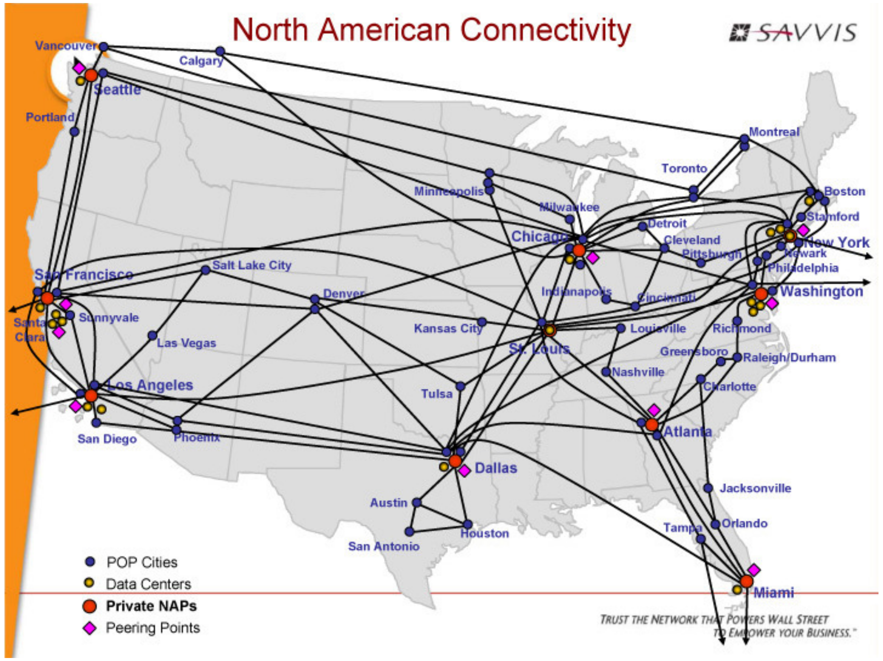
Our Roadmap

- What is the maximum flow problem?
- Ford-Fulkerson algorithm: using a residual network
- Edmonds-Karp algorithm: using BFS paths to run faster
- Matching: another application of maximum flow

Flow Networks in Real-Life!



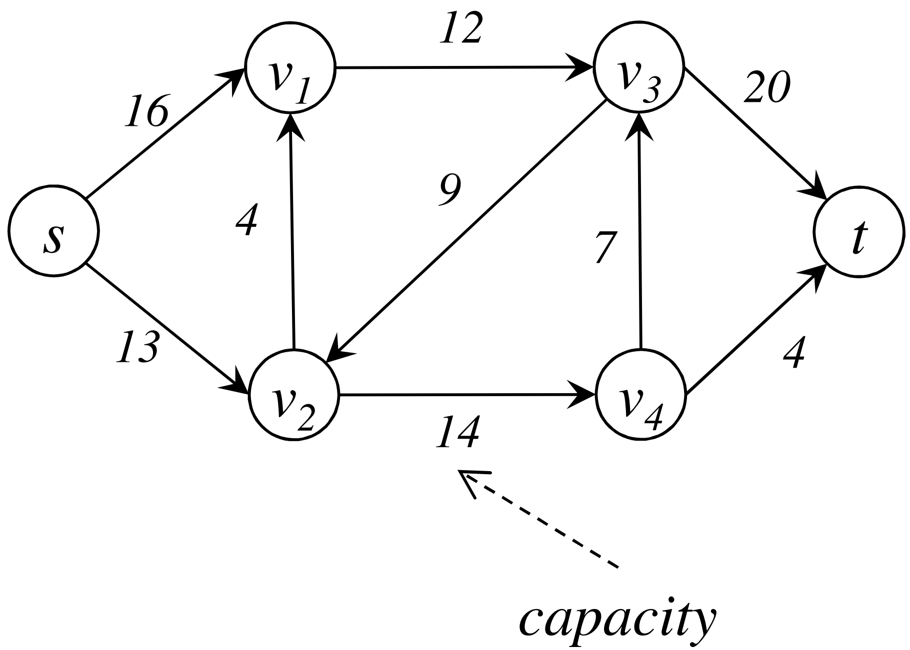
Adapted from Google Images

Flow Networks

- Applications of flow network:
  - Traffic flow, electric grid, communication network, assembly line, etc.
- What are their common features?
  - A directed graph
  - Each edge has a **capacity**
    - E.g., bandwidth, cable diameter, road lanes
  - **Flow**: the flowing rate of "material" on an edge
    - E.g., data bits per second, current flow per second, cars per second
  - Source(s): the producer(s) of materials
  - Sink(s): the consumer(s) of materials

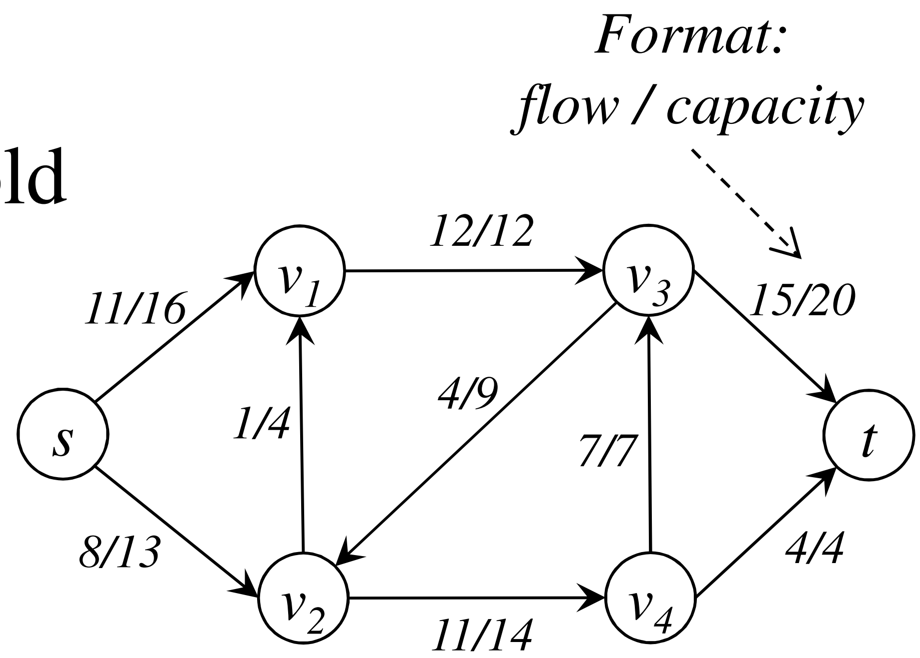
Flow Networks: Definitions

- **Flow network** $G = (V, E)$
- V is a set of vertices
  - Producer: a **source** vertex s
  - Consumer: a **sink** vertex t
- E is a set of edges
  - Let (u, v) be an edge in E
  - It has a **capacity** $c(u, v)$, and a **flow** $f(u, v)$
  - Both capacity and flow are non-negative
  - Only allow directed edges, i.e., cannot have both edges (u, v) and (v, u) in E



