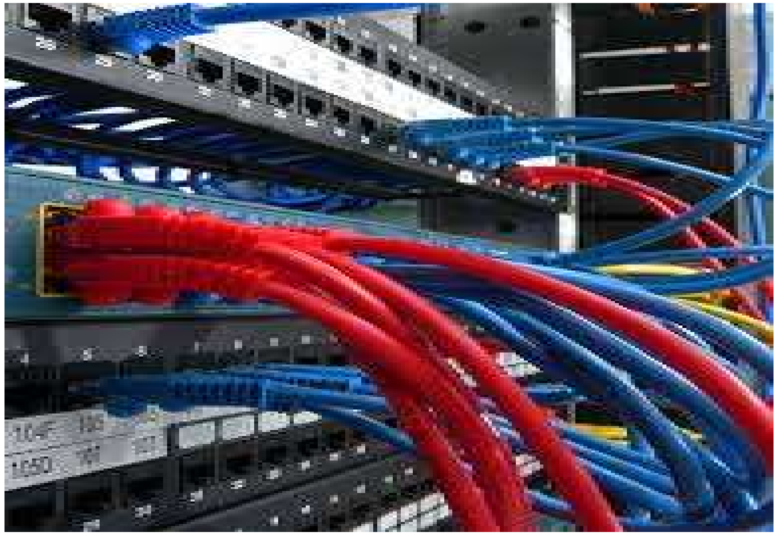
12: **Maximum Flow**

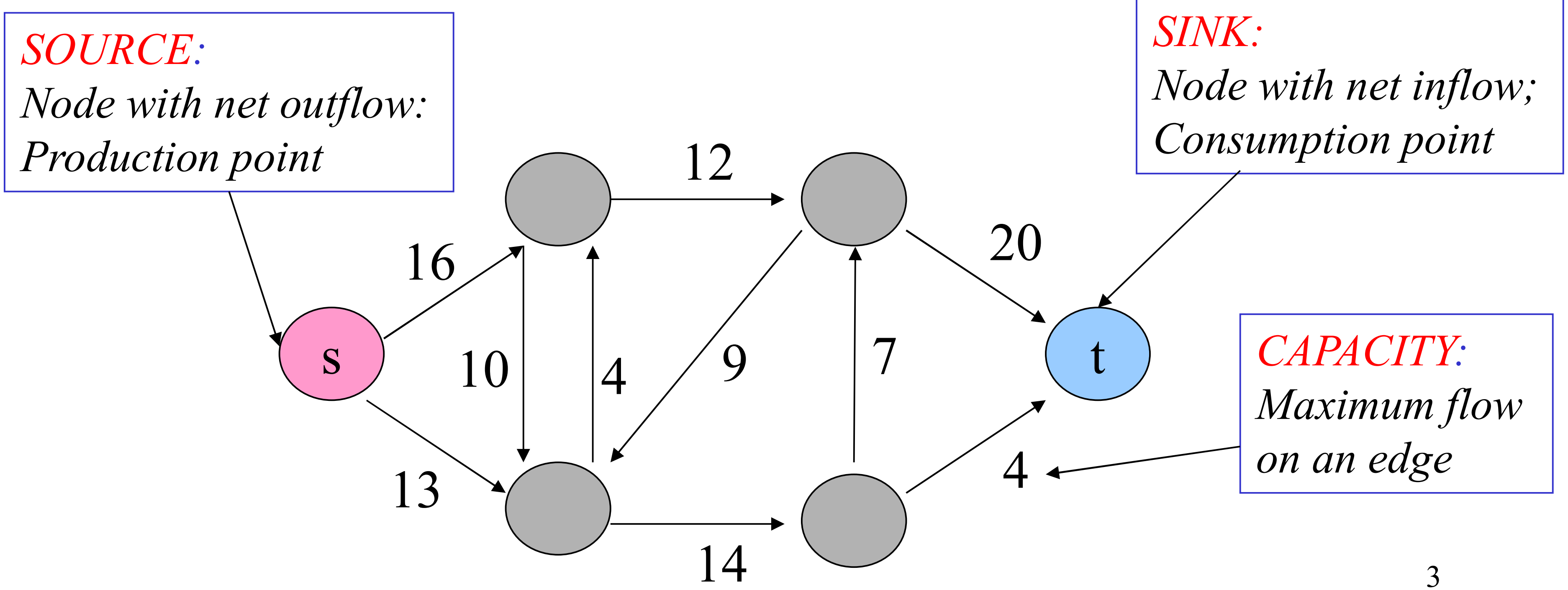
- Maximum Flow Problem
- The Ford-Fulkerson method
- Maximum bipartite matching


Google
I-71/670 Interchange Design Build Project



Flow networks:

- A flow network G=(V,E): a directed graph, where each edge (u,v)∈E has a nonnegative capacity c(u,v)>=0.
- If (u,v)∉E, we assume that c(u,v)=0.
- two distinct nodes: **a source s and a sink t.**



Flow:

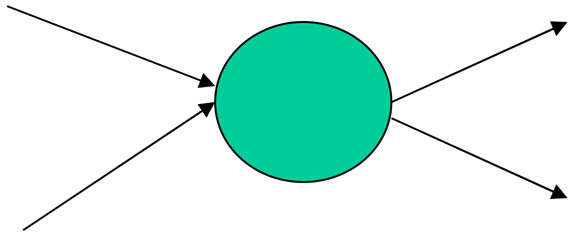
- Given: G=(V, E): a flow network with capacity function c.

  s---the source and t--- the sink.
- A flow in G: a real-valued function *f*: E →R satisfying the following two properties:
- Capacity constraint: For all u,v ∈V, we require

$$f(u,v) \leq c(u,v).$$

- Flow conservation: For all v ∈V-{s,t}, we require

$$\sum_{e.in.v} f(e) = \sum_{e.out.v} f(e)$$



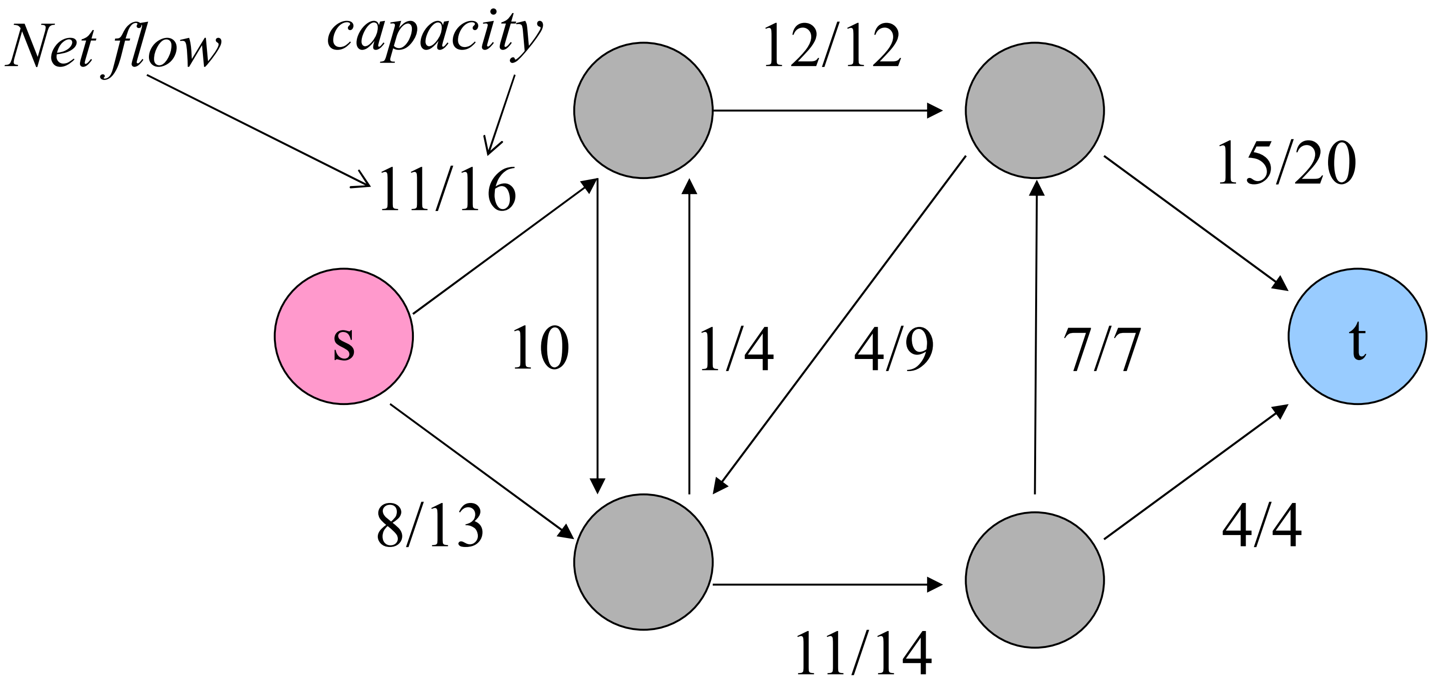
Net flow and value of a flow f:

- The quantity f (u,v) is called the net flow from vertex u to vertex v.
- The value of a flow is defined as

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

  – The total flow from the **source** (s) to any other vertices.

  – The same as the total flow from any vertices to **the sink (t).**



A flow f in G with value $\left|f\right| = 19$.

Maximum-flow problem:

- Given a flow network G with source s and sink t
- Find a flow of maximum value from s to t.

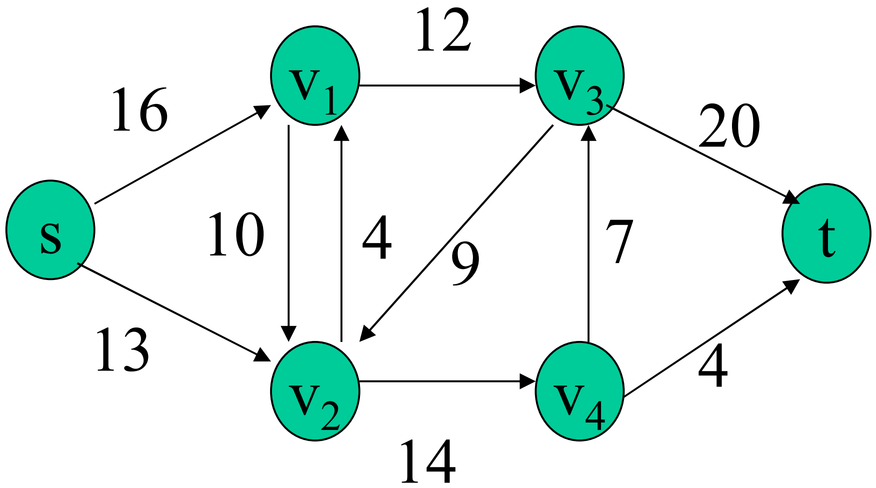
- How to solve it efficiently?

The Ford-Fulkerson method:

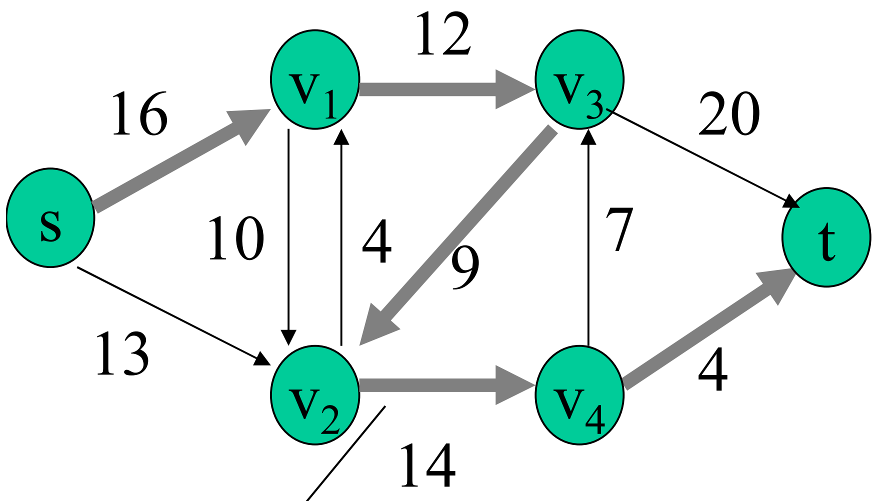
- Ford-Fulkerson method
  - it is a “method” rather than an “algorithm” because it encompasses several implementations with different running times.
  - The Ford-Fulkerson method depends on three important ideas:
    - residual networks
    - augmenting paths, and
    - cuts.

