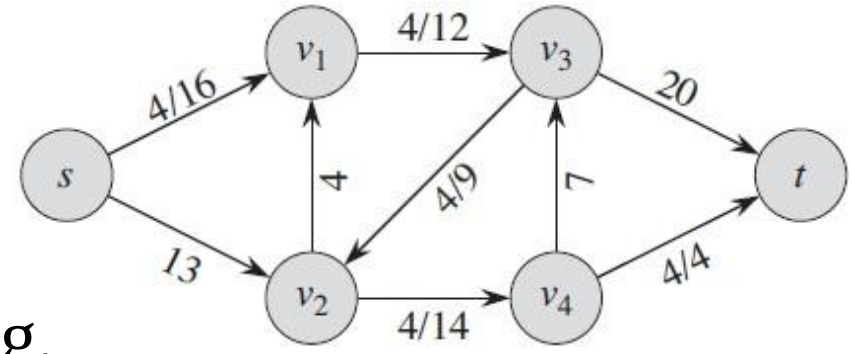
Flow = 4

- Initial flow = 0. Thus original network and initial residual network is same.
- Path: $s \rightarrow v_1 \rightarrow v_3 \rightarrow v_2 \rightarrow v_4 \rightarrow t$.
- Residual capacity: min(16, 12, 9, 14, 4) = 4.
- All edges in path exists in E, so add 4 to the initial flow for all the edges in the path (0 + 4 = 4).



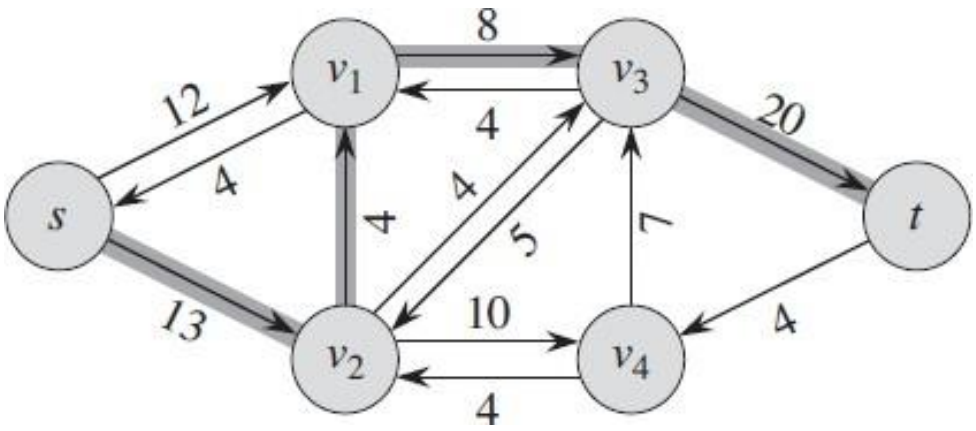


Contd…

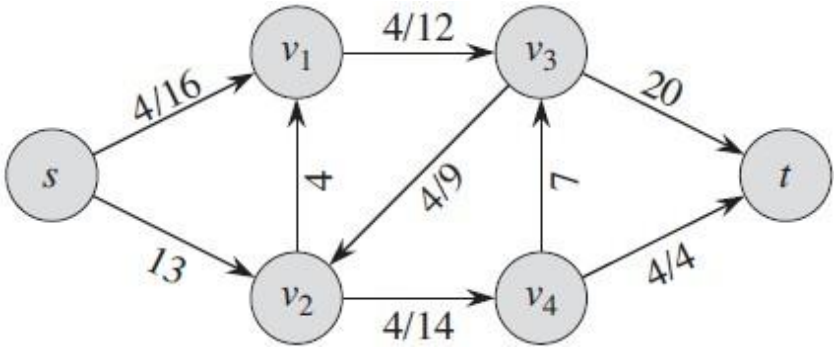


- Update residual network using,

$$c_f(u,v) = \begin{cases} c(u,v) - f(u,v) & \text{if } (u,v) \in E, \\ f(v,u) & \text{if } (v,u) \in E, \\ 0 & \text{otherwise.} \end{cases}$$

