算法设计与分析

Lecture 9: Network Flow and Matching

卢杨

厦门大学信息学院计算机科学系

luyang@xmu.edu.cn

MAX-FLOW PROBLEM

Network

Examples of a network

- Liquids flowing through pipes
- Parts through assembly lines
- Current through electrical network
- Information through communication network
- Signal through neural network in our brain
- Goods transported on the road…

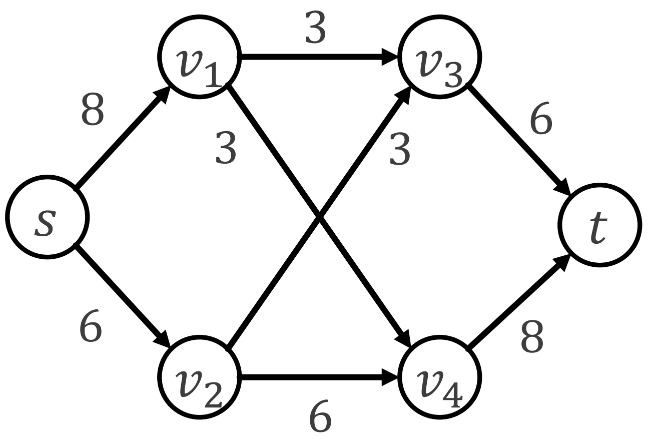
Network

Definition 9.1

A network $G = (V, E)$ is a directed graph in which each edge $(u, v) \in E$ has a nonnegative capacity (容量) $c(u, v) \geq 0$. If $(u, v) \notin E$, we assume that $c(u, v) = 0$.

We distinguish two vertices in a flow network: a source (源点) s and a sink (汇点) t, where s only has outedges (出边) and s only has inedges (入边).

Every vertex lies on some path from the source to the sink.

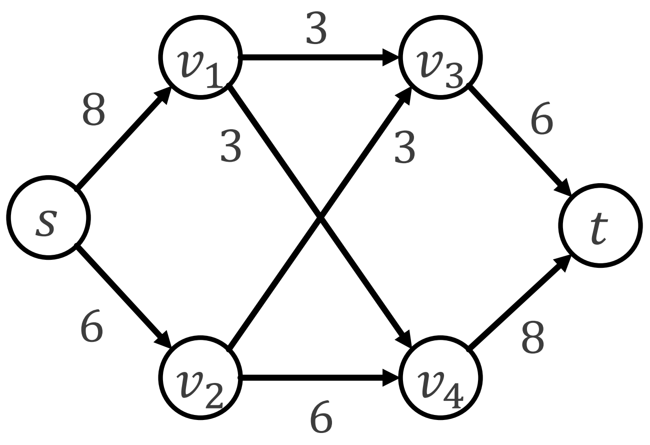


$$c(v_1, v_3) = 3$$
$$c(v_1, v_2) = 0$$

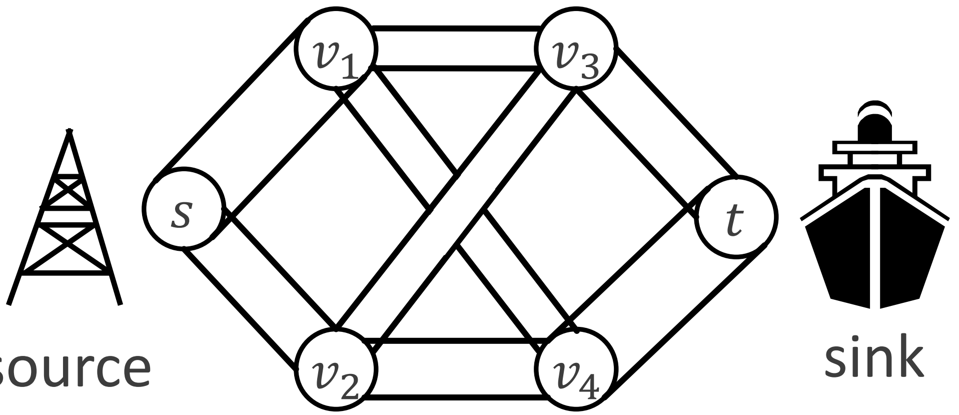
厦门大学信息学院 SCHOOL OF INFORMATICS XIAMEN UNIVERSITY 厦门大学计算机科学系 Computer Science Department of Xiamen University

