Network Flow

CS31005: Algorithms-II

Autumn 2020

IIT Kharagpur

Network Flow

- Models the flow of items through a network
- Example
  - Transporting goods through the road/rail/air network
  - Flow of fluids (oil, water,..) through pumping stations and pipelines
  - Packet transfer in computer networks
  - Many others in a variety of fields…
- Has many different versions with wide practical applicability
- We will study the maximum flow problem

The Maximum Flow Problem

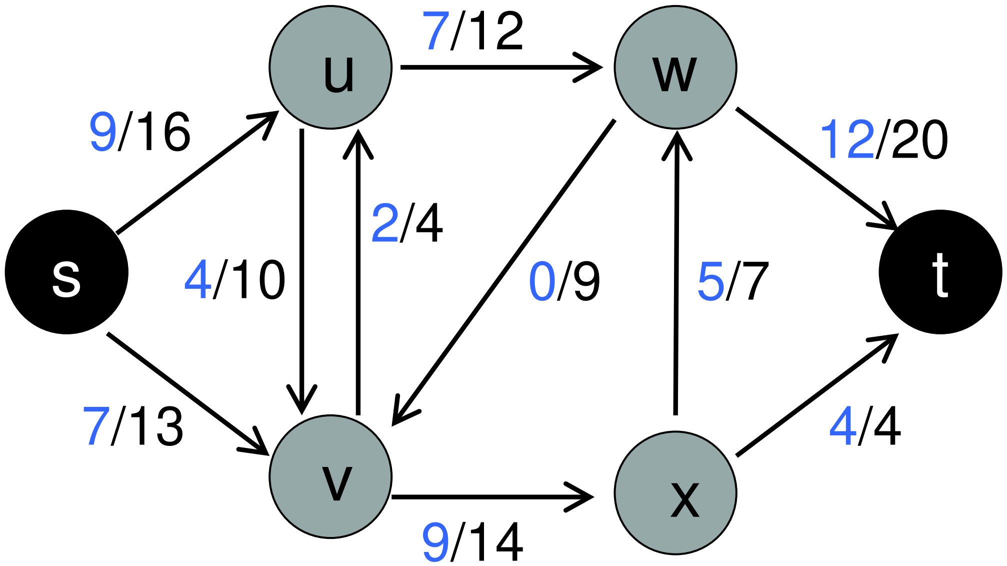
- Input: a directed graph $G = (V, E)$ with
  - Each edge $(u,v) \in E$ has a capacity $c(u, v) \geq 0$
  - Two distinguished vertices s (source) and t (sink)
- Output: Flow in G, a function $f: E \to \mathbb{R}$ such that
  - $0 \leq f(u,v) \leq c(u,v)$ for each (u,v) in E (capacity constraint)
  - $\sum_{u \in V, (u, v) \in E} f(u,v) = \sum_{w \in V, (v, w) \in E} f(v, w)$ for all v in $V \setminus \{s, t\}$ (flow conservation constraint)
- Easy to see that this means total flow leaving s must be the total flow entering t
- Flow satisfying the two constraints is called a feasible flow

- Value of the flow in the network

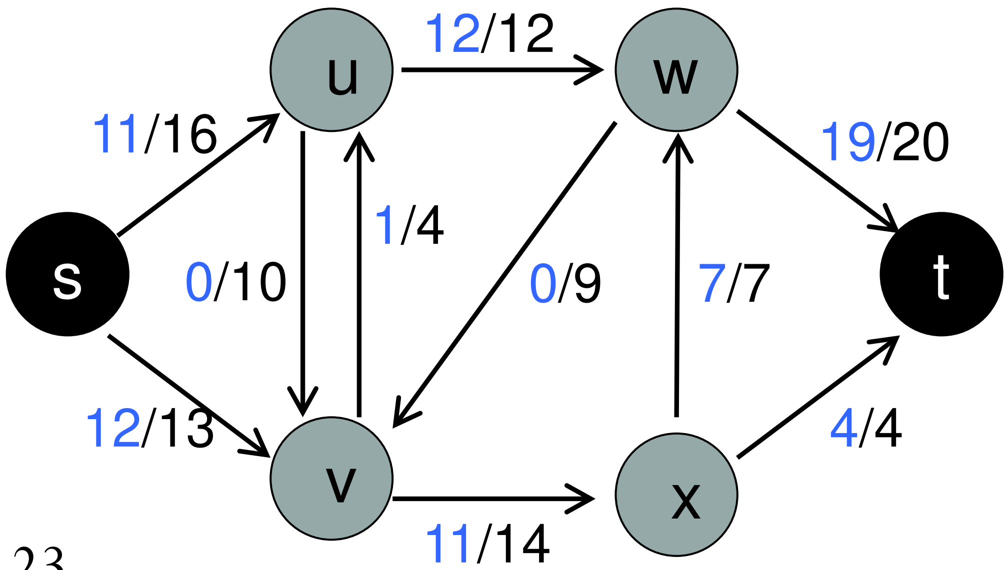
$$|f| = \sum_{u \in V, (s, u) \in E} f(s,u) = \sum_{u \in V, (u, t) \in E} f(u, t)$$

- Maximum Flow Problem: Find a feasible flow f such that the |f| is maximum among all possible feasible flows
- The assigned flow values on edges can model amount of goods in a transportation network, oil in a pipeline network, packets in a computer network along road/pipeline/link etc. to maximize the total amount of items moved from a source to a destination