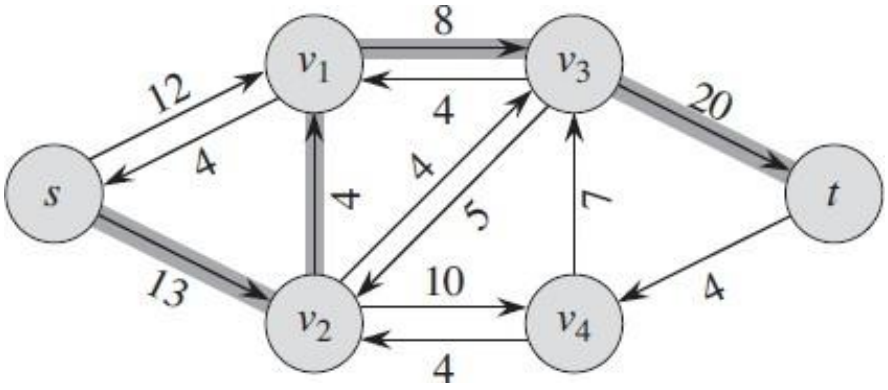


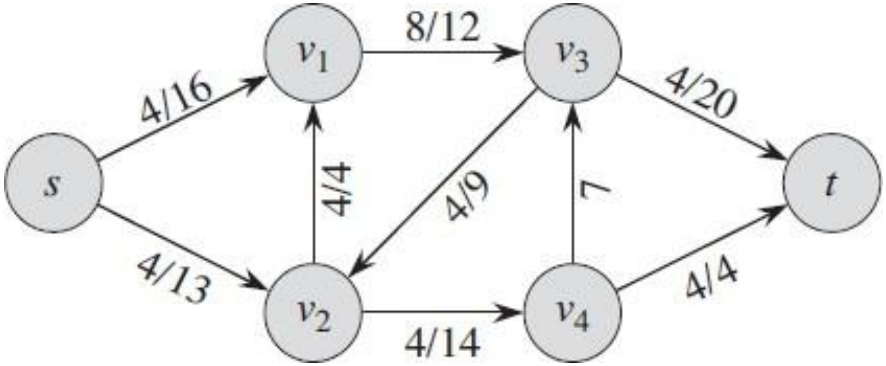
Contd…



- Path: $s \rightarrow v_2 \rightarrow v_1 \rightarrow v_3 \rightarrow t$.
- Residual capacity: min(13, 4, 8, 20) = 4.
- All edges in path exists in E, so add 4 to the flow for all the edges in the path.

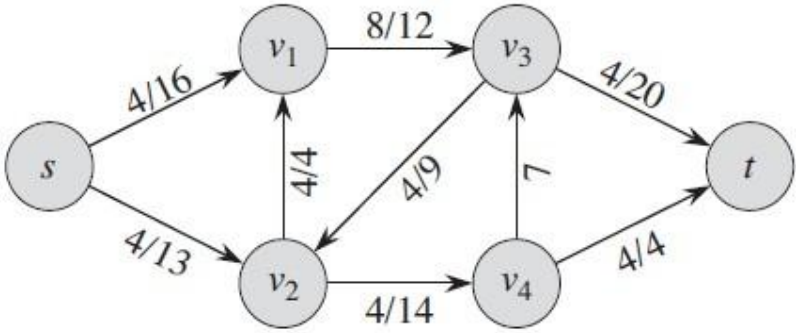
Flow = 4 + 4





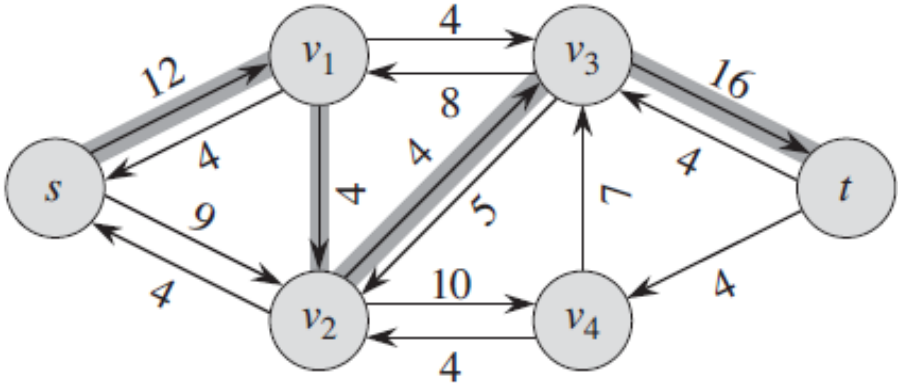
Contd…

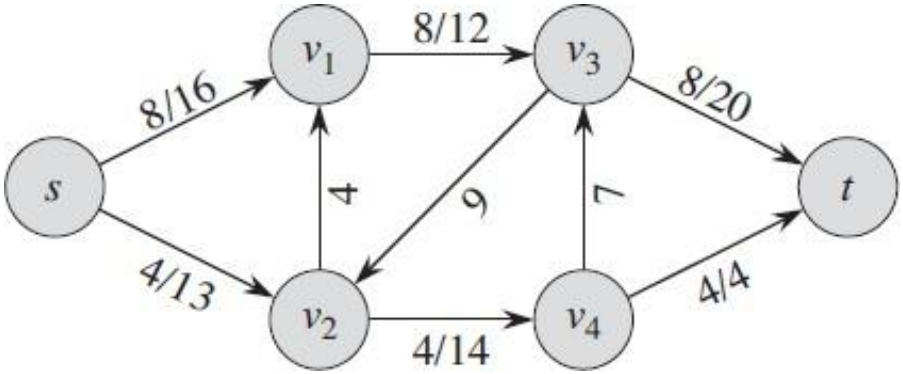
```
for each edge (u, v) in p
    if (u, v) ∈ E
        (u, v).f = (u, v).f + c_f(p)
    else (v, u).f = (v, u).f − c_f(p)
```



- Update residual network.
- Path: $s \rightarrow v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow t$.
- Residual capacity: min(12, 4, 4, 16) = 4.
- Edges (v_1,v_2) and (v_2,v_3) in path doesn't exist in E, so subtract residual capacity (4) from the previous flow for the edges (v_2,v_1) and (v_3,v_2). For both the edges updated flow is 4 – 4 = 0.