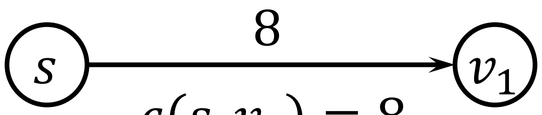
Capacity

Flow network

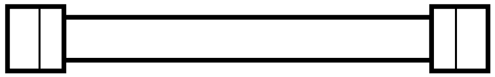


Example: Oil Pipeline

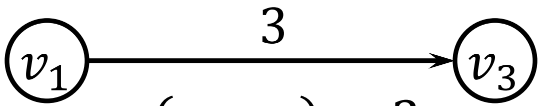




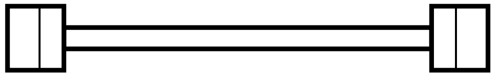
$c(s, v_1) = 8$



Big pipe



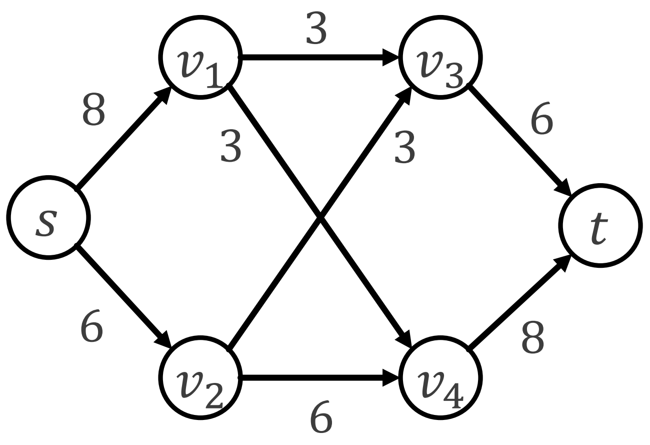
$c(v_1, v_3) = 3$



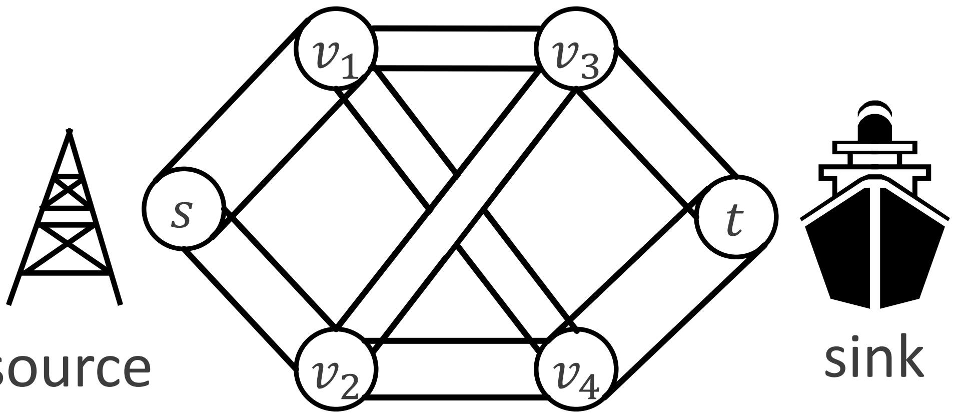
Small pipe

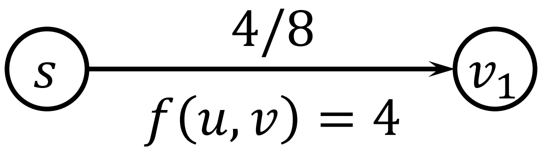
Flow

Flow network



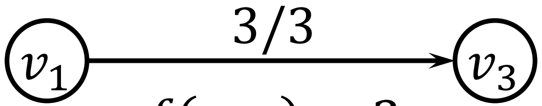
Example: Oil Pipeline