Continue:

- 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*

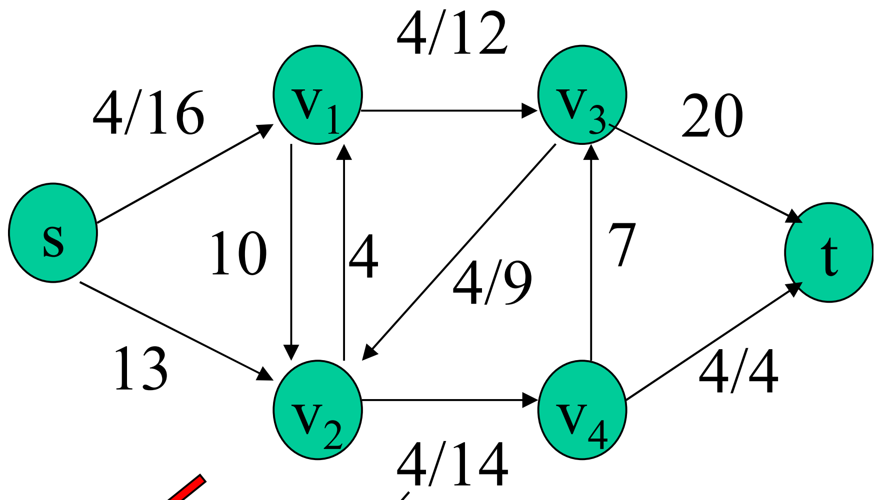

16
12
20
10
4
9
7
13
14
4
s
v1
v2
v3
v4
t

Example

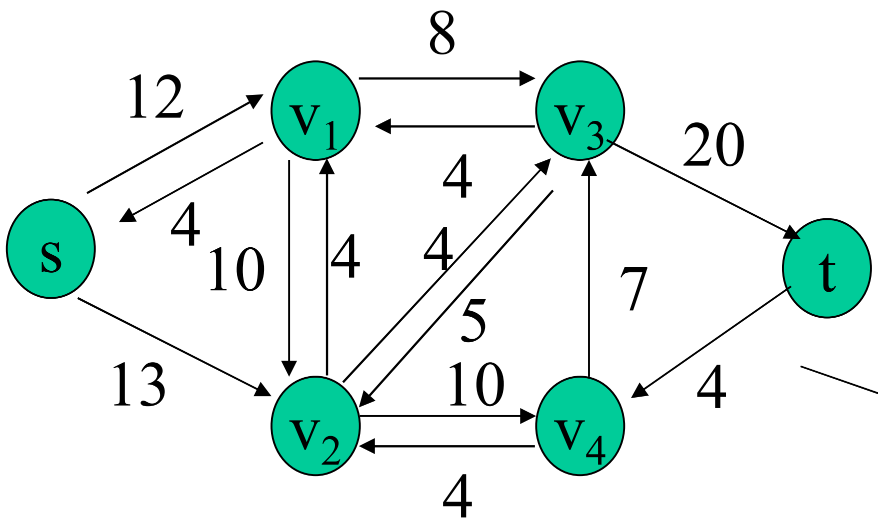


(a)

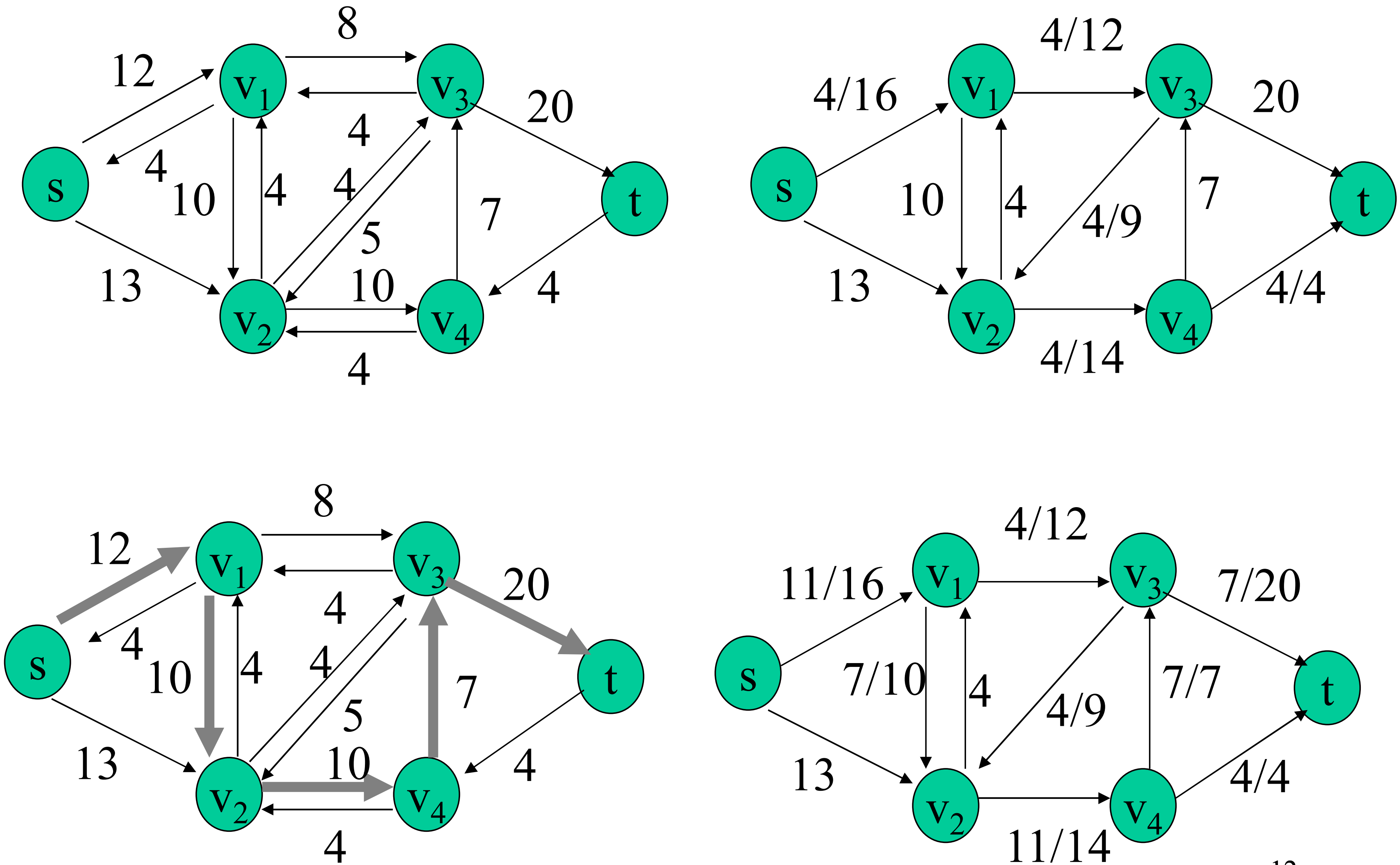
Bold path: augment path



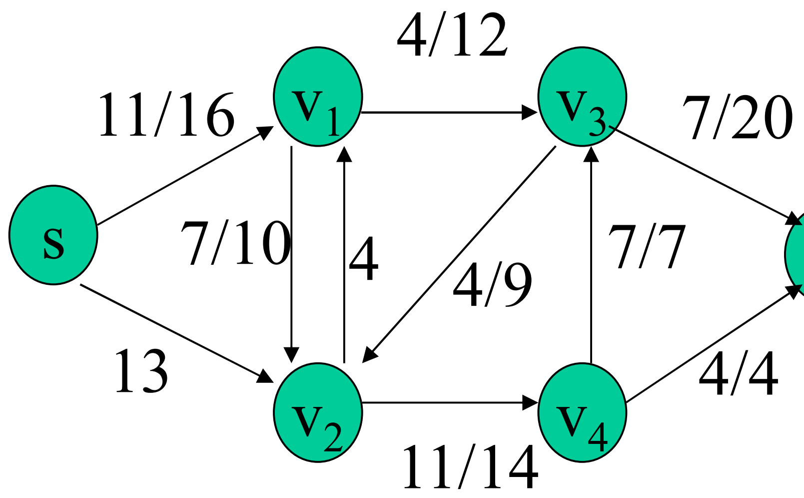
Net flow *Flow*



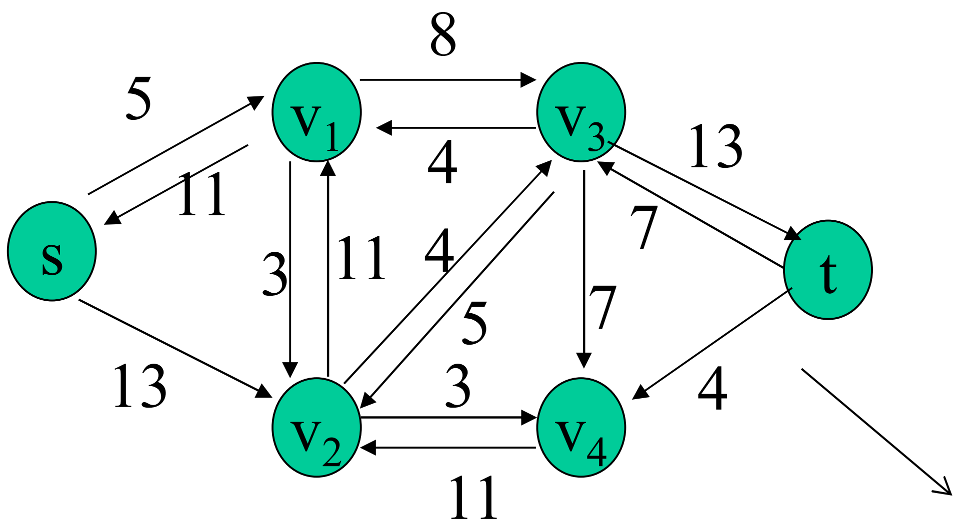
Residual network



s
v1
v2
v3
v4
t
12
4
10
4
8
4
4
5
13
10
4
20
7
4
4/16
4/12
20
10
4
4/9
7
13
4/14
4/4
12
8
4
4
10
4
5
13
10
4
20
7
4
11/16
4/12
7/20
7/10
4
4/9
7/7
13
11/14
4/4



Net flow network



(b)

Residual network

Residual networks:

- Given a flow network and a flow, the **residual network** consists of edges that can admit more net flow.
- f: a flow in G.
- The residual capacity of (u,v), given by:
  - **$c_f(u,v) = c(u,v) - f(u,v)$**
  - ***in the other direction***
  - **$c_f(v,u) = c(v, u) + f(u, v)$.**

Fact 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 flow sum f+f' is a flow in G with value .
- f+f': the flow in the same direction will be added.

  the flow in different directions will be cancelled.

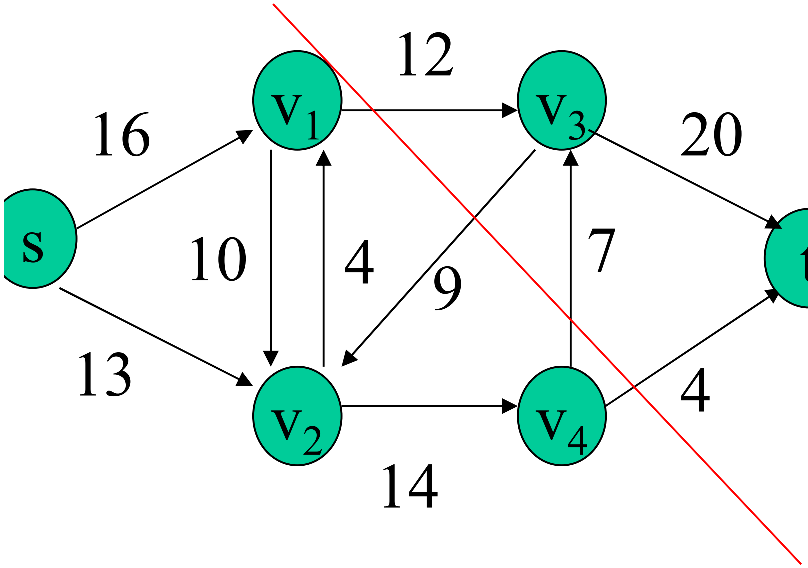
Augment paths:

- Given a flow network G=(V,E) and a flow f, an augment path is a simple path from s to t in the residual network G_f.