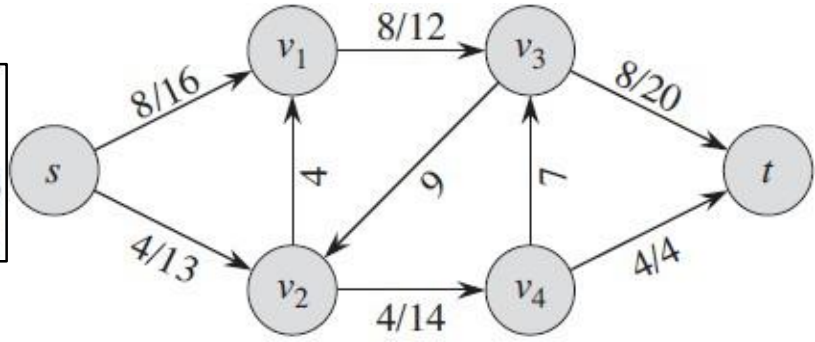
Flow = 4 + 4 + 4





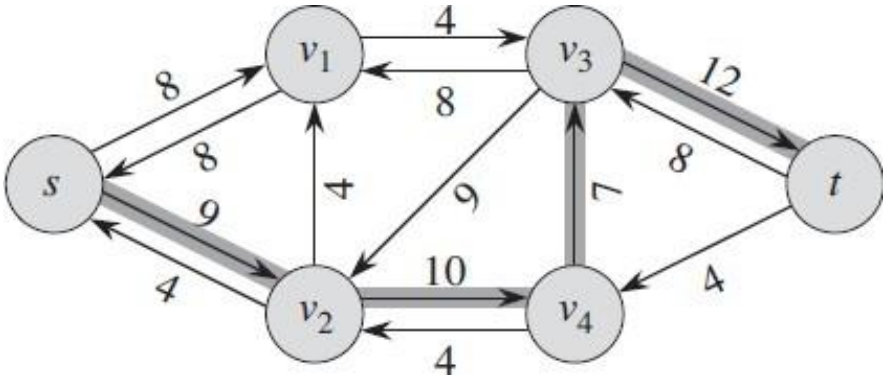
Contd…

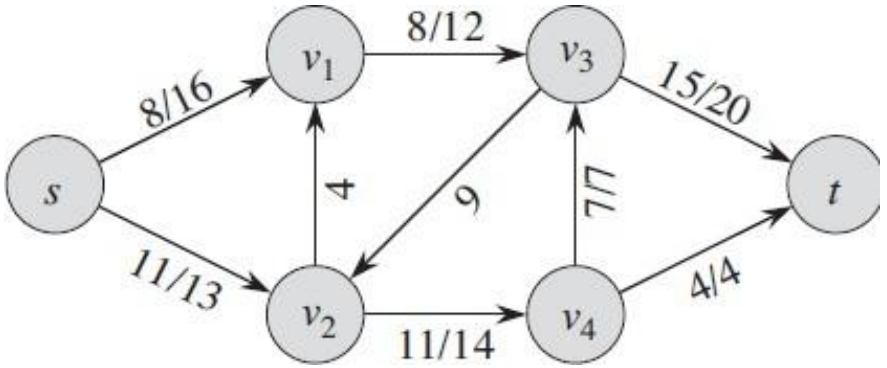
```
for each edge (u, v) in p
    if (u, v) ∈ E
        (u, v).f = (u, v).f + c_f(p)
    else (v, u).f = (v, u).f − c_f(p)
```



- Update residual network.
- Path: $s \rightarrow v_2 \rightarrow v_4 \rightarrow v_3 \rightarrow t$.
- Residual capacity: min(9, 10, 7, 12) = 7.
- All edges in path exists in E, so add 7 to the flow for all the edges in the path.