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

Flow below capacity

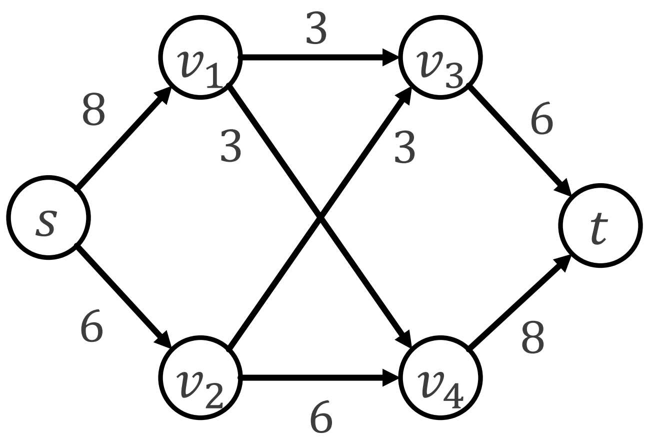


$$f(u,v) = 3$$

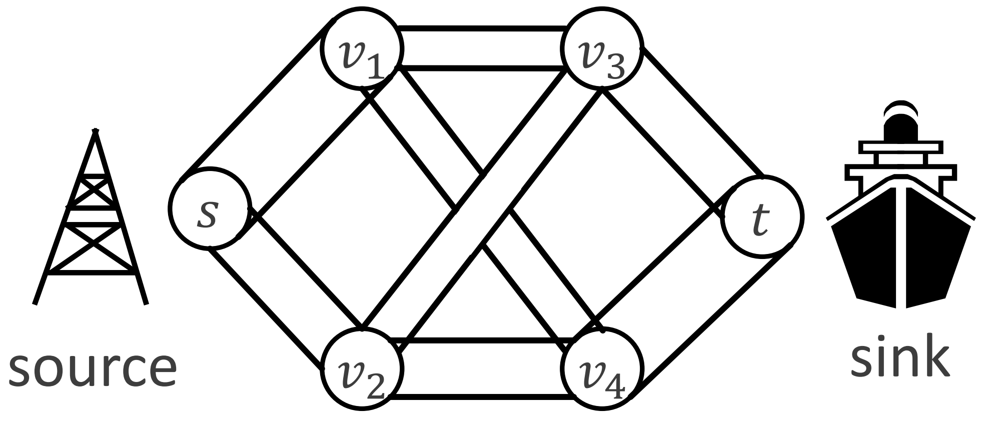
Maximum flow

Max-Flow Problem

Flow network



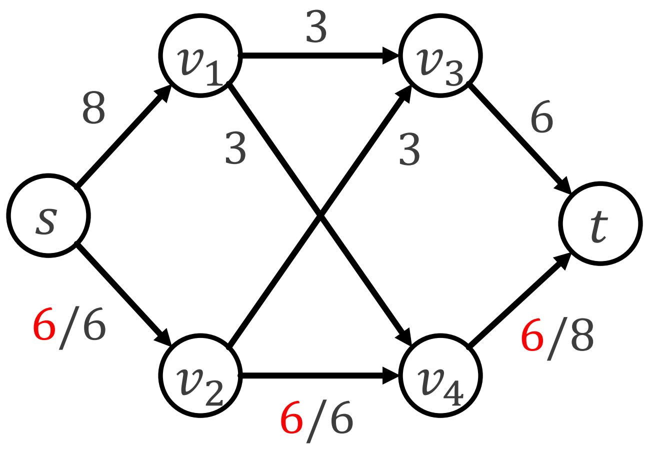
Example: Oil Pipeline



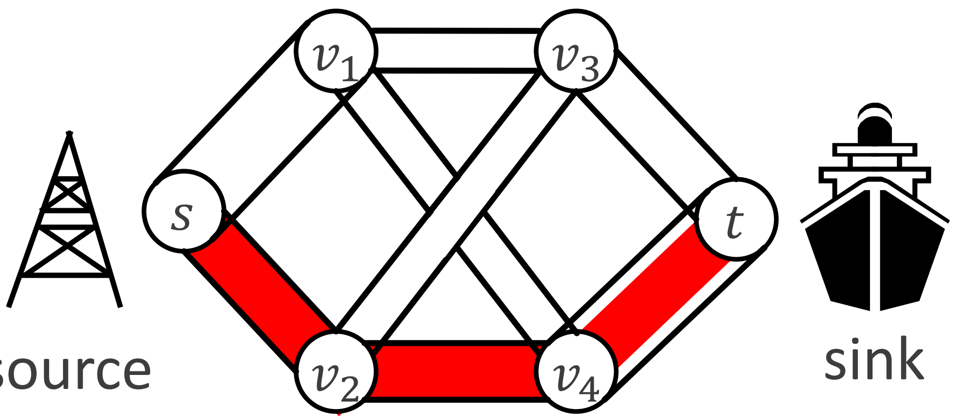
- Informal definition of the max-flow problem: What is the greatest rate at which material can be shipped from the source to the sink without violating any capacity constraints?