Example



A feasible flow with |f| = 16



A maximum flow with |f| = 23

Algorithms for Maximum Flow

- Follows two broad approaches
  - The Ford-Fulkerson Method
    - Originally proposed by Ford and Fulkerson in 1956
    - Actually defines a method, the original paper did not specify any particular implementation of some steps
    - Many algorithms proposed later following the method, with specific implementations of steps
  - Preflow-Push Method
    - Presented by Andrew Goldberg and Robert Tarjan in 1986 (ACM STOC, later detailed journal version in JACM in 1988)
    - A totally different approach from the Ford-Fulkerson methods

Ford-Fulkerson Method

- Before starting the algorithm, we first give an equivalent modelling of the problem by
  - Extending the domain of capacity c and flow f to $V \times V$ (instead of keeping to E only)
  - Modifying the constraints appropriately

- Capacity c: $V \times V \rightarrow R$ such that $c(u,v) = 0$ if (u,v) not in E
- Flow $f : V \times V \rightarrow R$ satisfying:
  - Capacity constraint: For all $u, v \in V$, $f(u,v) \leq c(u,v)$
  - Skew symmetry: For all $u, v \in V$, $f(u,v) = -f(v,u)$
  - Flow conservation: For all $u \in V - \{s, t\}$, $\sum_{v \in V} f(u,v) = 0$

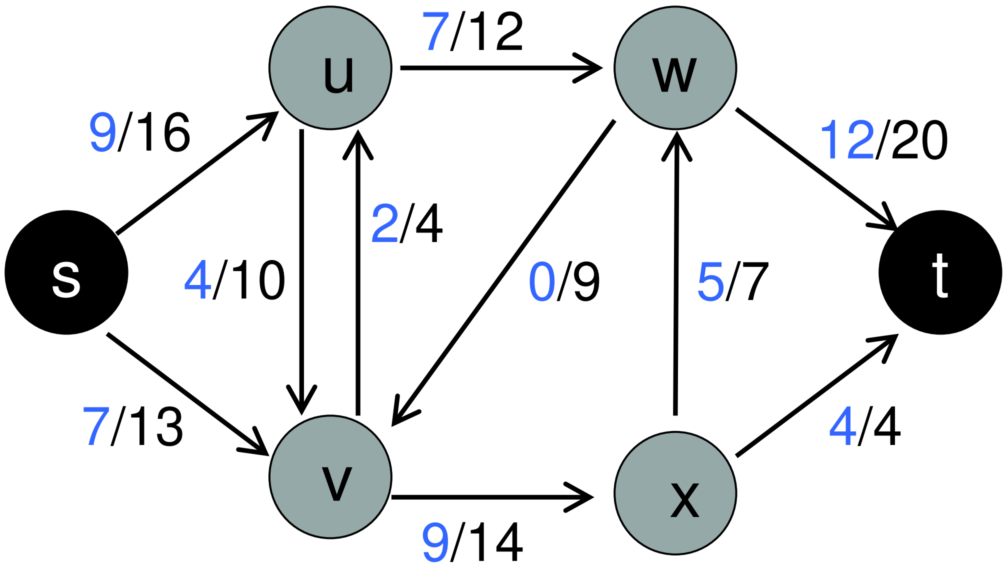
The value of the flow f is defined to be $|f| = \sum_{v \in V} f(s,v)$

The maximum flow problem is to find the flow with maximum value (same as before)

- What does this mean? Consider different possibilities for a pair (u,v)
  - None of the edges (u,v) or (v,u) exist
    - So $c(u,v) = c(v,u) = 0$
    - So $f(u,v) = f(v,u)$ must be 0 as otherwise capacity constraint and skew symmetry are violated
  - Only one of the edges exist (say (u,v))
    - So $c(u,v) \geq 0$ and $c(v,u) = 0$
    - If $f(u,v) = 0$, then $f(v,u) = 0$ (skew symmetry)
    - If $f(u,v) > 0$, then $f(v,u) < 0$ (skew symmetry)
    - If $f(u,v) < 0$ then $f(v,u) > 0$ (skew symmetry), But this violates capacity constraint for (v,u). So $f(u,v)$ cannot be negative

- Both the edges (u,v) and (v,u) exist
  - So $c(u,v) \geq 0$ and $c(v,u) \geq 0$
  - So seems like both f(u,v) and f(v,u) can be positive (by capacity constraint)
  - But that would break skew symmetry, so both cannot be positive
  - The way to think about it is to consider the "net flow"
    - If you ship 20 units from A to B and ship 5 units from B to A, the net flow into B is not 20, it is $20 - 5 = 15$. Similarly the net flow into A is not 5, but $(-20) + 5 = -15$, indicating it is actually an outflow
- In general, for any two vertices u, v, if $f(u,v) > 0$, then $f(v,u)$ must be < 0 (skew symmetry)