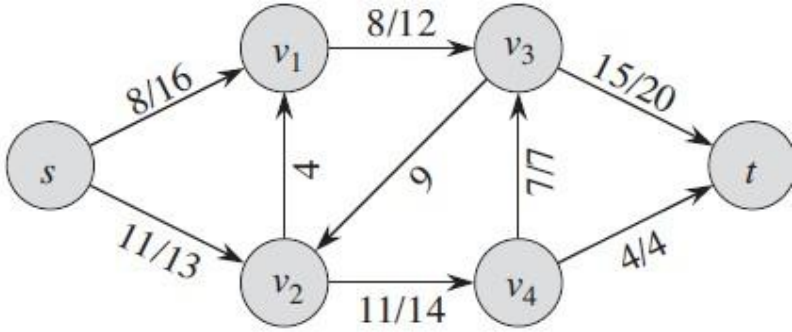
Flow = 4 + 4 + 4 + 7





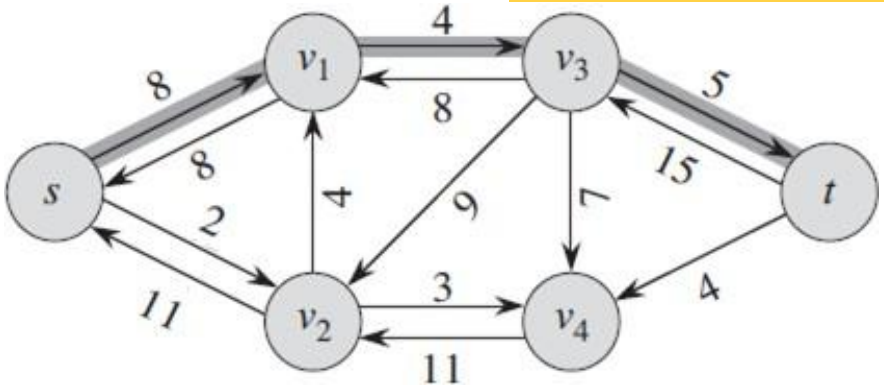
Contd…

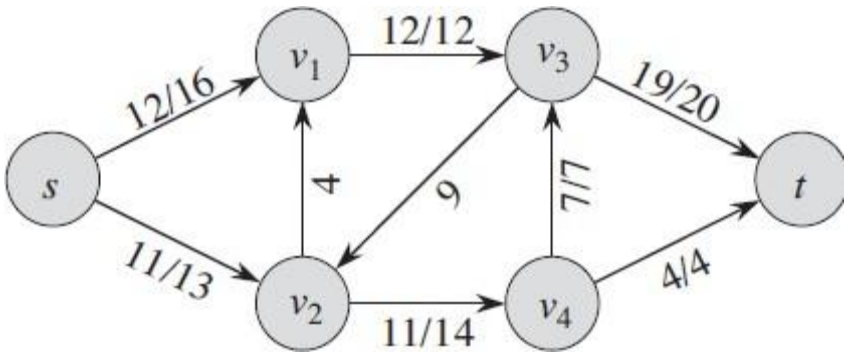
```
for each edge (u, v) in p
    if (u, v) ∈ E
        (u, v).f = (u, v).f + c_f(p)
    else (v, u).f = (v, u).f − c_f(p)
```



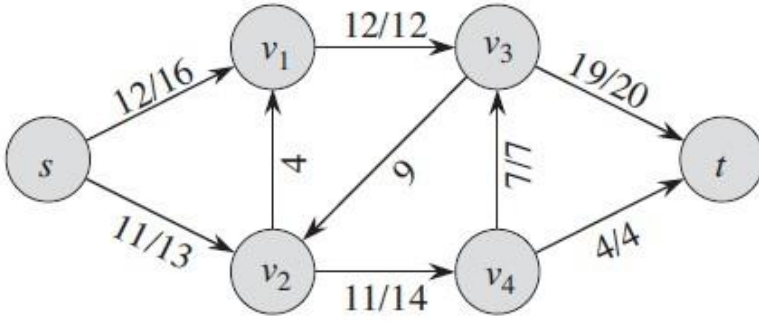
- Update residual network.
- Path: $s \rightarrow v_1 \rightarrow v_3 \rightarrow t$.
- Residual capacity: min(8, 4, 5) = 4.
- All edges in path exists in E, so add 4 to the flow for all the edges in the path.