Max-Flow Problem

Flow network



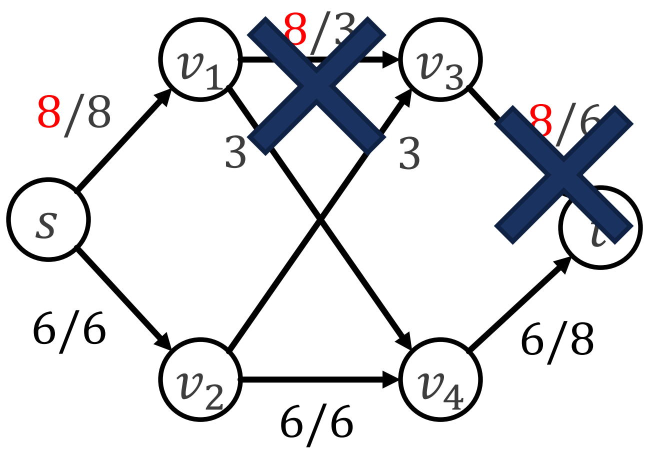
Example: Oil Pipeline



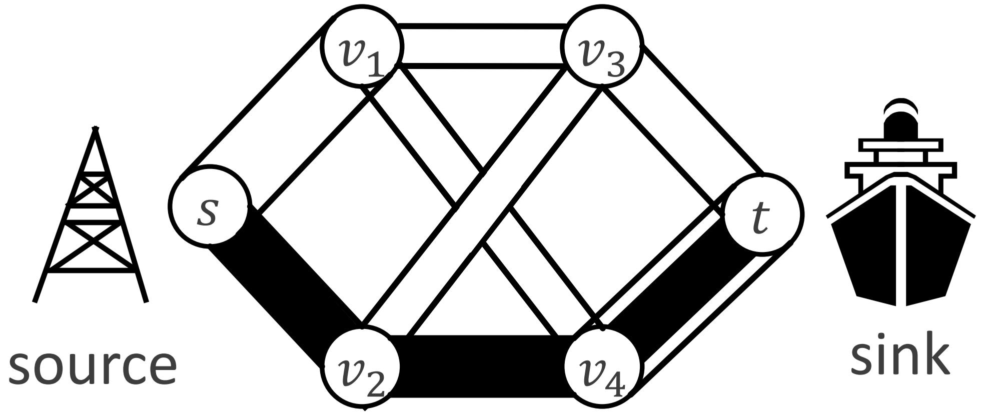
$\boxed{flow = 6}$

Max-Flow Problem

Flow network



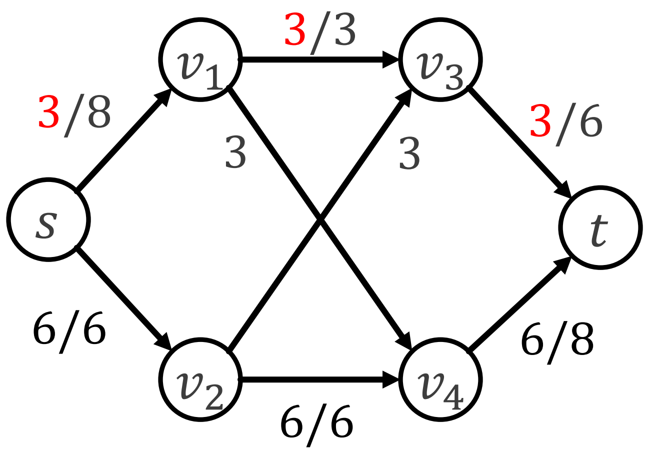
Example: Oil Pipeline