Example



$f(s, u) = 9, \quad f(u, s) = -9$

$f(s, v) = 7, \quad f(v, s) = -7$

$f(u,w) = 7, \quad f(w,u) = -7$

$f(u,v) = 4 - 2 = 2$

$f(v, u) = 2 - 4 = -2$

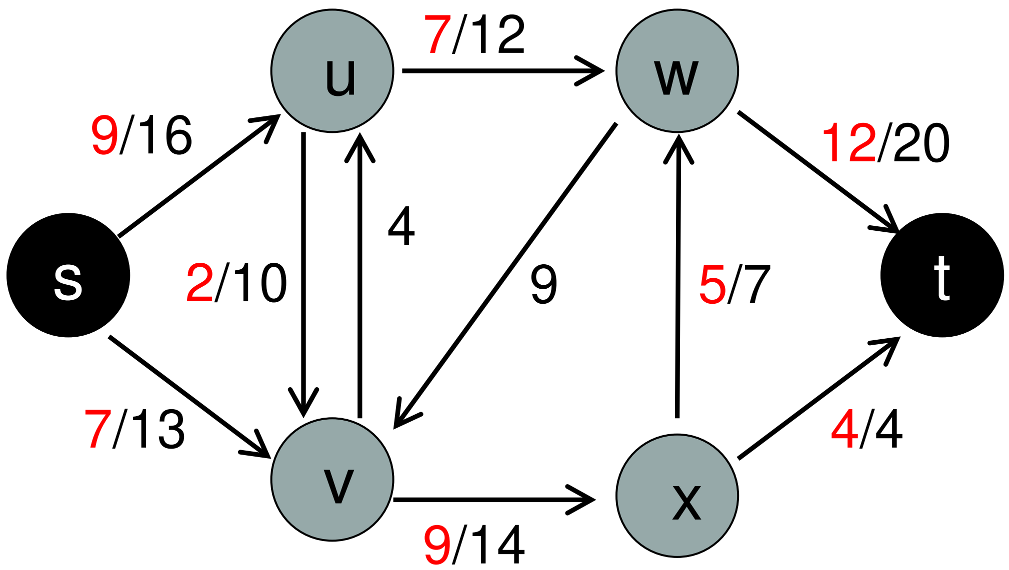
$f(v,x) = 9, \quad f(x,v) = -9$

$f(w,v) = 0, \quad f(v, w) = 0$

$f(u, x) = 0, \quad f(x, u) = 0$

similar for other pairs in $V \times V$

- With our new definition of flow, we will represent the graph to show f values on edges in red (not necessarily actual shipments)
- Also, we will only show positive f values on the edges of the graph
  - So for edges (v,u) and (w,v), we do not show the f values because f(v,u) = – 2 and f(w,v) = 0



- Did we lose anything from the earlier model?
  - For edges (u,v) and (v,u) (i.e for the case when edges exist in both direction between a pair of vertices), we are now representing only the net flow, not how exactly the net flow is achieved
    - For example, the net flow of 2 from u to v could have been achieved in different ways like "ship 6 units from u to v and 4 units from v to u", "ship 2 units from u to v and 0 units from v to u",…..
- So this model is not exactly equivalent to the model we had,
  - For the earlier model, actual shipments are the flow f
  - but ok as in practice as no need to ship in both directions
- If you have edge only in one direction, f will show the actual shipment

Residual Network

- Let f be a flow in a flow network $G = (V, E)$ with source s and sink t.
- Residual capacity of (u,v) = amount of additional flow that can be pushed from a node u to node v before exceeding the capacity c(u,v)

$$c_f(u, v) = c(u, v) – f(u, v)$$

- The residual graph of G induced by f is $G_f = (V, E_f)$, where

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

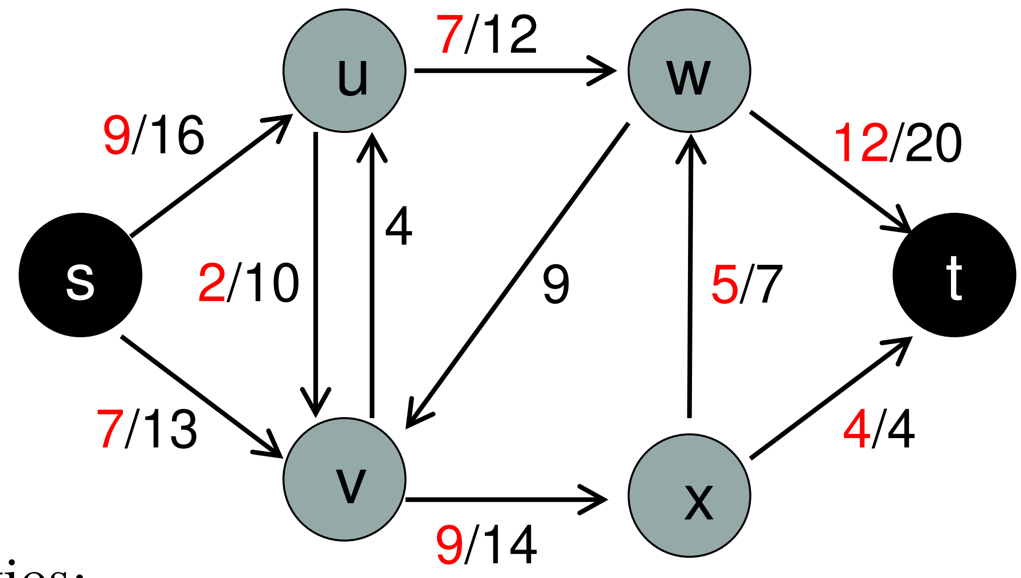
Edges of the residual graph are called residual edges, with capacity c_f

- Augmenting path: a simple path from source s to sink t in the residual graph G_f
- Residual capacity of an augmenting path p

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

$c_f(p)$ gives the maximum amount by which the flow on each edge in the path p can be increased

Example



- Residual capacities:

  $c_f(s,u) = 16 - 9 = 7$, $\quad c_f(u,s) = 0 - (-9) = 9$

  $c_f(s,v) = 13 - 7 = 6$, $\quad c_f(v,s) = 0 - (-7) = 7$

  $c_f(u,v) = 10 - 2 = 8$, $\quad c_f(v,u) = 4 - (-2) = 6$

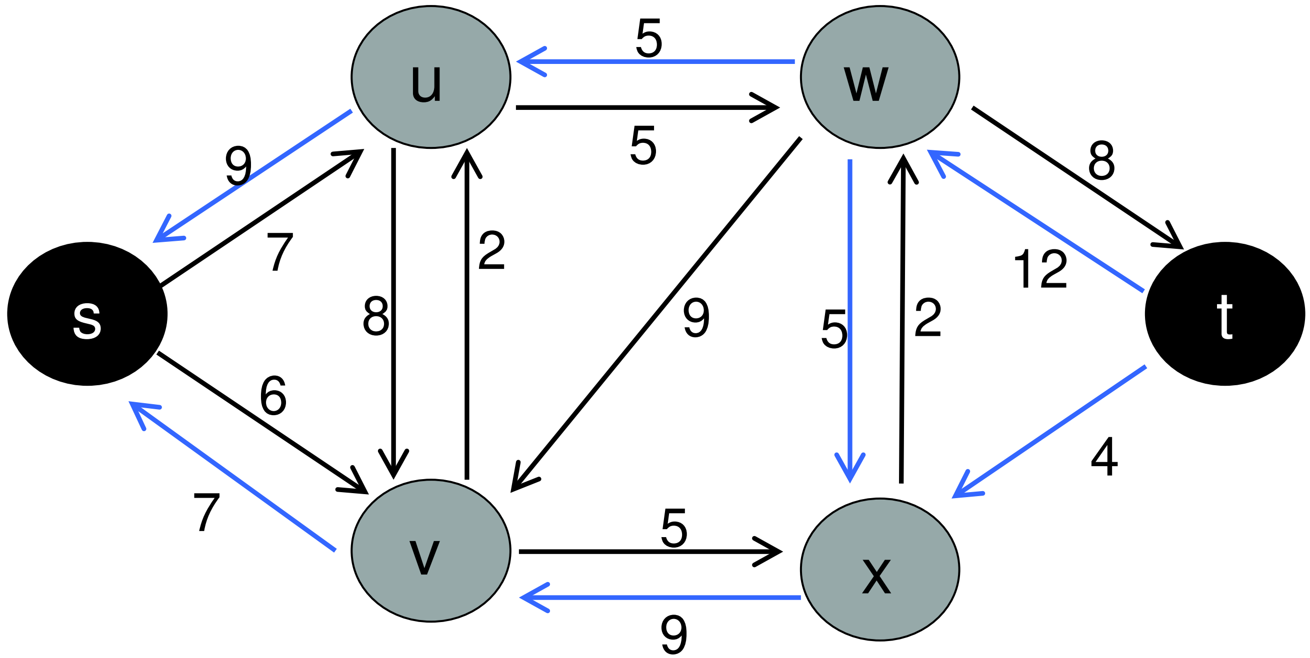
  $c_f(u,w) = 12 - 7 = 5$, $\quad c_f(w,u) = 0 - (-7) = 5$

  $c_f(w,v) = 9 - 0 = 9$, $\quad c_f(v,w) = 0 - 0 = 0$

  $c_f(x,t) = 4 - 4 = 0$, $\quad c_f(t,x) = 0 - 0 = 0$

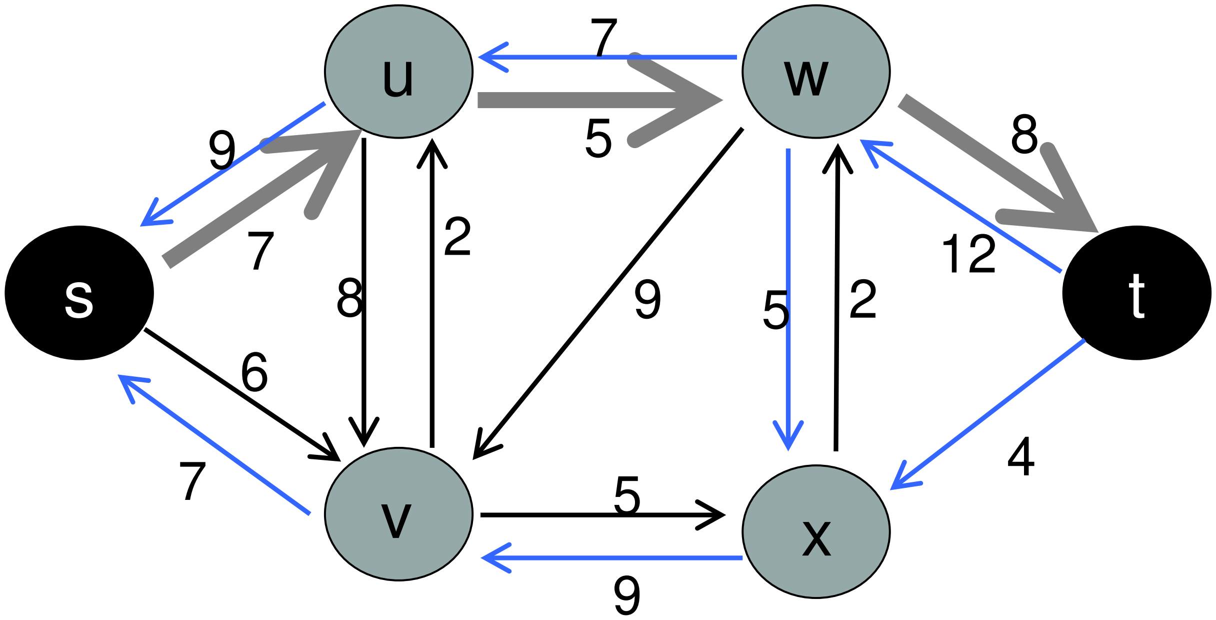
  and so on for the other pairs
- For any a, b in V, $c_f(a,b) = 0$ if neither (a,b) nor (b,a) is an edge (as c and f are both 0 for such pairs), so we do not look at them

- Residual Graph (edges with 0 residual capacity are never shown)



- Note that residual graph may have edges where G did not (shown in color blue)
- It also may NOT have edges where G has one, ex. (x,t)
    - The residual capacity of the edge is 0
    - Such edges are called saturated

