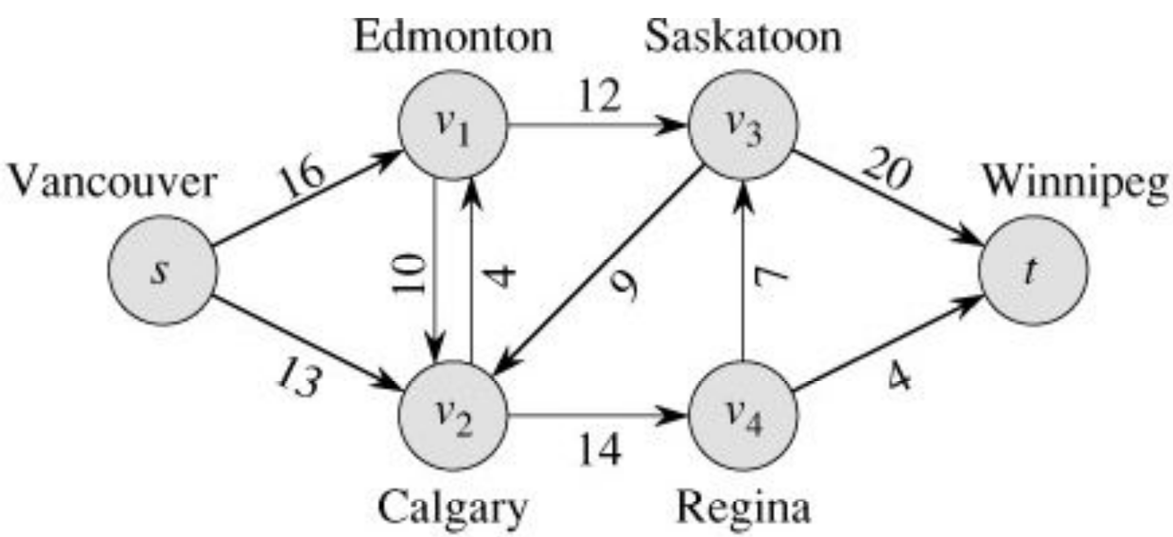
NETWORK FLOW

Network Flow

- Instance:
  - A Network is a directed graph G
  - Edges represent pipes that carry flow
  - Each edge $\langle u,v \rangle$ has a maximum capacity $c_{\langle u,v \rangle}$
  - A source node s in which flow arrives
  - A sink node t out which flow leaves

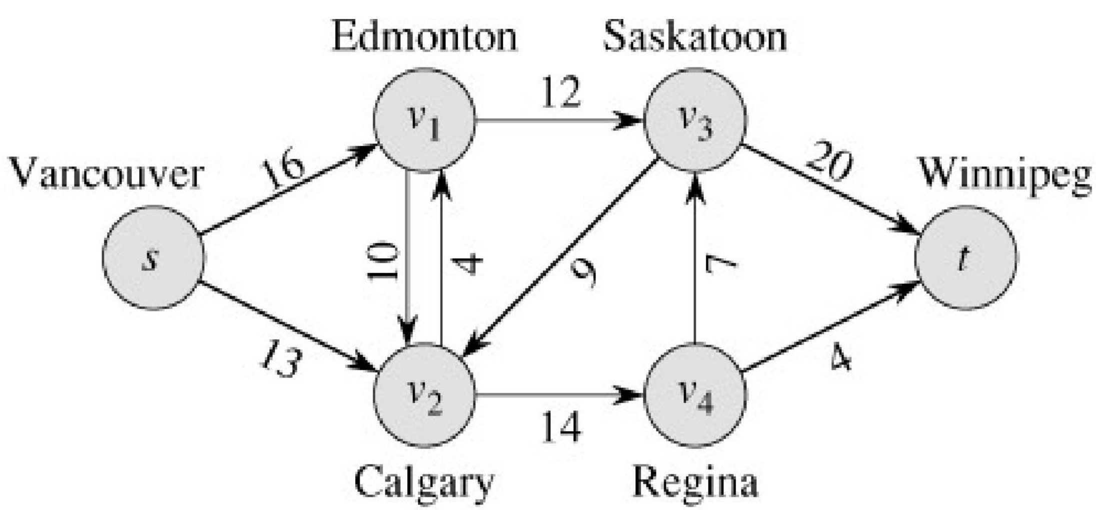
The Problem

- Use a graph to model material that flows through conduits.
- Each edge represents one conduit, and has a capacity, which is an upper bound on the flow rate = units/time.
- Can think of edges as pipes of different sizes.
- Want to compute max rate that we can ship material from a designated source to a designated sink.



What is Network Flow?

- Each edge (u,v) has a nonnegative capacity c(u,v).
- If (u,v) is not in E, assume c(u,v)=0.
- We have a source s, and a sink t.
- Assume that every vertex v in V is on some path from s to t.
- $c(s,v_1)=16$; $c(v_1,s)=0$; $c(v_2,v_3)=0$



Flow in a Flow Network

– A <u>flow</u> in the network is an integer-valued function f defined on the edges of G satisfying $0 \leq f(i,j) \leq c(i,j)$ for every edge (i,j) in E.

What is a Flow in a Network?

- For each edge (u,v), the flow f(u,v) is a real-valued function that must satisfy 3 conditions:

Capacity constraint: $\forall u,v \in V,\ f(u,v) \leq c(u,v)$

Skew symmetry: $\forall u,v \in V,\ f(u,v) = -f(v,u)$

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

- Note that skew symmetry condition implies that f(u,u)=0.

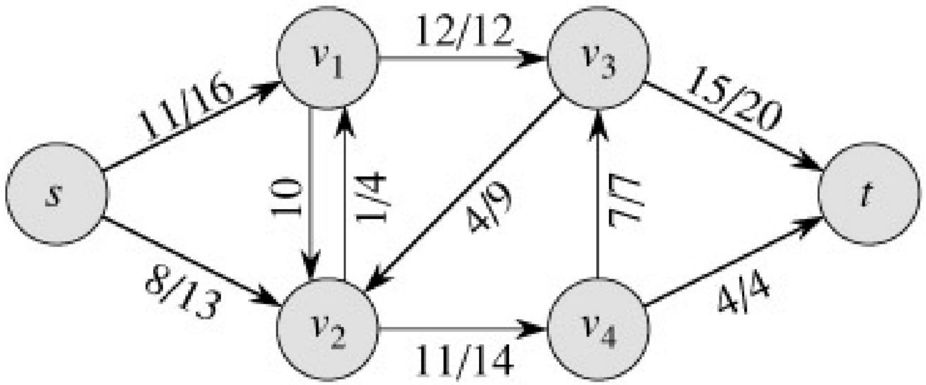
The Value of a flow

- The value of a flow is given by

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

- This is the total flow leaving s = the total flow arriving in t.

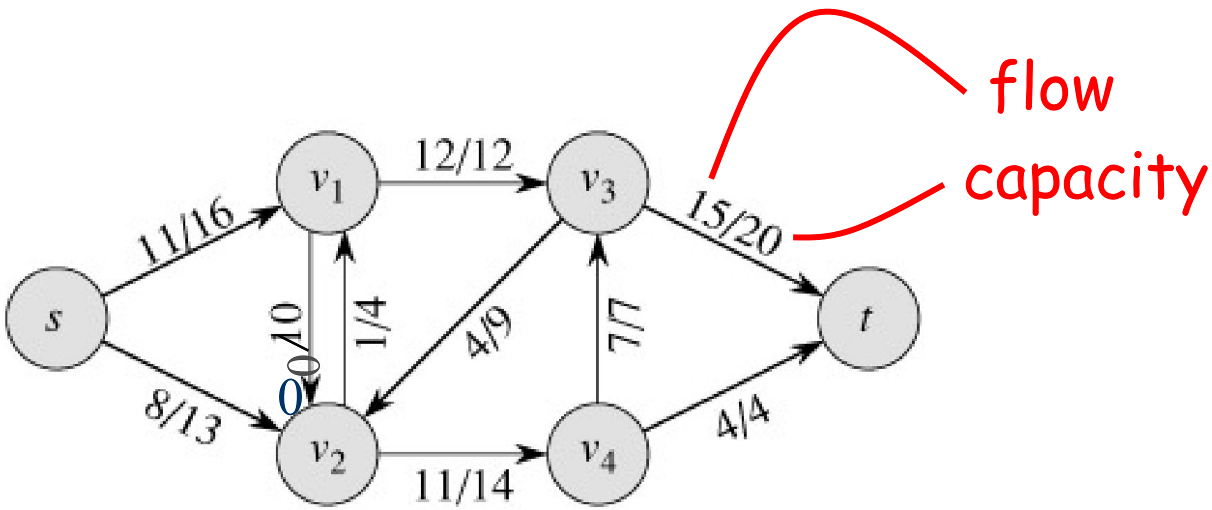
Example:



$$|f| = f(s, v_1) + f(s, v_2) + f(s, v_3) + f(s, v_4) + f(s, t) =$$
$$11 + 8 + 0 + 0 + 0 = 19$$

$$|f| = f(s, t) + f(v_1, t) + f(v_2, t) + f(v_3, t) + f(v_4, t) =$$
$$0 + 0 + 0 + 15 + 4 = 19$$