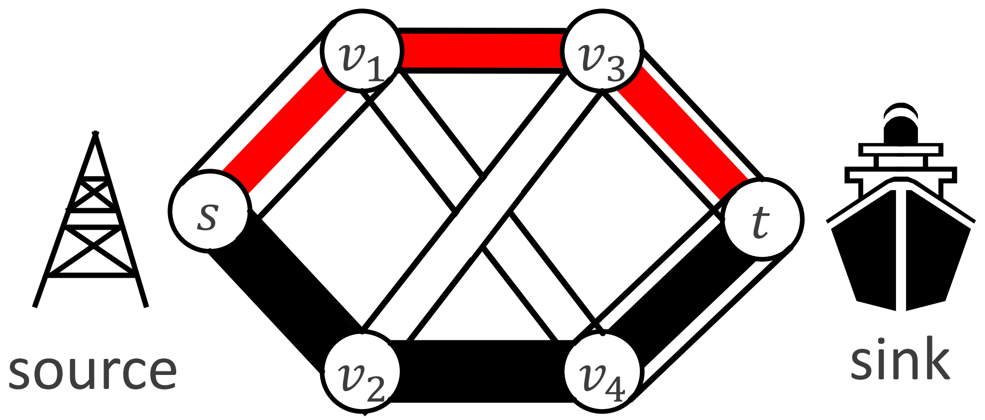
Flow can't exceed capacity!

Max-Flow Problem

Flow network



Example: Oil Pipeline

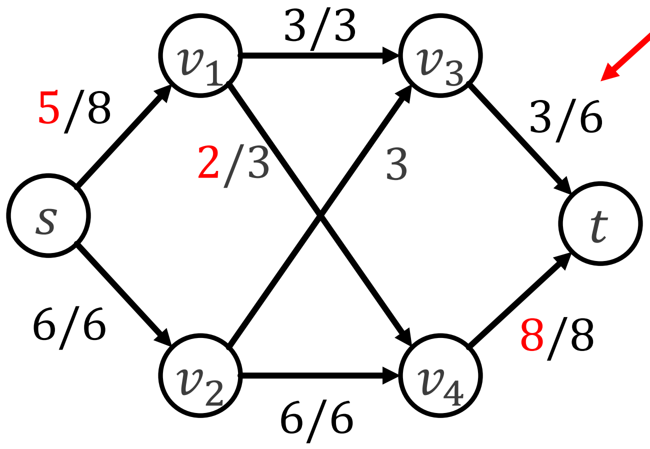


$$\boxed{flow = 9}$$

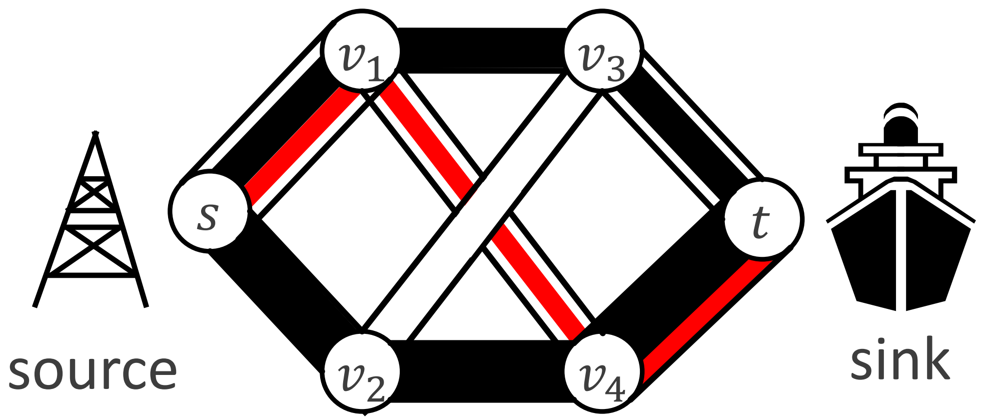
Max-Flow Problem

Can we improve over here?

