Graph Algorithms (Chapter 26)
Maximum Flow

University of Stavanger

Graph Algorithms (Chapter 26) Maximum Flow

Goals

The maximum flow problem

Antiparallel edges issues

Multiple sources and sinks

The Ford-Fulkerson method

Residual networks

Augmenting paths

Max-flow min-cut theorem

Forld-Flkerson algorithm and its runtime

Edmonds-Karp algorithm and its runtime

Summary

Graph Algorithms (Chapter 26)
Maximum Flow Problem

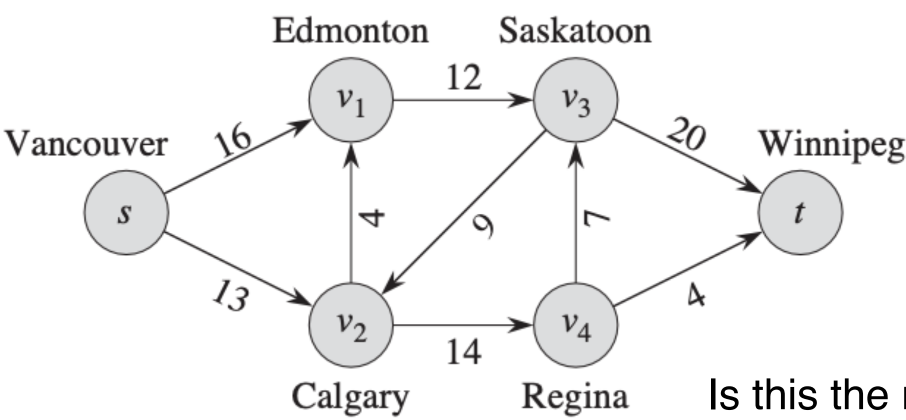
Given a directed graph $G = (V, E)$ where:

- If $(u, v) \in E$ then $(v, u) \notin E$ (antiparallel edge, we will fix this!)
- Each $(u, v) \in E$, has a capacity $c(u, v) \in \mathbb{R}^+$
- There is a source vertex s and a sink vertex t
- A flow along any edge $f(u, v)$ must satisfy:
  - Capacity constraint: $0 \leq f(u, v) \leq c(u, v)$
  - Flow conservation: $\forall u \in V - \{s, t\} \sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v)$
  - Goal: $max \sum_{v \in adj^+[s]} f(s, v)$, or $max \sum_{v \in adj^-[t]} f(v, t)$
  - Note: adj^+ for outgoing edges, adj^- incoming edges

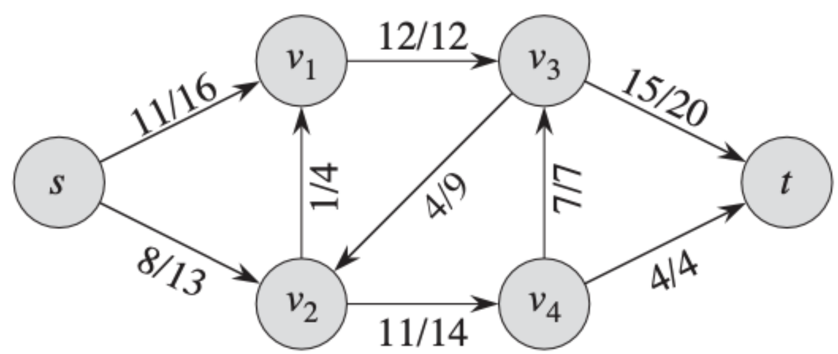
Graph Algorithms (Chapter 26)
Maximum Flow Problem

Example

- A flow along any edge $f(u,v)$ must satisfy:
  - Capacity constraint: $0 \leq f(u,v) \leq c(u,v)$
  - Flow conservation: $\forall u \in V - \{s,t\} \sum_{v \in V} f(v,u) = \sum_{v \in V} f(u,v)$
  - Goal: $max \sum_{v \in adj^+[s]} f(s,v)$, or $max \sum_{v \in adj^-[t]} f(v,t)$
  - Note: adj^+ for outgoing edges, adj^- incoming edges



Is this the maximum flow?

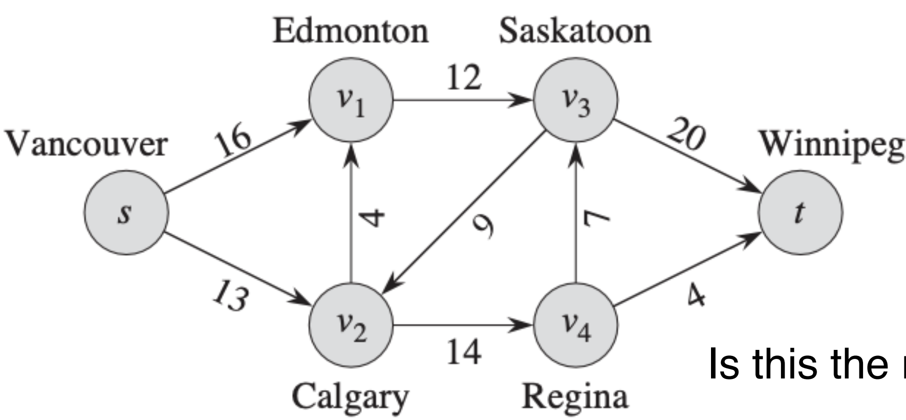


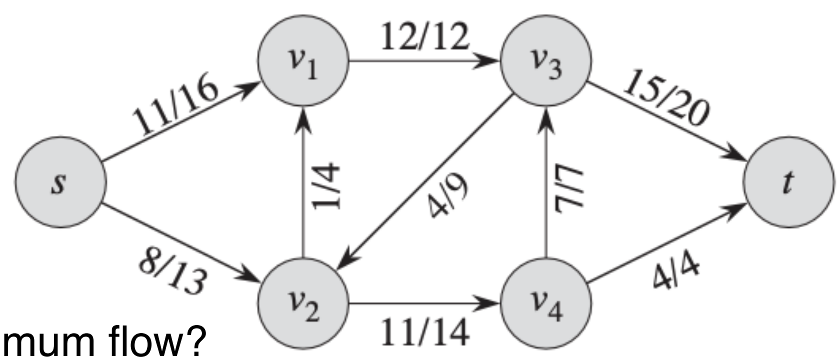
Graph Algorithms (Chapter 26)
Maximum Flow Problem

University of Stavanger

Example

- A flow along any edge $f(u,v)$ must satisfy:
  - Capacity constraint: $0 \le f(u,v) \le c(u,v)$
  - Flow conservation: $\forall u \in V - \{s,t\} \; \sum_{v \in V} f(v,u) = \sum_{v \in V} f(u,v)$
  - Goal: $max \sum_{v \in adj^{+}[s]} f(s,v)$, or $max \sum_{v \in adj^{-}[t]} f(v,t)$
  - Note: adj^{+} for outgoing edges, adj^{-} incoming edges

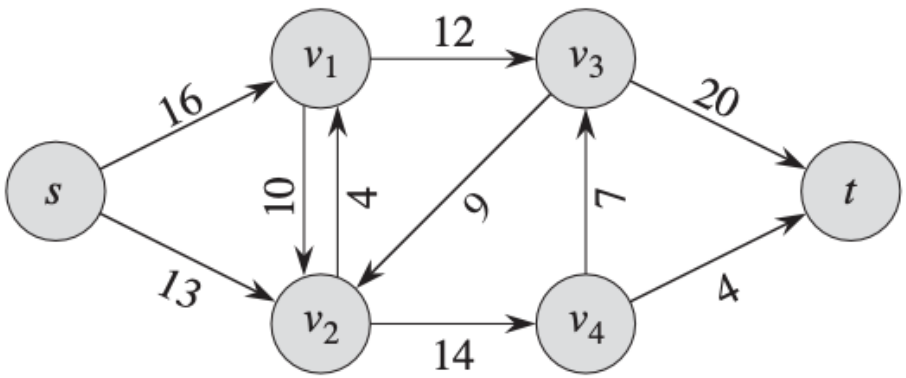




Is this the maximum flow?
NO!

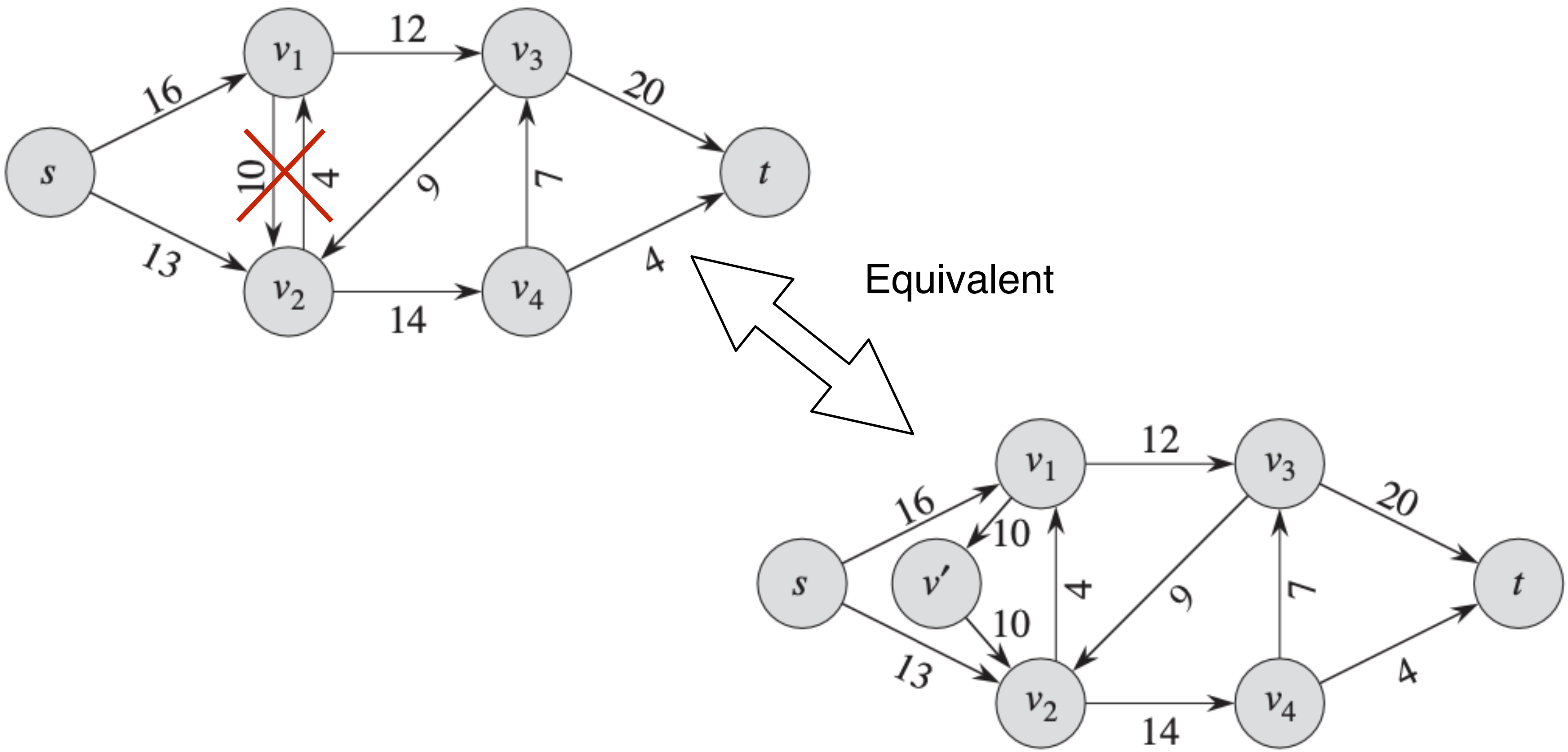
Graph Algorithms (Chapter 26)
Antiparallel edges

If $(u, v) \in E$ then $(v, u) \notin E$ (antiparallel edge, we will fix this!)