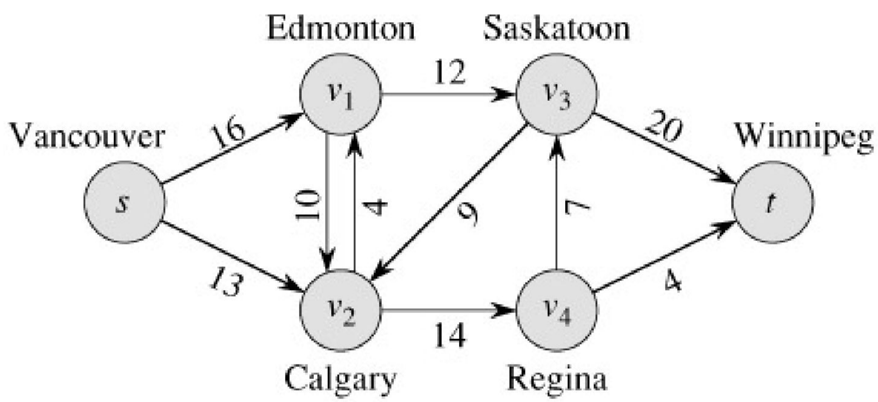
Example of a Flow



- $f(v_2, v_1) = 1$, $c(v_2, v_1) = 4$.
- $f(v_1, v_2) = -1$, $c(v_1, v_2) = 10$.
- f(v3, s) + f(v3, v1) + f(v3, v2) + f(v3, v4) + f(v3, t) =
- 0 + (-12) + 4 + (-7) + 15 = 0

A flow in a network

- We assume that there is only flow in one direction at a time.

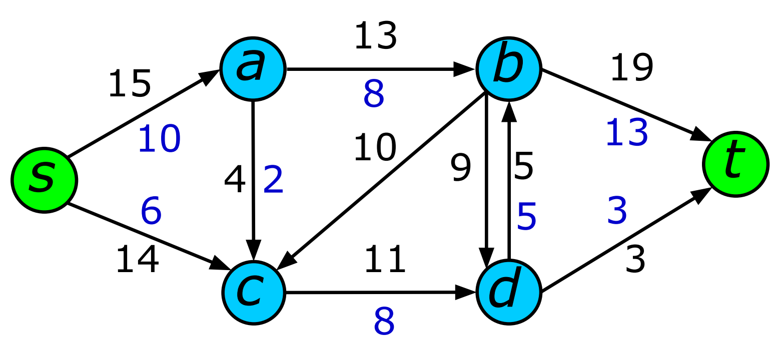


- Sending 7 trucks from Edmonton to Calgary and 3 trucks from Calgary to Edmonton has the same net effect as sending 4 trucks from Edmonton to Calgary.

Maximum flow

- *What do we want to maximize?*
  - ***Value*** of the flow *f*:

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



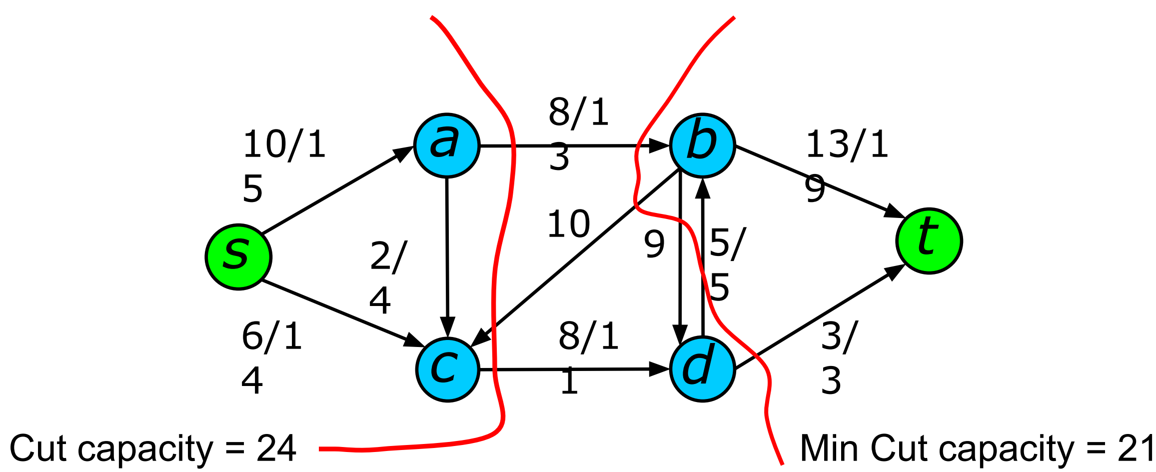
Some Lemmas:

- Prove that, $f(s,V)=f(V,t)$ *[pg-649]*
- [Lemma26.2] Prove that, $|f+f'|=|f|+|f'|$
- *Lemma: 26.3*
- *Lemma 26.4* $|f'| = |f| + |f_p| \geq |f|$
- *Lemma: 26.5* $|f| = f(S,T)$
- *Lemma: 26.6* $|f| \leq c(S,T)$

Cuts

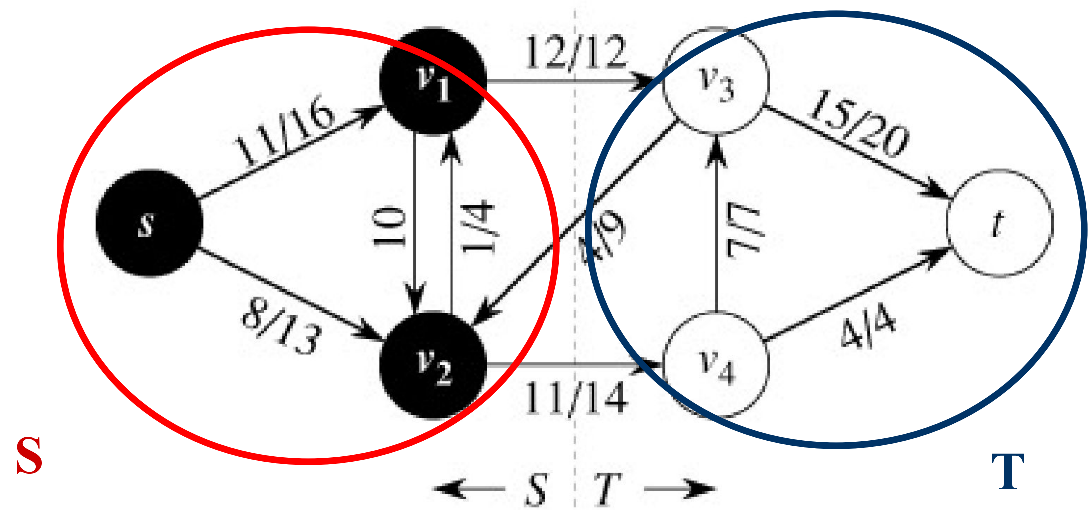
- A **cut** is a partition of V into S and T = V – S, such that $s \in$ S and $t \in$ T
- The **net flow** (f(S,T)) through the cut is the sum of flows f(u,v), where $s \in$ S and $t \in$ T
  - Includes negative flows back from T to S
- The **capacity** (c(S,T)) of the cut is the sum of capacities c(u,v), where $s \in$ S and $t \in$ T
  - The sum of positive capacities
- **Minimum cut** – a cut with the smallest capacity of all cuts.

  |f|= f(S,T) i.e. the value of a max flow is equal to the capacity of a min cut.