Flow Networks: Definitions

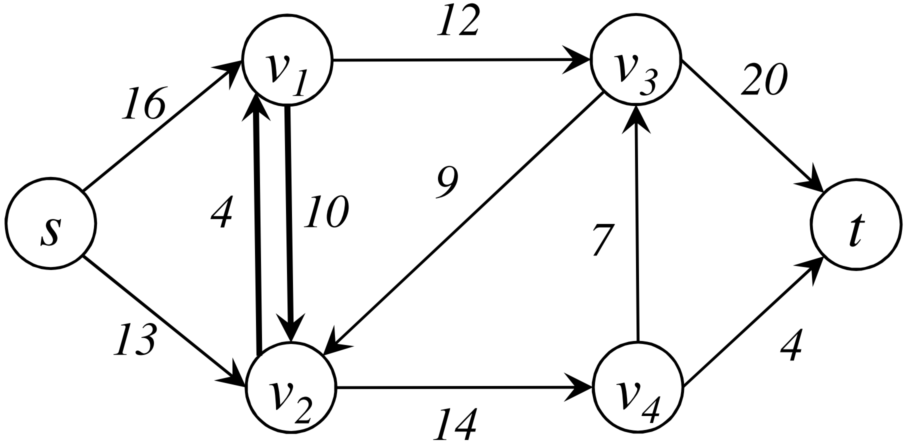
- Two flow properties must hold
- **Capacity constraint**
  - for any edge (u, v) in E, $c(u, v) \geq f(u, v) \geq 0$
- **Flow conservation**
  - Flow-in equals flow-out
  - for any vertex u in $V - \{s, t\}$, $\Sigma_{v \in V} f(v, u) = \Sigma_{v \in V} f(u, v)$
- Note: for an edge (u, v) not in E
  - We define $c(u, v) = f(u, v) = 0$



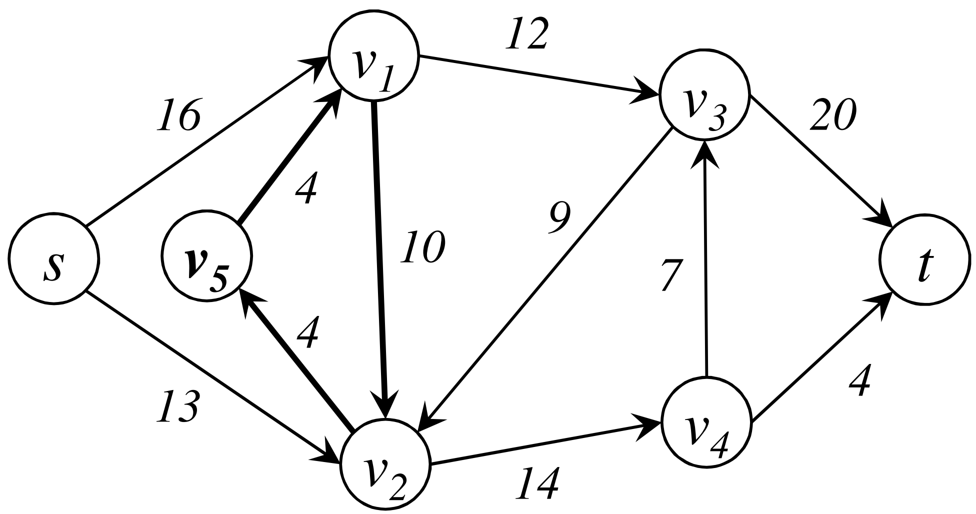
Let's check the flow conservation at vertex v_2

Flow Networks: Modeling

- Recall that a flow network does not allow both edges (u, v) and (v, u) to be in E
- How do we model a network that contains edges in both directions?
  - Example: edges (v_1, v_2) and (v_2, v_1)
- Just add a dummy vertex on one such edge



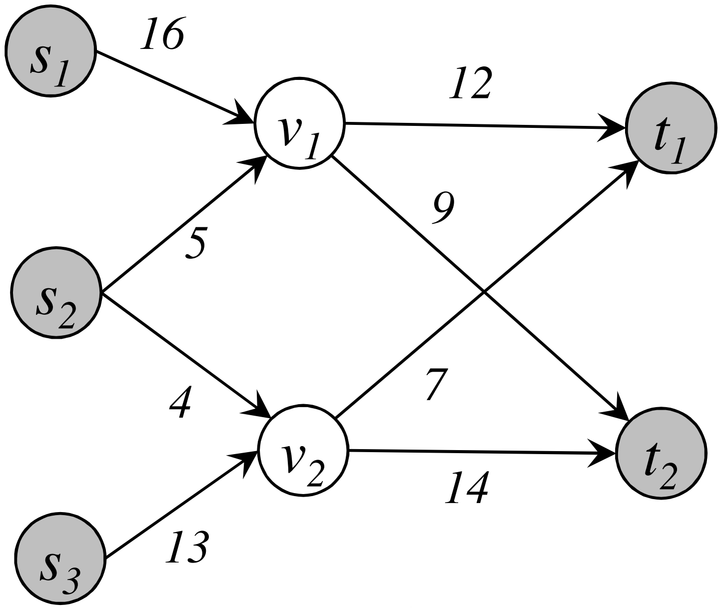
original network



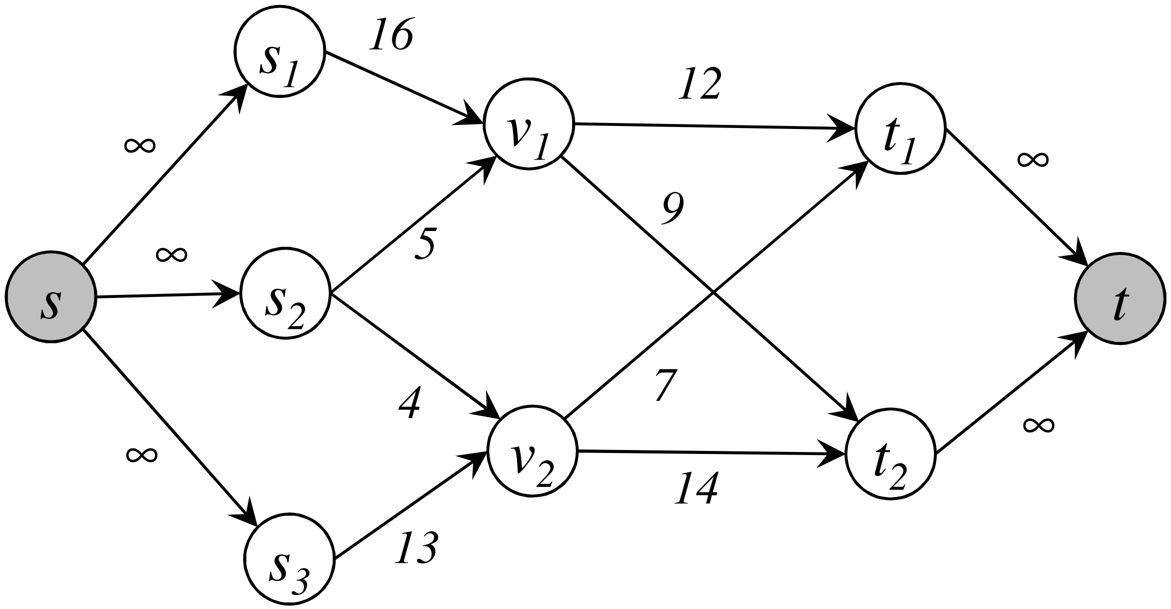
converted flow network

Flow Networks: Modeling

- Recall that a flow network has one source *s* and one sink *t*
- How do we model a network that has multiple sources and multiple sinks?
- Add a final source *s* and a final sink *t*
    - Link them to original sources and sinks by edges with capacity ∞