- Augmenting Path – path from s to t



- One path shown in bold grey, <s,u,w,t> with residual capacity = min(7, 5, 8) = 5
  - We can increase ("augment") the flow on each edge of the path by 5 to get a new feasible flow with higher value

Ford-Fulkerson Algorithm

1. Start with a feasible flow f (usually f=0 for all (u,v))
2. Create the residual graph G_f
3. Find an augmenting path p in G_f
4. Augment the flow in G
5. Repeat 2-4 until there is no augmenting path

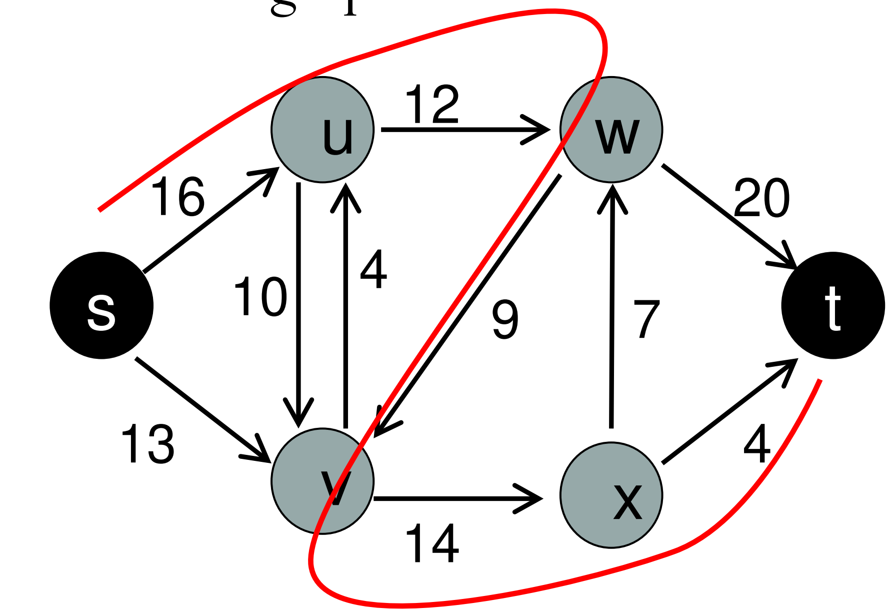
```
FORD-FULKERSON-METHOD(G, s, t)
initialize flow f to 0
while there exists an augmenting path p
      do augment flow f along p
return f
```

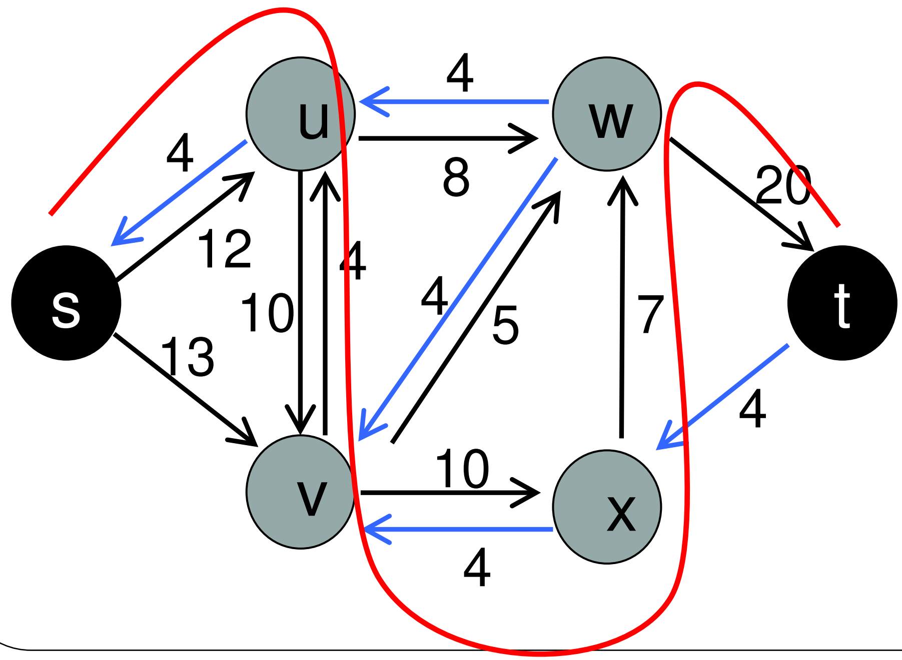
- Augmenting the flow along path p with residual capacity c

```
FORD-FULKERSON(G, s, t)
for each edge (u, v) ∈ E[G]
      do f[u, v] ← 0
         f[v, u] ← 0
while there exists a path p from s to t in the residual network G_f
      do c_f(p) ← min {c_f(u, v) : (u, v) is in p}
         for each edge (u, v) in p
               do f[u, v] ← f[u, v] + c_f(p)
                  f[v, u] ← −f[u, v]
```

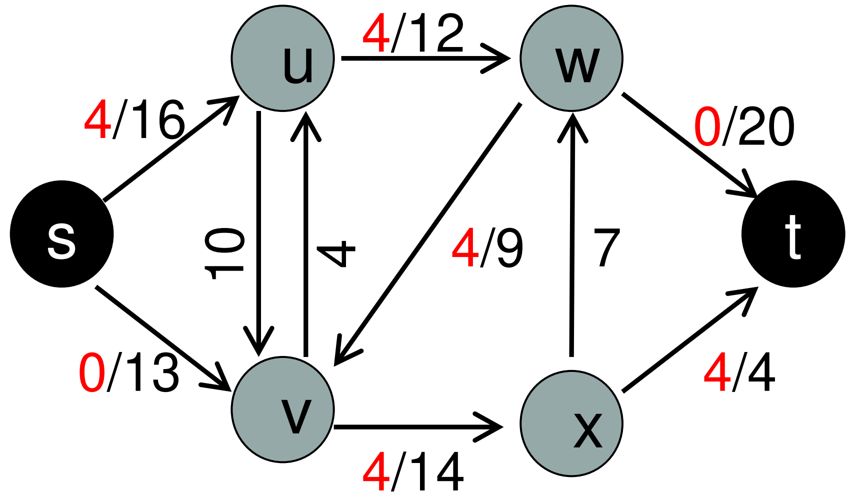
  - Note that either (u,v) or (v,u) must be an edge in G (or (u.v) cannot be in G_f)
  - If (u,v) is an edge, this increases f (u,v)
  - If (u,v) is not an edge, this actually decreases f(v,u)