The Net Flow Through a Cut(S,T)

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

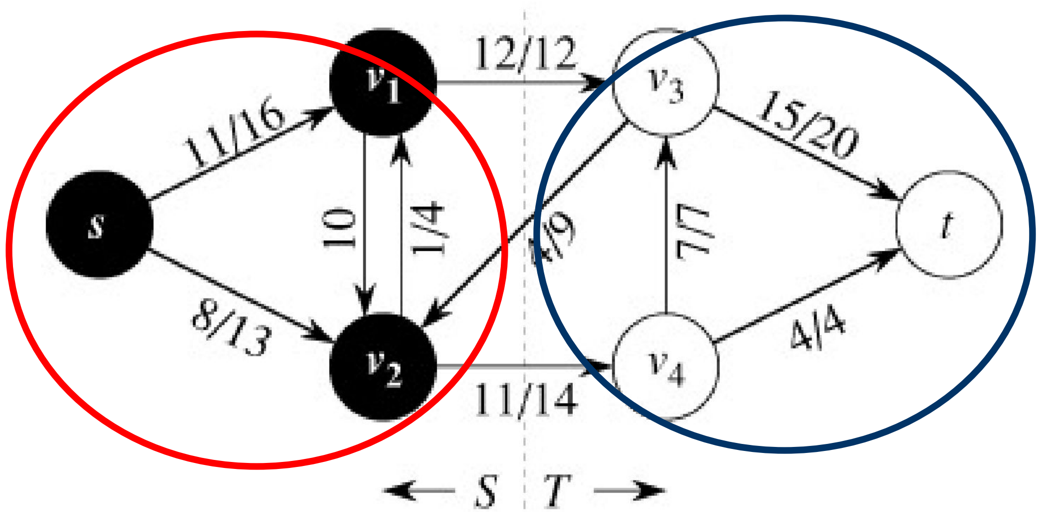


$$f(S,T) = 12 – 4 + 11 = 19$$

The value of any flow f in a flow network G is bounded from above by the capacity of any cut of G.

The Capacity of Cut(S,T)

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



c(S,T)= 12+ 14 = 26

Maxflow-Mincut Theorem

- *Max-flow min-cut theorem*:
  - If f is the flow in G, 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

The Ford-Fulkerson Method

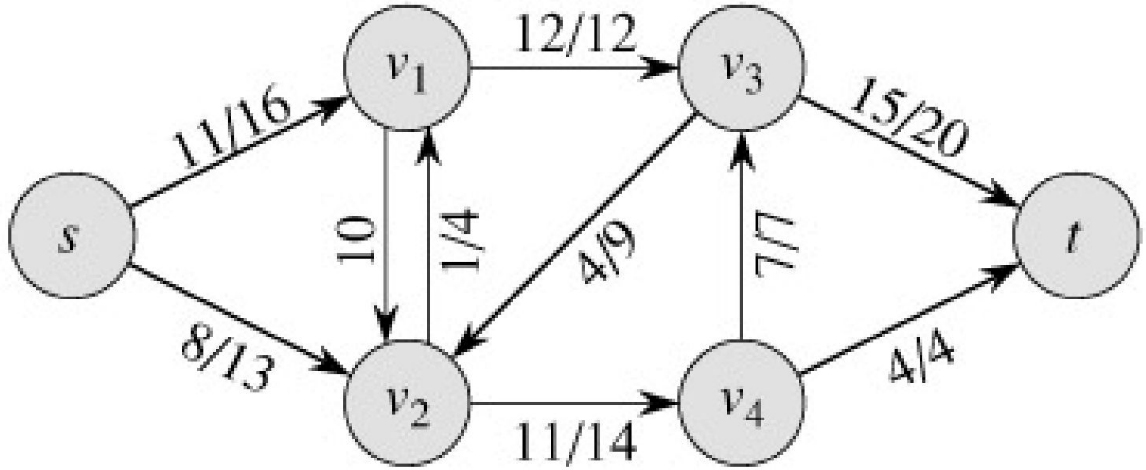
- Try to improve the flow, until we reach the maximum.
- The residual capacity of the network with a flow f is given by:

$$c_f(u,v) = c(u,v) - f(u,v)$$

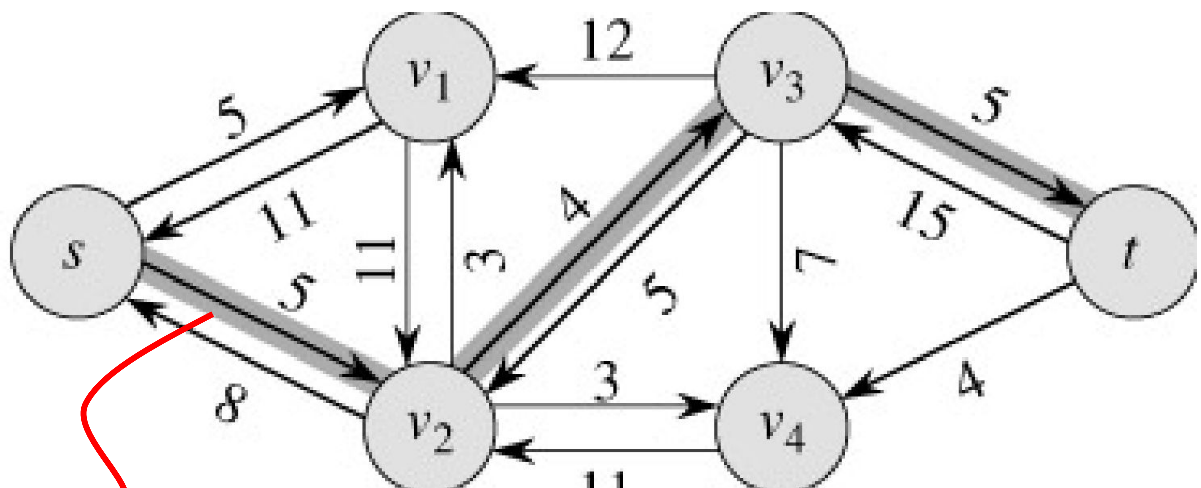
Always nonnegative (why?)

Example of residual capacities

Network:



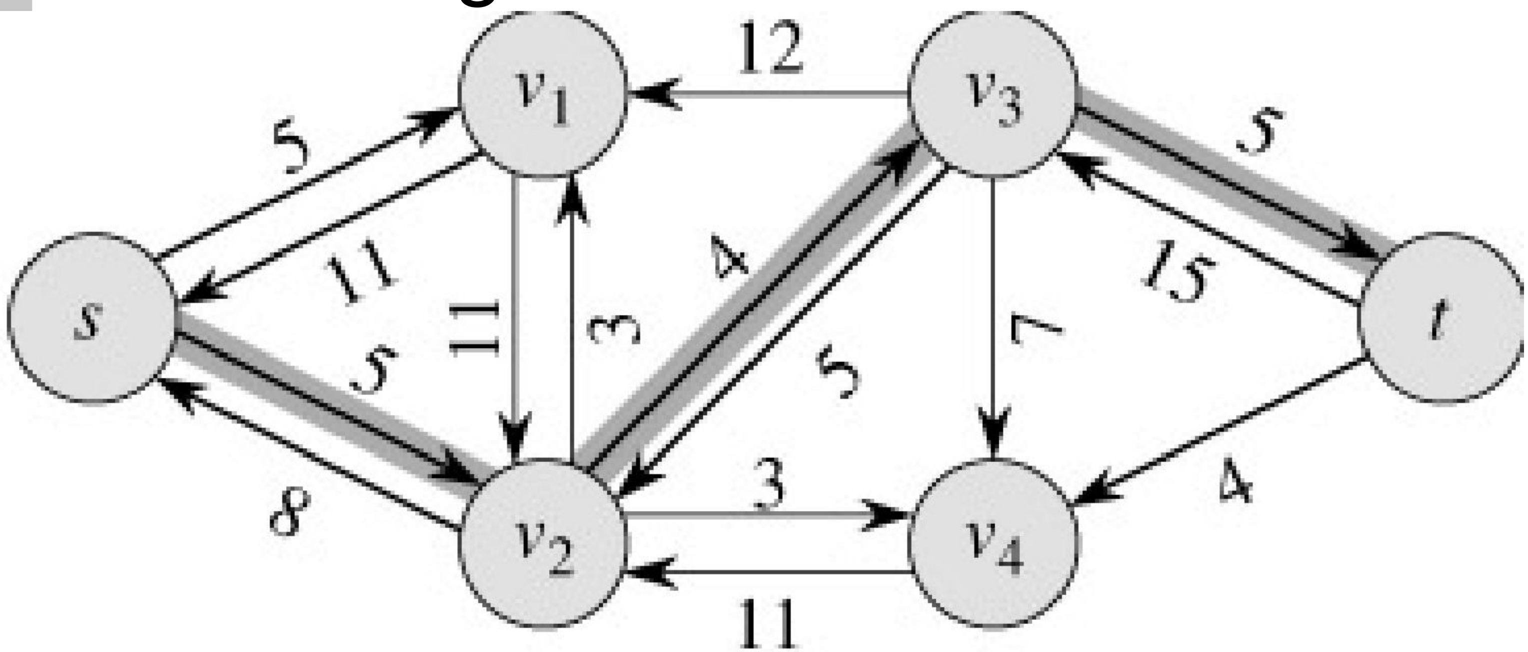
Residual Network:



Augmenting path

The residual network

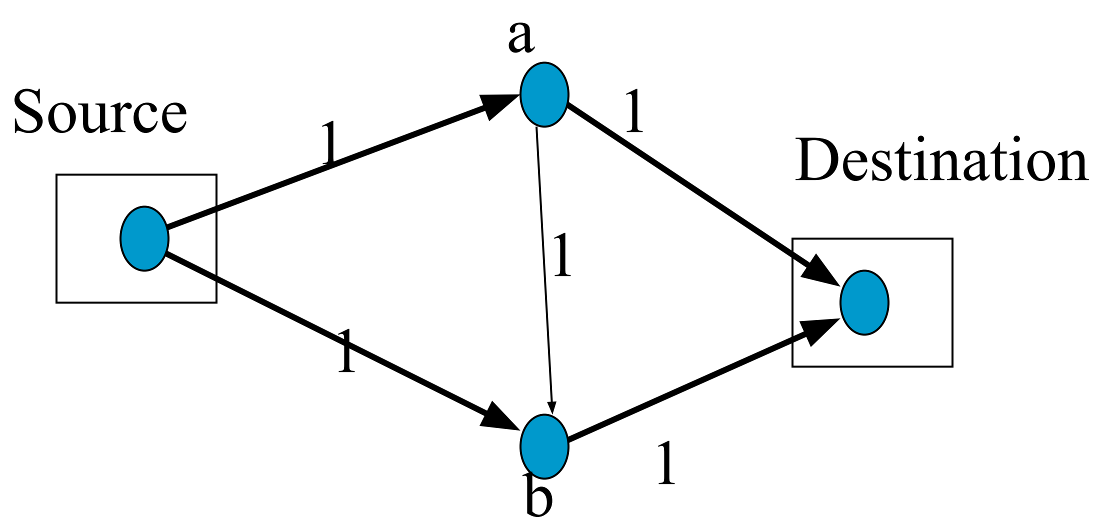
- The edges of the residual network are



Why do we need residual networks?

- Residual networks allow us to reverse flows if necessary.
- If we have taken a bad path then residual networks allow one to detect the condition and reverse the flow.
- A bad path is one which overlaps with too many other paths.

Example



Paths source, a, destinations and source, b destination gives a flow of 2 units.

Path source, a, b, destination overlaps with both the optimal paths.

If we initially choose source, a, b, destination as our path, then no greedy strategy will be able to augment the network flow any further (unless we use residual edges which allows recovery)

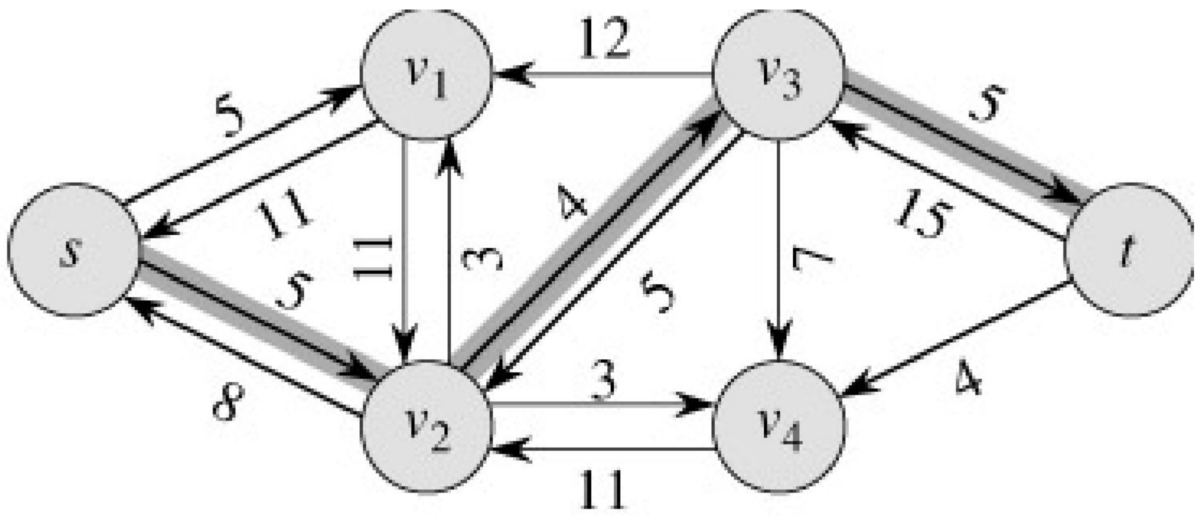
Verify how we recover in spite of the initial bad choice, if we use the residual network to augment flows.

Augmenting Paths

- An augmenting path p is a simple path from s to t on the residual network.
- We can put more flow from s to t through p.
- We call the maximum capacity by which we can increase the flow on p the residual capacity of p.

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

Augmenting Paths - example

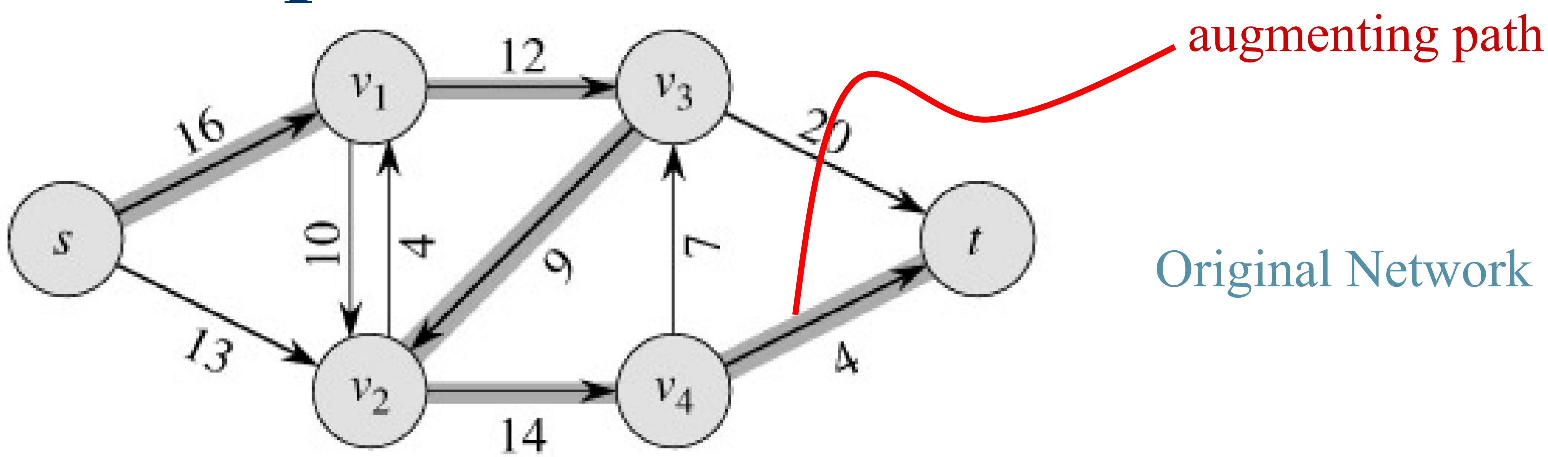


- The residual capacity of p = 4.
- Can improve the flow along p by 4.

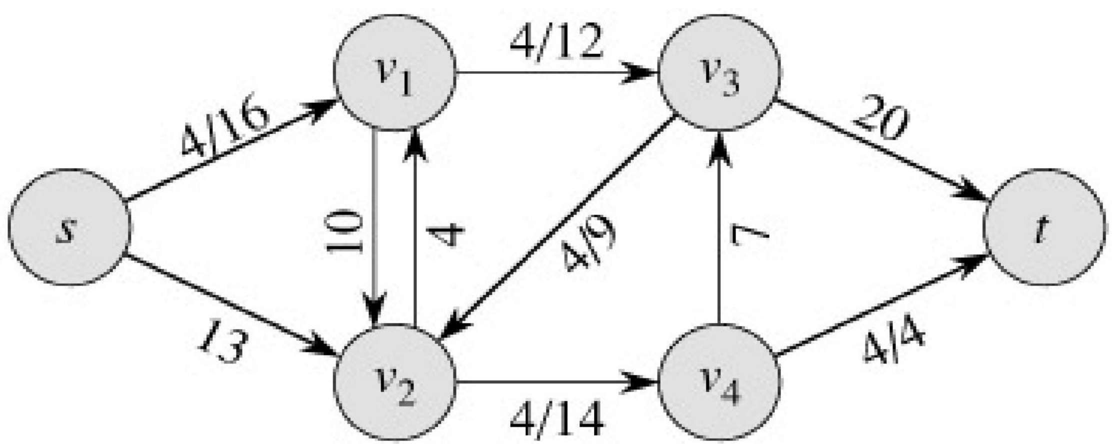
Ford-Fulkerson Method

```
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
```