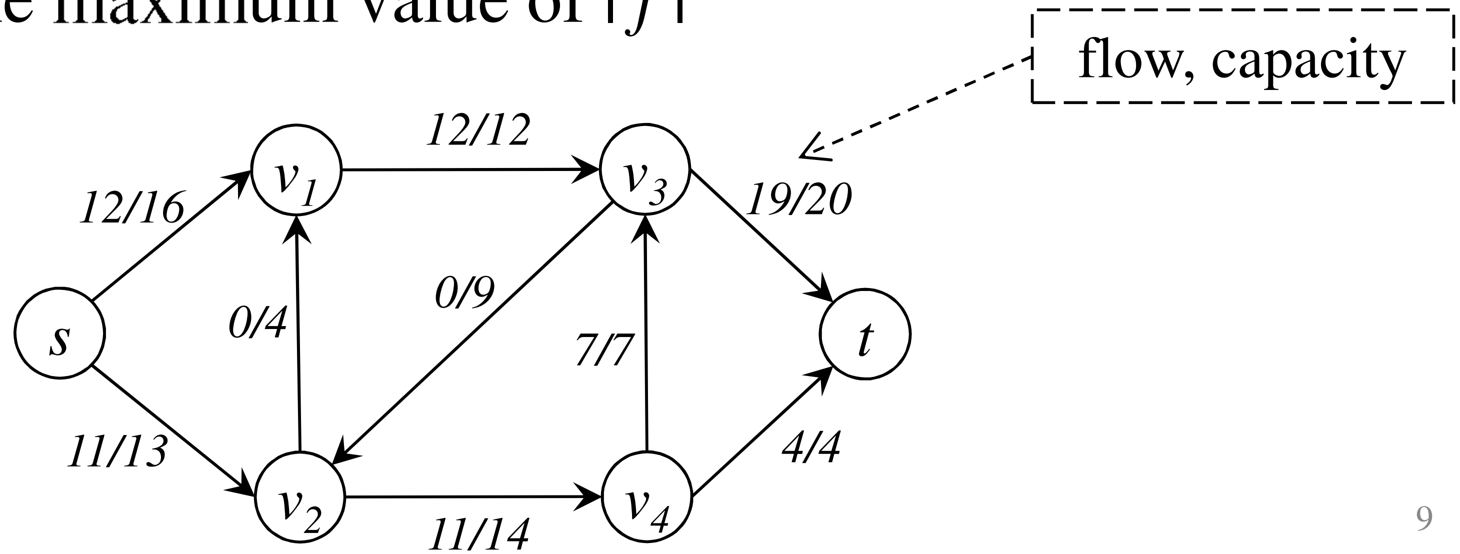
original network



converted flow network

The Maximum Flow Problem

- $|f|$ denotes the value of a flow f
  - $|f|$ = flow out of the source – flow into the source
  - $|f| = \Sigma_{v\in V} f(s, v) - \Sigma_{v\in V} f(v, s)$
  - Example: $|f| = (12 + 11) - 0 = 23$
- The **maximum flow** problem
  - Given a flow network G, with source s and sink t, find the maximum value of $|f|$

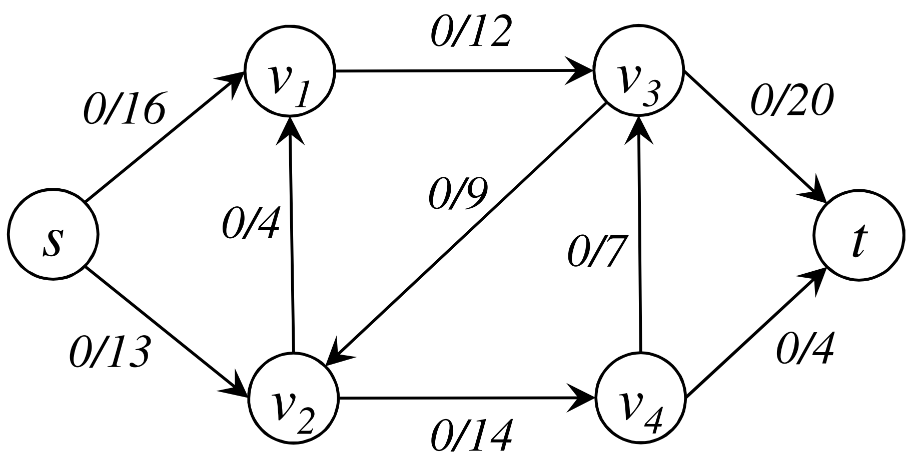


Our Roadmap

- What is the maximum flow problem?
- Ford-Fulkerson algorithm: using a residual network
- Edmonds-Karp algorithm: using BFS paths to run faster
- Matching: another application of maximum flow

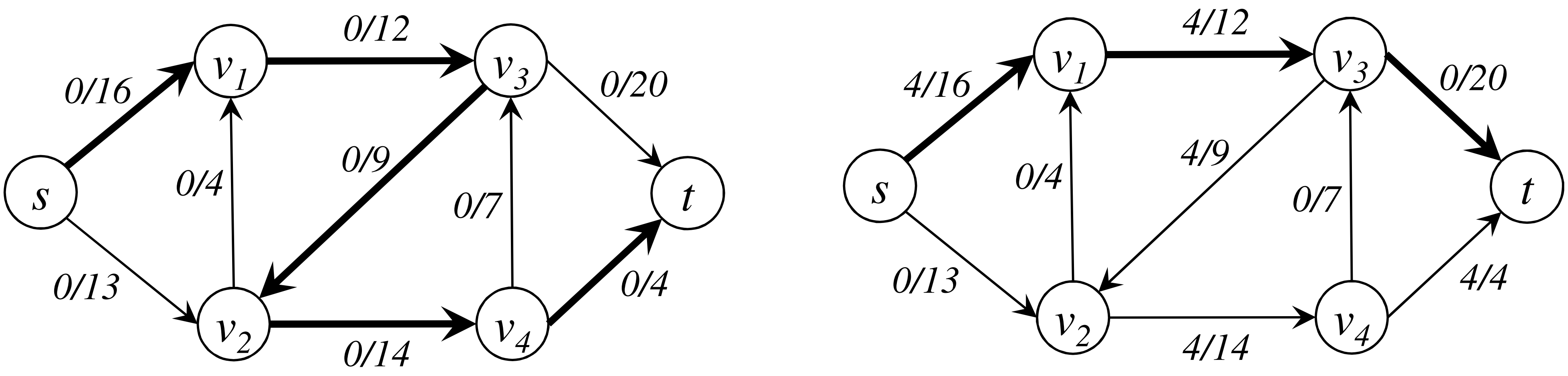
Basic Method

- Basic method for solving the maximum flow problem
  - 1. Find a path from *s* to *t*
  - 2. Increase the flow value of the path
  - 3. Repeat until no more path can be found
- Does this method compute the maximum flow correctly?
  - Why? Why Not?
- Let's look at an example …