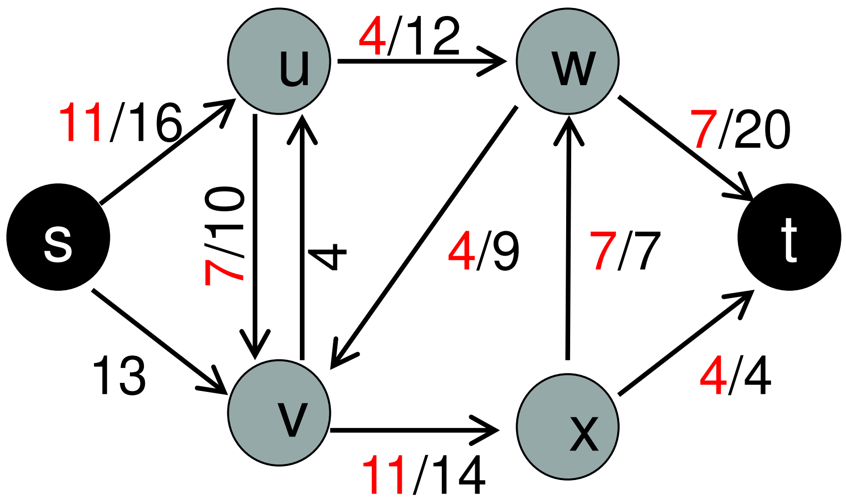
Residual graph

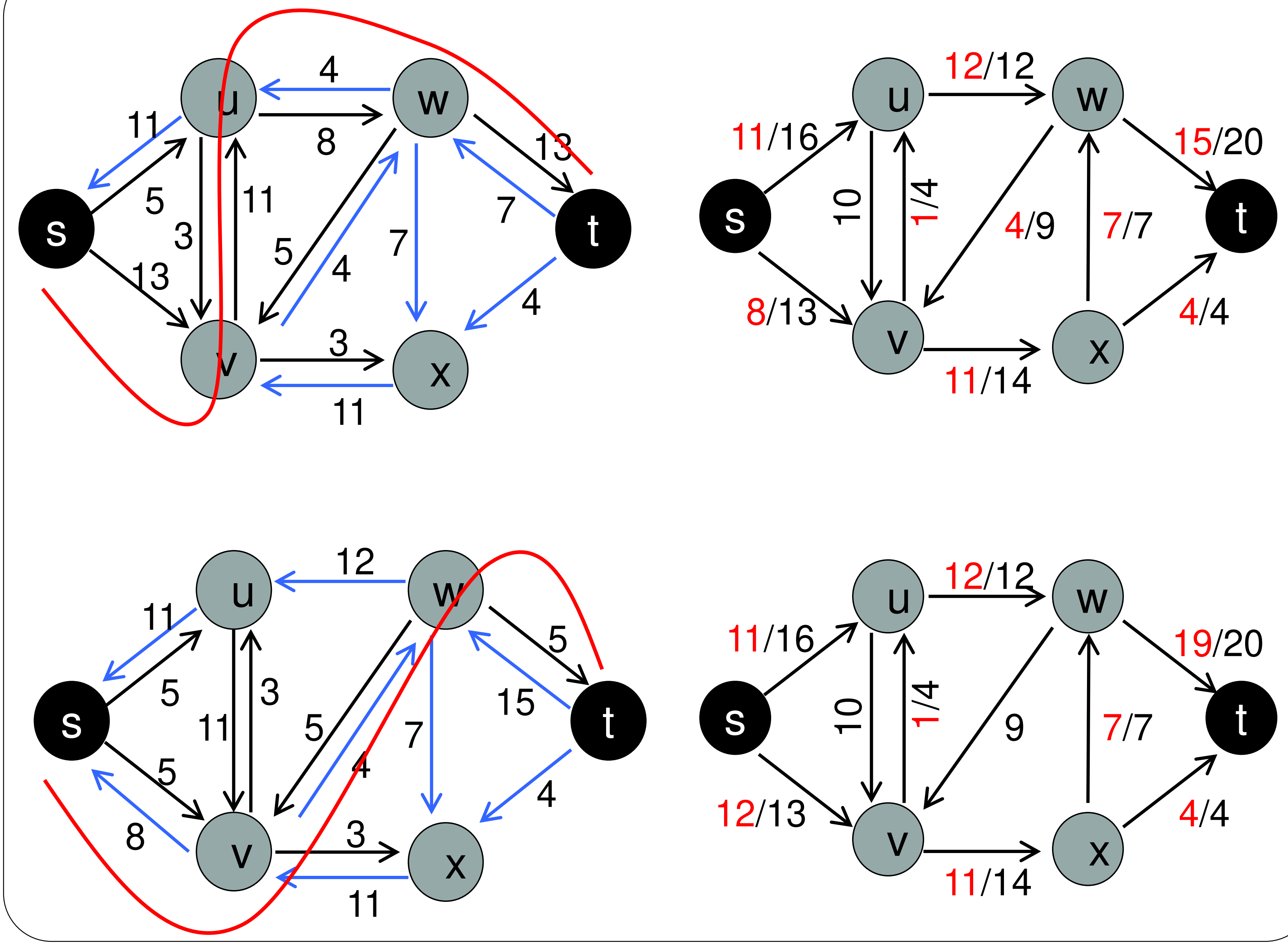




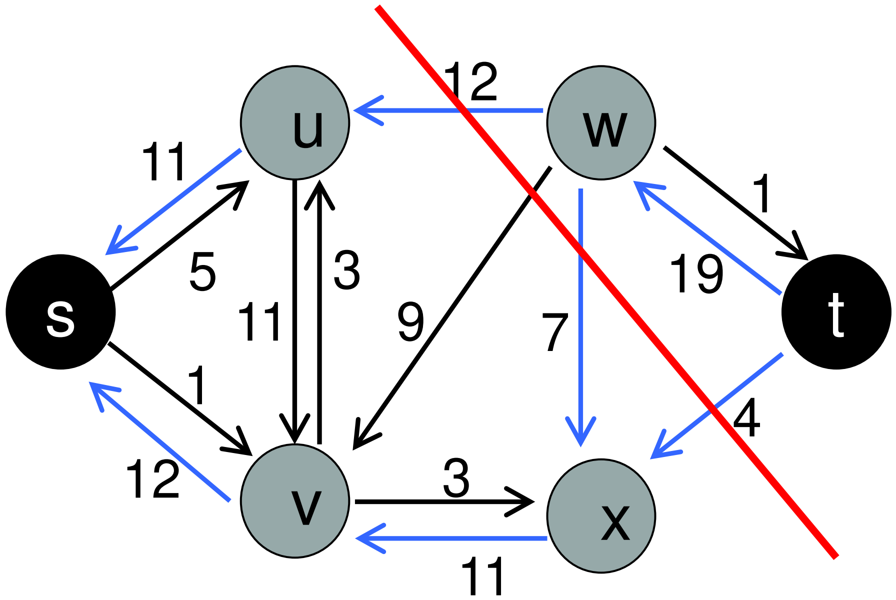
Flow Assignment







u
w
s
t
v
x
4
8
11
5
3
13
11
5
4
7
7
13
4
3
11
12/12
11/16
15/20
10
1/4
4/9
7/7
8/13
4/4
11/14
12
11
5
11
3
5
4
7
15
5
4
8
3
11
12/12
11/16
19/20
10
1/4
9
7/7
12/13
4/4
11/14

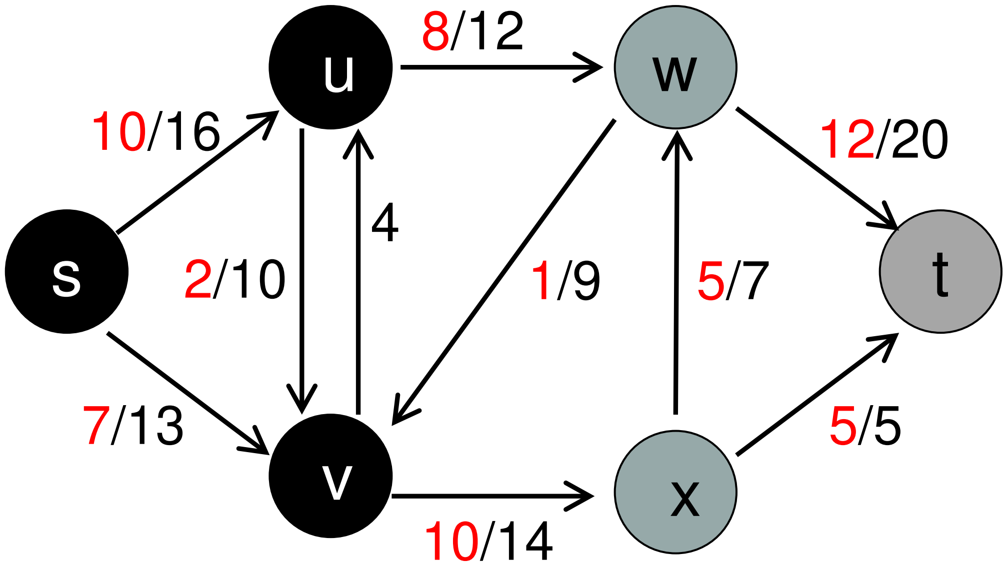


No augmenting path in the residual graph, so stop

Maximum Flow $|f| = 23$

Proof of Correctness

- We first need some definitions
  - A cut (S, T) of a flow network G = (V, E) is a partition of V into S and T = V – S, such that $s \in S$ and $t \in T$
  - If f is a flow then the net flow across the cut (S, T) , f(S, T), is the sum of the flows (f) of all pairs (u,v) with u in S and v in T
  - The capacity of the cut (S, T), c(S, T), is the sum of the capacities of all edges (u,v) with u in S and V in T
    - Of course, $f(S, T) \leq c(S, T)$
  - A minimum cut of a network is a cut whose capacity is minimum over all possible cuts of the network



- Consider the cut (S={s, u, v}, T={w, x, t})
- f(S,T) = f(u,w) + f(v,w) + f(v,x)

  = 8 + (-1) + 10 = 17
- c(S,T) = c(u,w) + c(v,x) = 12 + 14 = 26

Lemma 1: Let f be a flow in a network G with source s and sink t, and let (S, T) be a cut of G. Then the net flow across (S, T) is $f(S, T) = |f|$.