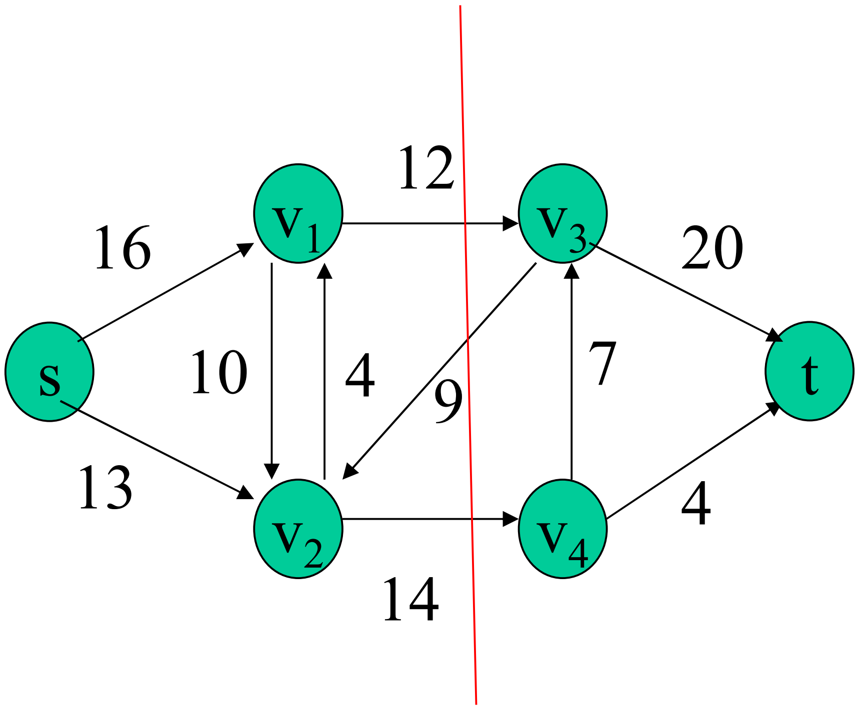
- Residual capacity of p : the maximum amount of net flow that we can ship along p, i.e.,
  $c_f(p)=\min\{c_f(u,v):(u,v) \text{ is on } p\}$.

The residual capacity is 1.

Cut and flow



$|f| \leq 12+7+4,$



$|f| \leq 12+14,$

Theorem max $|f| = min\ |cut|$

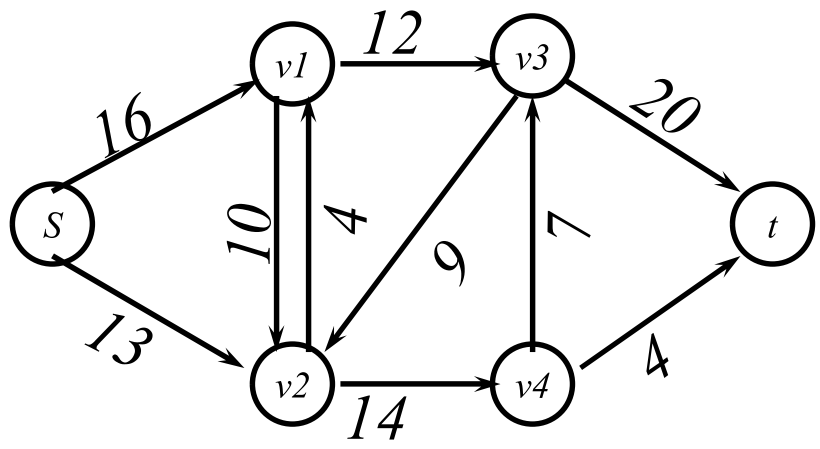
The basic Ford-Fulkerson algorithm:

- 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 $f[u,v] \leftarrow f[u,v]+c_f(p)$
-

The basic Ford Fulkerson algorithm

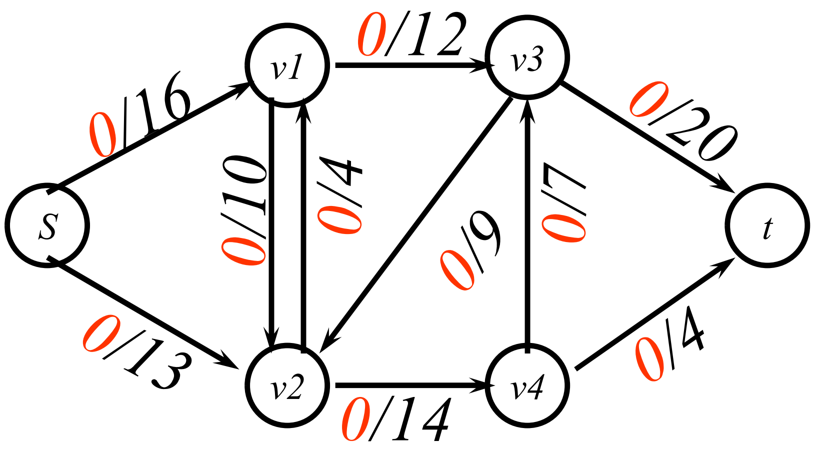
example of an execution

Network



The basic Ford Fulkerson algorithm

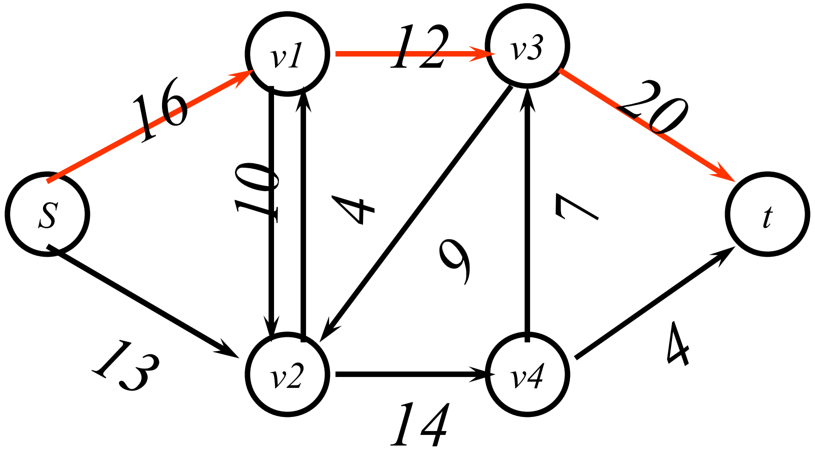
example of an execution




```
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) ∈ p}
     for each edge (u, v) in p
         do f [u, v] = f [u, v] + c_f(p)
```

The basic Ford Fulkerson algorithm

example of an execution

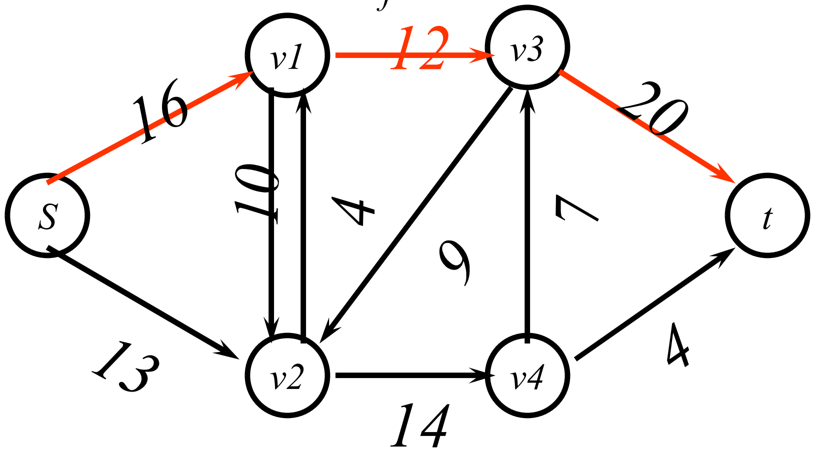


for *each edge (u, v)* $\in$ *E [G]*
  do f [u, v] = 0
    f [v, u] = 0
while *there exists a path p from s to t in the residual network* G_f
  do $c_f(p) = \min\{c_f(u, v) \mid (u, v) \in p\}$
    for *each edge (u, v) in p*
      do $f[u, v] = f[u, v] + c_f(p)$

The basic Ford Fulkerson algorithm

example of an execution

(residual) network G_f




```
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 Gf
  do cf(p) = min{cf(u, v) | (u, v) ∈ p}
     for each edge (u, v) in p
         do f [u, v] = f [u, v] + cf(p)
```

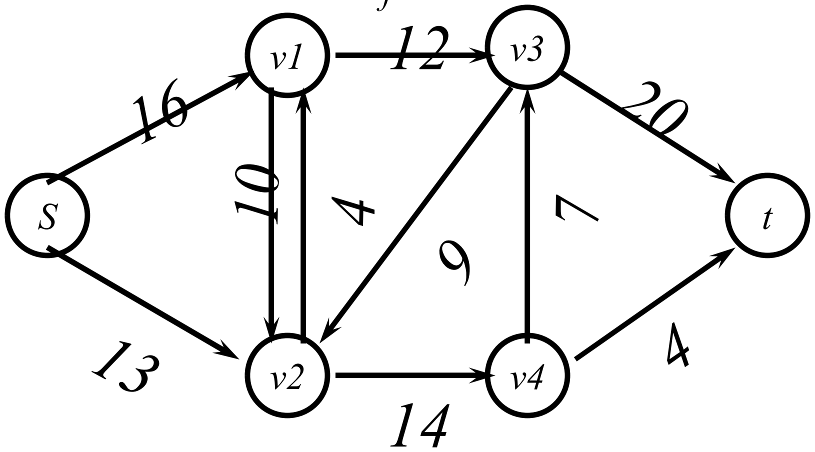
temporary variable:

$c_f(p) = 12$

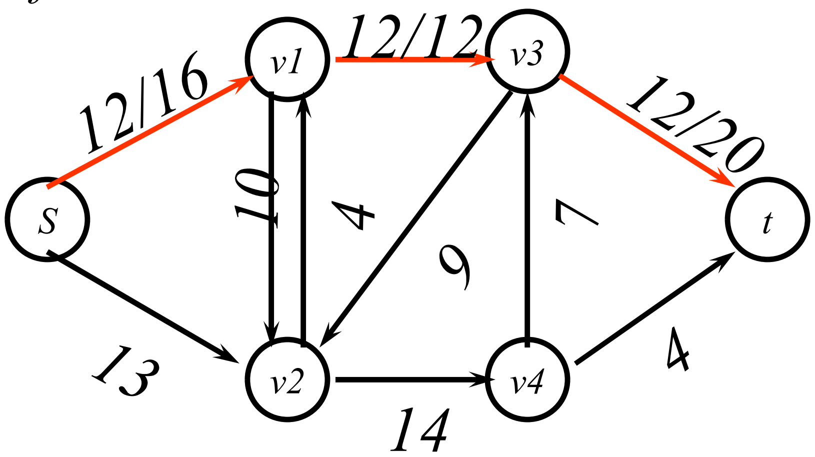
The basic Ford Fulkerson algorithm

example of an execution

(residual) network G_f



new flow network G




```
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 Gf
    do cf(p) = min{cf(u, v) | (u, v) ∈ p}
        for each edge (u, v) in p
            do f [u, v] = f [u, v] + cf(p)
```

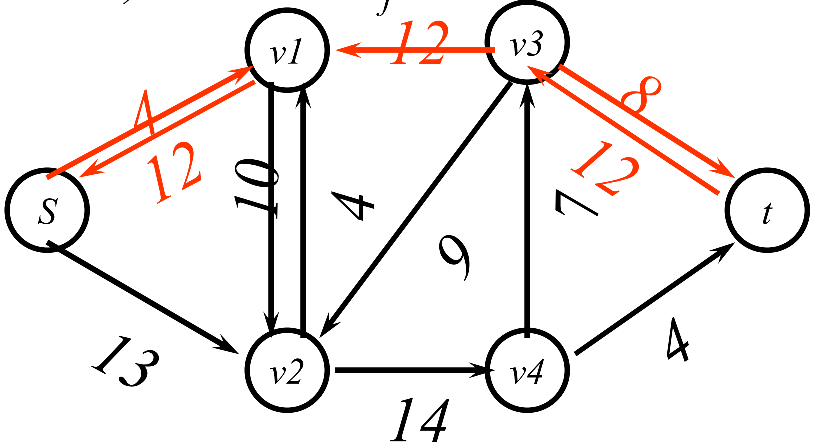
temporary variable:

$c_f(p) = 12$

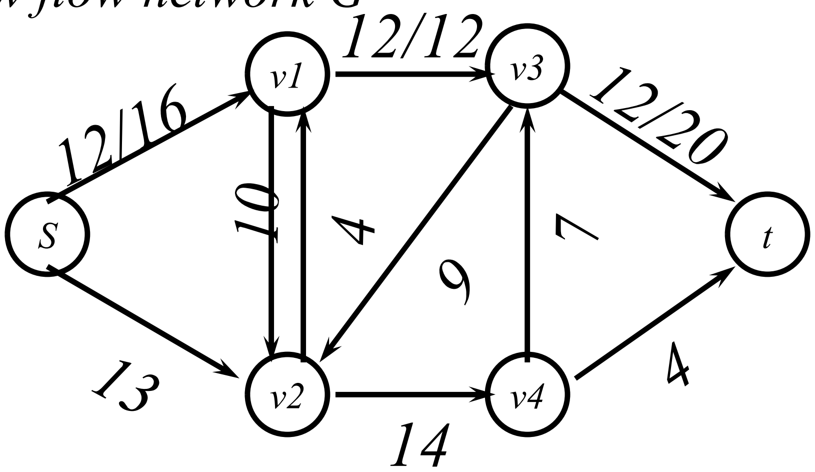
The basic Ford Fulkerson algorithm

example of an execution

(residual) network G_f



new flow network G

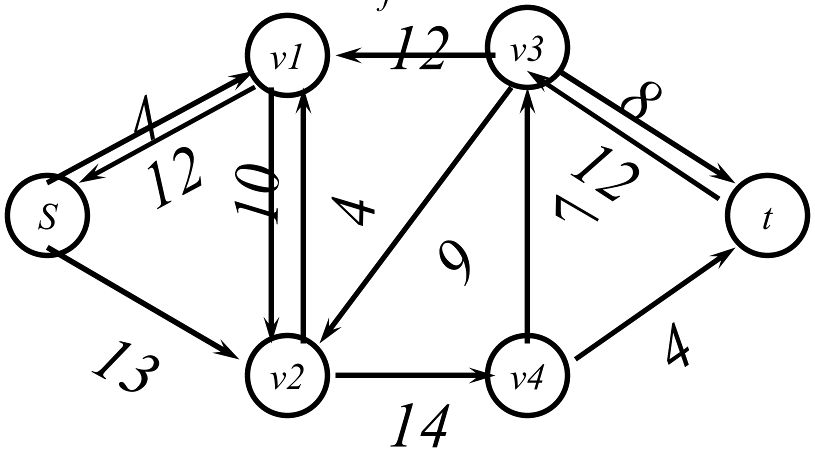


for *each edge (u, v) $\in$ E [G]*
  do f [u, v] = 0
    f [v, u] = 0
while *there exists a path p from s to t in the residual network G_f*
  do $c_f(p) = \min\{c_f(u, v) \mid (u, v) \in p\}$
    for *each edge (u, v) in p*
      do f [u, v] = f [u, v] + $c_f(p)$

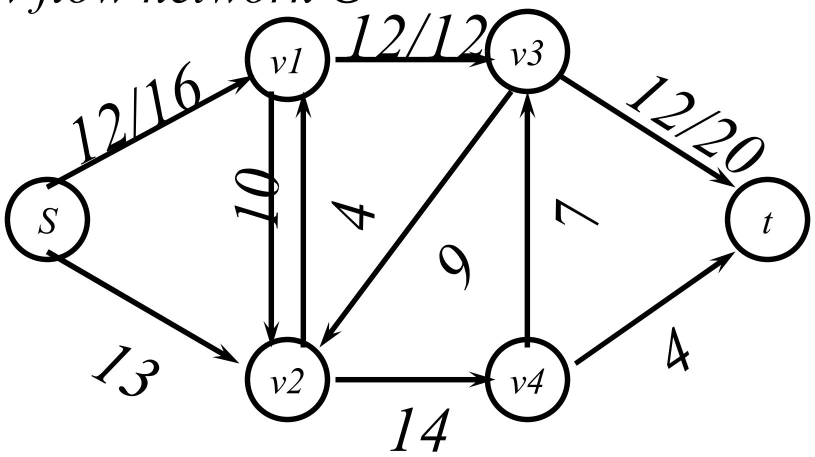
The basic Ford Fulkerson algorithm

example of an execution

(residual) network G_f



new flow network G

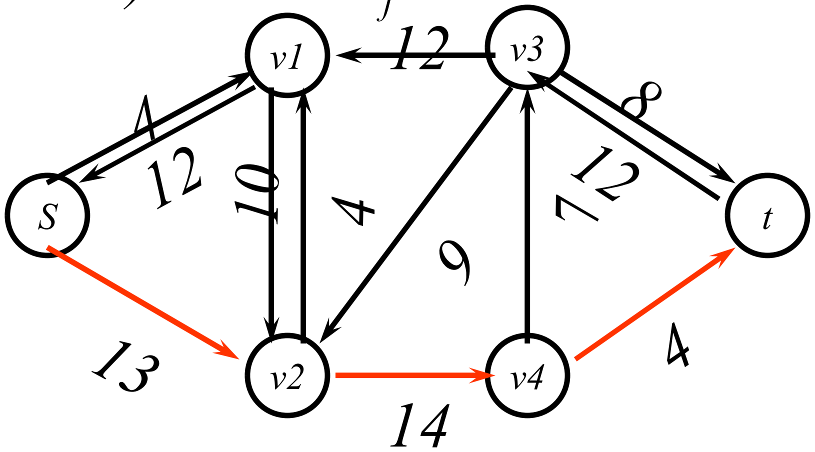


for *each edge (u, v)* $\in$ *E [G]*
    do f [u, v] = 0
        f [v, u] = 0
while *there exists a path p from s to t in the residual network G_f*
    do $c_f(p) = \min\{c_f(u, v) \mid (u, v) \in p\}$
        for *each edge (u, v) in p*
            do f [u, v] = f [u, v] + $c_f(p)$

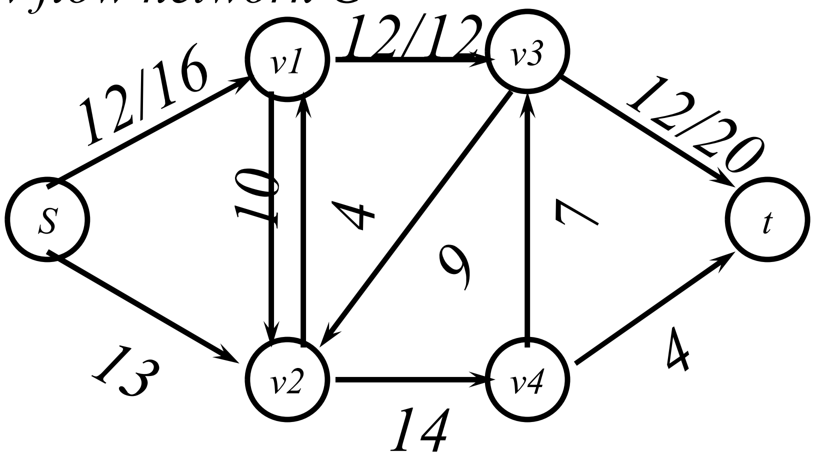
The basic Ford Fulkerson algorithm

example of an execution

(residual) network G_f



new flow network G

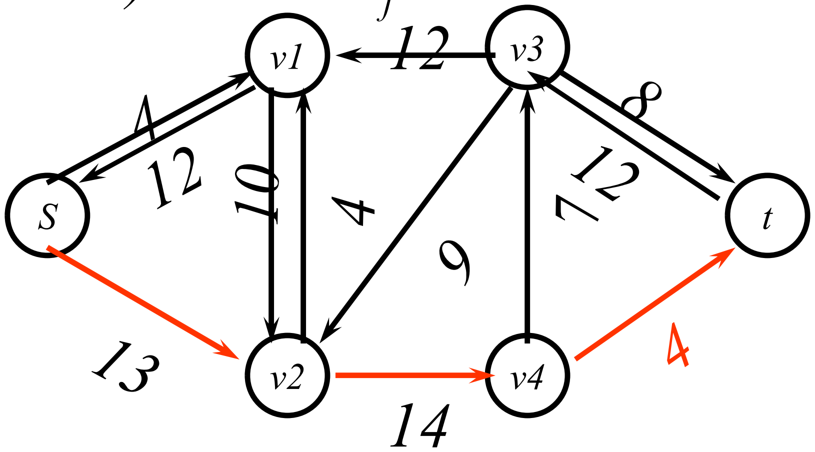


for *each edge (u, v)* $\in$ *E [G]*
    do f [u, v] = 0
        f [v, u] = 0
while *there exists a path p from s to t in the residual network G_f*
    do $c_f(p) = \min\{c_f(u, v) \mid (u, v) \in p\}$
        for *each edge (u, v) in p*
            do $f[u, v] = f[u, v] + c_f(p)$

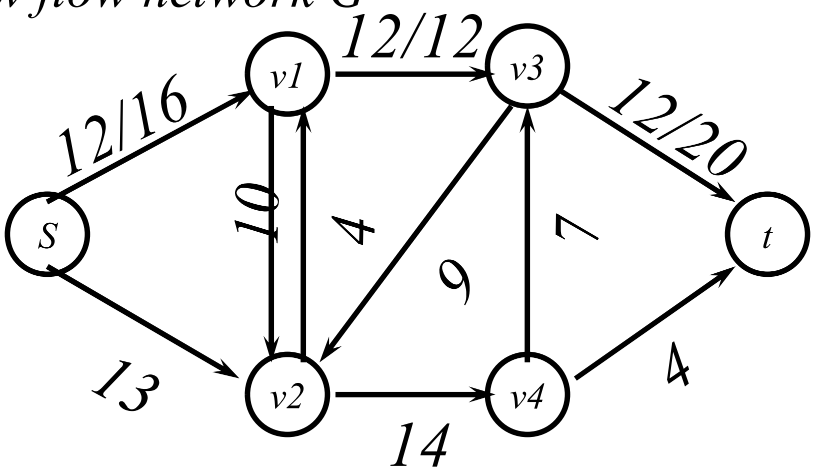
The basic Ford Fulkerson algorithm

example of an execution

(residual) network G_f



new flow network G



for *each edge (u, v)* $\in$ *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 $c_f(p) = \min\{c_f(u, v) \mid (u, v) \in p\}$
    for *each edge (u, v) in p*
      do f [u, v] = f [u, v] + $c_f(p)$

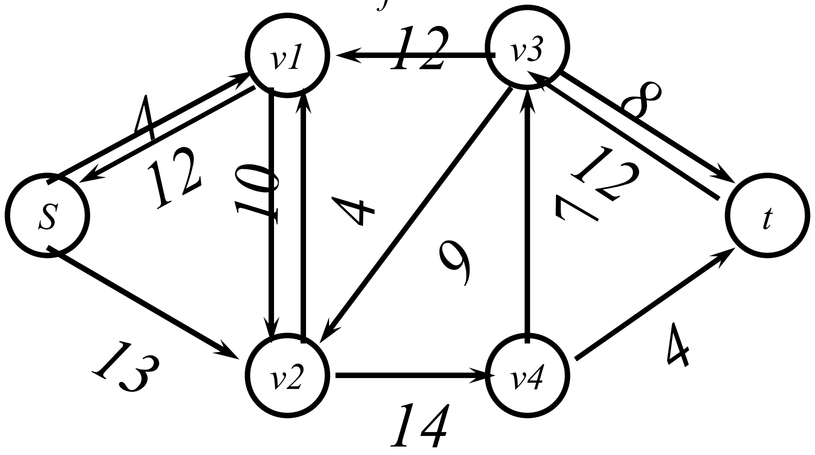
temporary variable:

$c_f(p) = 4$

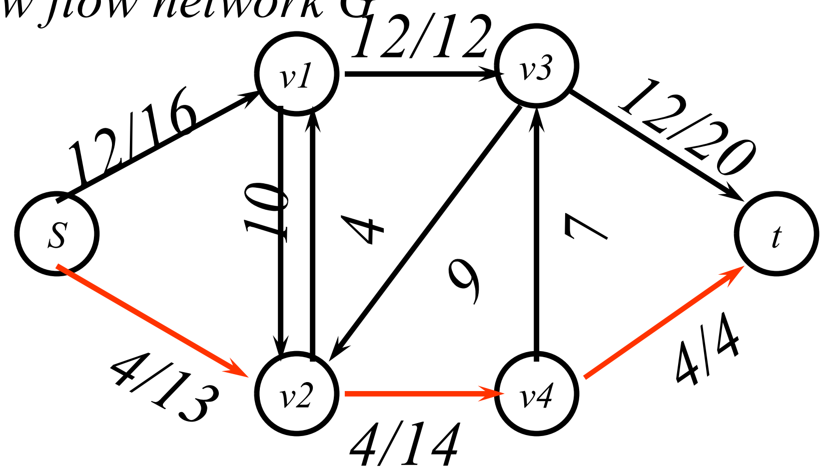
The basic Ford Fulkerson algorithm

example of an execution

(residual) network G_f



new flow network G



for *each edge (u, v) $\in$ 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 $c_f(p) = \min\{c_f(u, v) \mid (u, v) \in p\}$
        for *each edge (u, v) in p*
            do $f[u, v] = f[u, v] + c_f(p)$

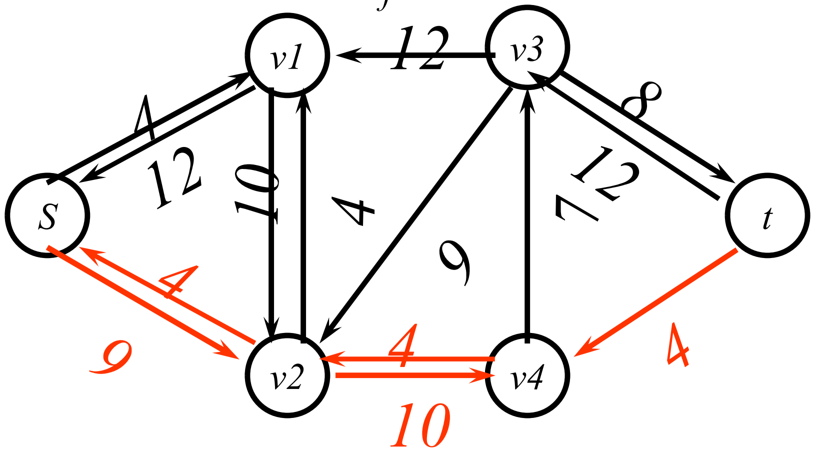
temporary variable:

$c_f(p) = 4$

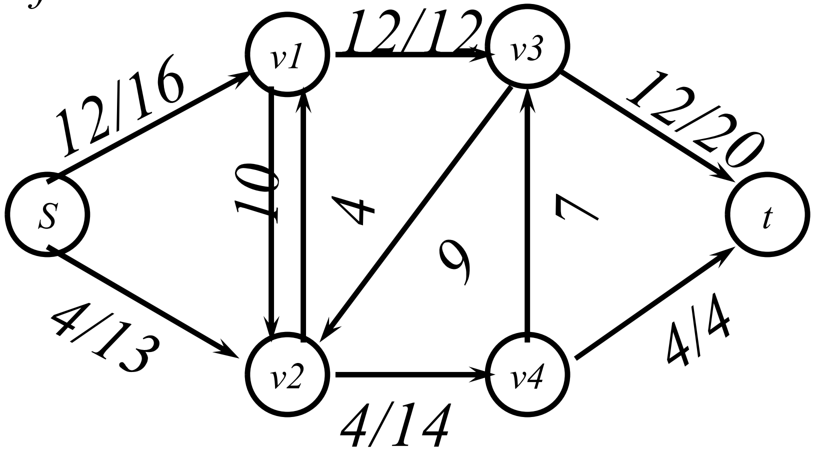
The basic Ford Fulkerson algorithm

example of an execution

(residual) network G_f



new flow network G

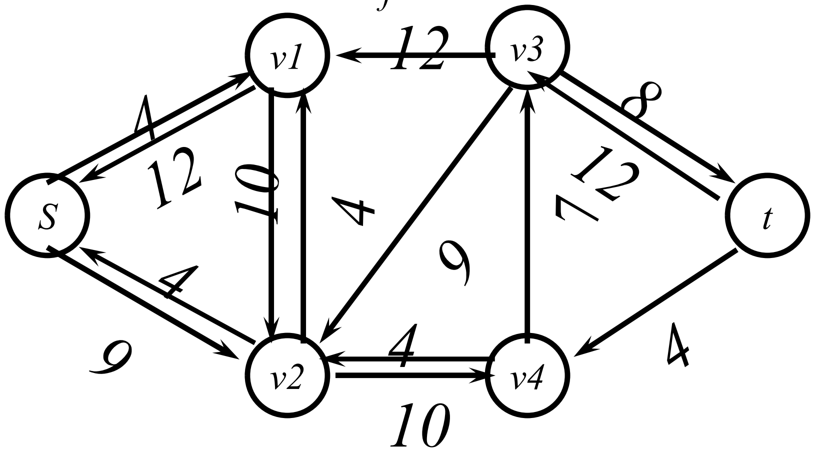



```
for each edge (u, v) ∈ E [G]
    do f [u, v] = 0
         f [v, u] = 0
while there exists a path p from s to t
in the residual network Gf
    do cf(p) = min{cf(u, v) | (u, v) ∈ p}
         for each edge (u, v) in p
              do f [u, v] = f [u, v] + cf(p)
```

The basic Ford Fulkerson algorithm

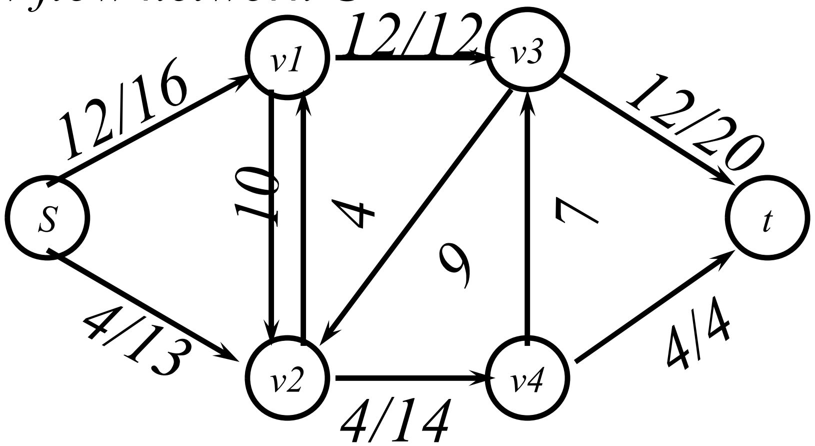
example of an execution

(residual) network G_f




```
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 Gf
    do cf(p) = min{cf(u, v) | (u, v) ∈ p}
       for each edge (u, v) in p
           do f [u, v] = f [u, v] + cf(p)
```

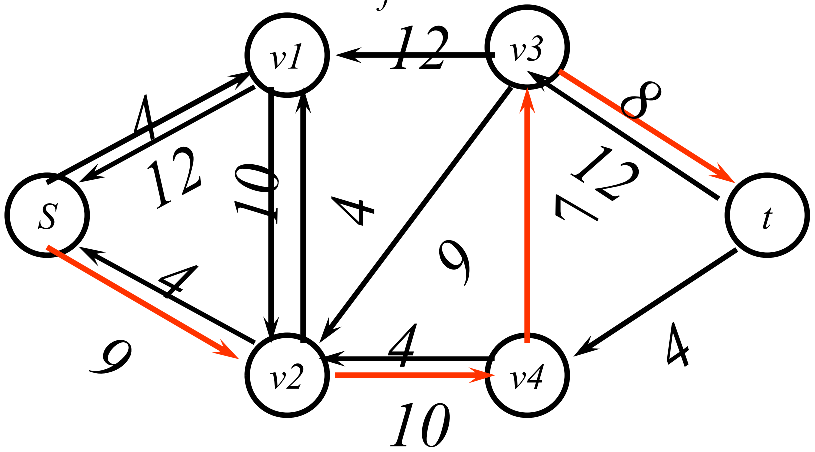
new flow network G



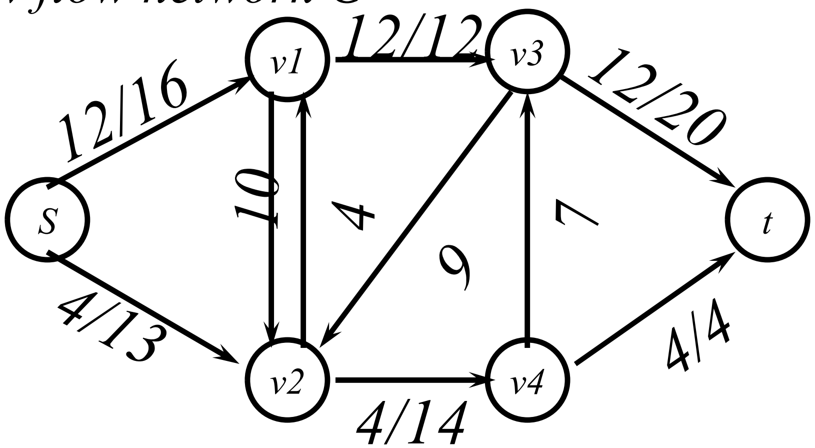
The basic Ford Fulkerson algorithm

example of an execution

(residual) network G_f



new flow network G

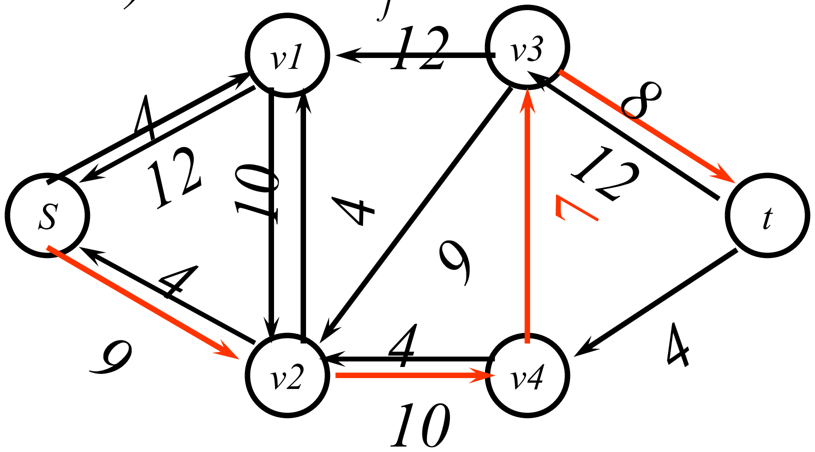



```
for each edge (u, v) ∈ E [G]
    do f [u, v] = 0
       f [v, u] = 0
while there exists a path p from s to t
in the residual network Gf
    do cf(p) = min{cf(u, v) | (u, v) ∈ p}
       for each edge (u, v) in p
           do f [u, v] = f [u, v] + cf(p)
```

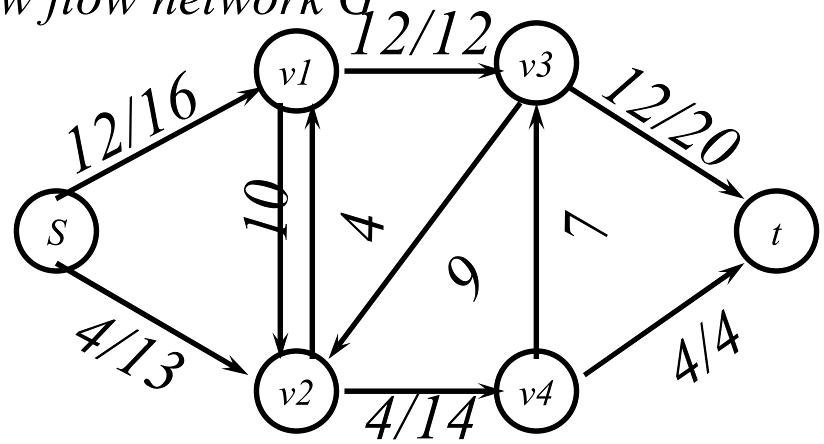
The basic Ford Fulkerson algorithm

example of an execution

(residual) network G_f



new flow network G




```
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 Gf
    do cf(p) = min{cf(u, v) | (u, v) ∈ p}
       for each edge (u, v) in p
           do f [u, v] = f [u, v] + cf(p)
```

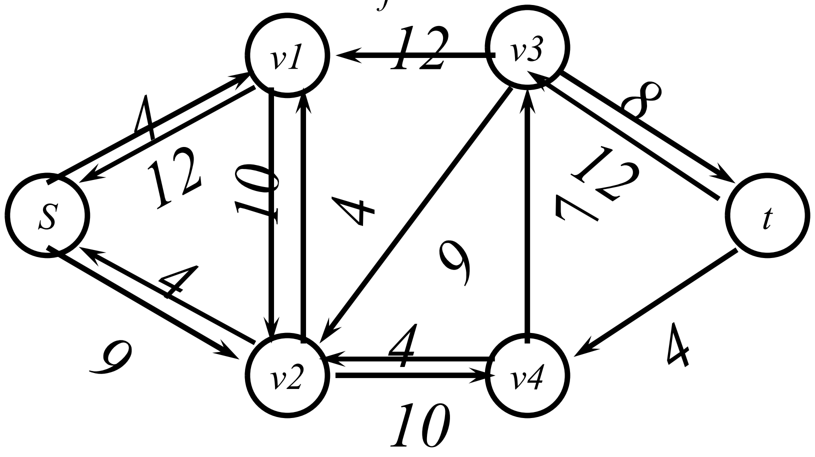
temporary variable:

$c_f(p) = 7$

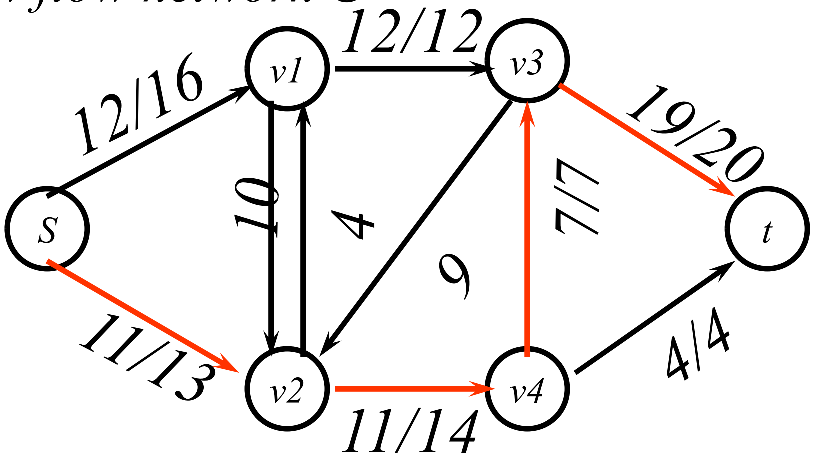
The basic Ford Fulkerson algorithm

example of an execution

(residual) network G_f



new flow network G



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 $c_f(p) = \min\{c_f(u, v) \mid (u, v) \in p\}$
    for *each edge (u, v) in p*
      do $f[u, v] = f[u, v] + c_f(p)$