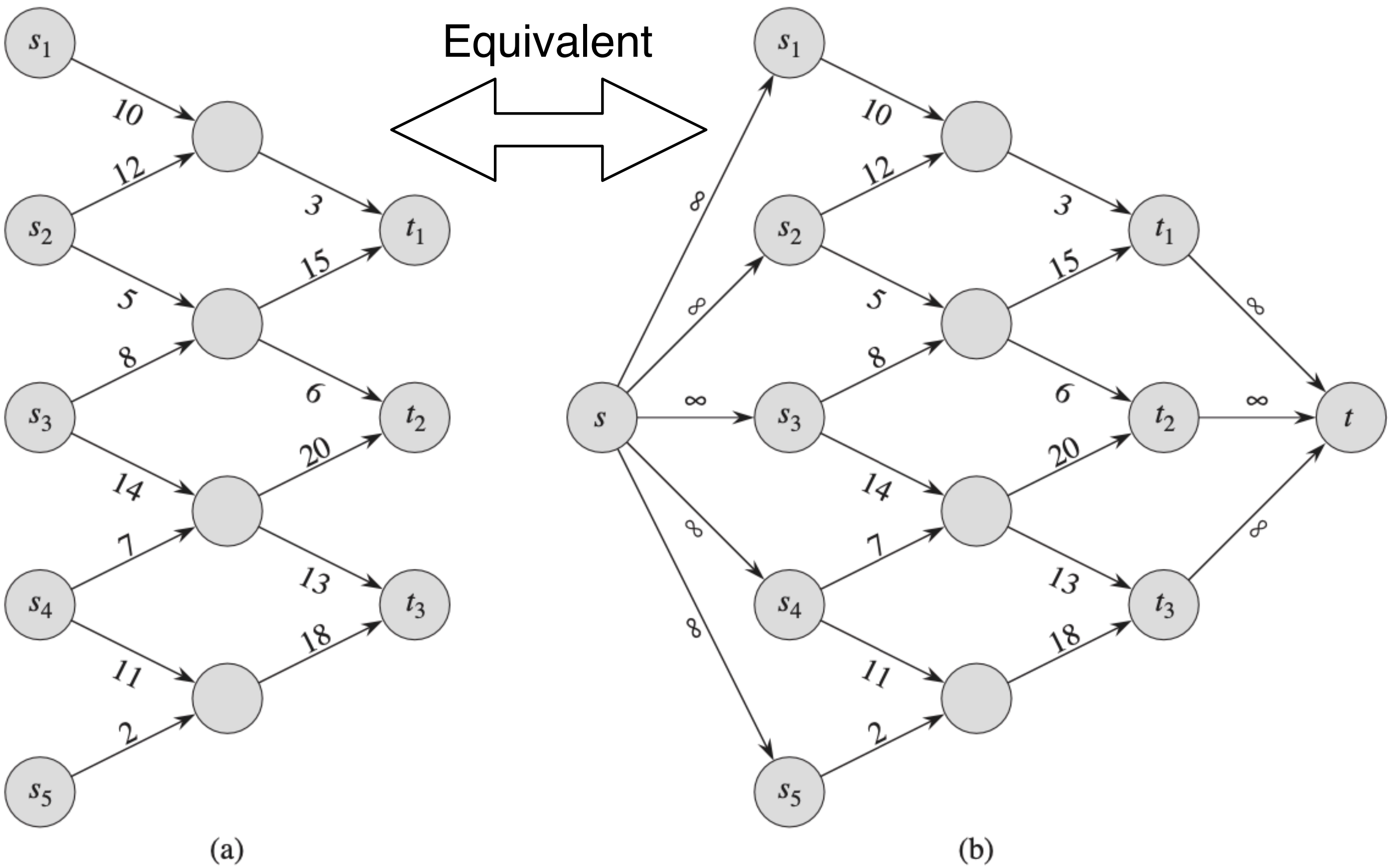


Graph Algorithms (Chapter 26)
Antiparallel edges

If $(u, v) \in E$ then $(v, u) \notin E$ (antiparallel edge, we will fix this!)



Graph Algorithms (Chapter 26)
Multiple sources and sinks



Graph Algorithms (Chapter 26)
Ford-Fukerson Method

Graph Algorithms (Chapter 26)
Ford-Fukerson Method

University of Stavanger

Intuition

As long as there is a path from source to sink on the “residual network” increase the flow (augmenting path) on that path.

Residual network?

Augmenting path?

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

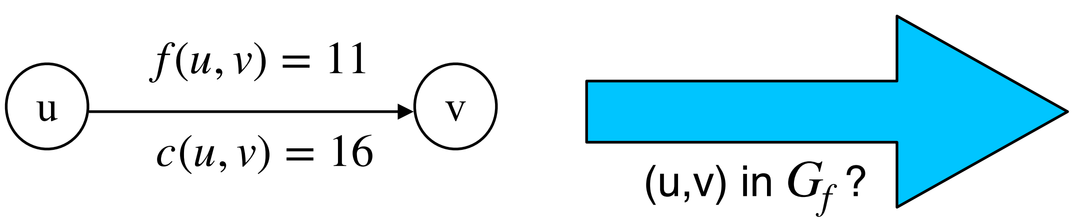
Graph Algorithms (Chapter 26)
Ford-Fukerson Method

University of Stavanger

Residual network

Given a network G and a flow f, the residual network G_f consists of edges representing how we can change the flow on them.

- G_f may contain edges from G with residual capacity $c_f(u,v) = c(u,v) - f(u,v)$ if $c_f(u,v)$ is positive
- G_f may contain edges that are not in G to represent a possible decrease of flow with $c_f(u,v) = f(u,v)$. Explanation!



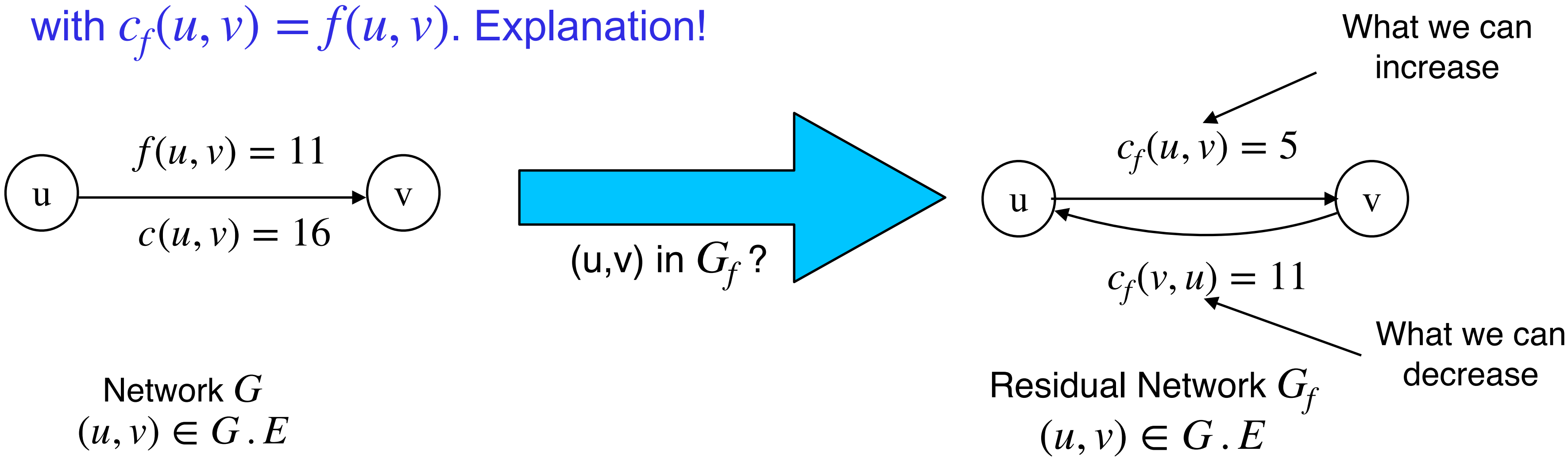
Network G
$(u,v) \in G.E$

Graph Algorithms (Chapter 26)
Ford-Fukerson Method

Residual network

Given a network G and a flow f, the residual network G_f consists of edges representing how we can change the flow on them.

- G_f may contain edges from G with residual capacity $c_f(u,v) = c(u,v) - f(u,v)$ if $c_f(u,v)$ is positive
- G_f may contain edges that are not in G to represent a possible decrease of flow with $c_f(u,v) = f(u,v)$. Explanation!

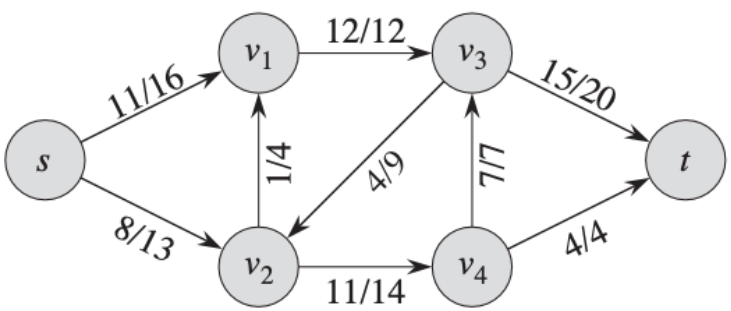


Graph Algorithms (Chapter 26)
Ford-Fukerson Method

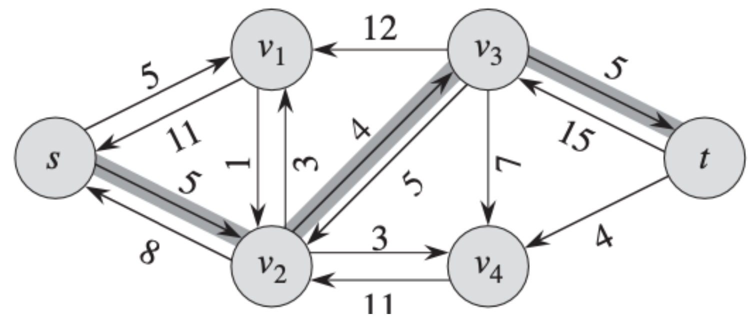
University of Stavanger

Residual network Example

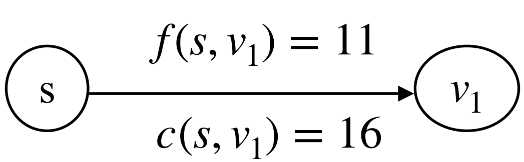
Network G



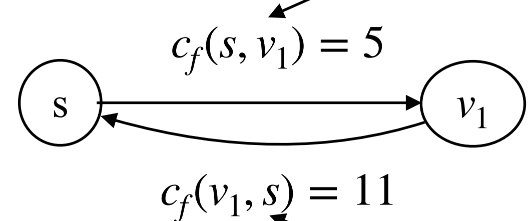
Residual Network G_f



What we can increase



(s, v_1) update



What we can decrease

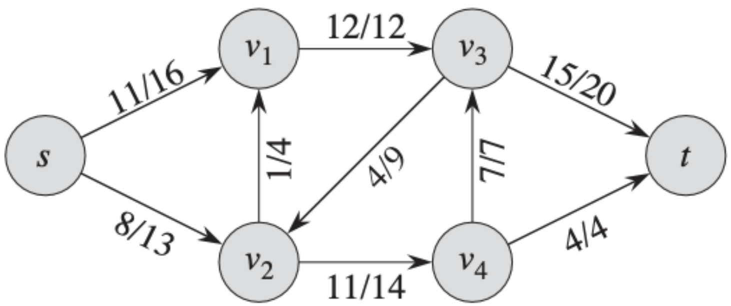
Network G
$(u, v) \in G.E$

Residual Network G_f
$(u, v) \in G.E$

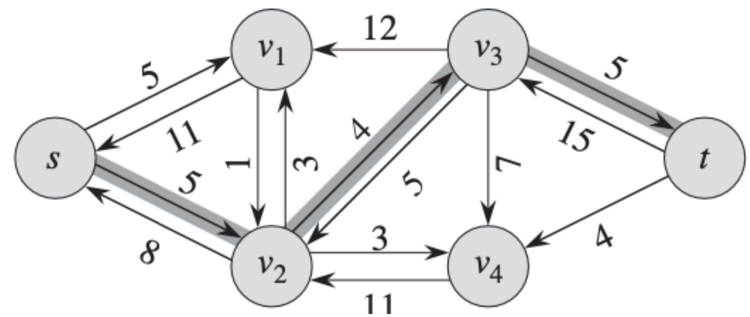
Graph Algorithms (Chapter 26)
Ford-Fukerson Method