Flow = 4 + 4 + 4 + 7 + 4



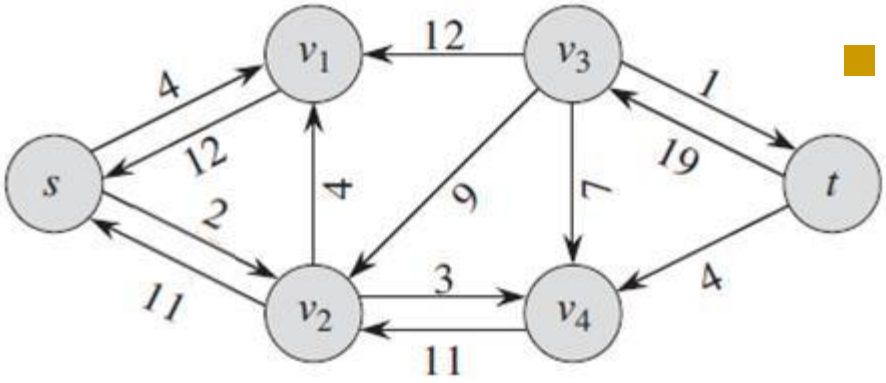


Contd…



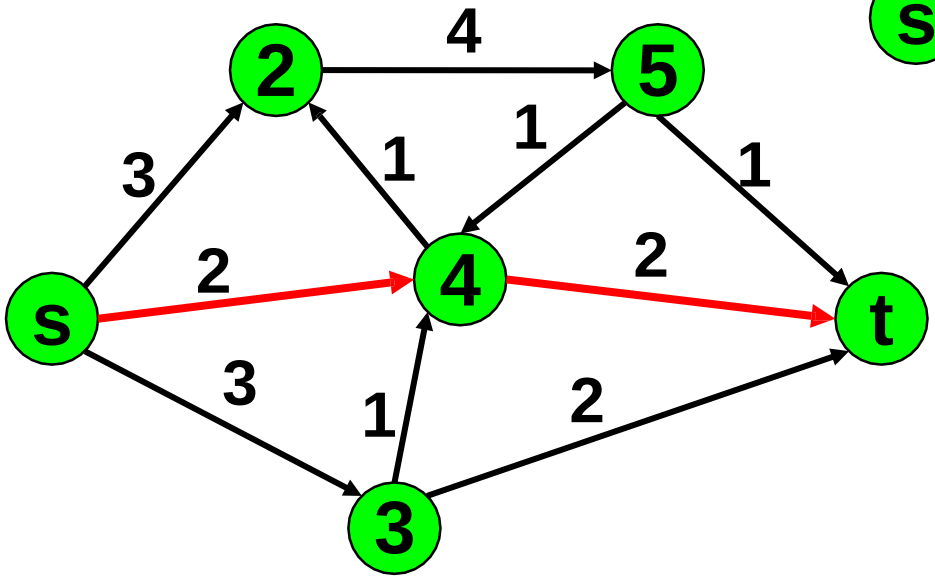
- Update residual network.
- No path exists in the residual network from *s* to *t*.
- Loop terminates and the final flow is

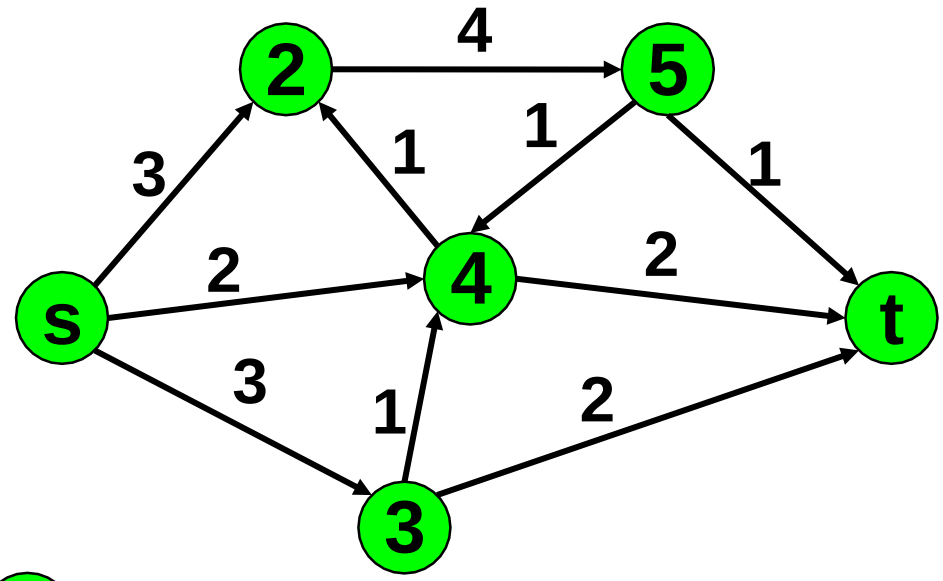
Flow = 4 + 4 + 4 + 7 + 4 = 23

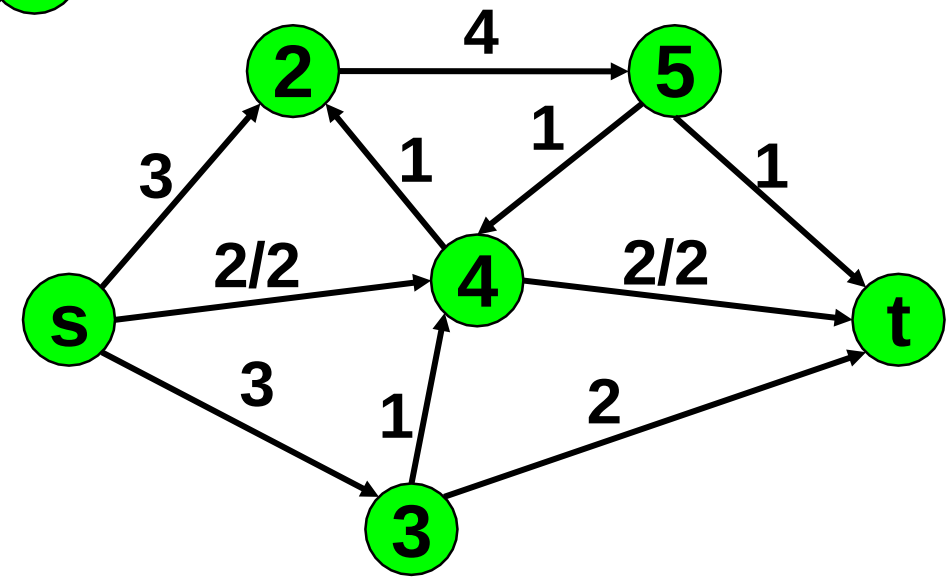


- $|f| = 23$.
- $c(\{s, v_1, v_2, v_4\}, \{v_3, t\}) = 23$.