Proof:

$$
\begin{aligned}
f(S, T) &= f(S,V) - f(S,S) \\
&= f(S,V) \\
&= f(s, V) + f(S - s, V) \\
&= f(s,V) \\
&= |f|
\end{aligned}
$$

Lemma 1 implies that the net flow across *any* cut is the same (= value of flow).

Corollary 2: The value of any flow f in a flow network G is bounded from above by the capacity of any cut of G, and hence by the capacity of the minimum cut.

Theorem 3 (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. $|f|$ = capacity of the minimum cut

Proof:

1 implies 2 is obvious, as otherwise $|f|$ can be increased by increasing the flow along the augmenting path

2 implies 3:

Suppose that G_f has no augmenting paths. Let

$S = \{v \in V$: there exists a path from s to v in $G_f\}$ and

$T = V - S$.

Then (S,T) is a cut as s is in S and t is not in S as there is no path from s to t in G_f.

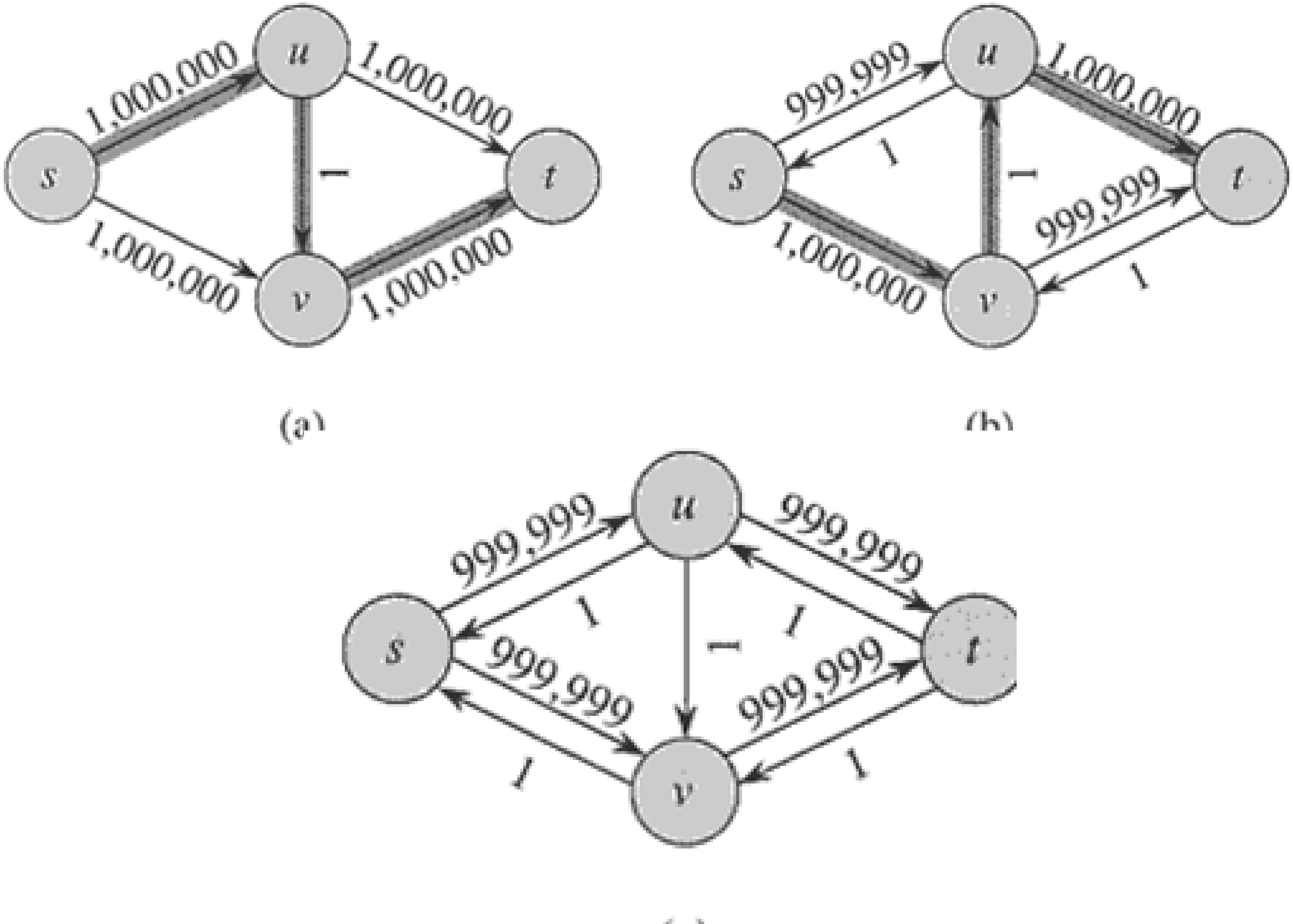
For any $u \in S$ and $v \in T$, we have $f(u,v) = c(u,v)$ as otherwise (u,v) is in G_f, which would mean v is in S, which is a contradiction. Therefore, by Lemma 1, $|f| = f(S,T) = c(S,T)$

3 implies 1: By corollary 2, $|f| \leq c(S,T)$ for all cuts (S,T). Then, $|f| = c(S,T)$ implies $|f|$ is a maximum flow.

Time Complexity

- Original Ford-Fulkerson algorithm does not specify how to find an augmenting path
  - Can find in any order
- Assume all capacities are integer
- Let $f^* =$ maximum flow
- Lines 1-3 (Initialization) takes $O(|E|)$ time
- No. of times the while loop (no. of times an augmenting path is found) is executed is bounded above by $|f^*|$
  - As $|f|$ increases by at least 1 in each augmentation
- Each iteration of the while loop takes $O(|E|)$ time
- So worst case time complexity $O(|E||f^*|)$
  - This is not polynomial, it is pseudo-polynomial

- This bound is tight



(c)