Residual network Example

Network G

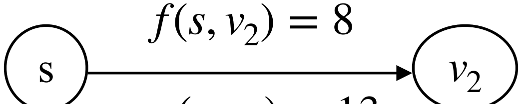


Residual Network G_f



What we can increase

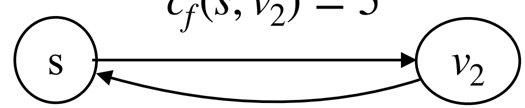
$f(s, v_2) = 8$



$c(s, v_2) = 13$

(s, v_2) update

$c_f(s, v_2) = 5$



$c_f(v_2, s) = 8$

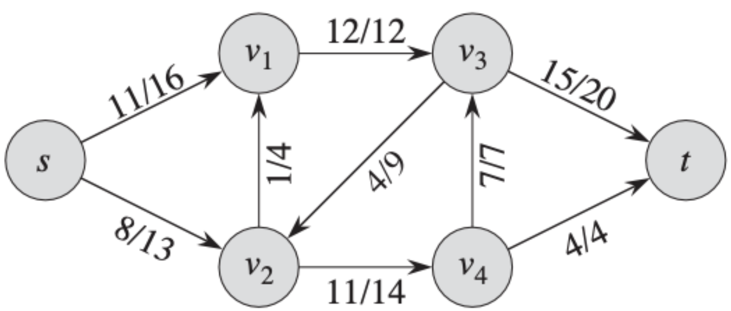
What we can decrease

Network G
$(u, v) \in G.E$

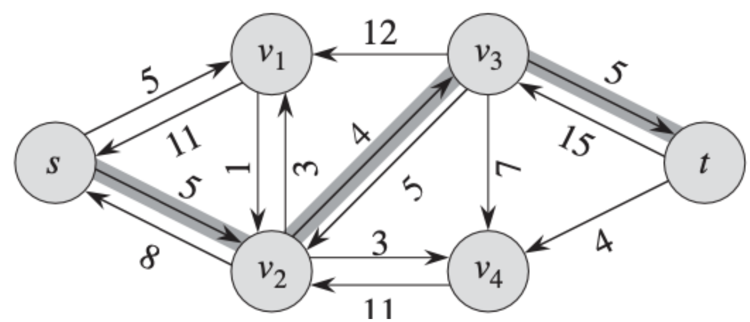
Residual Network G_f
$(u, v) \in G.E$

Residual network Example

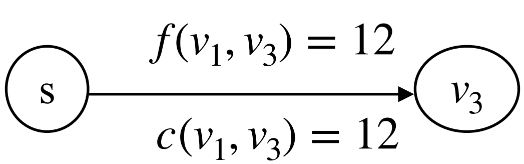
Network G



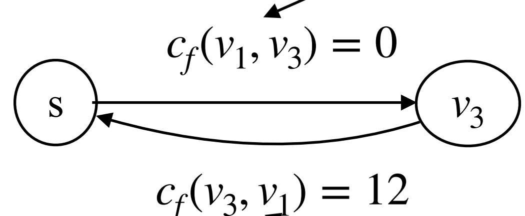
Residual Network G_f



What we can increase



(v_1, v_3) update



What we can decrease

Network G
$(u, v) \in G.E$

Residual Network G_f
$(u, v) \in G.E$

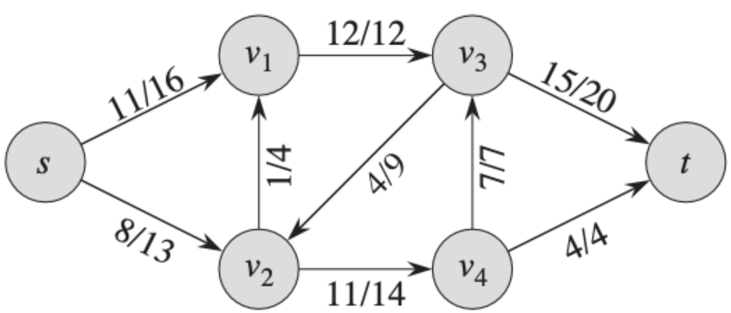
Graph Algorithms (Chapter 26)
Ford-Fukerson Method

Augmenting paths

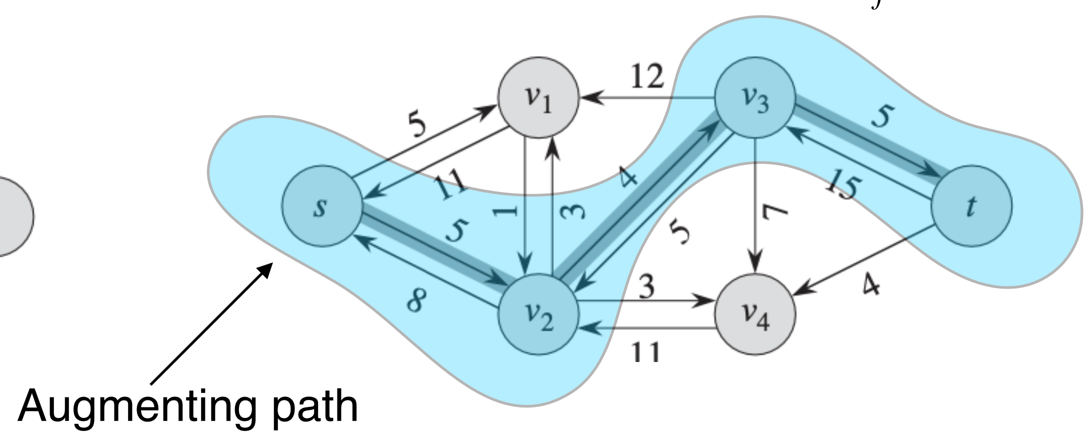
An augmenting path p is a simple path (no repeated vertices) from s to t in the residual network G_f. The residual capacity $c_f(p)$ is the maximum flow by which we can increase the flow on each edge in p.

How much $c_f(p)$ can be without violating the capacity constraints?

Network G



Residual Network G_f



Graph Algorithms (Chapter 26) Ford-Fukerson Method

University of Stavanger

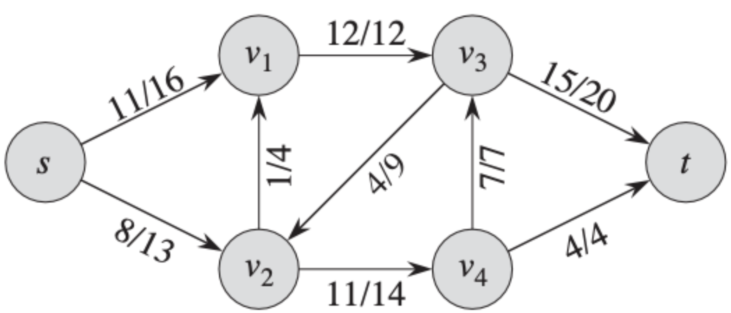
Augmenting paths

An augmenting path p is a simple path (no repeated vertices) from s to t in the residual network G_f. The residual capacity $c_f(p)$ is the maximum flow by which we can increase the flow on each edge in p.

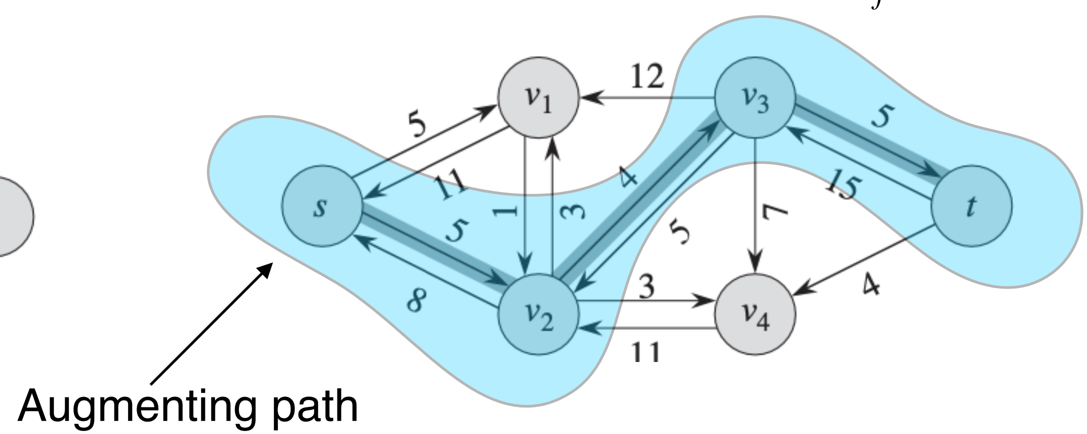
How much $c_f(p)$ can be without violating the capacity constraints?

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

Network G



Residual Network G_f



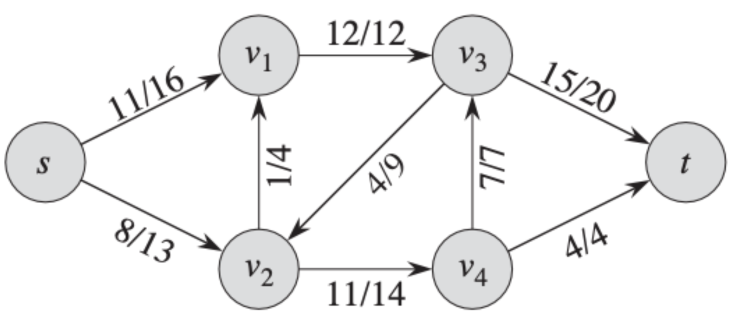
Graph Algorithms (Chapter 26)
Ford-Fukerson Method

Augmenting paths

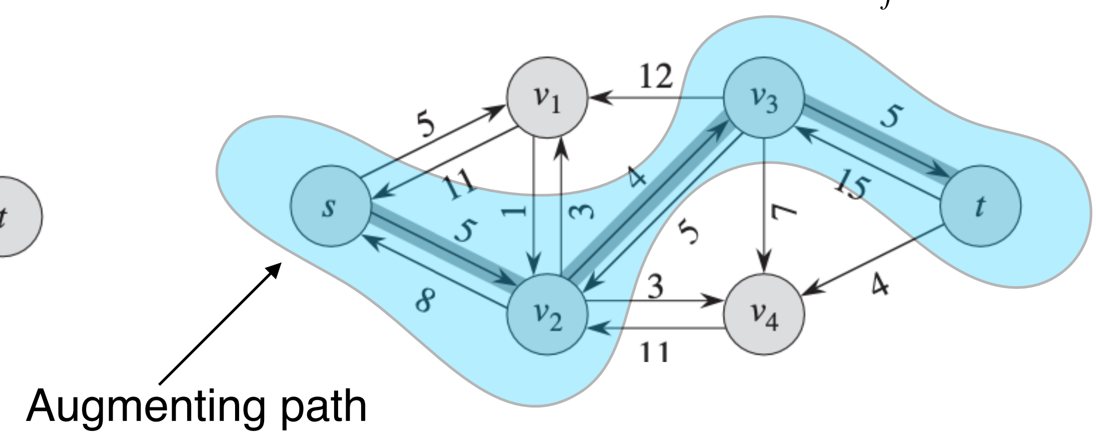
We can thus say $\forall (u, v) \in p, f(u, v) += c_f(p)$ will be possible.

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

Network G



Residual Network G_f



Graph Algorithms (Chapter 26)
Ford-Fukerson Method

University of Stavanger

Augmenting paths useful definitions (related to coming algorithms)

Lemma 26.1

Let $G = (V, E)$ be a flow network with source s and sink t, and let f be a flow in G. Let G_f be the residual network of G induced by f, and let f' be a flow in G_f. Then the function $f \uparrow f'$ defined in equation (26.4) is a flow in G with value $|f \uparrow f'| = |f| + |f'|$.

Lemma 26.2

Let $G = (V, E)$ be a flow network, let f be a flow in G, and let p be an augmenting path in G_f. Define a function $f_p : V \times V \to \mathbb{R}$ by

$$f_p(u, v) = \begin{cases} c_f(p) & \text{if } (u, v) \text{ is on } p \,, \\ 0 & \text{otherwise} \,. \end{cases} \tag{26.8}$$

Then, f_p is a flow in G_f with value $|f_p| = c_f(p) > 0$. ∎

Corollary 26.3

Let $G = (V, E)$ be a flow network, let f be a flow in G, and let p be an augmenting path in G_f. Let f_p be defined as in equation (26.8), and suppose that we augment f by f_p. Then the function $f \uparrow f_p$ is a flow in G with value $|f \uparrow f_p| = |f| + |f_p| > |f|$.

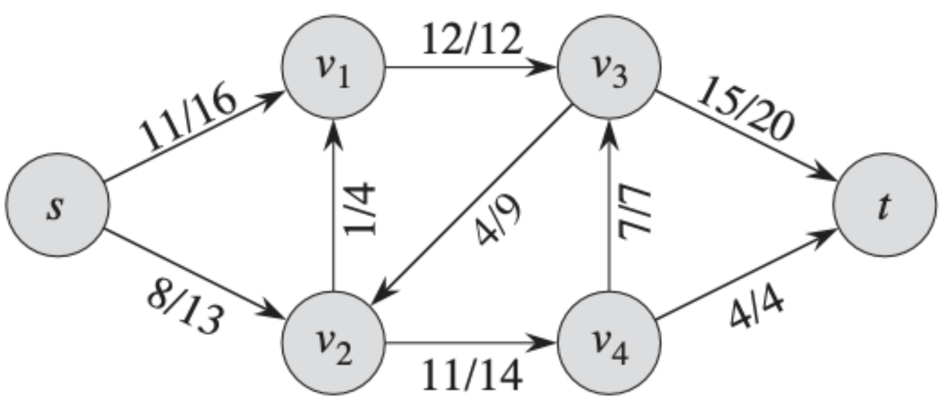
Proof Immediate from Lemmas 26.1 and 26.2. ∎

Graph Algorithms (Chapter 26)
Ford-Fukerson Method

University of Stavanger

Cuts of flow networks

- A $cut(S,T)$ of $G=(V,E)$ is a partition of V into S and T, such that $s \in S, t \in T$
- The net flow across the cut $f(S,T) = \sum_{u \in S, v \in T} f(u,v) - \sum_{u \in S, v \in T} f(v,u)$
- The cat capacity $c(S,T) = \sum_{u \in S, v \in T} c(u,v)$