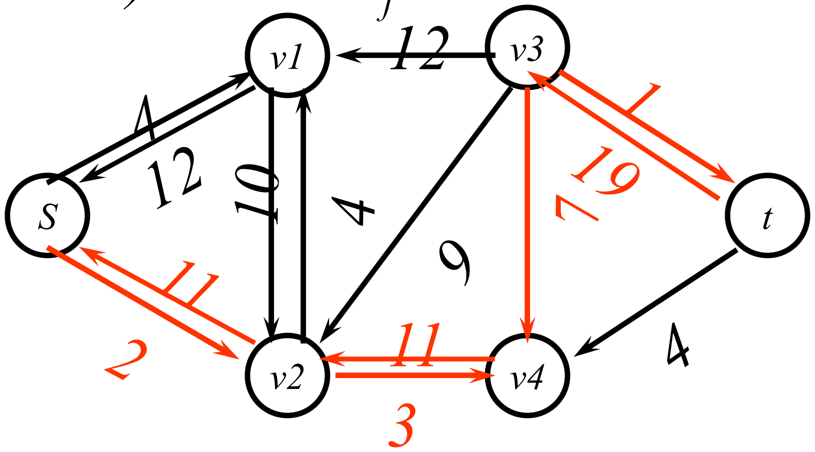
temporary variable:

$c_f(p) = 7$

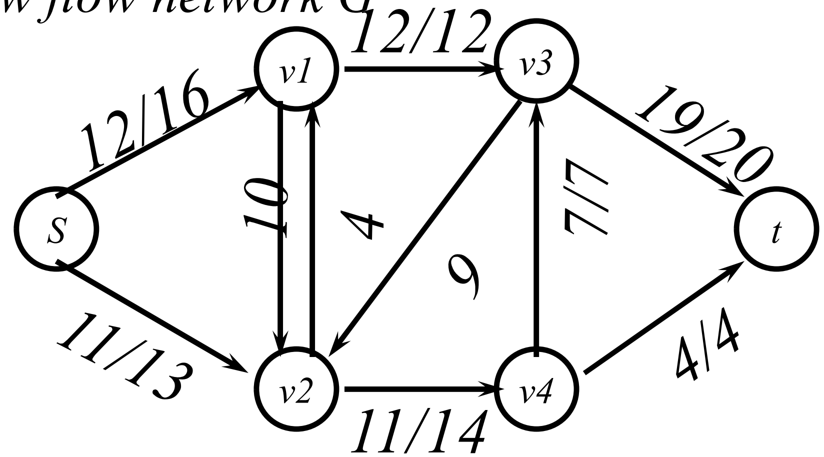
The basic Ford Fulkerson algorithm

example of an execution

(residual) network G_f



new flow network G

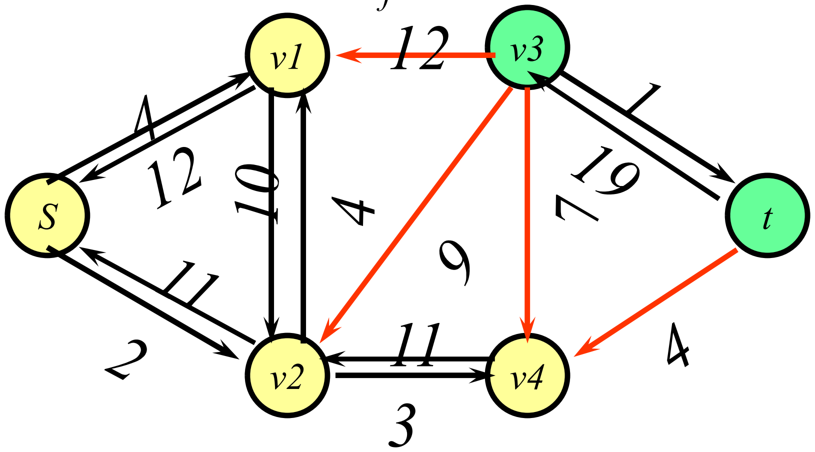


for *each edge (u, v)* $\in$ *E [G]*
  do f [u, v] = 0
    f [v, u] = 0
while *there exists a path p from s to t in the residual network* G_f
  do $c_f(p) = \min\{c_f(u, v) \mid (u, v) \in p\}$
    for *each edge (u, v) in p*
      do $f[u, v] = f[u, v] + c_f(p)$

The basic Ford Fulkerson algorithm

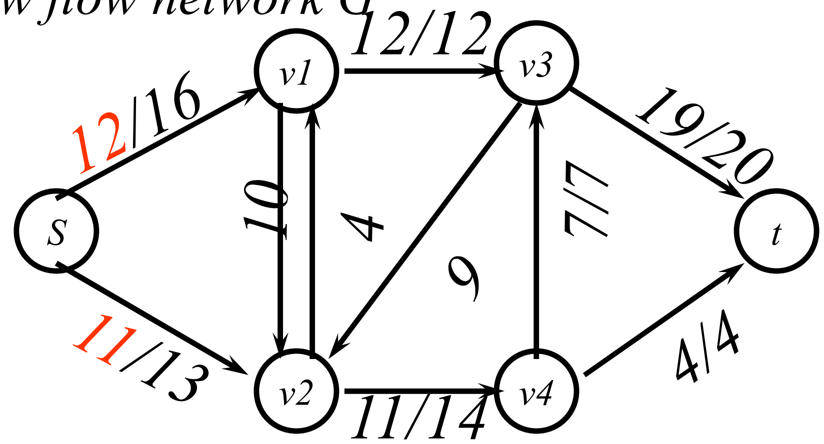
example of an execution

(residual) network G_f




```
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) ∈ p}
       for each edge (u, v) in p
           do f [u, v] = f [u, v] + c_f(p)
```

new flow network G



Finally we have:

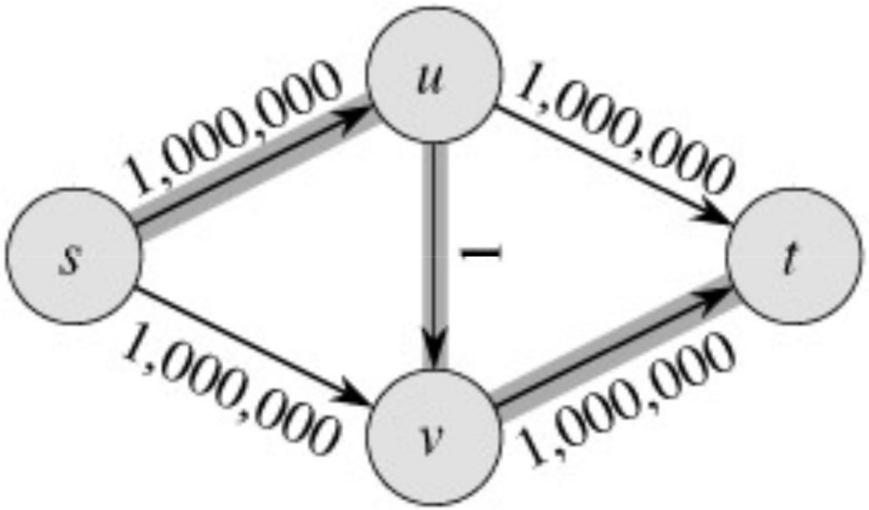
$|f| = f(s, V) = 23$

Time complexity:

- If each $c(e)$ is an *integer*, then time complexity is $O(|E|f^*)$, where f^* is the maximum flow.
- Reason: each time the flow is increased by at least one.
- This might not be a polynomial time algorithm since f^* can be represented by $\log(f^*)$ bits. So, the input size might be $\log(f^*)$.

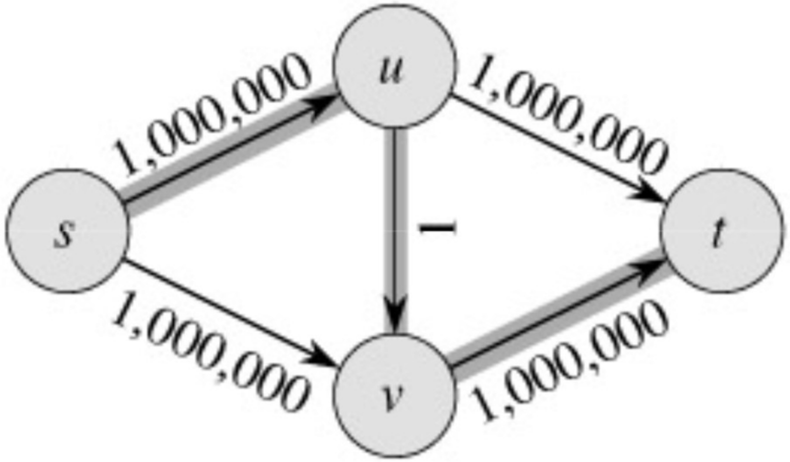
The Basic Ford-Fulkerson Algorithm

- With time O (E |f*|), the algorithm is **not** polynomial.
- Ford-Fulkerson may perform very badly