Example 2

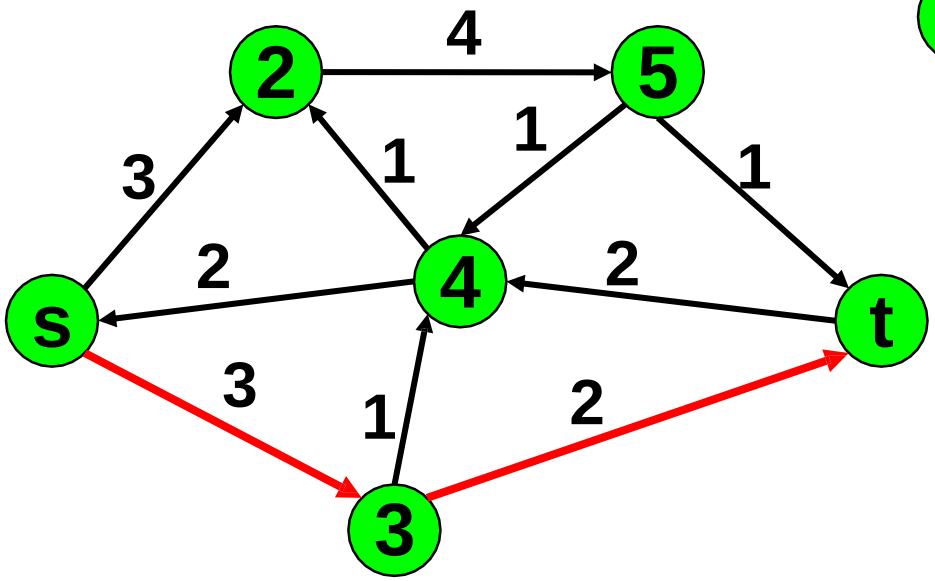


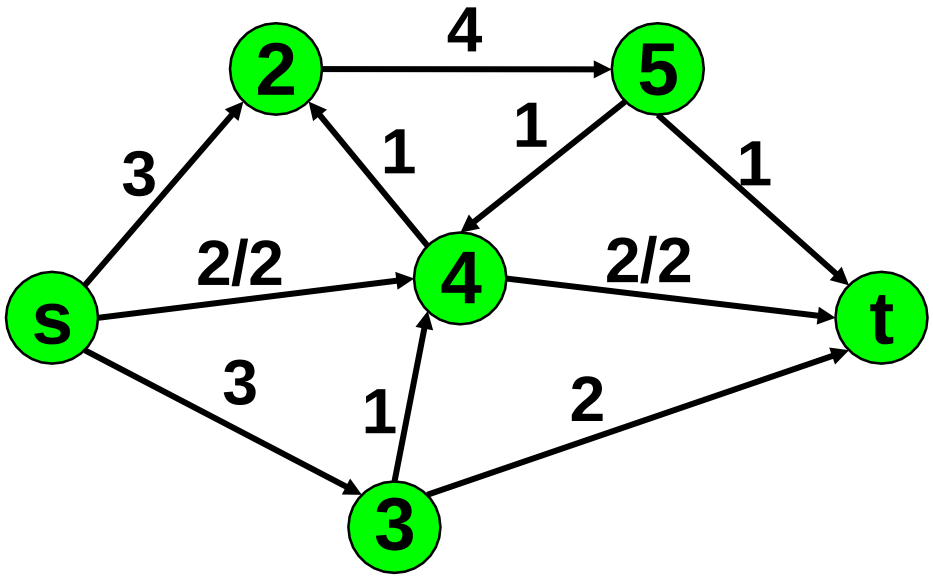


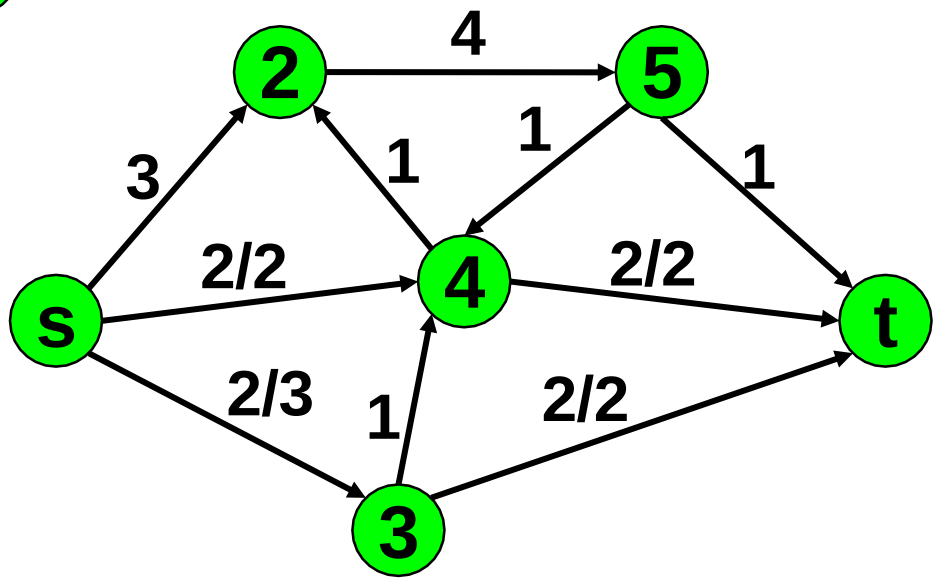


Flow = 2

Contd…

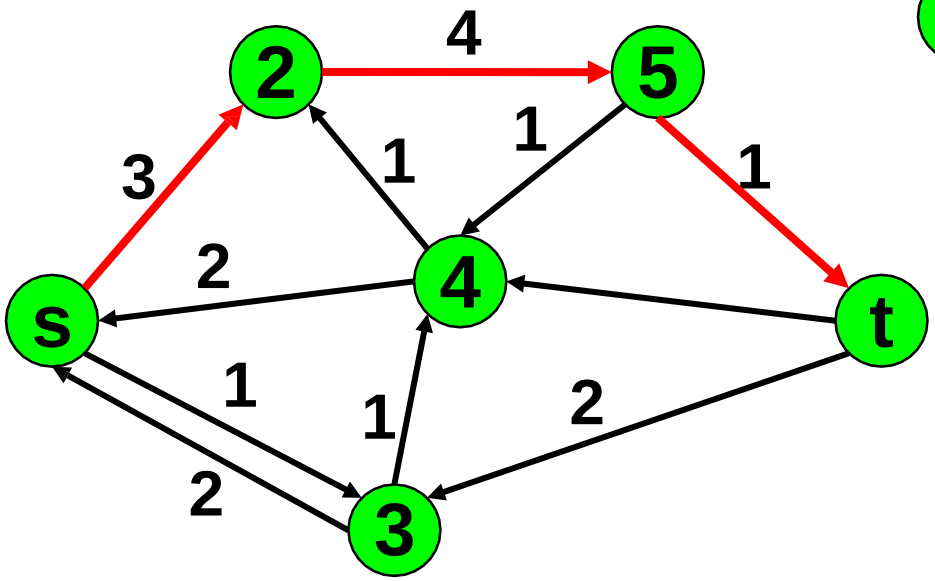


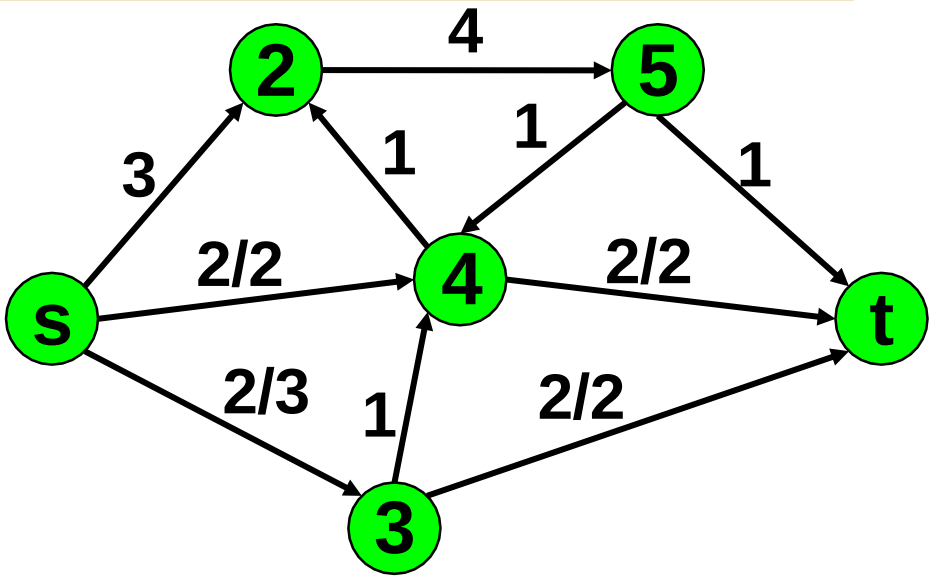


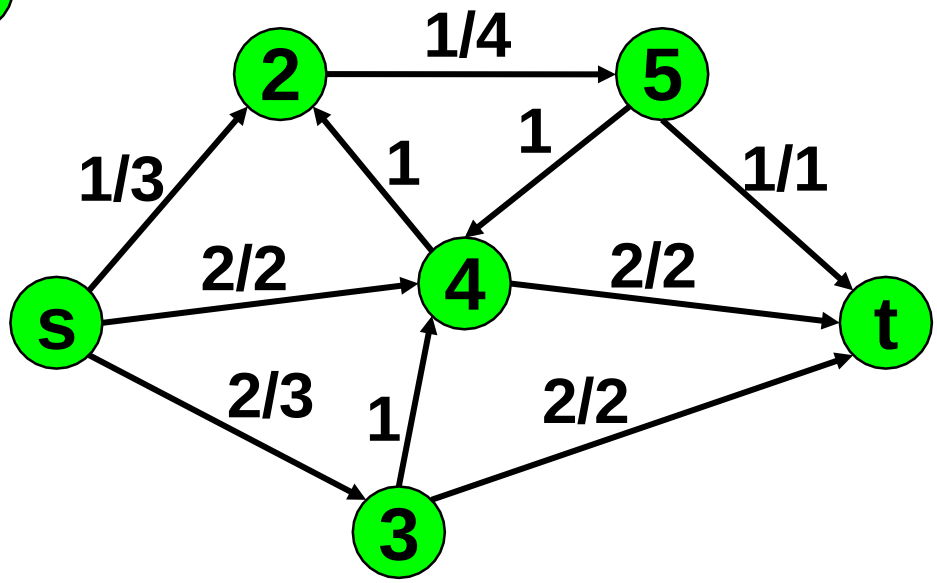


Flow = 2 + 2

Contd…

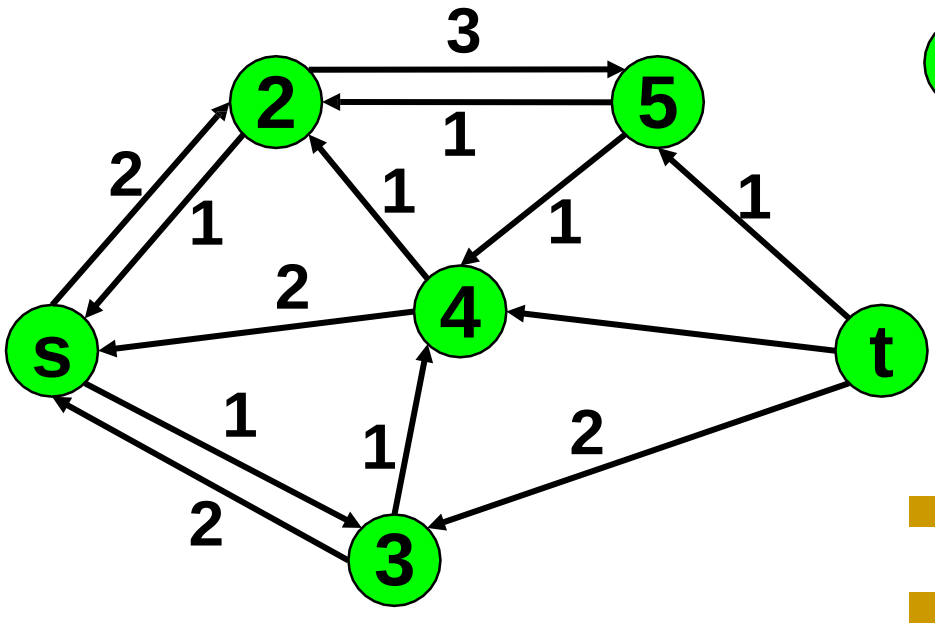


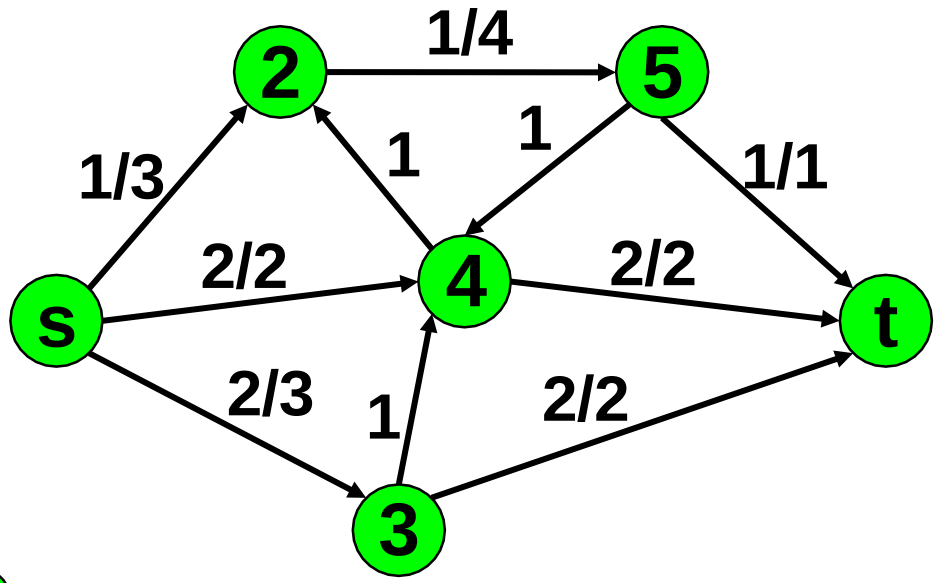




Flow = 2 + 2 + 1

Contd…





- $|f| = 5$.
- $c(\{s,2,3,4,5\},\{t\}) = 5$.

Flow = 2 + 2 + 1 = 5

Analysis

- If f^* is the maximum flow, then algorithm executes while loop of lines 3–8 at most $|f^*|$ times, assuming flow increases by at least one unit in each iteration.
- The time to find a path in a residual network is O(E).
- The time to update capacity and flow values is O(1).
- Each iteration of while loop thus takes O(E) time, as does the initialization in lines 1–2.
- Thus, the total running time of the FORD-FULKERSON algorithm is $O(E|f^*|)$.

Edmonds-Karp algorithm

- It's similar to a Ford-Fulkerson method.
- It chooses the augmenting path as a shortest path from s to t in the residual network.
- Edmonds-Karp algorithm runs in $O(VE^2)$ time.

Applications

- Application area of max flow min cut is very vast.

- Interested students may refer the document available at http://www.cs.princeton.edu/~wayne/cs423/lectures/max-flow-applications