Basic Method: Example

- Iteration 1
  - Choose the path $<s, v_1, v_3, v_2, v_4, t>$
  - How large can the value of the flow become?
  - Increase the flow of the path by 4
- Iteration 2
  - Choose the path $<s, v_1, v_3, t>$, its *min. residual capacity* is 12–4=**8**
  - Increase the flow of the path by 8



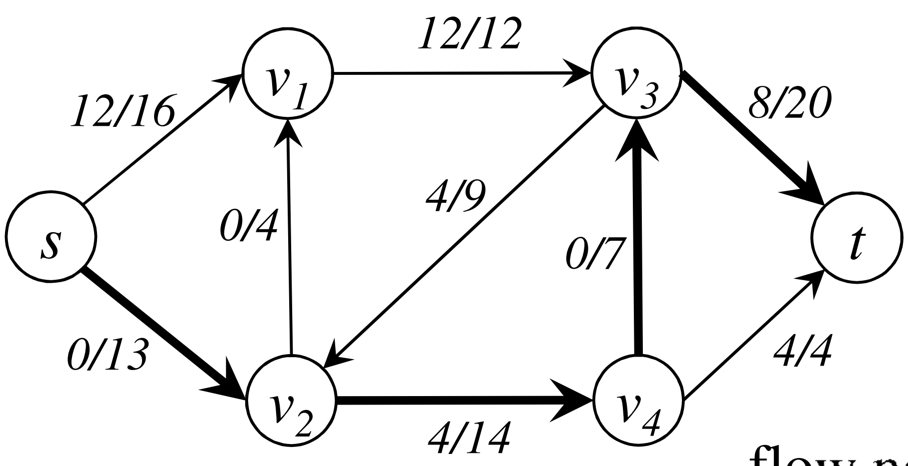
flow network *G*

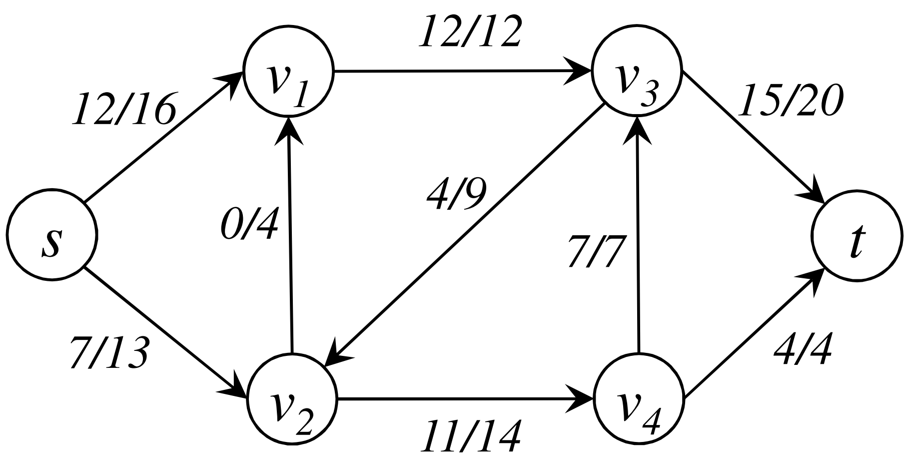
Basic Method: Example

- Iteration 3
  - Choose the path $\langle s, v_2, v_4, v_3, t\rangle$, its min. residual capacity is 7
  - Increase the flow of the path by 7
- Iteration 4
  - We cannot choose any path now. Why?

See page 9 …

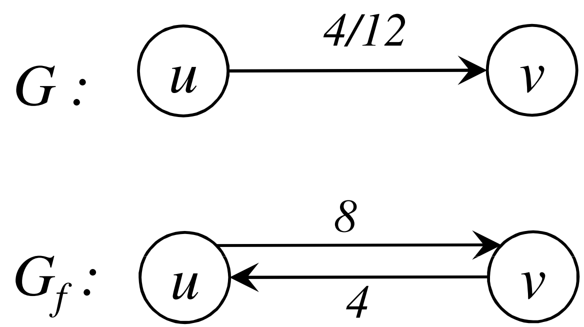
  - The flow value is: 12+7=19. Is this really the maximum flow?
- We need a method to "**cancel**" flow that blocks our way!





flow network G

Residual Network



- A residual network $G_f = (G.V, E_f)$
  - Defined by a flow network G and a flow f
  - G_f has the same set of vertices as G
  - E_f contains every edge (u, v) that satisfies $c_f(u, v) > 0$
  - Given an edge (u, v) in E_f, its **residual capacity** $c_f(u, v)$ is the amount of flow allowed to be taken

$$c_f(u, v) = c(u, v) - f(u, v) \qquad \text{if } (u, v) \in G.E$$
$$c_f(v, u) = f(u, v) \qquad \text{if } (u, v) \in G.E$$