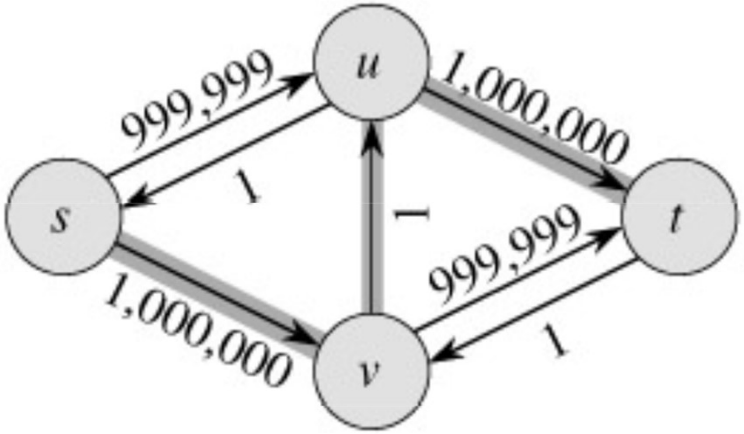


|f|=2,000,000*

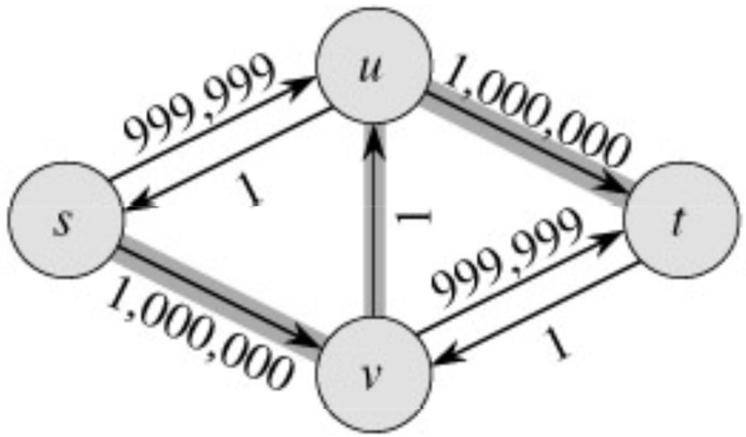
Run Ford-Fulkerson on this example



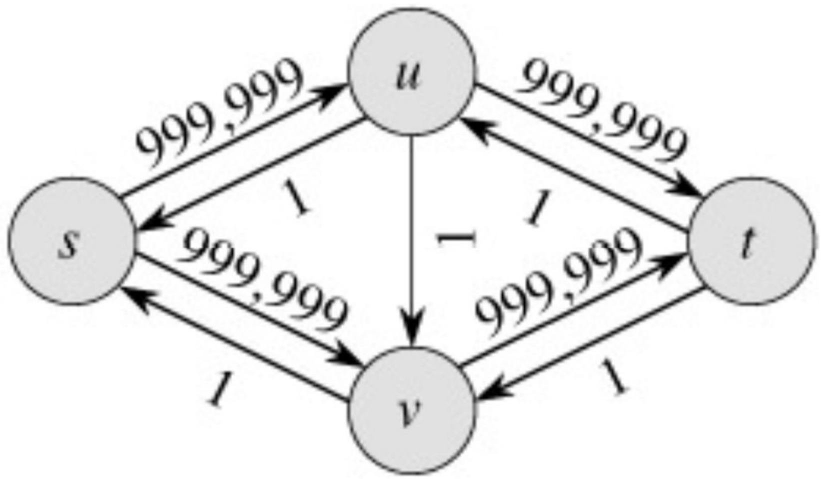
Augmenting Path



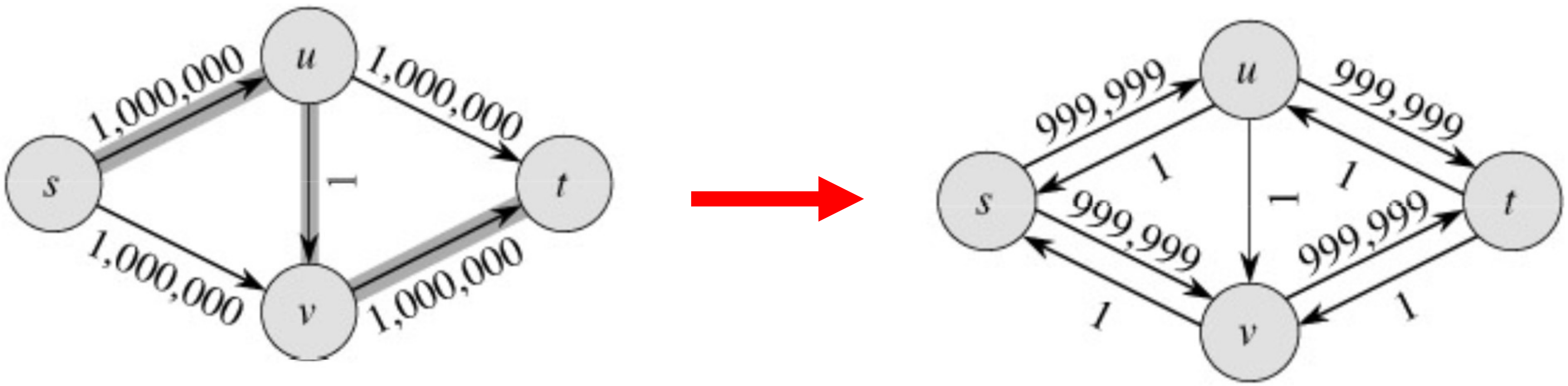
Residual Network



Augmenting Path



Residual Network

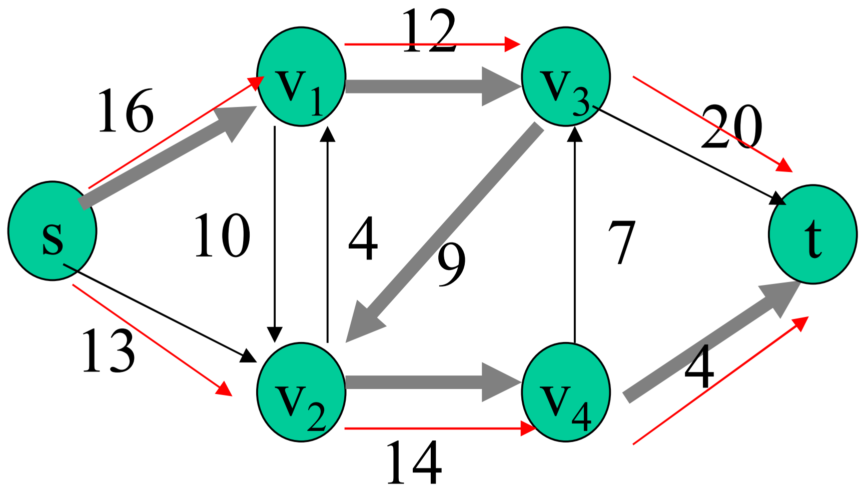


- Repeat 999,999 more times…
- Can we do better than this?

The Edmonds-Karp algorithm

- Find the augment path using breadth-first search.
- Breadth-first search gives the shortest path for graphs (Assuming the length of each edge is 1.)
- Time complexity of Edmonds-Karp algorithm is $O(V*E^2)$.
- The proof is very hard and is not required here.

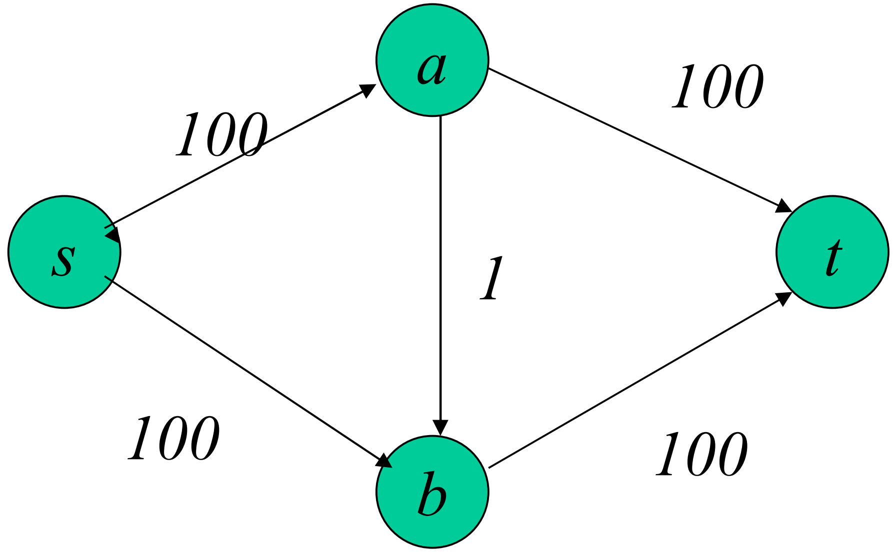
Breadth-first search



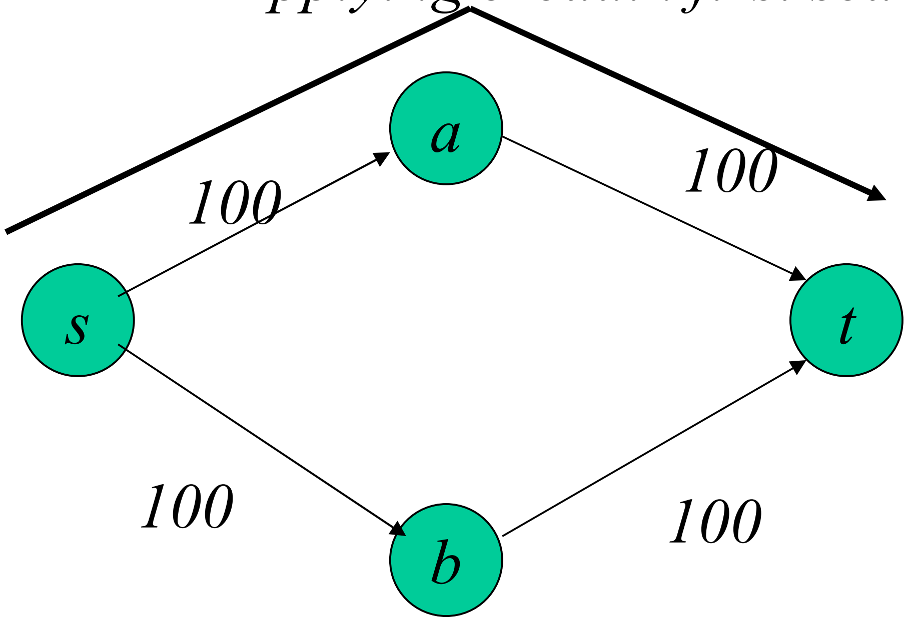
Path : s->v1->v3->t   *Path: s->v2->v4->t.*

(a)

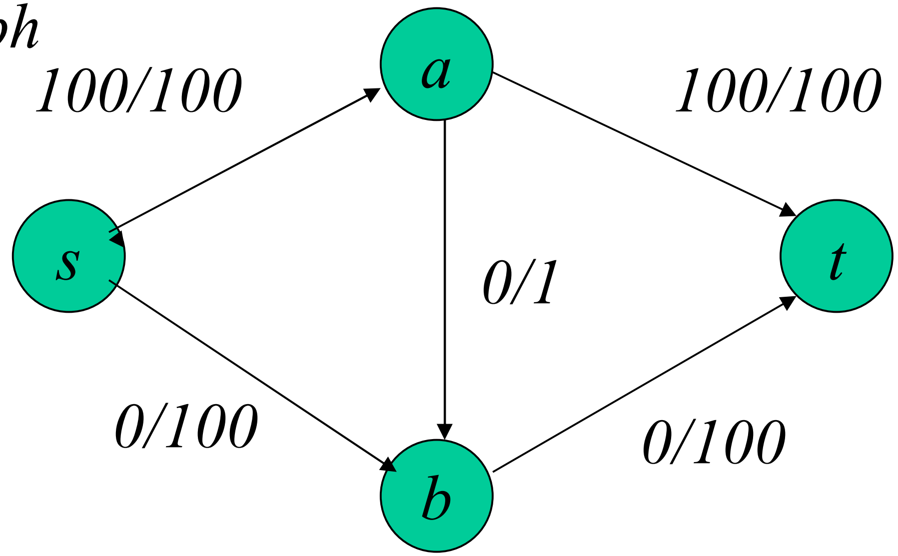
Example :