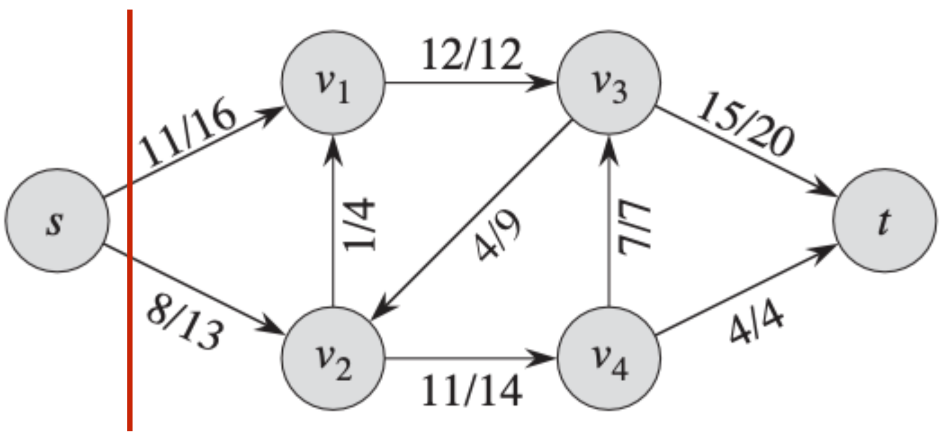
Graph Algorithms (Chapter 26)
Ford-Fukerson Method

University of Stavanger

Cuts of flow networks

- A $cut(S, T)$ of $G = (V, E)$ is a portions of V into S and T, such that $s \in S, t \in T$
- The net flow across the cut $f(S, T) = \sum_{u \in S, v \in T} f(u, v) - \sum_{u \in S, v \in T} f(v, u)$
- The cat capacity $c(S, T) = \sum_{u \in S, v \in T} c(u, v)$

$\leftarrow S \mid T \rightarrow$



Example cut 1

$S = \{?\}, T = \{?\}$

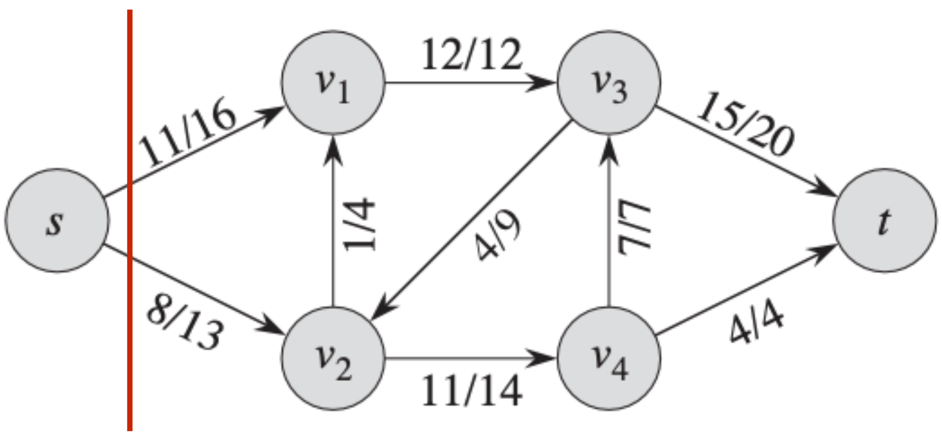
$f(S, T) = ?$

$c(S, T) = ?$

Cuts of flow networks

- A $cut(S, T)$ of $G = (V, E)$ is a portions of V into S and T, such that $s \in S, t \in T$
- The net flow across the cut $f(S, T) = \sum_{u \in S, v \in T} f(u, v) - \sum_{u \in S, v \in T} f(v, u)$
- The cut capacity $c(S, T) = \sum_{u \in S, v \in T} c(u, v)$

$\leftarrow S \mid T \rightarrow$



Example cut 1

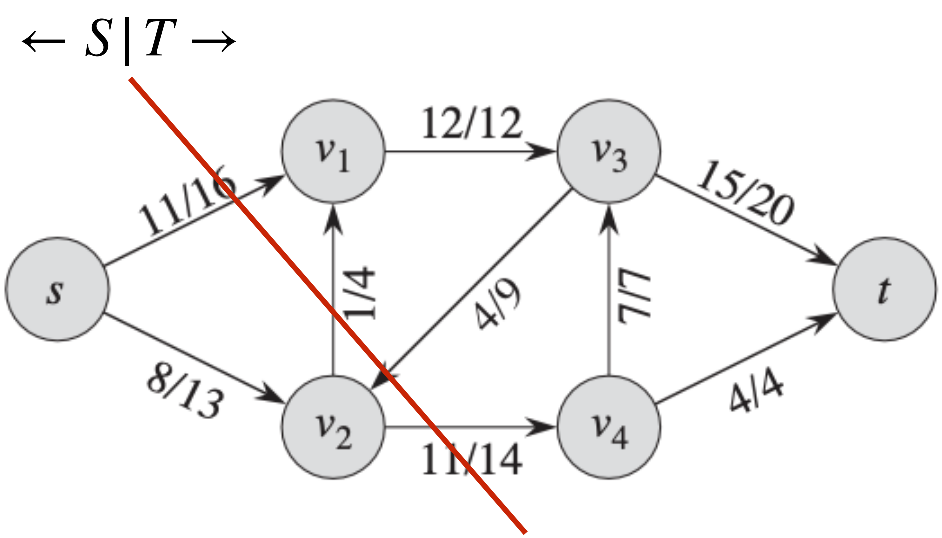
$S = \{s\}, T = \{v_1, v_2, v_3, v_4, t\}$

$f(S, T) = 11 + 8 = 19$

$c(S, T) = 16 + 13 = 29$

Cuts of flow networks

- A $cut(S, T)$ of $G = (V, E)$ is a portions of V into S and T, such that $s \in S, t \in T$
- The net flow across the cut $f(S, T) = \sum_{u \in S, v \in T} f(u, v) - \sum_{u \in S, v \in T} f(v, u)$
- The cat capacity $c(S, T) = \sum_{u \in S, v \in T} c(u, v)$



Example cut 2

$S = \{?\}, T = \{?\}$

$f(S, T) = ?$

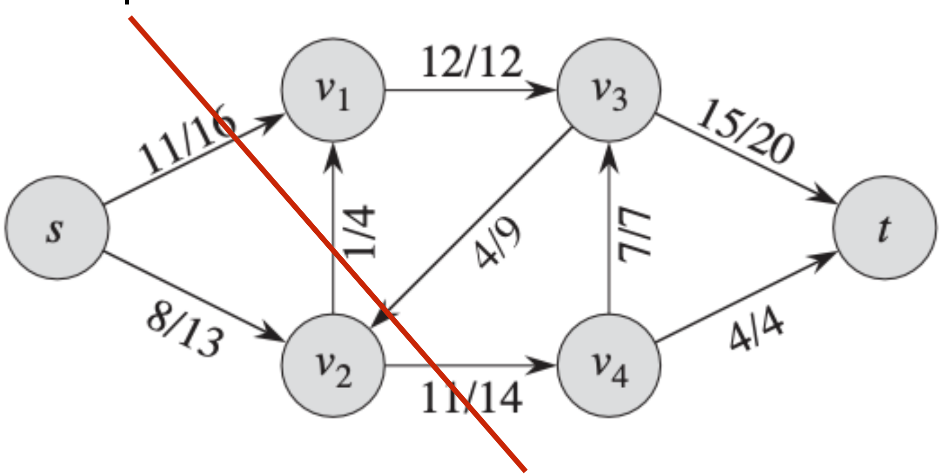
$c(S, T) = ?$

University of Stavanger

Cuts of flow networks

- A $cut(S, T)$ of $G = (V, E)$ is a portions of V into S and T, such that $s \in S, t \in T$
- The net flow across the cut $f(S, T) = \sum_{u \in S, v \in T} f(u, v) - \sum_{u \in S, v \in T} f(v, u)$
- The cat capacity $c(S, T) = \sum_{u \in S, v \in T} c(u, v)$

$\leftarrow S \,|\, T \rightarrow$



Example cut 2

$S = \{s, v_2\}, T = \{v_1, v_3, v_4, t\}$

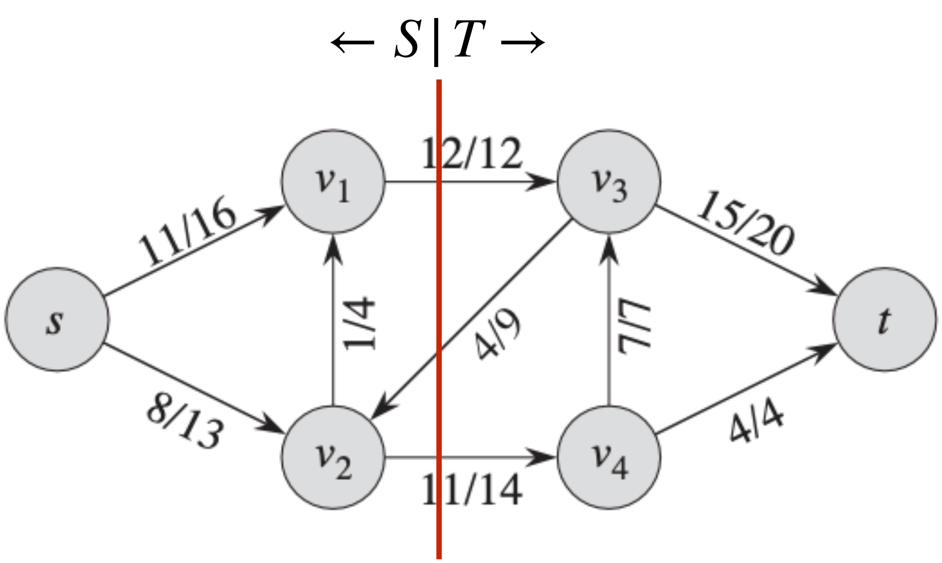
$f(S, T) = 11 + 1 + 11 - 4 = 19$

$c(S, T) = 16 + 14 + 4 = 34$

University of Stavanger

Cuts of flow networks

- A $cut(S, T)$ of $G = (V, E)$ is a portions of V into S and T, such that $s \in S, t \in T$
- The net flow across the cut $f(S, T) = \sum_{u \in S, v \in T} f(u, v) - \sum_{u \in S, v \in T} f(v, u)$
- The cat capacity $c(S, T) = \sum_{u \in S, v \in T} c(u, v)$



Example cut 3

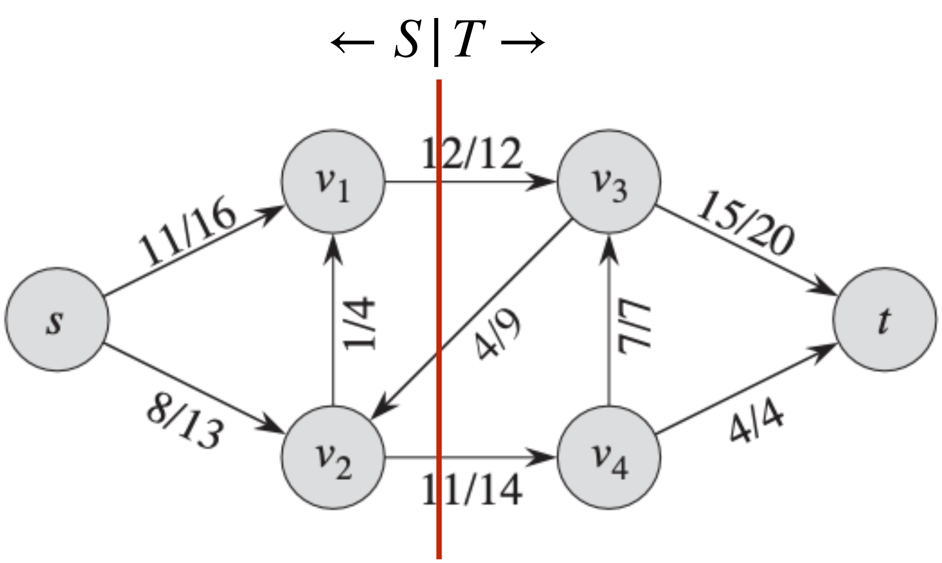
$S = \{?\}, T = \{?\}$

$f(S, T) = ?$

$c(S, T) = ?$

Cuts of flow networks

- A $cut(S,T)$ of $G=(V,E)$ is a portions of V into S and T, such that $s \in S, t \in T$
- The net flow across the cut $f(S,T) = \sum_{u \in S, v \in T} f(u,v) - \sum_{u \in S, v \in T} f(v,u)$
- The cat capacity $c(S,T) = \sum_{u \in S, v \in T} c(u,v)$