Flow network



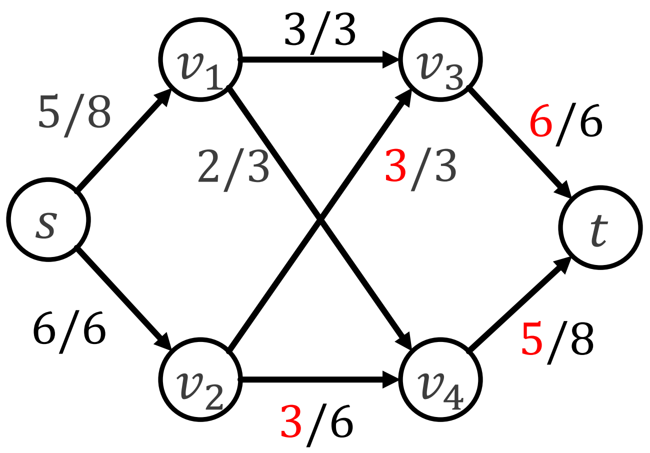
Example: Oil Pipeline



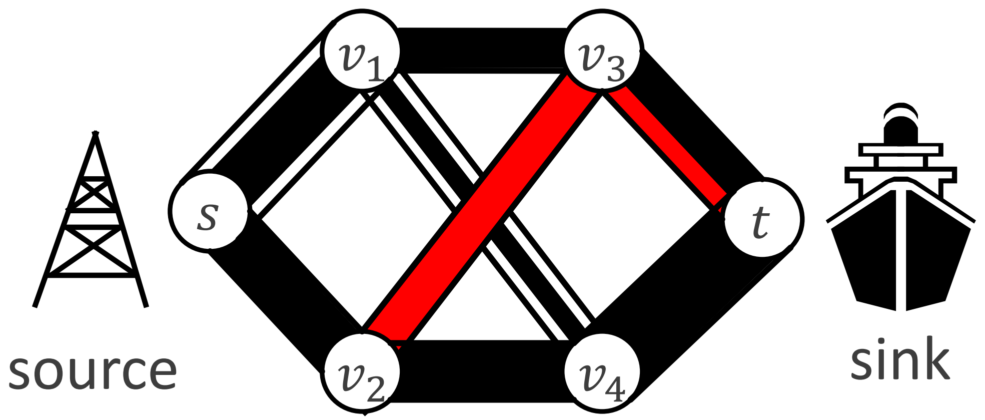
$$\boxed{flow = 11}$$

Max-Flow Problem

Flow network



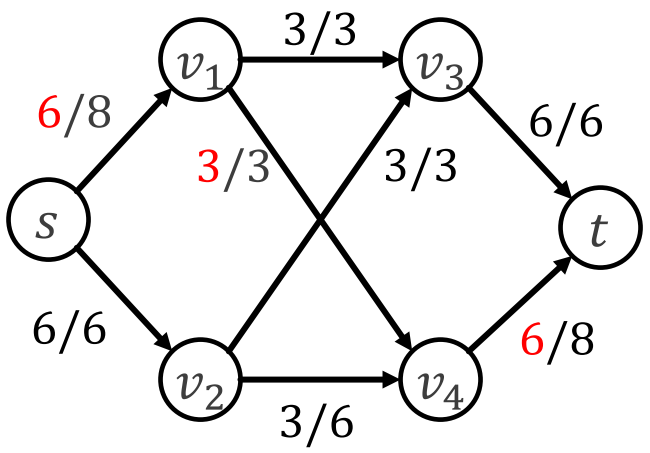
Example: Oil Pipeline



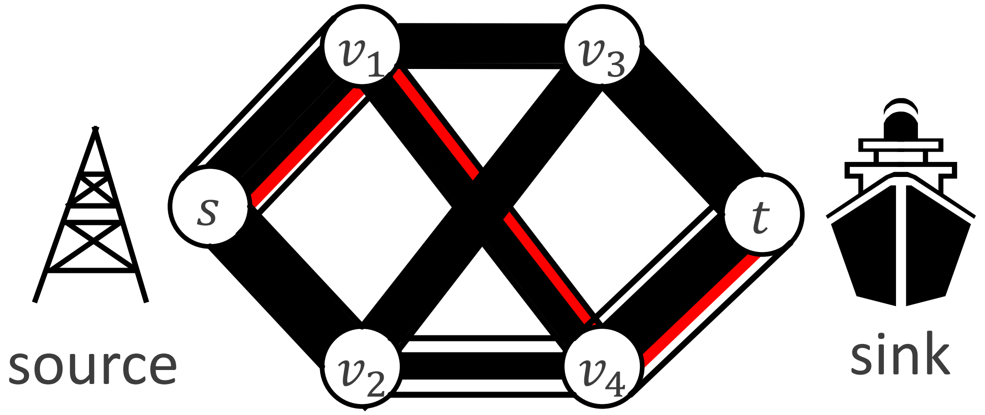
$$\boxed{flow = 11}$$

Max-Flow Problem

Flow network



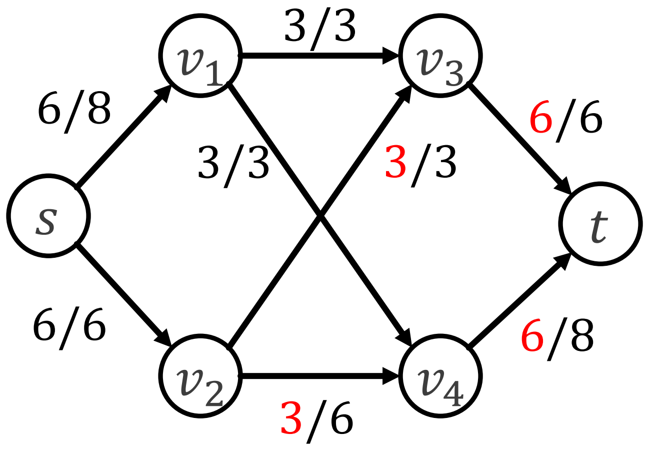
Example: Oil Pipeline



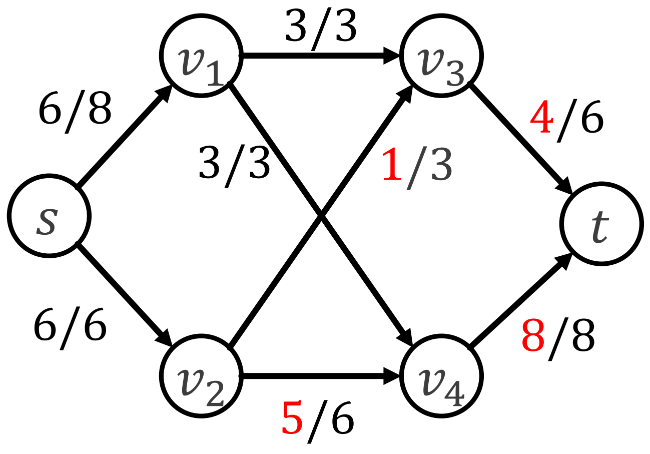
$$\boxed{flow = 12}$$

Max-Flow Problem

Flow network



Flow network



$$\boxed{flow = 12}$$

Flow

Definition 9.2