<u>Example</u>

$c(u, v) = 12$

$f(u, v) = 4$

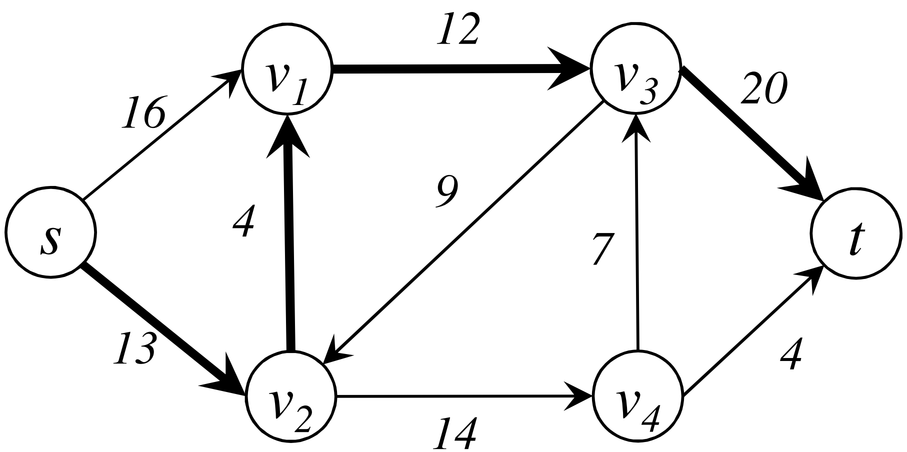
$c_f(u, v) = 12 - 4 = 8$

$c_f(v, u) = 4$

because $c(v, u) = 0$ and $f(v, u) = -f(u, v)$

Augmenting Path

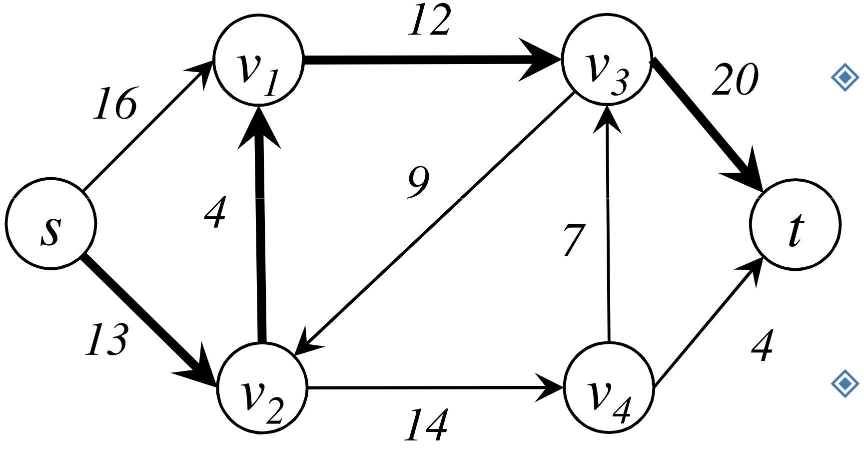
- What is an **augmenting path** p ?
  - A simple path (*no-cycle*) from s to t in the residual network G_f
- The **residual capacity** of p is:
  - $c_f(p) = \min\{\ c_f(u, v) : (u, v) \text{ is on } p\ \}$
- How do we add this flow to the network?



residual network G_f

What is the residual capacity of the black path ?

Adding a Flow to Residual Network



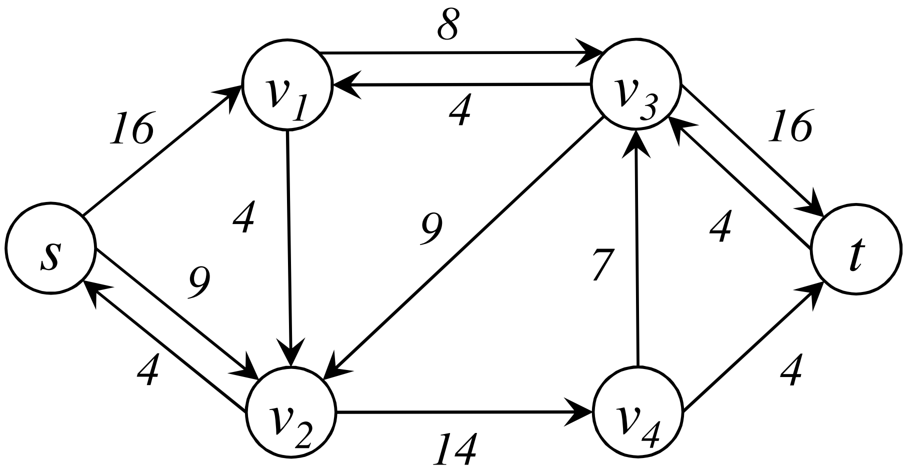
residual network G_f

- Given a flow f in G, and a flow f' in G_f
- **Augmenting flow** $(f \uparrow f')(u, v) =$

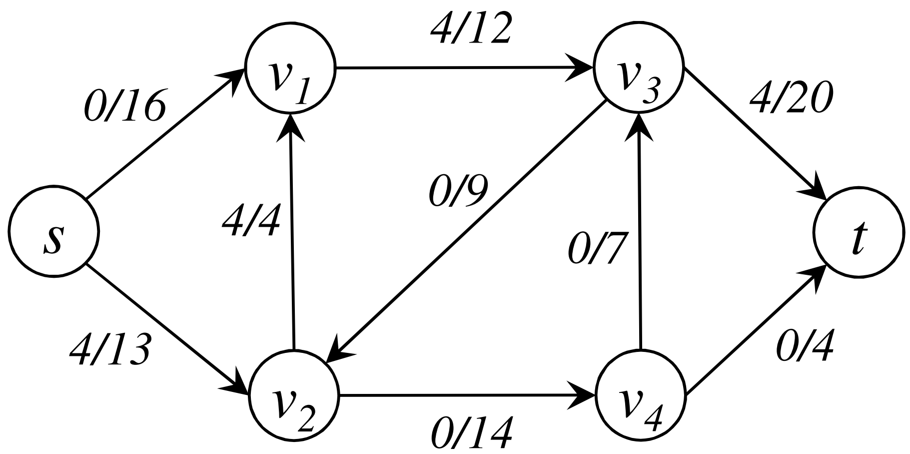
$f(u, v) + f'(u, v) - f'(v, u)$ if $(u, v) \in G.E$

0 otherwise

- Changes of edges on the residual network
  - Reduce res. capacity of a forward edge
  - Increase res. capacity of a reverse edge
  - Delete edges with “zero” res. capacity

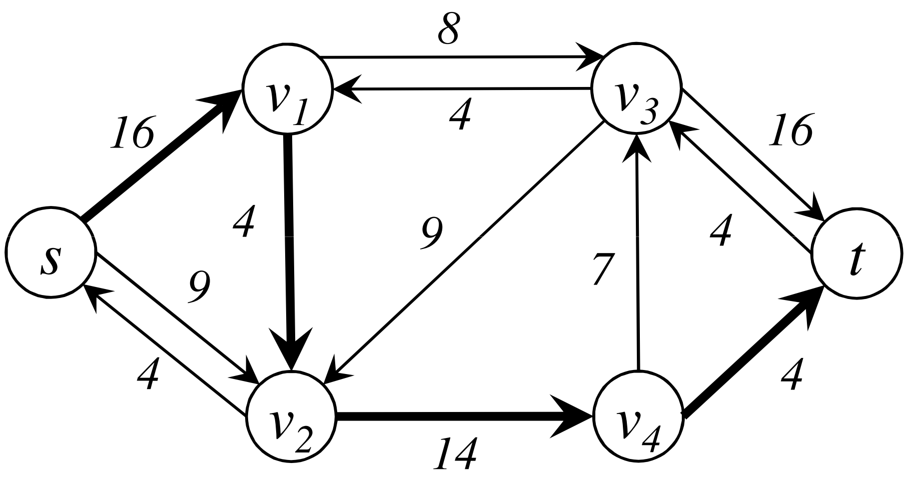


residual network G_f (updated)