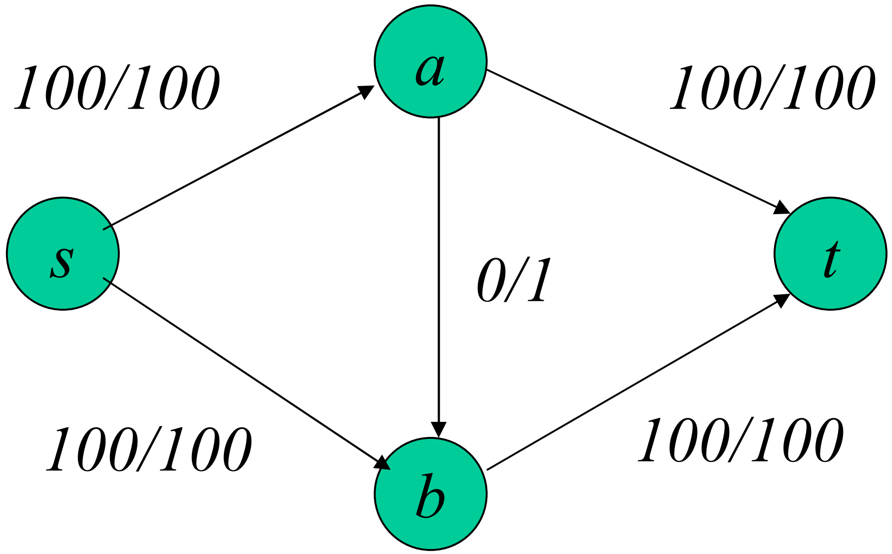
Applying breadth first search



Residual graph



Again Run BFS So the obvious path is s-b-t

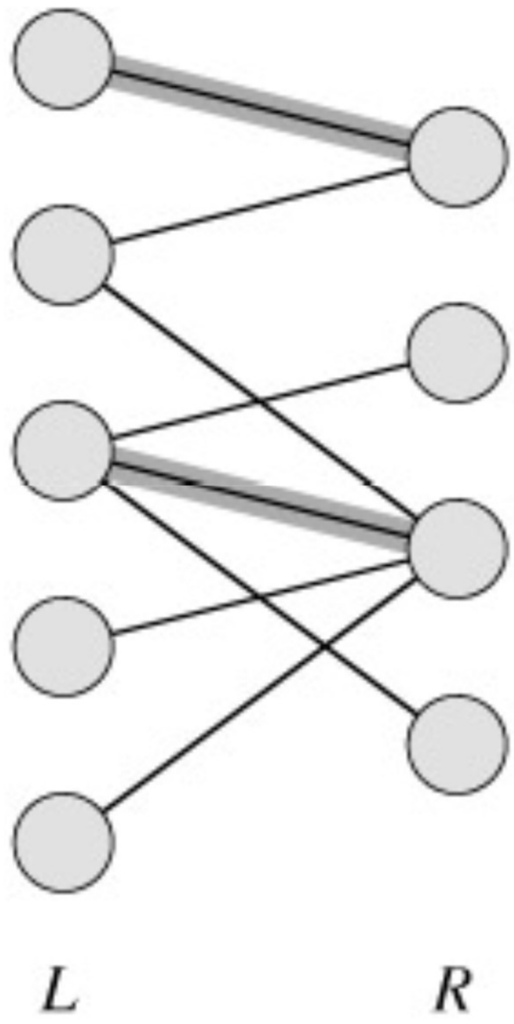


Maximum bipartite matching

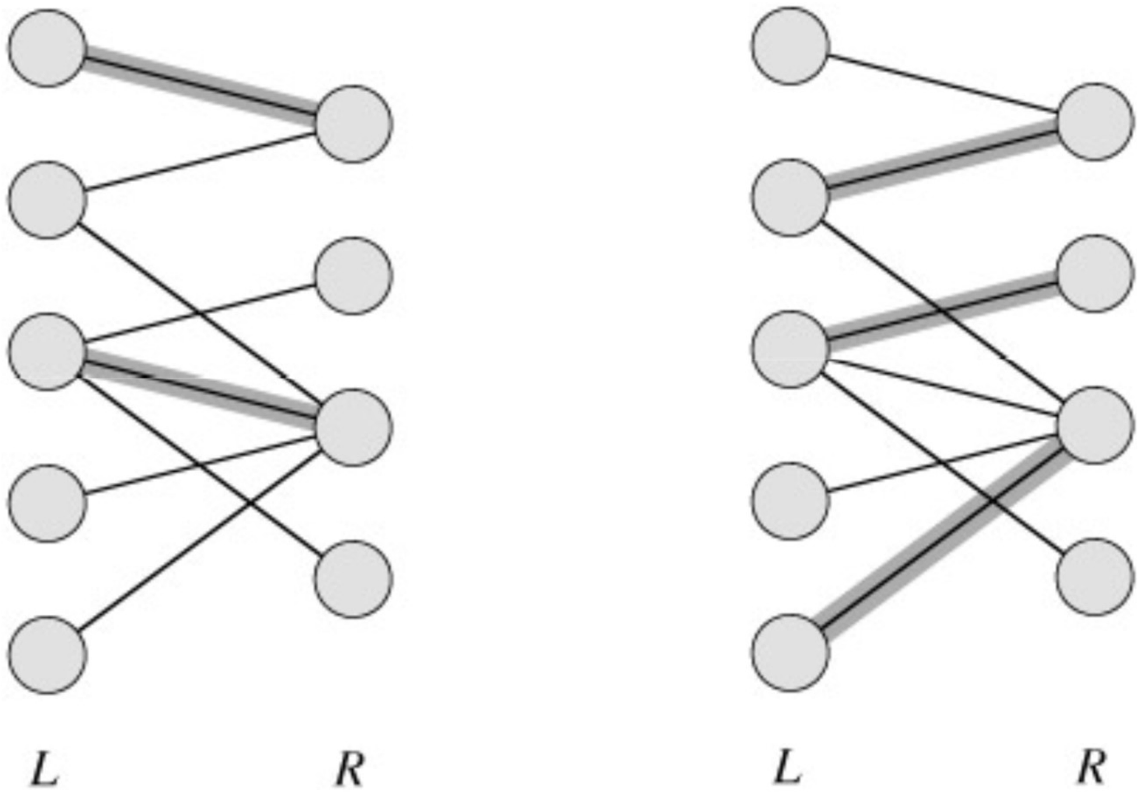
Maximum Bipartite Matching

- *A bipartite graph is a graph G=(V,E) in which V can be divided into two parts L and R such that every edge in E is between a vertex in L and a vertex in R.*

- *e.g. vertices in L represent skilled workers and vertices in R represent jobs. An edge connects workers to jobs they can perform.*

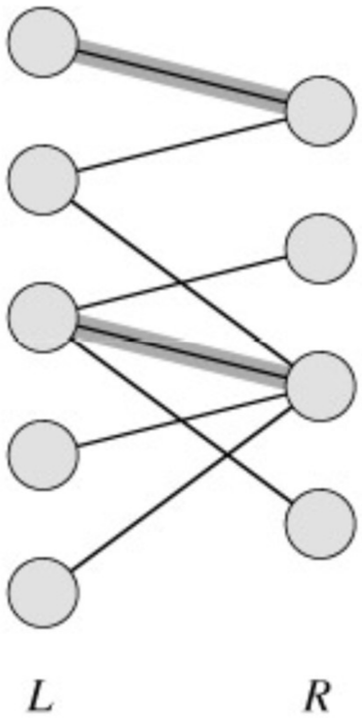


- A matching in a graph is a subset M of E, such that for all vertices v in V, at most one edge of M is incident on v.

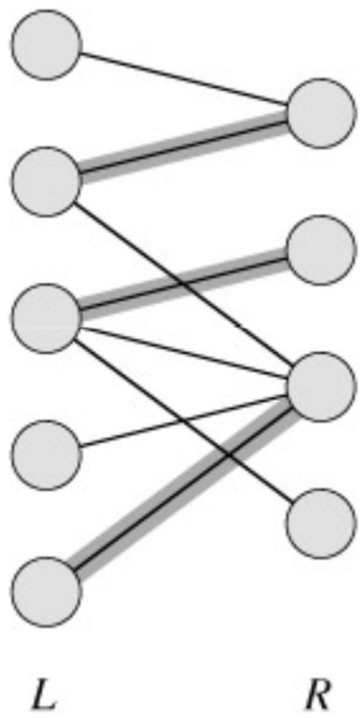


- A **maximum matching** is a matching of maximum cardinality (maximum number of edges).

not maximum



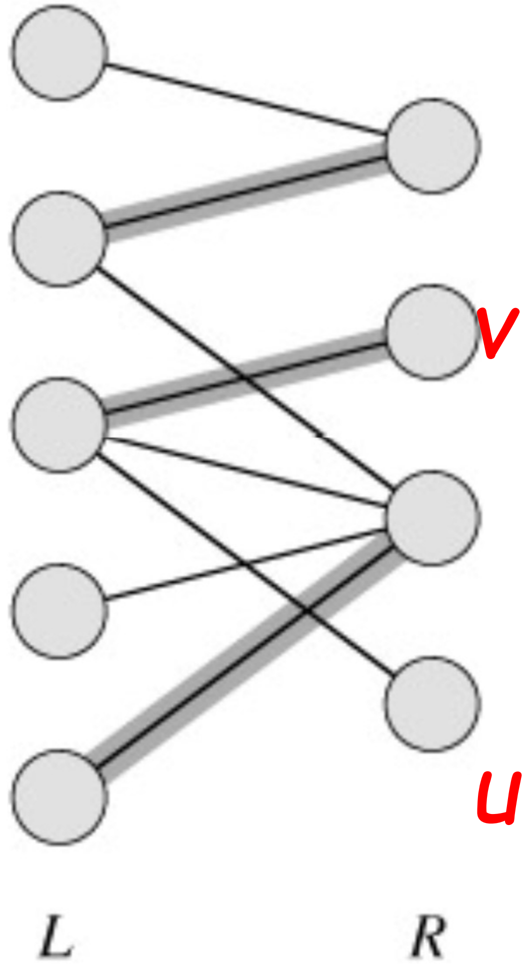
maximum



A Maximum Matching

- No matching of cardinality 4, because only one of v and u can be matched.

- In the workers-jobs example a max-matching provides work for as many people as possible.

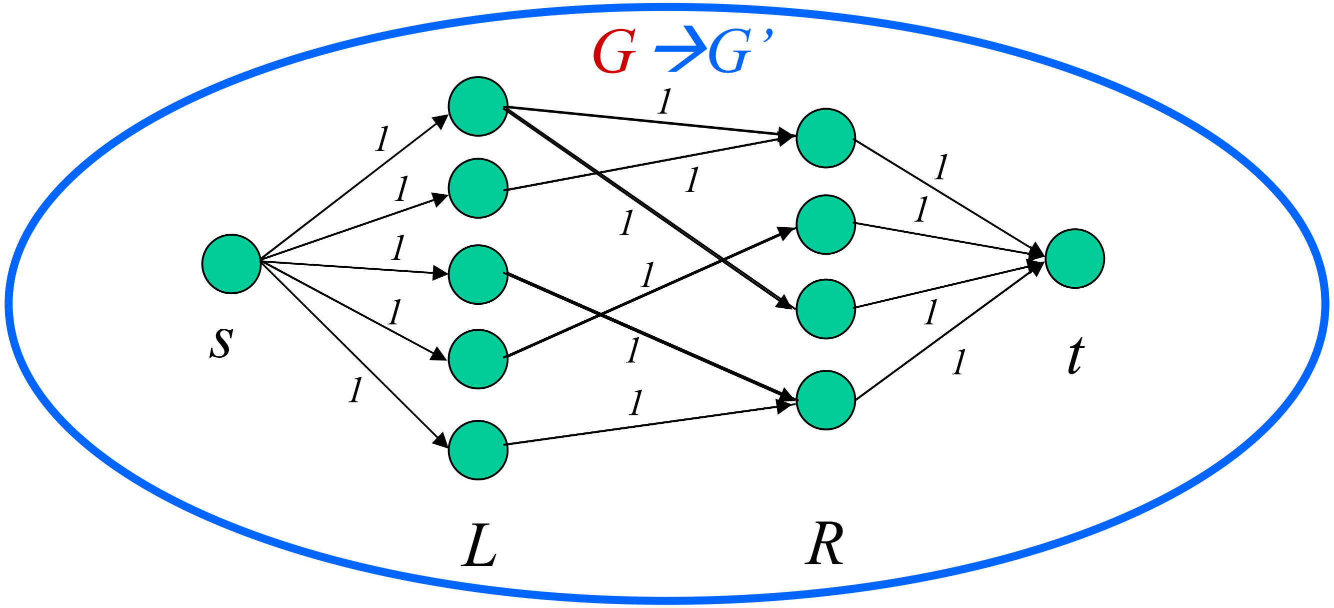


Solving the Maximum Bipartite Matching Problem

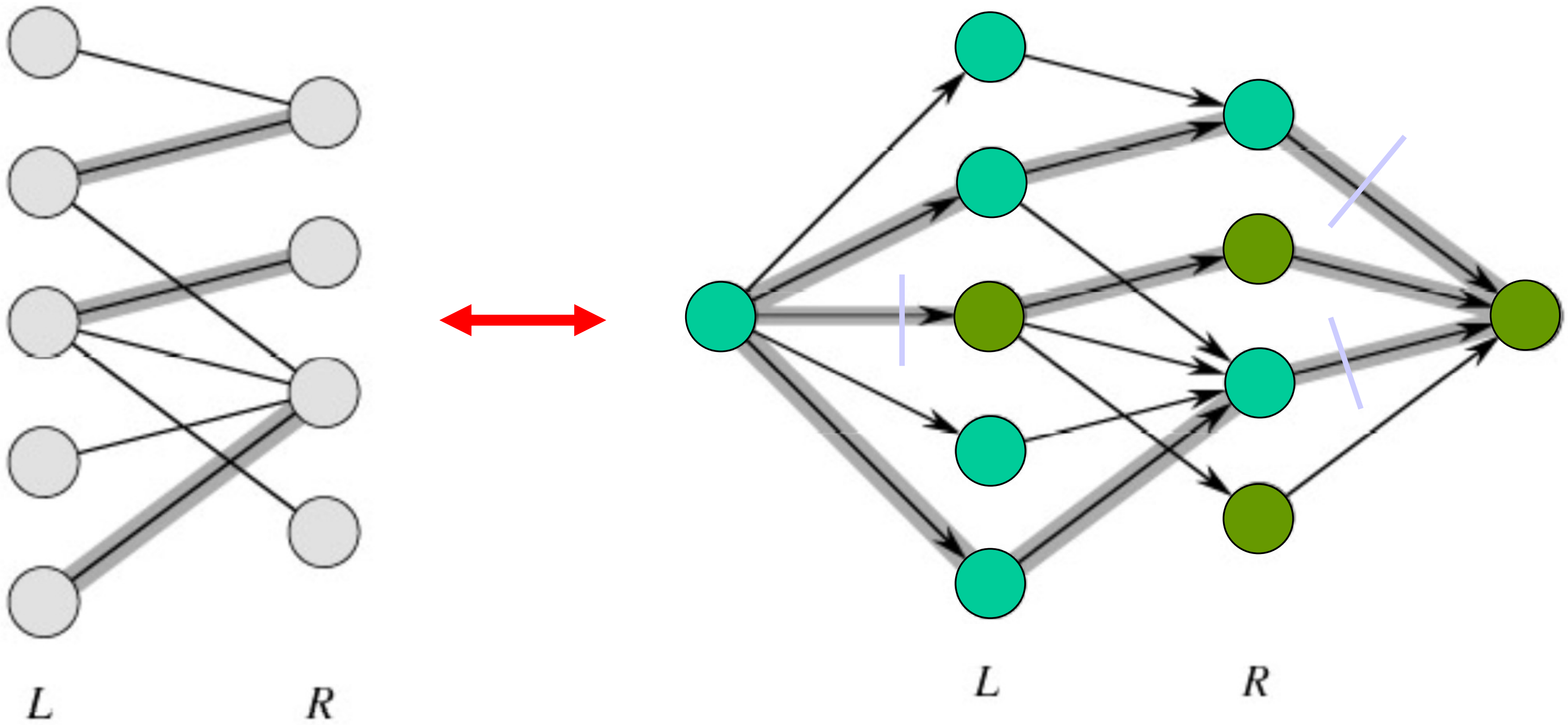
- Reduce the maximum bipartite matching problem on graph **G** to the max-flow problem on a corresponding flow network **G'**.
- Solve using Ford-Fulkerson method.

Corresponding Flow Network

- To form the corresponding flow network **G'** of the bipartite graph **G**:
  - Add a source vertex s and edges from s to L.
  - Direct the edges in E from L to R.
  - Add a sink vertex t and edges from R to t.
  - Assign a capacity of 1 to all edges.
- Claim: max-flow in **G'** corresponds to a max-bipartite-matching on **G**.



Example



$|M| = 3$ ←→ max flow $= |f| = 3$