Example

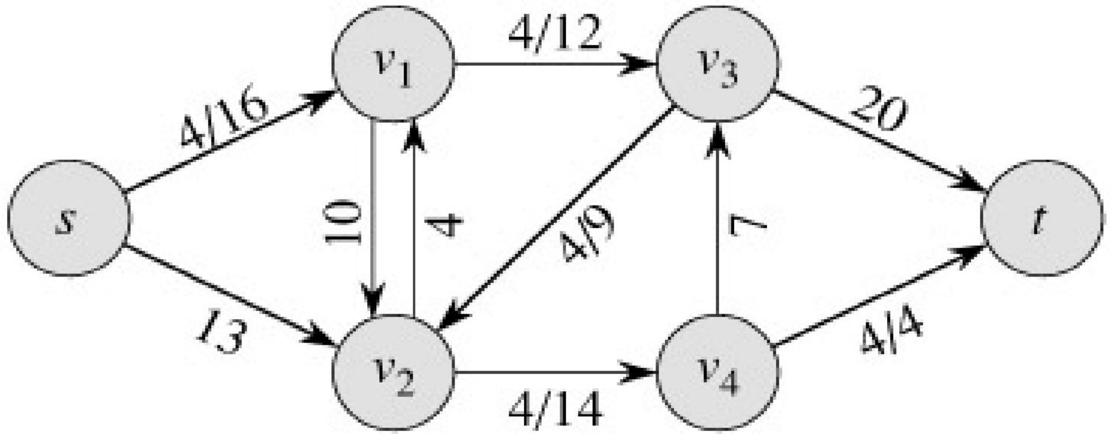


Original Network



Resulting Flow =4

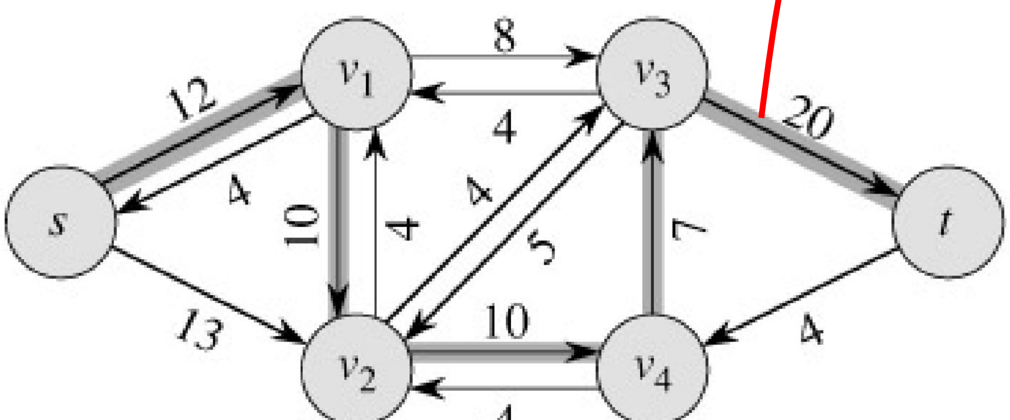
Example



Resulting

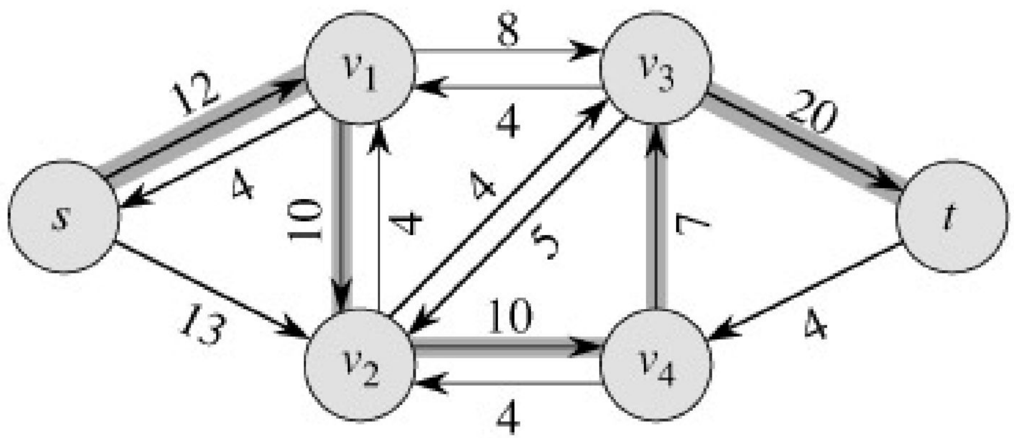
Flow = 4

augmenting path

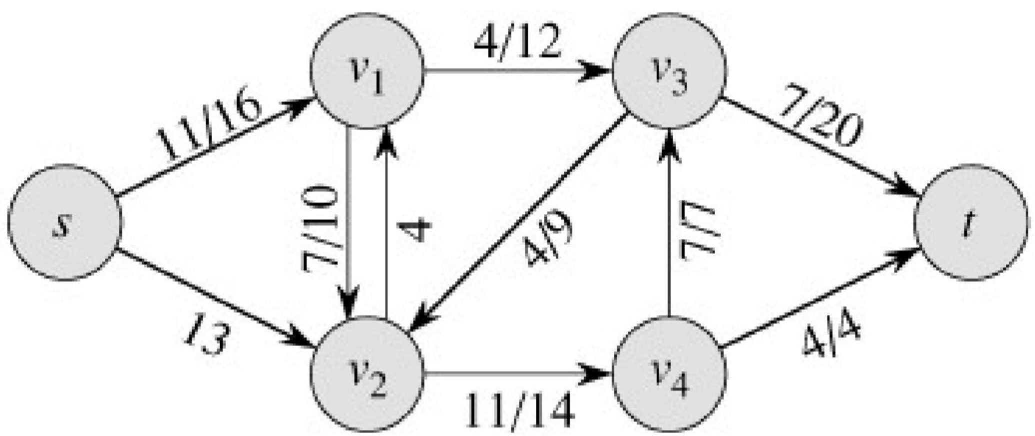


Residual Network

Example

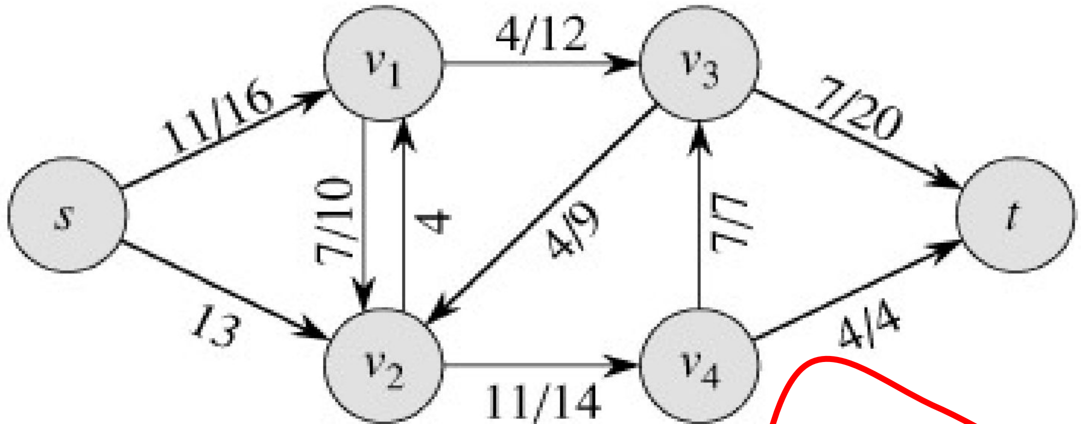


Residual Network



Resulting Flow =11

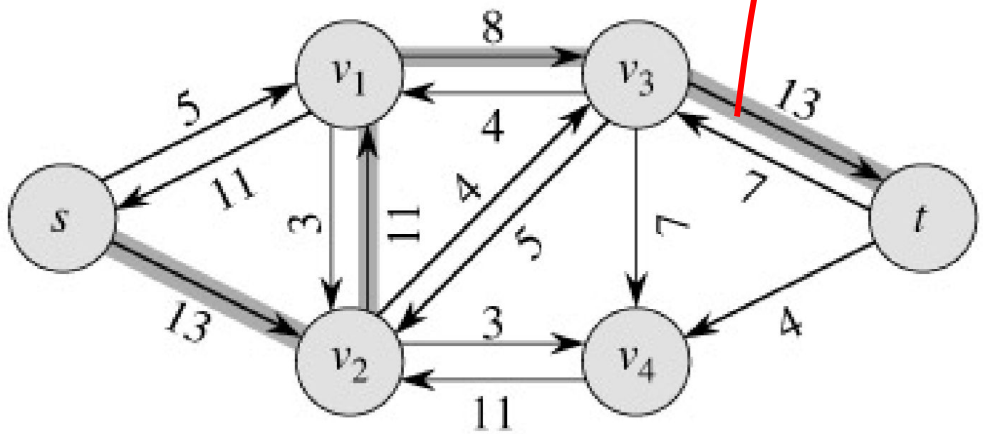
Example



Resulting

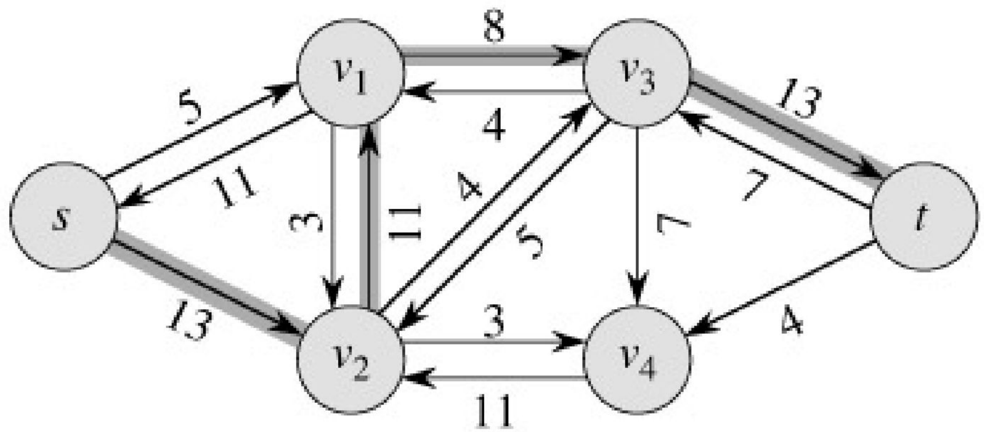
Flow =11

augmenting path

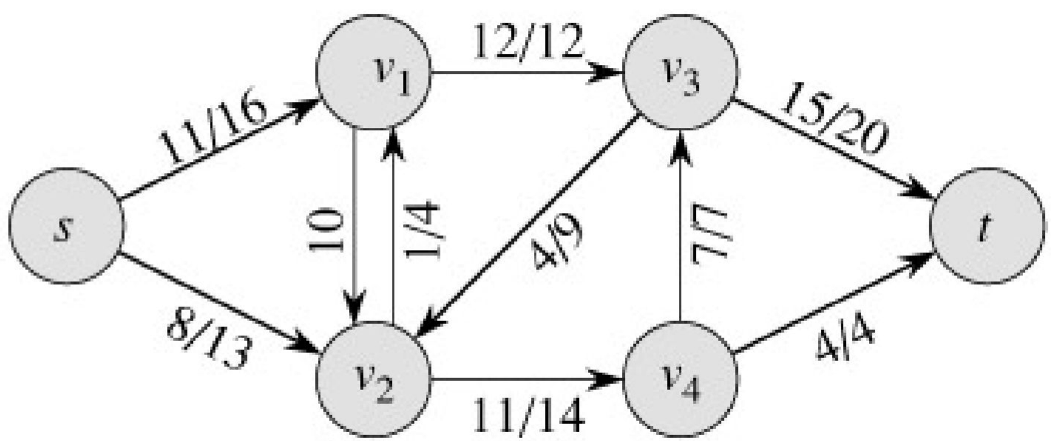


Residual Network

Example



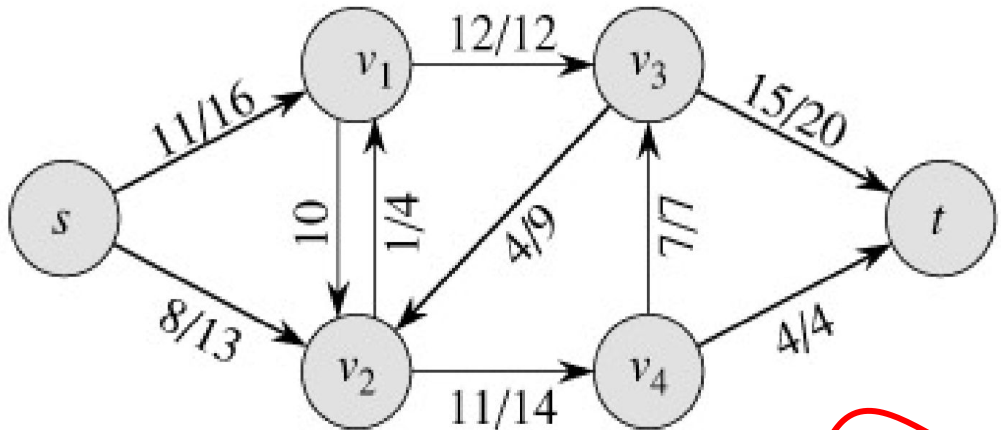
Residual Network



Resulting

Flow =19

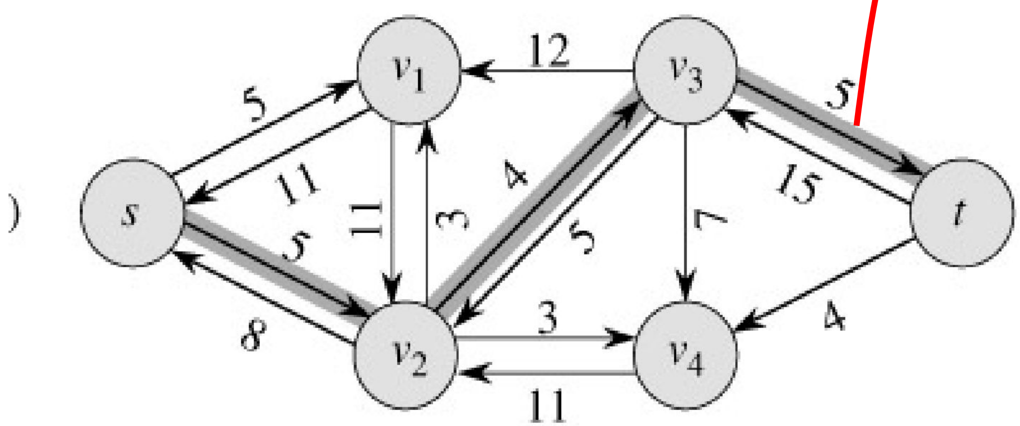
Example



Resulting

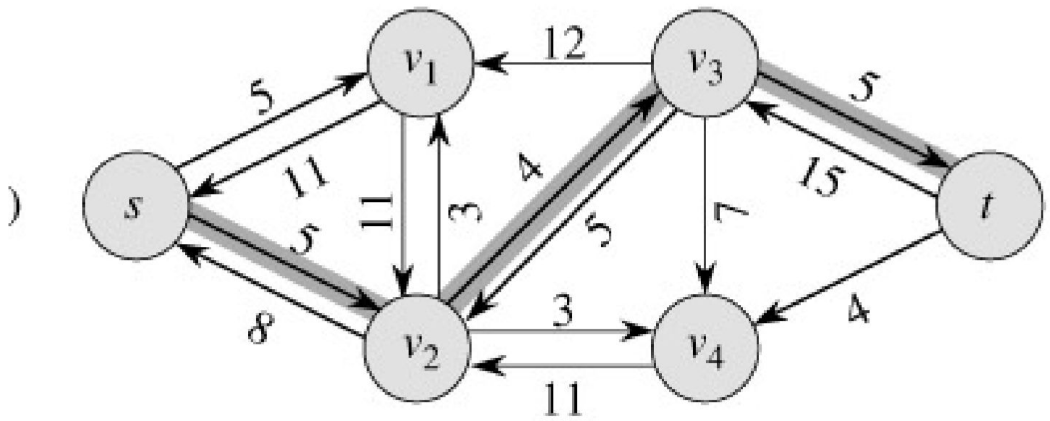
Flow = 19

augmenting path

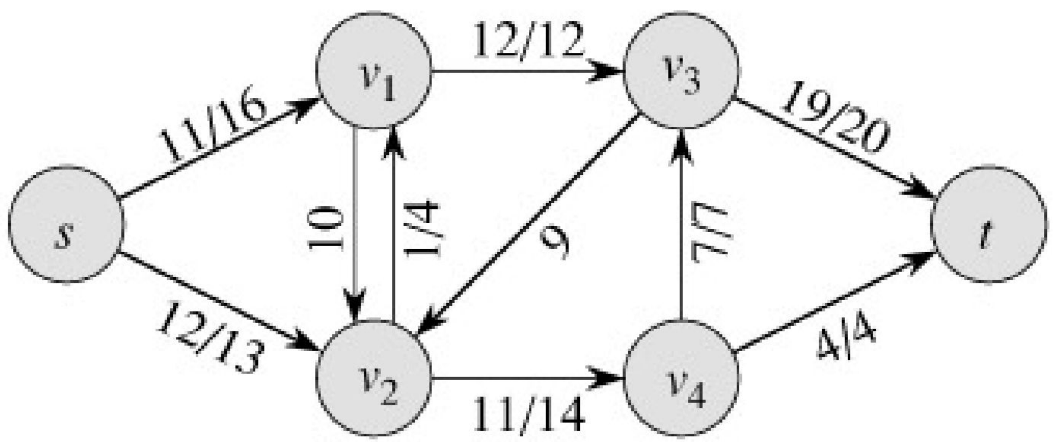


Residual Network

Example



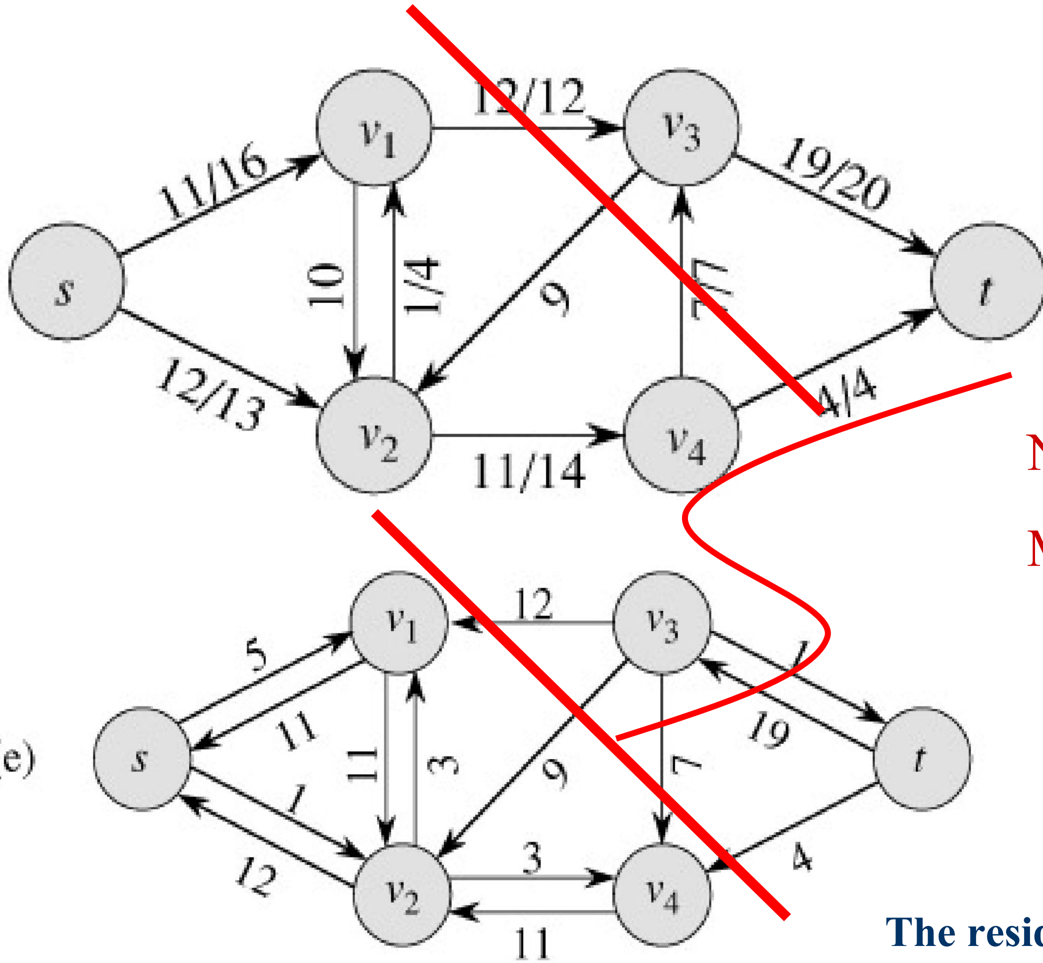
Residual Network



Resulting

Flow = 23