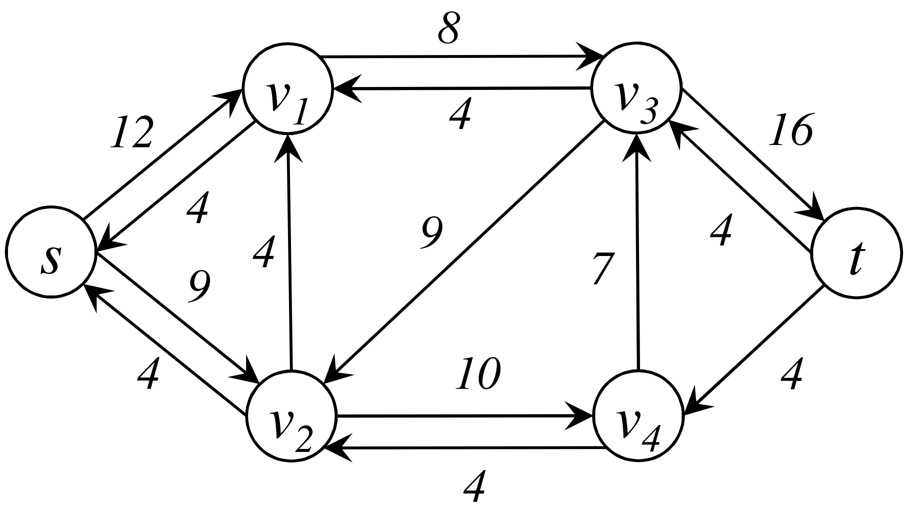
flow network G (updated)

Adding a Flow to Residual Network

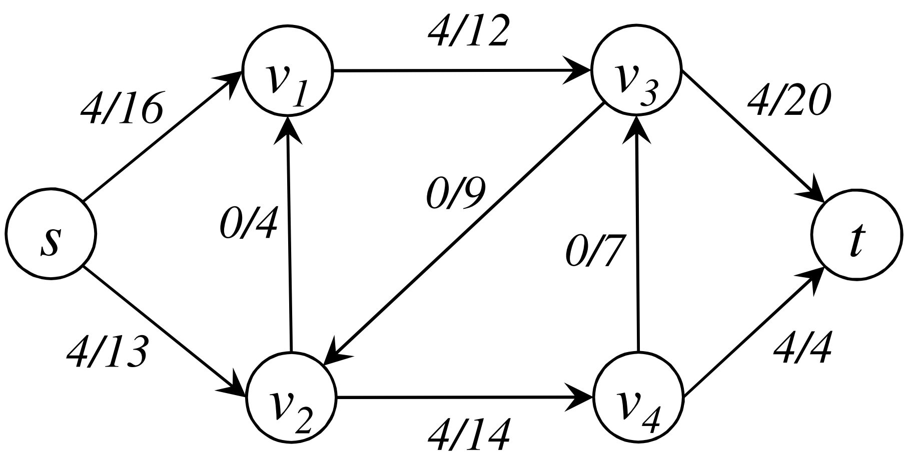


residual network G_f

- **Cancellation**: flow increase on a reverse edge (i.e., edge in G_f but not in G)
  - Example: edge (v_1, v_2) is a reverse edge
  - We have sent 4 units on (v_2, v_1)
  - Next, we will send 4 units on (v_1, v_2)
- *Why the cancellation is useful?*



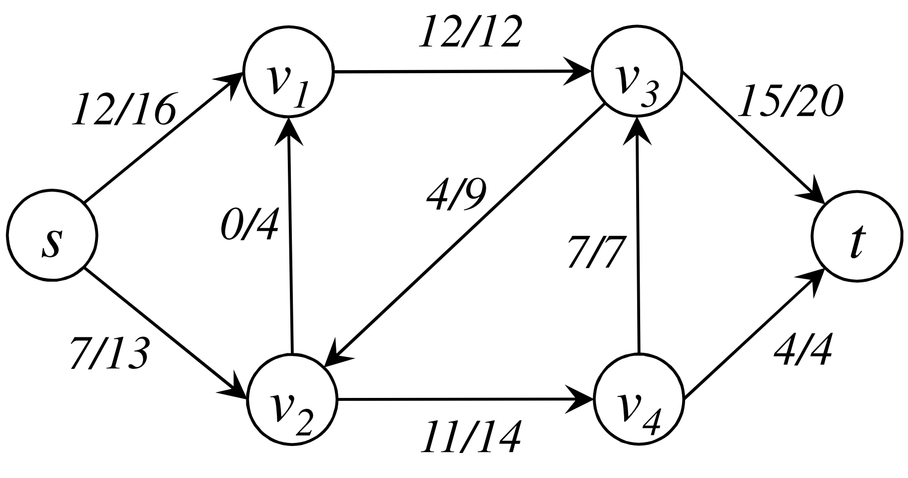
residual network G_f (updated)



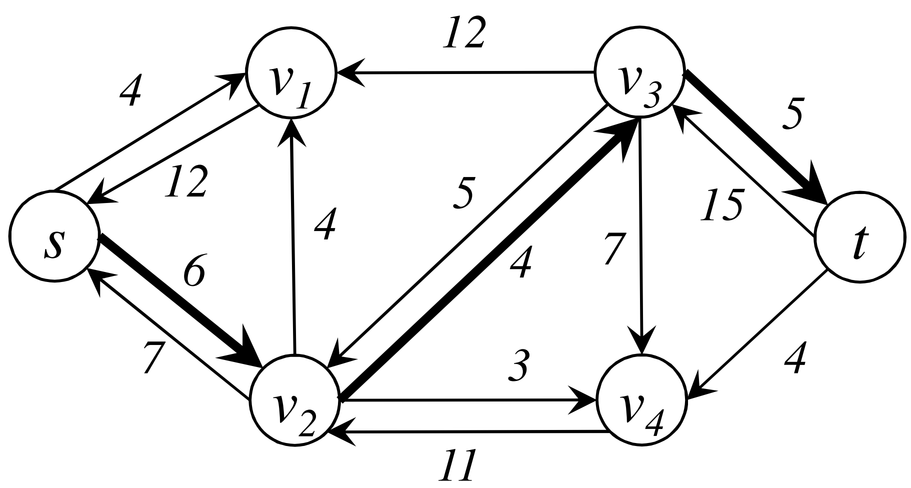
flow network G (updated)

Revisit: Why is the Residual Network Useful?