A real valued function $f: V \times V \rightarrow R$ in $G = (V, E)$ is called flow (流) if it satisfies 3 properties:

(1) Capacity constraint (容量约束): $\forall u, v \in V$:

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

(2) Skew symmetry (反对称): $\forall u, v \in V$:

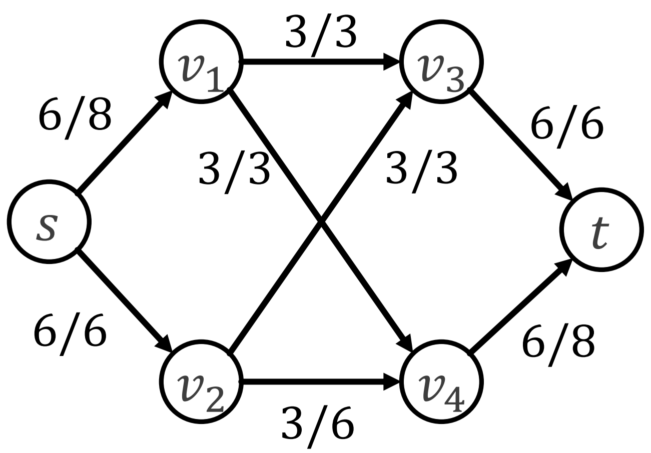
$$f(u, v) = -f(v, u).$$

(3) Flow conservation (流守恒): $\forall u \in V - \{s, t\}$:

$$\sum_{v \in V} f(u, v) = 0 .$$

where $f(u, v)$ is called the flow from vertex u to vertex v. If $f(u, v) = c(u, v)$, we call (u, v) a saturated edge (饱和边).

Flow Properties



We don't draw negative flow and zero capacity on the graph!