Example cut 3

$$S = \{s, v_1, v_2\}, T = \{v_3, v_4, t\}$$

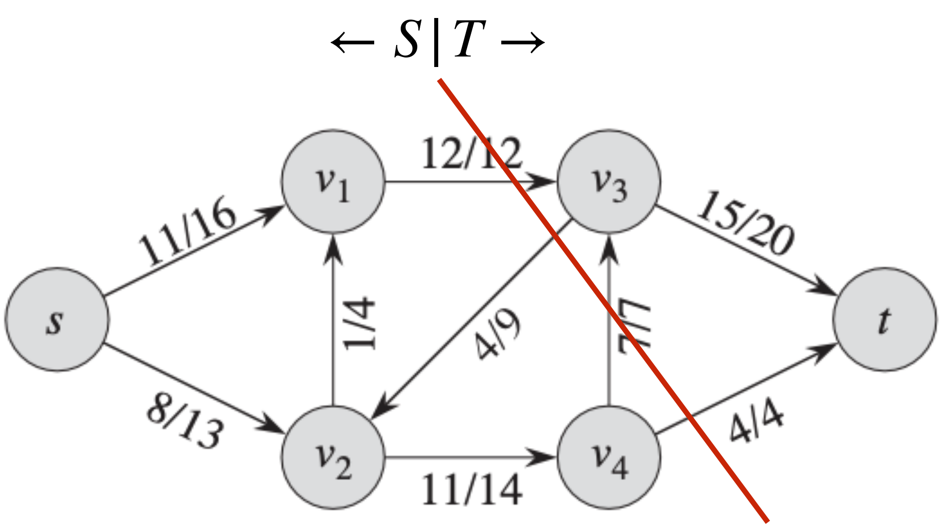
$$f(S,T) = 12 + 11 - 4 = 19$$

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

University of Stavanger

Cuts of flow networks

- A $cut(S, T)$ of $G = (V, E)$ is a portions of V into S and T, such that $s \in S, t \in T$
- The net flow across the cut $f(S,T) = \sum_{u \in S, v \in T} f(u,v) - \sum_{u \in S, v \in T} f(v,u)$
- The cat capacity $c(S,T) = \sum_{u \in S, v \in T} c(u,v)$



Example cut 3

$S = \{?\}, T = \{?\}$

$f(S,T) = ?$

$c(S,T) = ?$

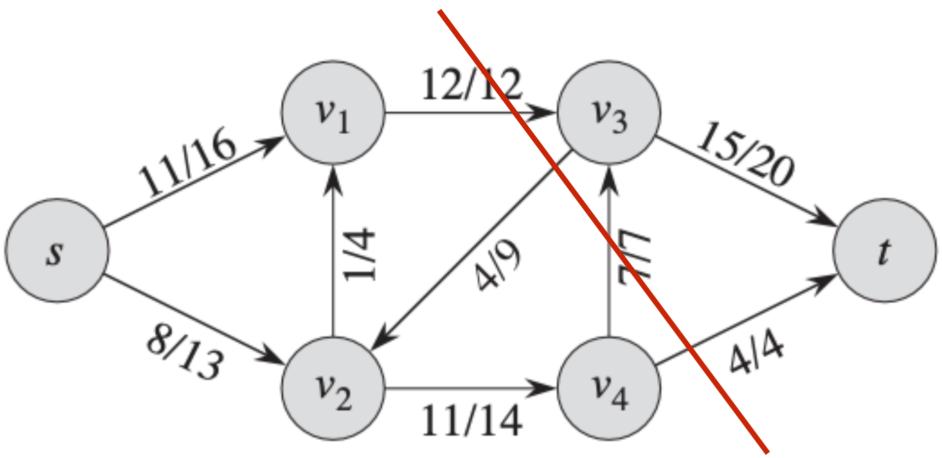
Graph Algorithms (Chapter 26)
Ford-Fukerson Method

University of Stavanger

Cuts of flow networks

- A $cut(S,T)$ of $G=(V,E)$ is a portions of V into S and T, such that $s \in S, t \in T$
- The net flow across the cut $f(S,T) = \sum_{u \in S, v \in T} f(u,v) - \sum_{u \in S, v \in T} f(v,u)$
- The cat capacity $c(S,T) = \sum_{u \in S, v \in T} c(u,v)$

$\leftarrow S \,|\, T \rightarrow$



Example cut 4

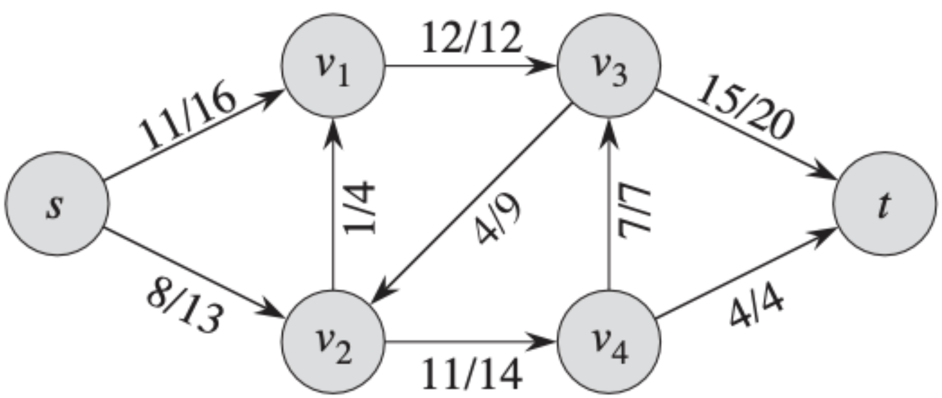
$$S = \{s, v_1, v_2, v_4\}, T = \{v_4, t\}$$

$$f(S,T) = 12 + 7 + 4 - 4 = 19$$

$$c(S,T) = 12 + 7 + 4 = 23$$

Cuts of flow networks

What is the maximum flow that a network can have?



Graph Algorithms (Chapter 26)
Ford-Fukerson Method

Cuts of flow networks

The maximum flow that a network can have is the minimum capacity cut!

Corollary 26.5

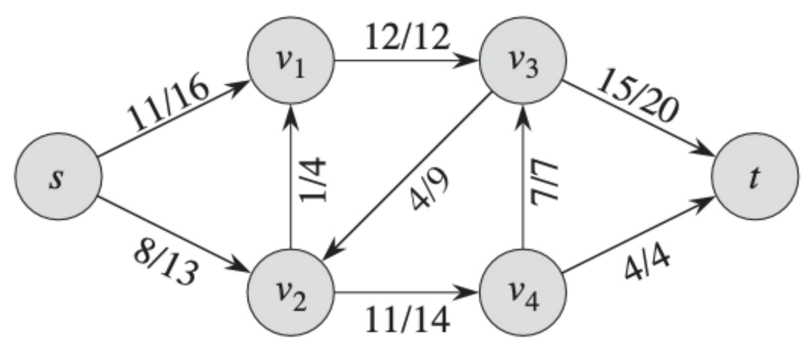
The value of any flow f in a flow network G is bounded from above by the capacity of any cut of G.

Theorem 26.6 (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.

MUST THEN BE THE MINIMUM CAPACITY CUT !!!



Graph Algorithms (Chapter 26)
Ford-Fukerson Method

University of Stavanger

Cuts of flow networks

The maximum flow that a network can have is the minimum capacity cut!

Corollary 26.5

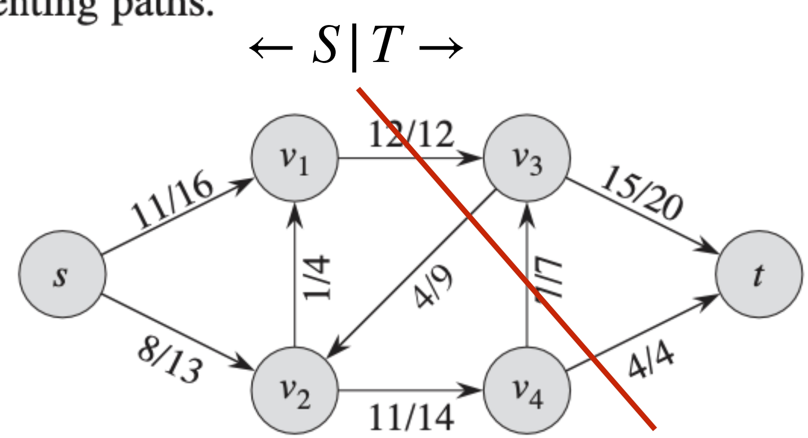
The value of any flow f in a flow network G is bounded from above by the capacity of any cut of G.

Theorem 26.6 (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.

MUST THEN BE THE MINIMUM CAPACITY CUT !!!



Graph Algorithms (Chapter 26)
Ford-Fulkerson Algorithm

FORD-FULKERSON(G, s, t)

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

Graph Algorithms (Chapter 26)
Ford-Fulkerson Algorithm

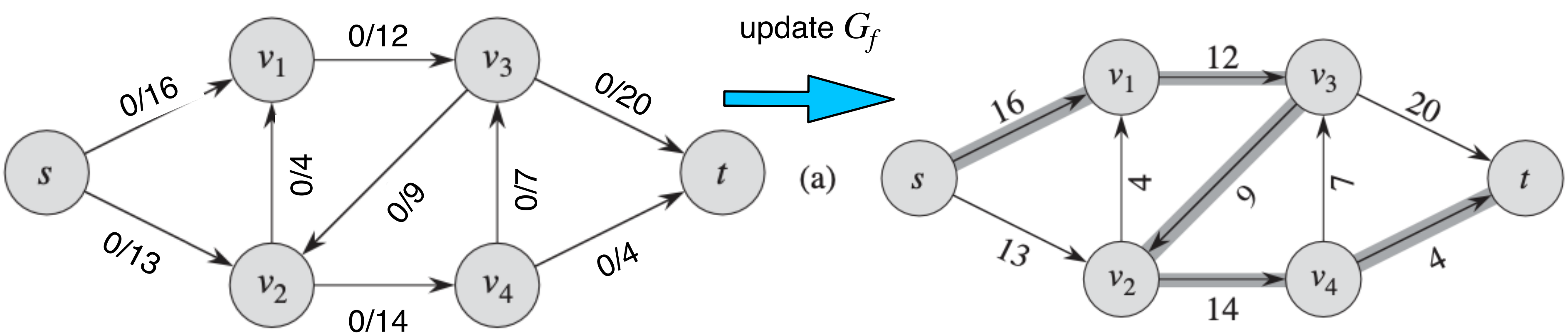
With DFS

FORD-FULKERSON(G, s, t)

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

Network G

Residual G_f and augmented path



Graph Algorithms (Chapter 26)
Ford-Fulkerson Algorithm

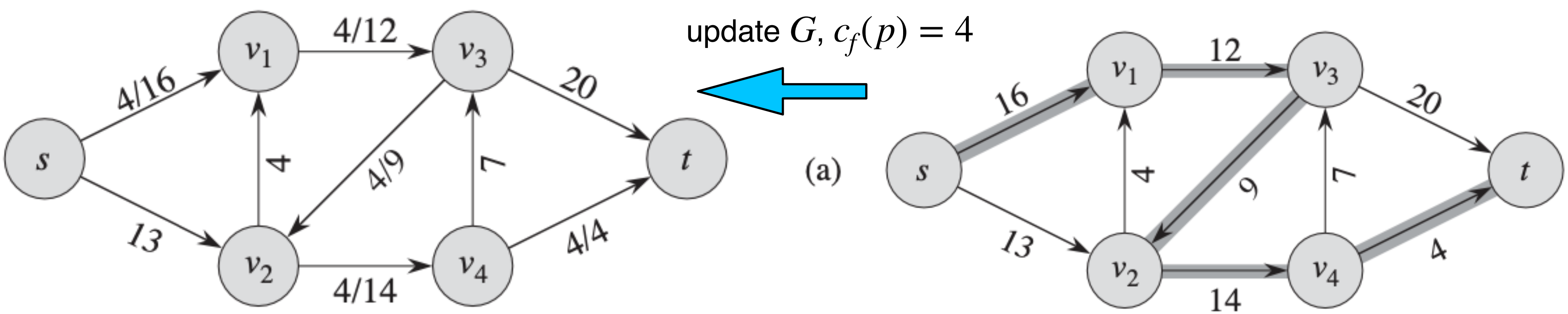
FORD-FULKERSON(G, s, t)

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

With DFS

Network G

Residual G_f and augmented path



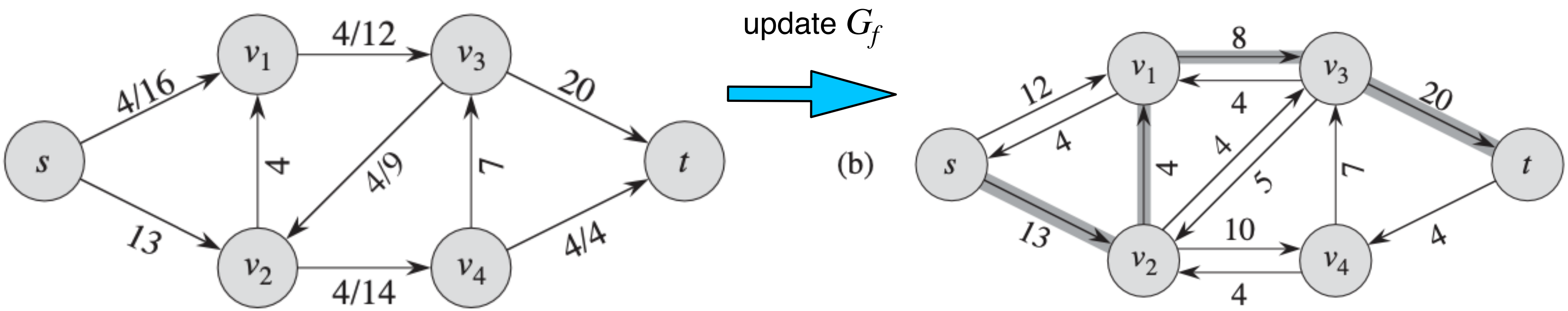
Graph Algorithms (Chapter 26)
Ford-Fulkerson Algorithm