- Can we add a flow in this flow network G ?
  - In G, there is NO path from s to t now
  - We cannot change any existing flow
- Can we add a flow in its residual network G_f ?
  - In G_f, there is still a path $<s, v_2, v_3, t>$
  - The cancellation effect automatically changes some existing flow



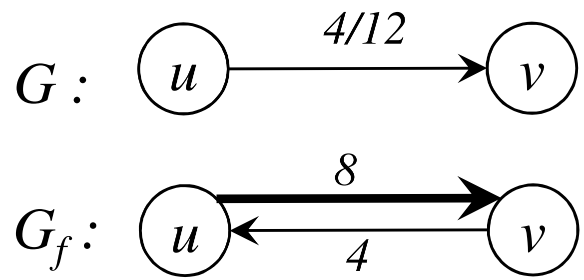
flow network G

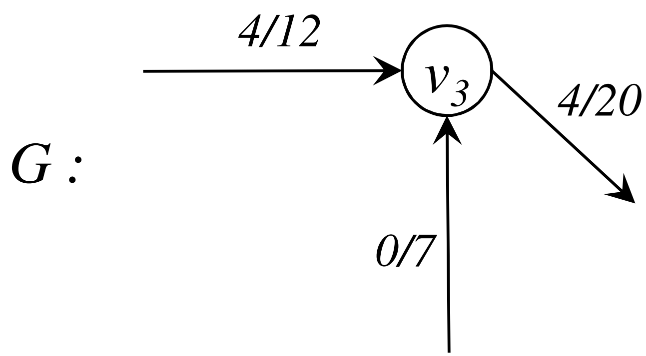


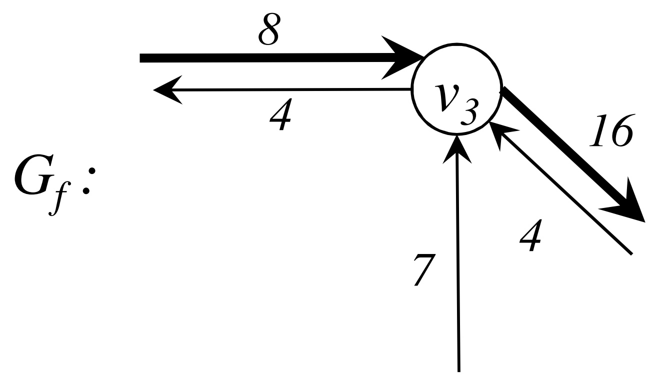
residual network G_f

Correctness of Augmenting Flow

- Let f be a flow in G, and f' be a flow in G_f
- Is it correct to add the flow f' to the flow f in G ?
  - The augmenting flow ($f \uparrow f'$) is a flow in G, and its flow value is: $|f \uparrow f'| = |f| + |f'|$
- In the appendix, we will show that:
  - Flow property: Capacity constraint
  - Flow property: Flow conservation
  - Flow value: $|f \uparrow f'| = |f| + |f'|$







Ford-Fulkerson Algorithm

Augmenting flow

$$(f \uparrow f')(u, v) = \begin{cases} f(u, v) + f'(u, v) - f'(v, u) & \text{if } (u, v) \in G.E \\ 0 & \text{otherwise} \end{cases}$$

Ford-Fulkerson(G, s, t)

```
1  for each edge (u, v) ∈ G.E
2      f(u, v) ← 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 on p }
5      for each edge (u, v) on p
6          if (u, v) ∈ G.E
7              f(u, v) ← f(u, v) + c_f(p)
8          else
9              f(v, u) ← f(v, u) − c_f(p)
```

Idea

- Lines 1-2: set the flow to zero
- Line 3: find a path from s to t in G_f
- Line 4: compute the path's flow value
- Lines 6-7: add the flow for an actual edge in G
- Lines 8-9: cancel the flow for a reverse edge
- Stop when there is no path from s to t in G_f
    - See the *correctness proof* in textbook

Ford-Fulkerson Algorithm

Ford-Fulkerson(G, s, t)

1 for each edge $(u, v) \in G.E$
2 $\quad f(u, v) \leftarrow 0$
3 while there exists a path p from s to t in the residual network G_f
4 $\quad c_f(p) \leftarrow \min\{\, c_f(u, v) : (u, v)$ is on $p \,\}$
5 $\quad$ for each edge (u, v) on p
6 $\quad\quad$ if $(u, v) \in G.E$
7 $\quad\quad\quad f(u, v) \leftarrow f(u, v) + c_f(p)$
8 $\quad\quad$ else
9 $\quad\quad\quad f(v, u) \leftarrow f(v, u) - c_f(p)$

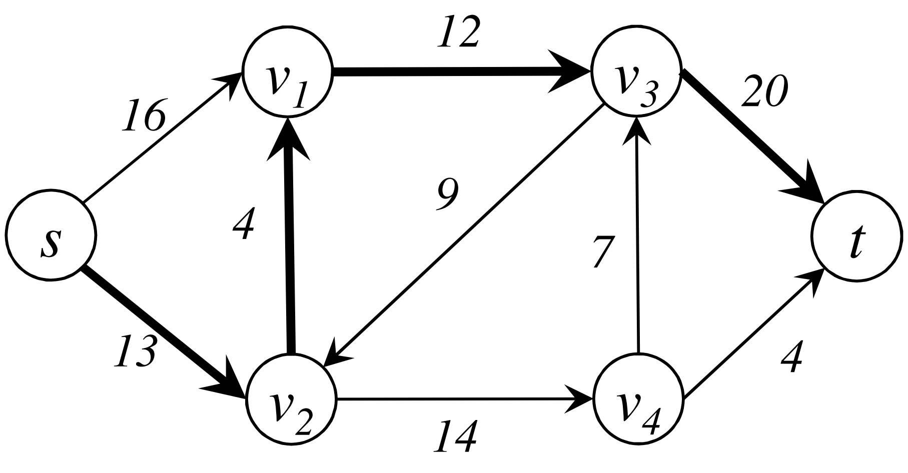
<u>Time complexity</u>

- To find a path p by graph traversal, it takes $O(|V| + |E|) = O(|E|)$ time
- Each outer loop (Lines 3-9) increases the flow value by at least 1
- Let $|f^*|$ be the maximum flow value
- Total time: $O(\, |E|\, |f^*|\,)$

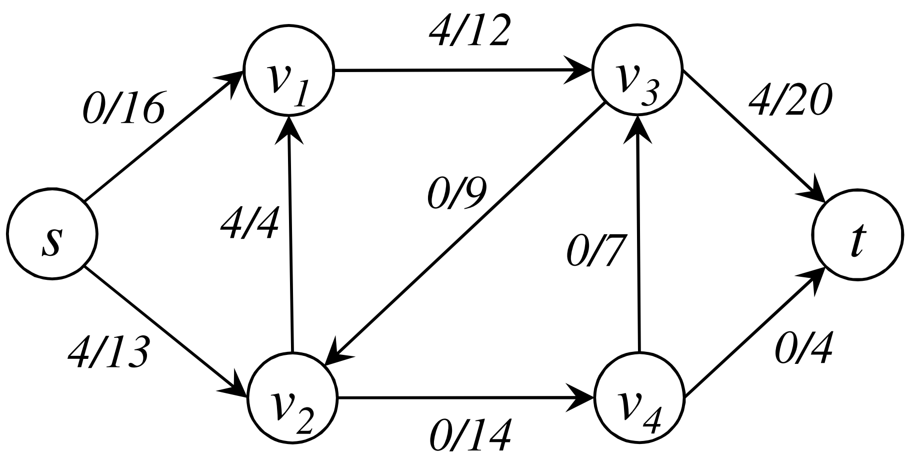
Ford-Fulkerson Algorithm: Example

Iteration 1

- 1. Choose a path from s to t, on the residual network G_f
  - E.g., the path $\langle s, v_2, v_1, v_3, t \rangle$, shown in bold type
- 2. The minimum residual capacity $c_f\,(u, v)$ on the path is 4
- 3. Update the flow on G
  - The flow on G_f will then be automatically updated