Example



(e)

Resulting

Flow = 23

No augmenting path:

Maxflow=23

Residual Network

The residual network Gf contains no augmented paths. So f is a maximum flow in G.

Ford-Fulkerson method, with details

```
Ford-Fulkerson(G,s,t)
1   for each edge (u,v)∈G.E do
2       f(u,v) = f(v,u) = 0
3   while ∃ path p from s to t in residual network G_f do
4       c_f = min{c_f(u,v): (u,v)∈p}
5       for each edge (u,v) in p do
6           f(u,v) = f(u,v) + c_f
7           f(v,u) = -f(u,v)
8   return f
```

The algorithms based on this method differ in how they choose *p* in step 3.

Time Analysis I

- A complete analysis establishing which specific method is best is a complex task, however, because their running times depend on
  - The number of augmenting paths needed to find a maxflow
  - The time needed to find each augmenting path

Analysis

FORD-FULKERSON(G, s, t)

for each edge $(u, v) \in E[G]$
  do $f[u, v] \leftarrow 0$
    $f[v, u] \leftarrow 0$

O(E)

while there exists a path p from s to t in the residual network G_f
  do $c_f(p) \leftarrow \min\{c_f(u, v) : (u, v) \text{ is in } p\}$
    for each edge (u, v) in p
      do $f[u, v] \leftarrow f[u, v] + c_f(p)$
        $f[v, u] \leftarrow -f[u, v]$

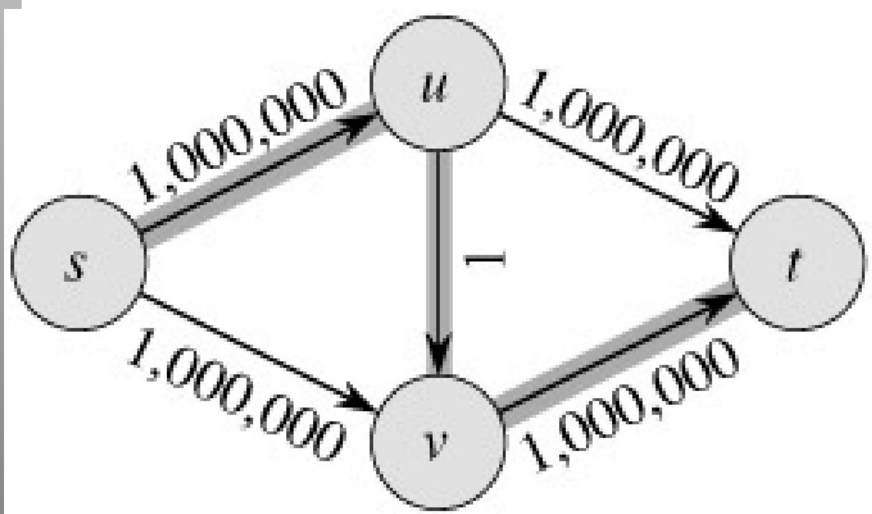
O(E)

Analysis

- If capacities are all integer, then each augmenting path raises $|f|$ by ≥ 1.
- If max flow is f^*, then need $\leq |f^*|$ iterations
  - **So time is $O(E|f^*|)$.**
- Note that this running time is not polynomial in input size. It depends on $|f^*|$, which is not a function of $|V|$ or $|E|$.
- If capacities are rational, can scale them to integers.
- If irrational, FORD-FULKERSON might never terminate!

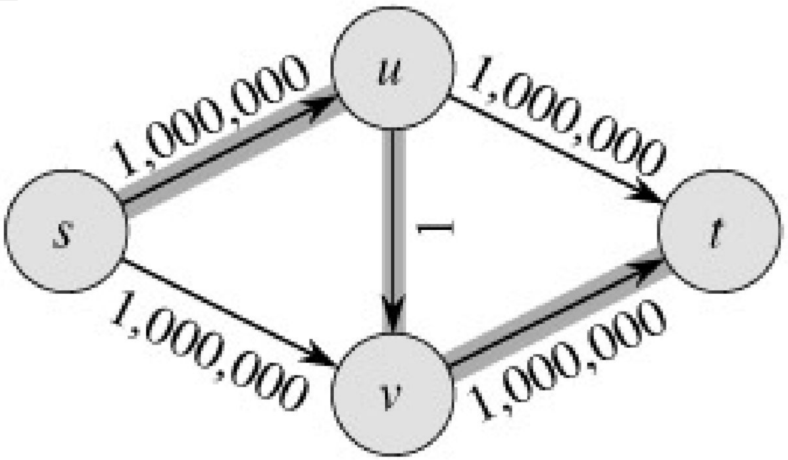
The basic Ford-Fulkerson Algorithm

- With time O (E |f*|), the algorithm is not polynomial.
- This problem is real: Ford-Fulkerson may perform very badly if we are unlucky:

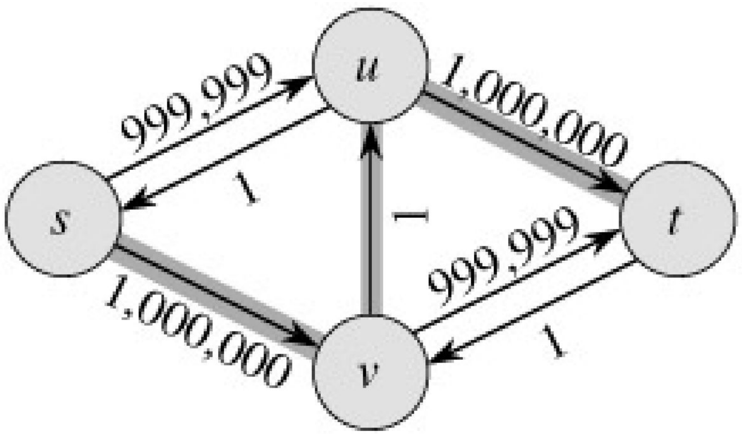


|f*|=2,000,000

Run Ford-Fulkerson on this example

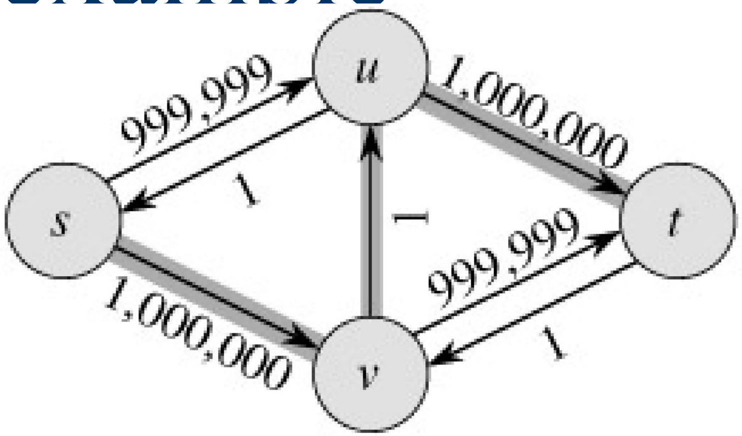


Augmenting Path

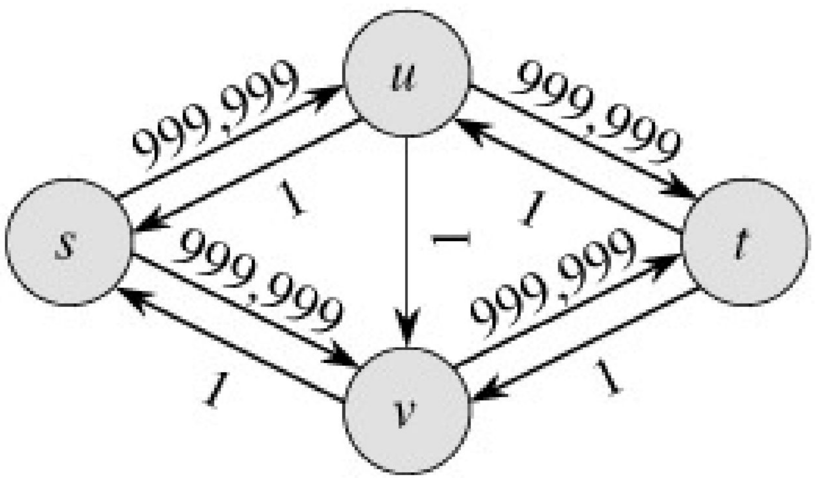


Residual Network

Run Ford-Fulkerson on this example

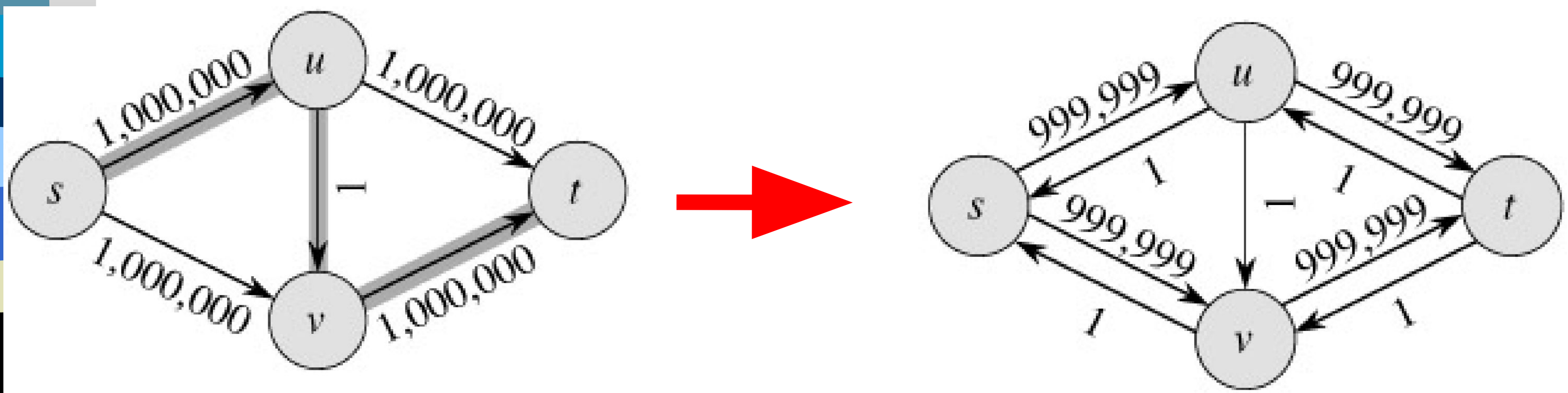


Augmenting Path



Residual Network

Run Ford-Fulkerson on this example



- Repeat 999,999 more times…

The Edmonds-Karp Algorithm

- A small fix to the Ford-Fulkerson algorithm makes it work in polynomial time.
- Specify how to compute the path in line 4.

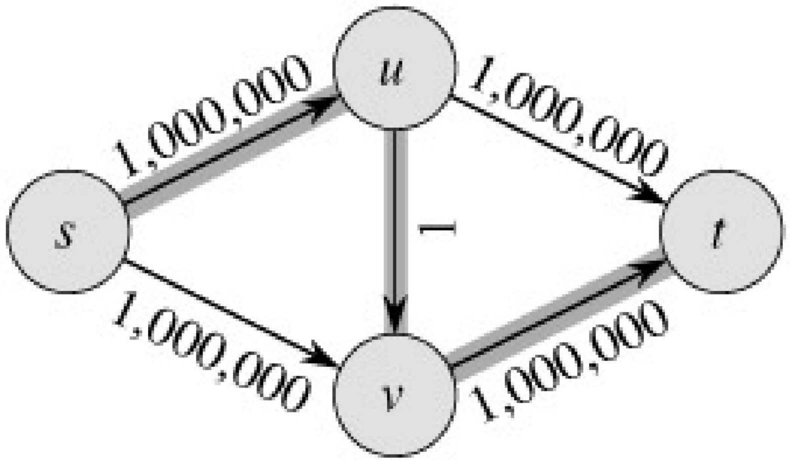
FORD-FULKERSON(G, s, t)

1 **for** each edge $(u, v) \in E[G]$
2   **do** $f[u, v] \leftarrow 0$
3         $f[v, u] \leftarrow 0$
4 **while** there exists a path p from s to t in the residual network G_f
5   **do** $c_f(p) \leftarrow \min\{c_f(u, v) : (u, v) \text{ is in } p\}$
6   **for** each edge (u, v) in p
7   **do** $f[u, v] \leftarrow f[u, v] + c_f(p)$
8                 $f[v, u] \leftarrow -f[u, v]$

The Edmonds-Karp Algorithm

- Compute the path in line 4 using breadth-first search on residual network.
- The augmenting path p is the shortest path from s to t in the residual network (treating all edge weights as 1).
- Runs in time $O(V E^2)$.

The Edmonds-Karp Algorithm - example



- Edmonds-Karp’s algorithm runs only 2 iterations on this graph.

Time Complexity

FORD-FULKERSON(G, s, t)

for each edge $(u, v) \in E[G]$
  do $f[u, v] \leftarrow 0$
   $f[v, u] \leftarrow 0$
while there exists a path p from s to t in the residual network G_f
  do $c_f(p) \leftarrow \min\{c_f(u, v) : (u, v) \text{ is in } p\}$
   for each edge (u, v) in p
    do $f[u, v] \leftarrow f[u, v] + c_f(p)$
     $f[v, u] \leftarrow -f[u, v]$

- Let, total number of flow augmentations performed by Edmonds-Karp algorithm is O(VE)
- BFS to find the augmented path – O(E)
- So, Total running time is $O(VE^2)$

Thm: Total number of flow augmentations is O(VE)

- Proof:

Conditions

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 augmented paths.
3. $|f| = c(S,T)$ for some cut (S,T) (a min-cut).

It is a flow since there is no augmented paths It is maximum since the sink is not reachable from the source