FORD-FULKERSON(G, s, t)

```
for each edge (u, v) ∈ G.E
    (u, v).f = 0
while there exists a path p from s to t in the residual network G_f
    c_f(p) = min{c_f(u, v) : (u, v) is in p}
    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)
```

With DFS

Network G — Residual G_f and augmented path

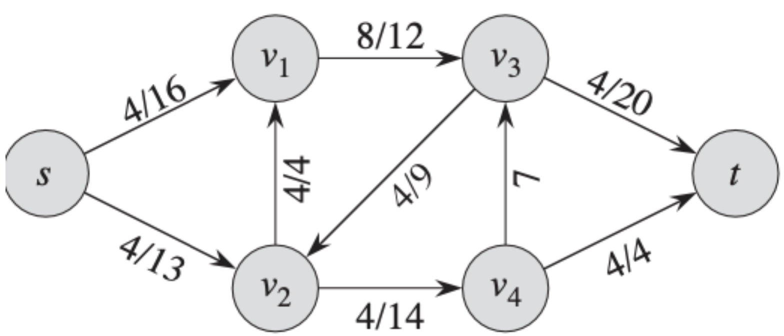


Graph Algorithms (Chapter 26)
Ford-Fulkerson Algorithm

With DFS

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

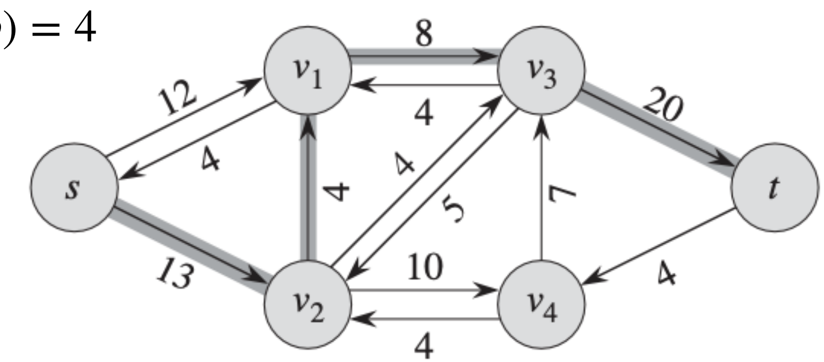
Network G



update G, $c_f(p) = 4$

Residual G_f and augmented path

(b)

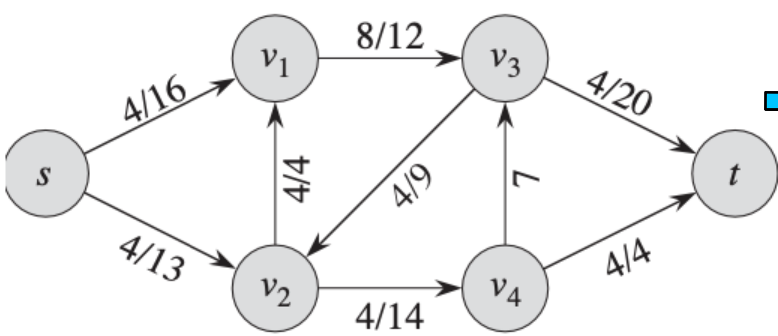


Graph Algorithms (Chapter 26) Ford-Fulkerson Algorithm


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

With DFS

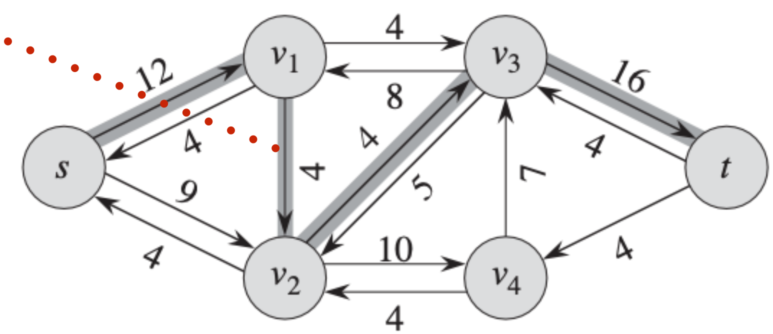
Network G



update G_f

Residual G_f and augmented path

(c)



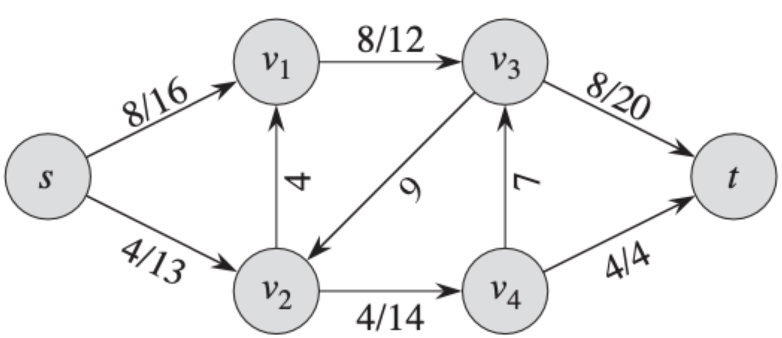
Graph Algorithms (Chapter 26)
Ford-Fulkerson Algorithm

FORD-FULKERSON(G, s, t)

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

With DFS

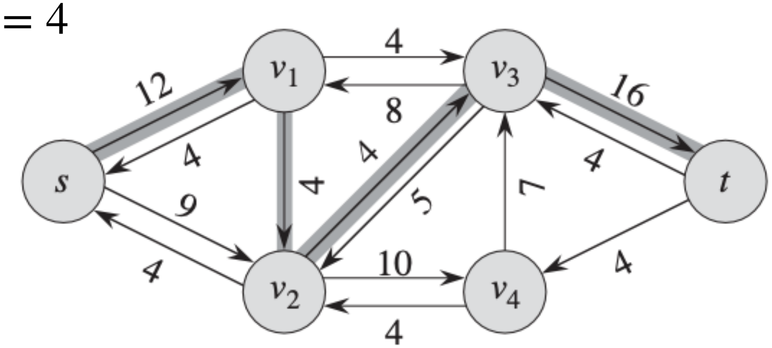
Network G



Update G, $c_f(p) = 4$

Residual G_f and augmented path

(c)



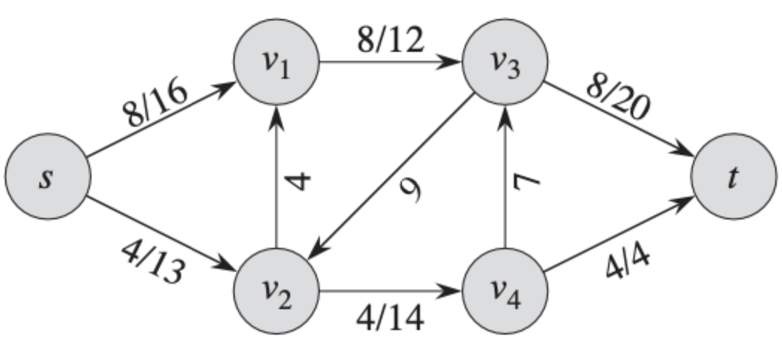
Graph Algorithms (Chapter 26)
Ford-Fulkerson Algorithm

FORD-FULKERSON(G, s, t)

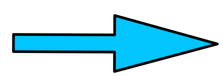
for each edge $(u, v) \in G.E$
    $(u, v).f = 0$
while there exists a path p from s to t in the residual network G_f
    $c_f(p) = \min\{c_f(u, v) : (u, v) \text{ is in } p\}$
    for each edge (u, v) in p
        if $(u, v) \in E$
            $(u, v).f = (u, v).f + c_f(p)$
        else $(v, u).f =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

With DFS

Network G

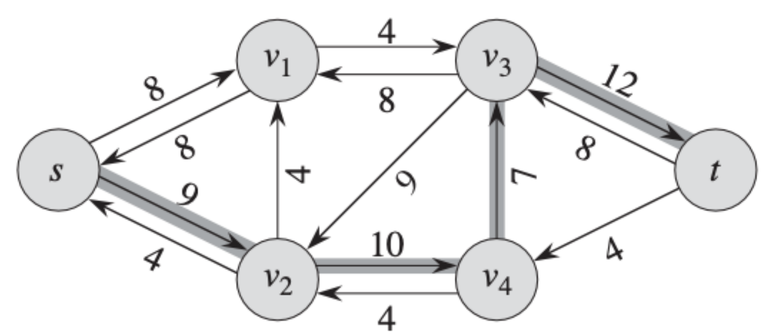


update G_f



(d)

Residual G_f and augmented path



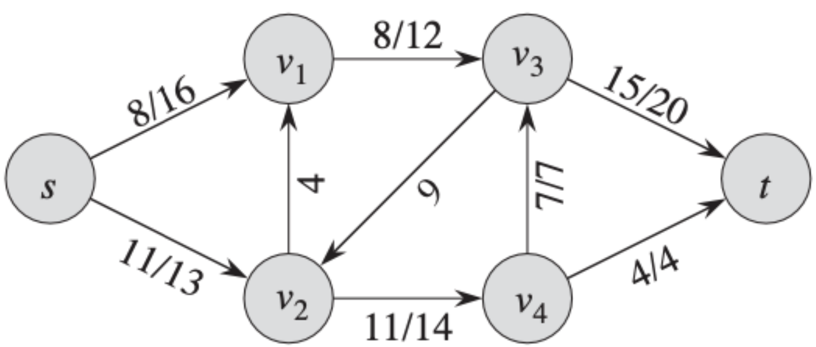
Graph Algorithms (Chapter 26)
Ford-Fulkerson Algorithm

With DFS

FORD-FULKERSON(G, s, t)

```
for each edge (u, v) ∈ G.E
    (u, v).f = 0
while there exists a path p from s to t in the residual network G_f
    c_f(p) = min {c_f(u, v) : (u, v) is in p}
    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)
```

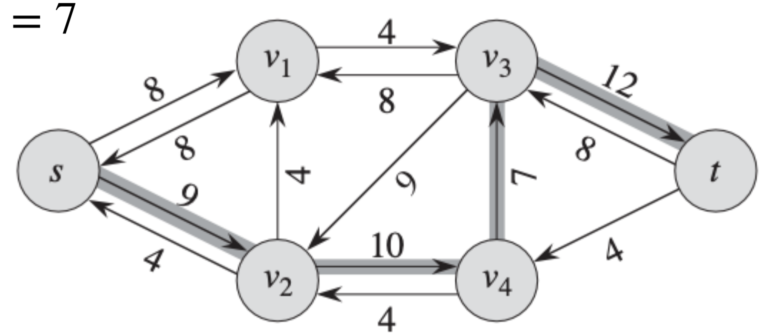
Network G



Update G, $c_f(p) = 7$

Residual G_f and augmented path

(d)



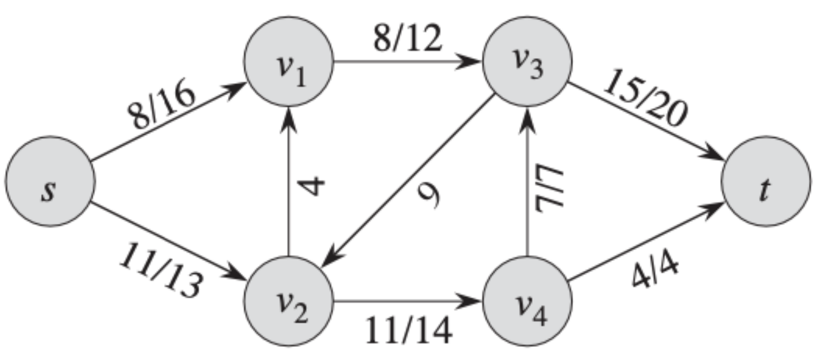
Graph Algorithms (Chapter 26)
Ford-Fulkerson Algorithm

FORD-FULKERSON(G, s, t)

```
for each edge (u, v) ∈ G.E
    (u, v).f = 0
while there exists a path p from s to t in the residual network G_f
    c_f(p) = min {c_f(u, v) : (u, v) is in p}
    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)
```

With DFS

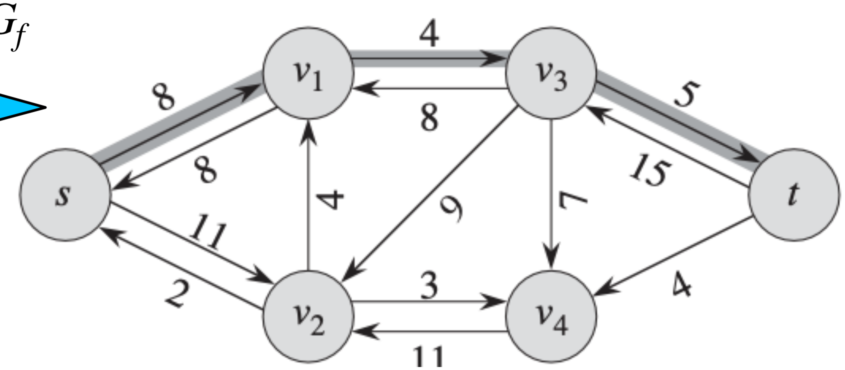
Network G



update G_f

Residual G_f and augmented path

(e)