residual network G_f



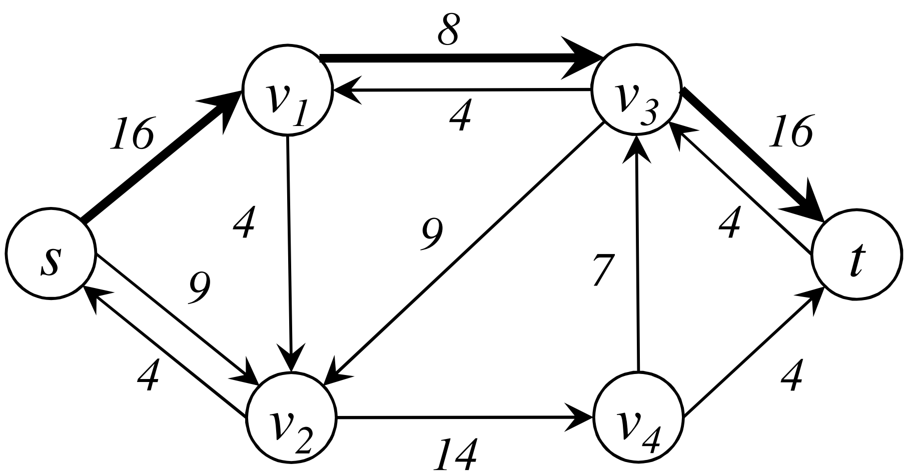
flow network G

What will the residual network become ?

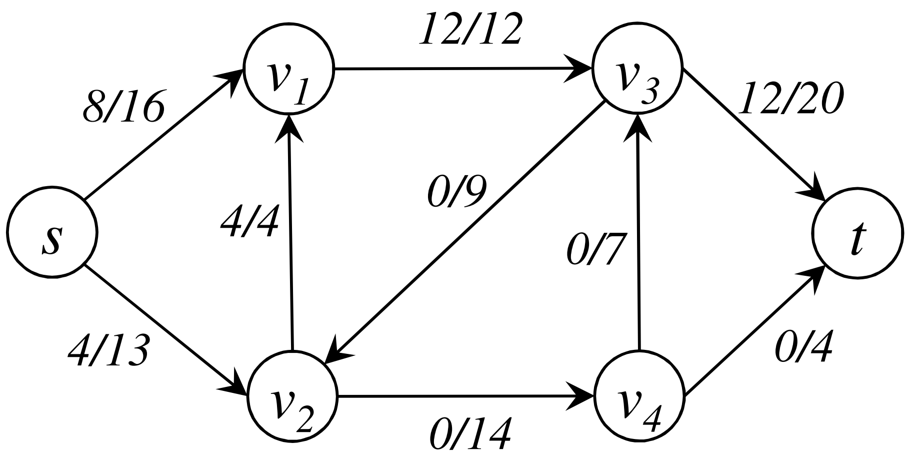
Ford-Fulkerson Algorithm: Example

<u>Iteration 2</u>

- 1. Choose a path from *s* to *t*, on the residual network G_f
  - E.g., the path $<s, v_1, v_3, t>$
- 2. The minimum residual capacity is 8
- 3. Update the flow



residual network G_f



flow network G

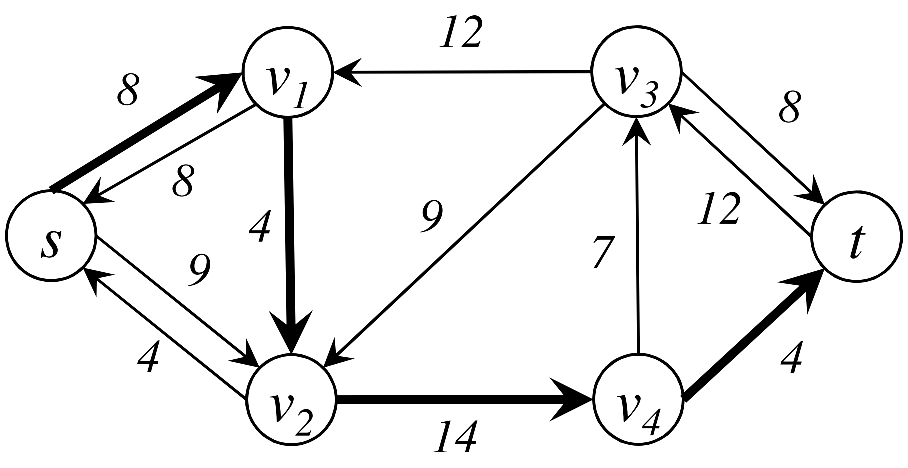
What will happen next ?

Ford-Fulkerson Algorithm: Example

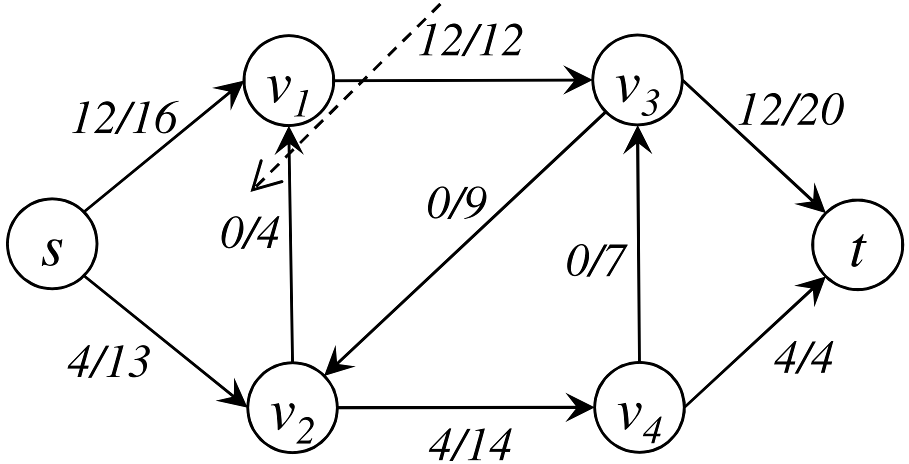
Iteration 3

- 1. Choose a path from s to t, on the residual network G_f
  - E.g., the path $\langle s, v_1, v_2, v_4, t \rangle$
- 2. The minimum residual capacity is 4
- 3. Update the flow

flow cancellation



residual network G_f



flow network G

What will happen next ?