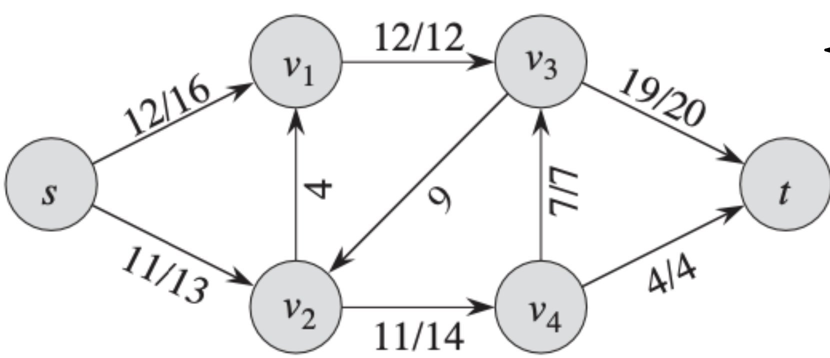


Graph Algorithms (Chapter 26)
Ford-Fulkerson Algorithm

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

With DFS

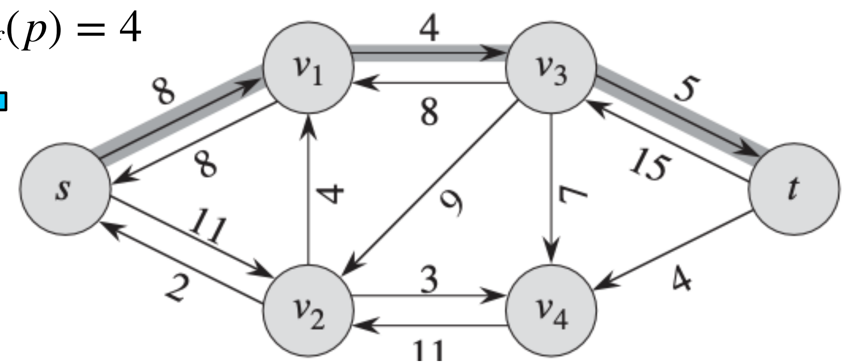
Network G



Update G, $c_f(p) = 4$

Residual G_f and augmented path

(e)

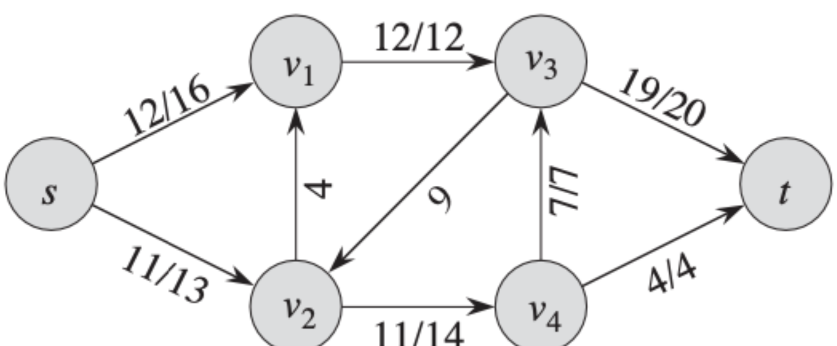


University of Stavanger

With DFS

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

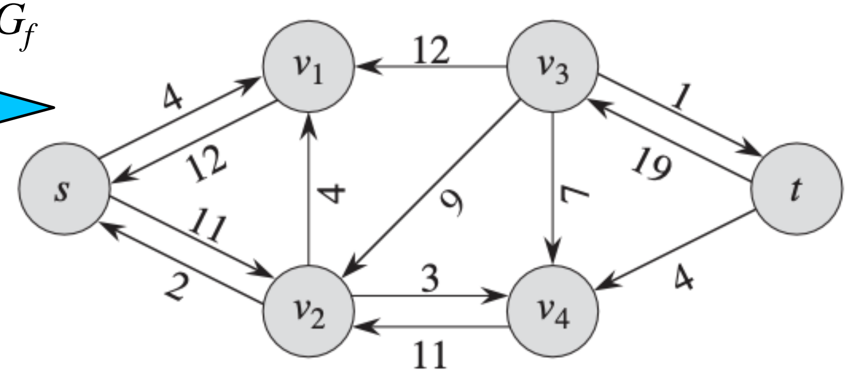
Network G



update G_f

Residual G_f and augmented path

(f)



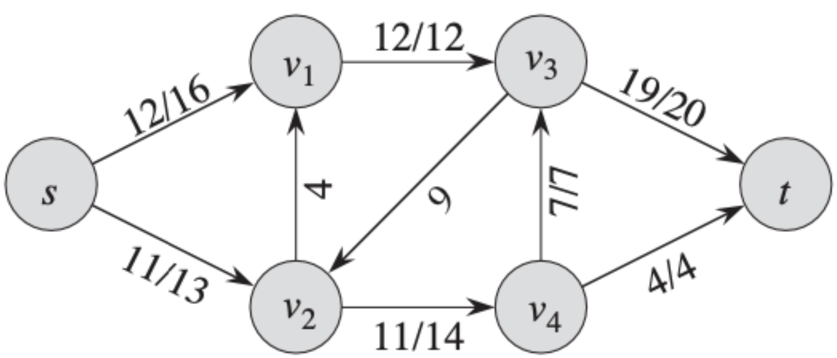
No more augmented paths (done!)

Graph Algorithms (Chapter 26)
Ford-Fulkerson Algorithm

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

With DFS

Network G



Test

What is minimum capacity cut?

What is the maximum flow?

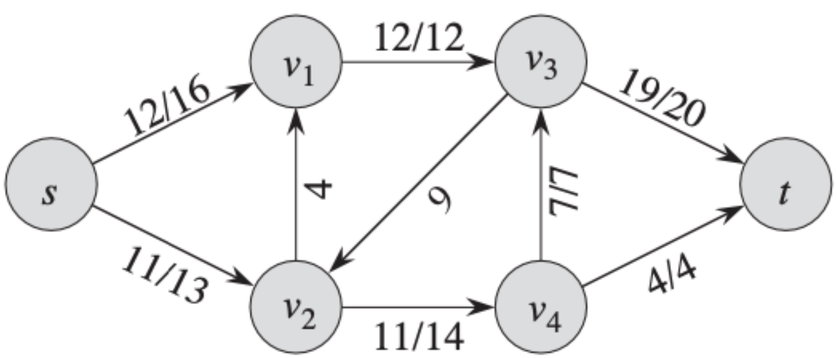
Graph Algorithms (Chapter 26)
Ford-Fulkerson Algorithm

FORD-FULKERSON(G, s, t)

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

With DFS

Network G



Runtime complexity

- Time to find a path $O(E + V) = O(E)$, when $|E| \geq |V|$ using DFS
- if $|f^*|$ notion of maximum flow / max step increase

$$O(E \cdot |f^*|)$$

Graph Algorithms (Chapter 26) Ford-Fulkerson Algorithm

University of Stavanger

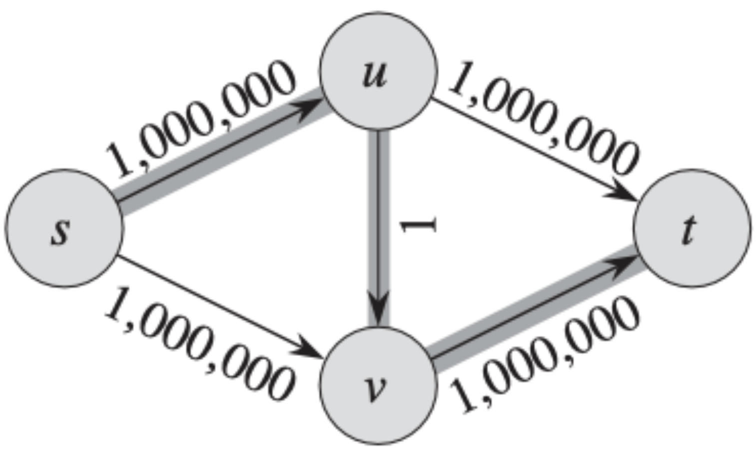
```
FORD-FULKERSON(G, s, t)
for each edge (u, v) ∈ G.E
    (u, v).f = 0
while there exists a path p from s to t in the residual network G_f
    c_f(p) = min{c_f(u, v) : (u, v) is in p}
    for each edge (u, v) in p
        if (u, v) ∈ E
            (u, v).f = (u, v).f + c_f(p)
        else (v, u).f =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

With DFS

How many times do we have to increase the flow ? if DFS finds:

$s - u - v - t$ followed by $s - v - u - t$

Network G



Graph Algorithms (Chapter 26)
Ford-Fulkerson Algorithm

University of Stavanger

FORD-FULKERSON(G, s, t)

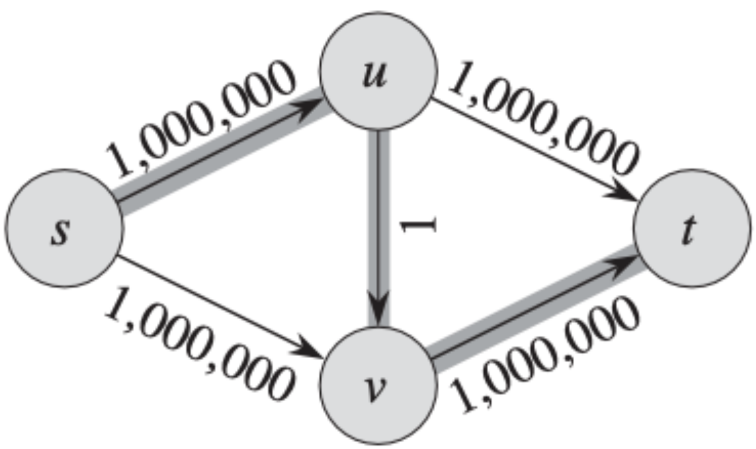
```
for each edge (u, v) ∈ G.E
    (u, v).f = 0
while there exists a path p from s to t in the residual network G_f
    c_f(p) = min {c_f(u, v) : (u, v) is in p}
    for each edge (u, v) in p
        if (u, v) ∈ E
            (u, v).f = (u, v).f + c_f(p)
        else (v, u).f = (v, u).f − c_f(p)
```

With DFS

How many times do we have to increase the flow ? if DFS finds:

$s - u - v - t$ followed by $s - v - u - t$

Network G



The maximum flow $f = 2.10^6$ increased by units of 1 gives us $|f^*| = 2.10^6$

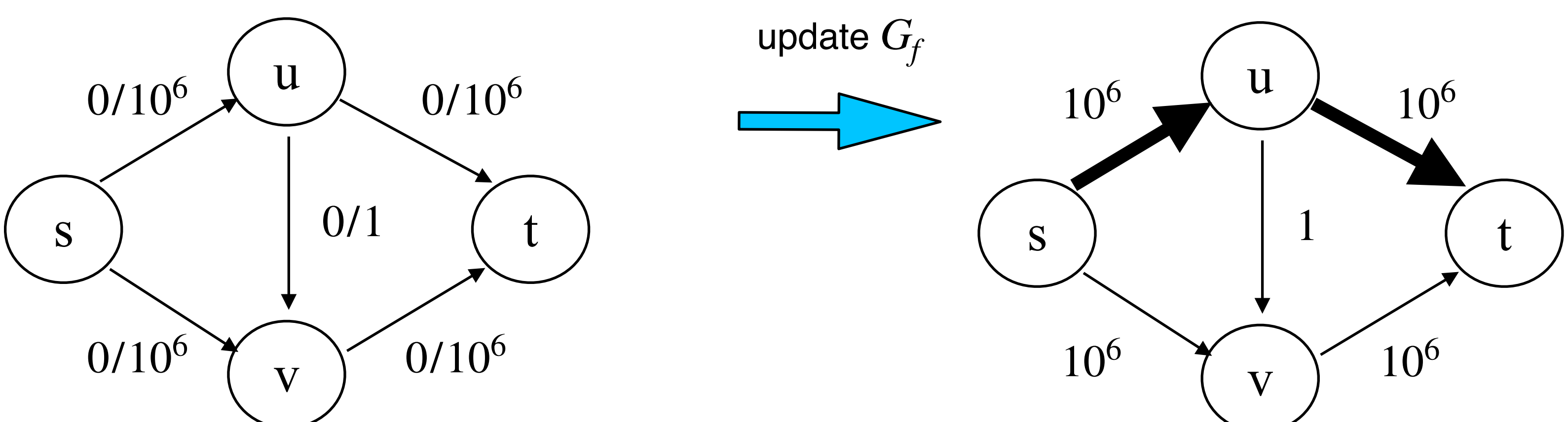
$O(E \cdot |f^*|)$

Graph Algorithms (Chapter 26)
Edmonds-Karp algorithm

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

With BFS

Network G | Residual G_f and augmented path

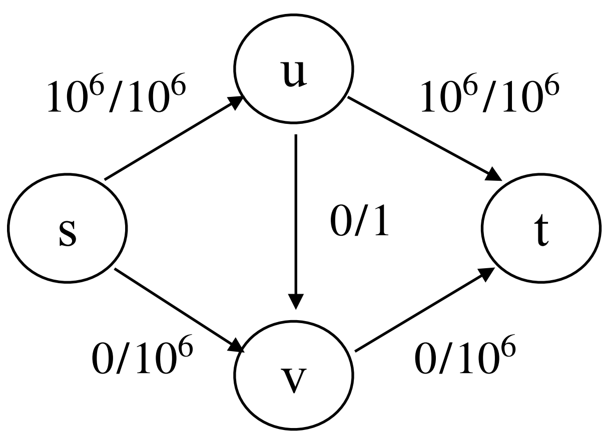


Graph Algorithms (Chapter 26) Edmonds-Karp algorithm


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

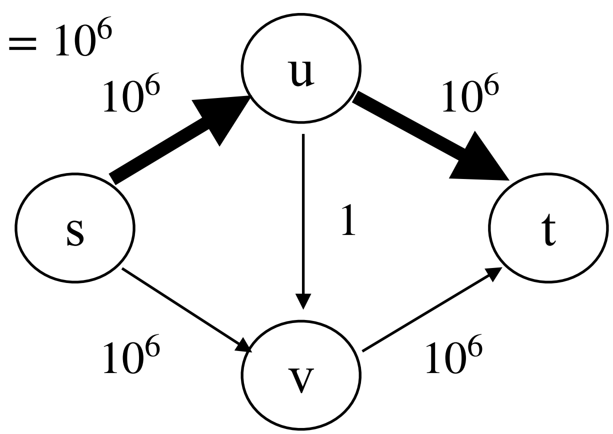
With BFS

Network G



Update G, $c_f(p) = 10^6$

Residual G_f and augmented path



Graph Algorithms (Chapter 26)
Edmonds-Karp algorithm

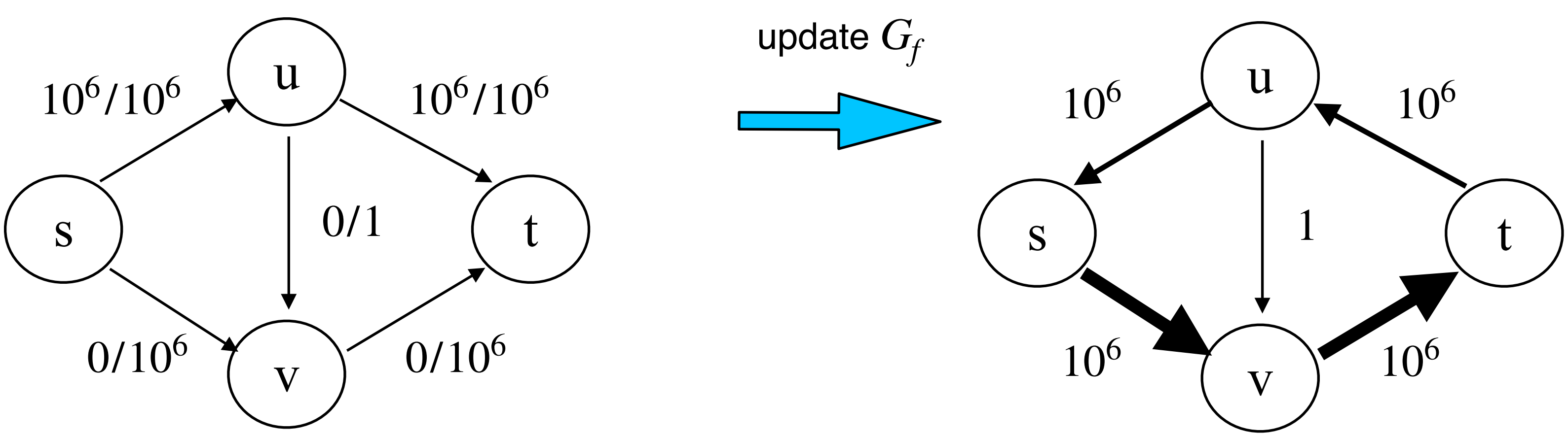
University of Stavanger

FORD-FULKERSON(G, s, t)

```
for each edge (u, v) ∈ G.E
    (u, v).f = 0
while there exists a path p from s to t in the residual network G_f
    c_f(p) = min {c_f(u, v) : (u, v) is in p}
    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)
```

With BFS

Network G — Residual G_f and augmented path



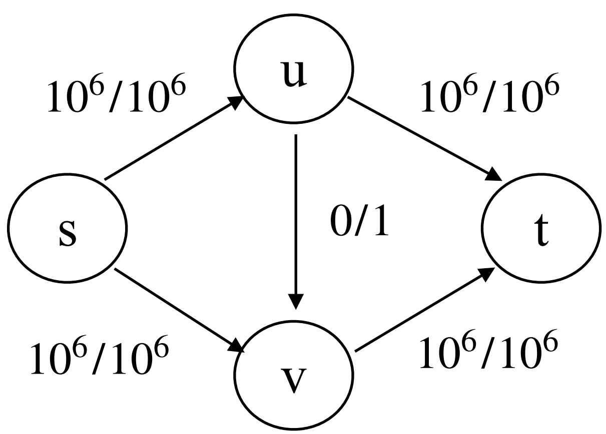
Graph Algorithms (Chapter 26)
Edmonds-Karp algorithm

FORD-FULKERSON(G, s, t)

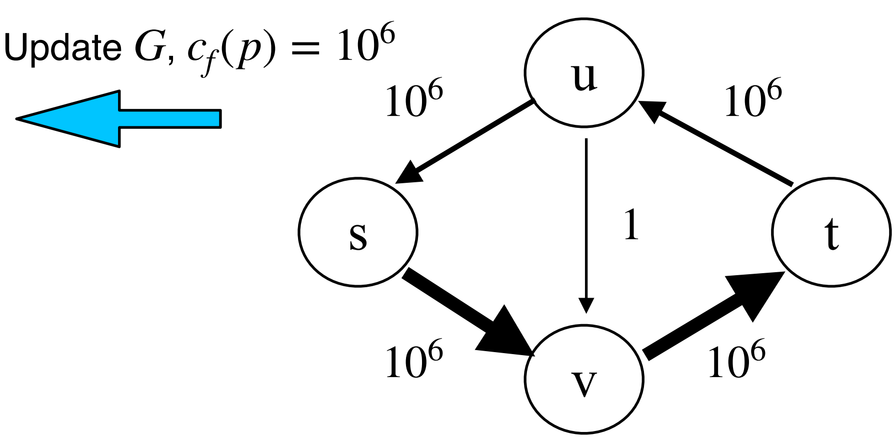
```
for each edge (u, v) ∈ G.E
    (u, v).f = 0
while there exists a path p from s to t in the residual network G_f
    c_f(p) = min{c_f(u, v) : (u, v) is in p}
    for each edge (u, v) in p
        if (u, v) ∈ E
            (u, v).f = (u, v).f + c_f(p)
        else (v, u).f = (v, u).f − c_f(p)
```

With BFS

Network G



Residual G_f and augmented path



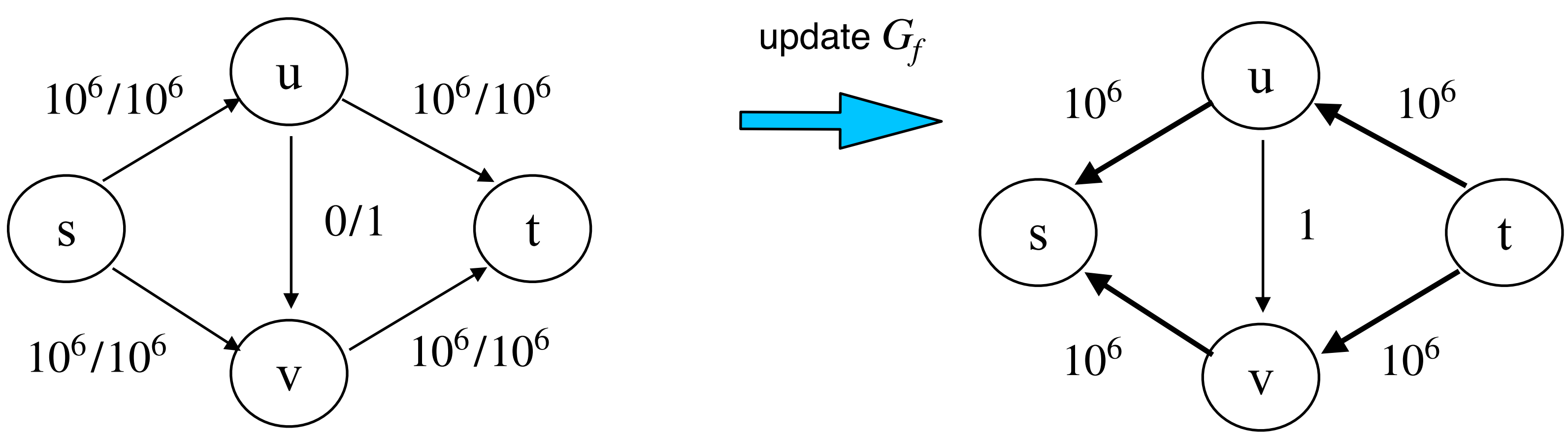
Graph Algorithms (Chapter 26)
Edmonds-Karp algorithm

FORD-FULKERSON(G, s, t)

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

With BFS

Network G | Residual G_f and augmented path



Graph Algorithms (Chapter 26) Edmonds-Karp algorithm

University of Stavanger

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

With BFS

Runtime complexity

- Time to find a path $O(E+V) = O(E)$, when $|E| \geq |V|$ using BFS
- Total number of flow augmentation $O(V.E)$ (read page 729)

$$O(V.E^2)$$

Why!?

Graph Algorithms (Chapter 26)
Edmonds-Karp algorithm

University of Stavanger

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

Runtime complexity

$O(V\,.\,E^2)$ Why!?

Lemma 26.7 states that the shortest paths keep increasing in size! Proof in page 728

Lemma 26.7

If the Edmonds-Karp algorithm is run on a flow network $G = (V, E)$ with source s and sink t, then for all vertices $v \in V - \{s, t\}$, the shortest-path distance $\delta_f(s, v)$ in the residual network G_f increases monotonically with each flow augmentation.

Graph Algorithms (Chapter 26)
Edmonds-Karp algorithm

FORD-FULKERSON(G, s, t)

```
for each edge (u, v) ∈ G.E
    (u, v).f = 0
while there exists a path p from s to t in the residual network G_f
    c_f(p) = min{c_f(u, v) : (u, v) is in p}
    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)
```

Runtime complexity

$O(V \,.\, E^2)$ Why!?

Lemma 26.8 states the number of flow augmentations $O(VE)$ in the worst case.

Proof on page 729

Theorem 26.8

If the Edmonds-Karp algorithm is run on a flow network $G = (V, E)$ with source s and sink t, then the total number of flow augmentations performed by the algorithm is $O(VE)$.

Edmonds-Karp algorithm

Proof direction of Lemma 26.8 states the number of flow augmentations $O(VE)$:

1. We use lemma 26.7: shortest path increases monotonically
2. An edge (u, v) that is augmented will disappear from the residual graph until the flow on it is decreased => (v, u) happens to appear on a subsequent shortest path.
3. Point 2, leads the fact that (u, v) can become critical again (minimum flow on the shortest path) at best 2 distances away from the previous time when it was.
4. This means that (u, v) can become critical at most the number of edges on the shortest path $|E|/2$, because the number of edges in a shortest path is at most $|V| - 1$, and edge can become critical every second time, we have $|V|/2 - 1$ —> $O(V)$
5. Every edge can eventually become critical $O(V)$ times, leading to $O(E.V)$ possible flow augmentations.

Graph Algorithms (Chapter 26) Edmonds-Karp algorithm

More formally (simplified)

1. $\delta_f(s,v) = \delta_f(s,u) + 1$ the first time (u,v) is critical (augmented and not any more in residual graph)
2. $\delta_{f'}(s,u) = \delta_f(s,u) + 2$ the next time (u,v) becomes critical
3. The maximum distance of the shortest path is at most $O(V)$
4. An edge can become critical $|V|/2 \rightarrow O(V)$ because of point 2
5. There are $|E| \rightarrow O(E)$ possible edges that can become critical.
6. Total: There are $O(E)$ edges than can become critical $O(V)$ times, i.e $O(E\,.\,V)$, each time requires a BFS search $O(E+V) -> O(E), |E| > |V|$———-Result: $O(V\,.\,E^2)$

Summary Chapter 26

The maximum flow problem

Antiparallel edges issues

Multiple sources and sinks

The Ford-Fulkerson method

Residual networks

Augmenting paths

Max-flow min-cut theorem

Forld-Flkerson algorithm and its runtime

Edmonds-Karp algorithm and its runtime

Exercices

Will come