For v_1:

$f(v_1, s) = -6$

$f(v_1, v_3) = 3$

$f(v_1, v_4) = 3$

$\sum_{v \in V} f(v_1, v) = 0$

For v_3:

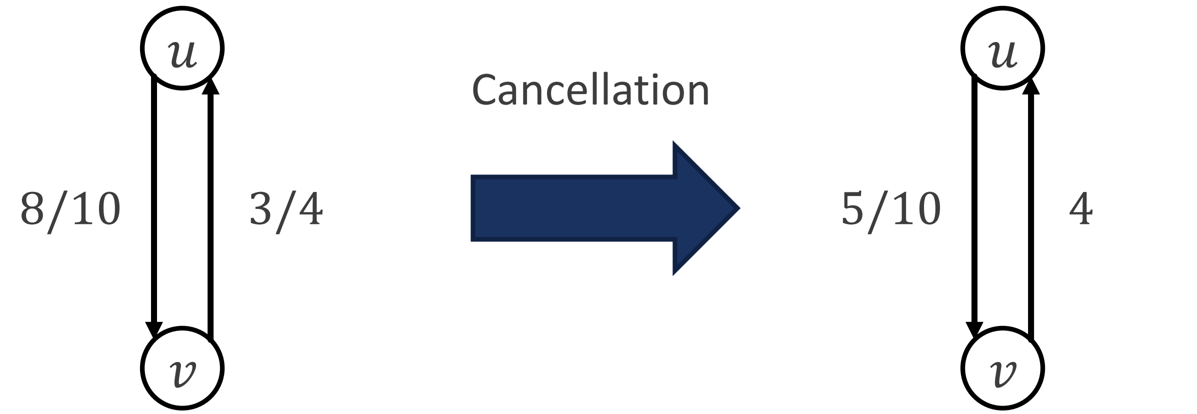
$f(v_3, v_1) = -3$

$f(v_3, v_2) = -3$

$f(v_3, t) = 6$

$\sum_{v \in V} f(v_3, v) = 0$

Cancellation



$$f(u,v) = 8$$
$$f(v,u) = 3$$

$$f(u,v) = 5$$
$$f(v,u) = -5$$

Not satisfy skew symmetry

Flow

Definition 9.2

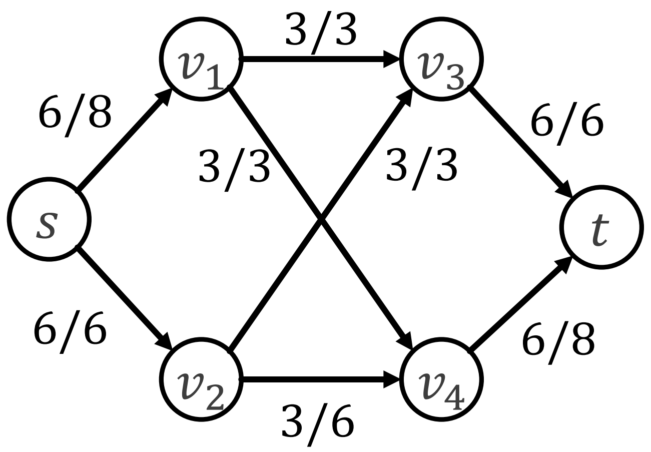
The value of a flow f (f的流量) is defined as

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

that is, the total flow out of the source s.

- Here, the $|\cdot|$ notation denotes flow value, not absolute value or cardinality.
- Formal definition of the max-flow problem: The max-flow problem is to find a valid flow for a given weighted directed graph G, that has the maximum value over all valid flows.

Flow



$$|f| = \sum_{v \in V} f(s,v) = f(s,v_1) + f(s,v_2) = 6 + 6 = 12.$$

厦门大学信息学院 SCHOOL OF INFORMATICS XIAMEN UNIVERSITY 厦门大学计算机科学系 Computer Science Department of Xiamen University

MAX-FLOW PROBLEM

THE FORD-FULKERSON METHOD

The Ford-Fulkerson Method

- The Ford-Fulkerson method is a way to find the max-flow.
- It has three important concepts:

  (1) Residual networks (剩余网络)

  (2) Augmenting paths (增广路径)

  (3) Cuts of flow networks (网络的割)

Residual Networks

Definition 9.3

Given a flow f on G with capacity function c, the residual capacity (剩余容量) function c_f on the set of pairs of vertices is defined as:

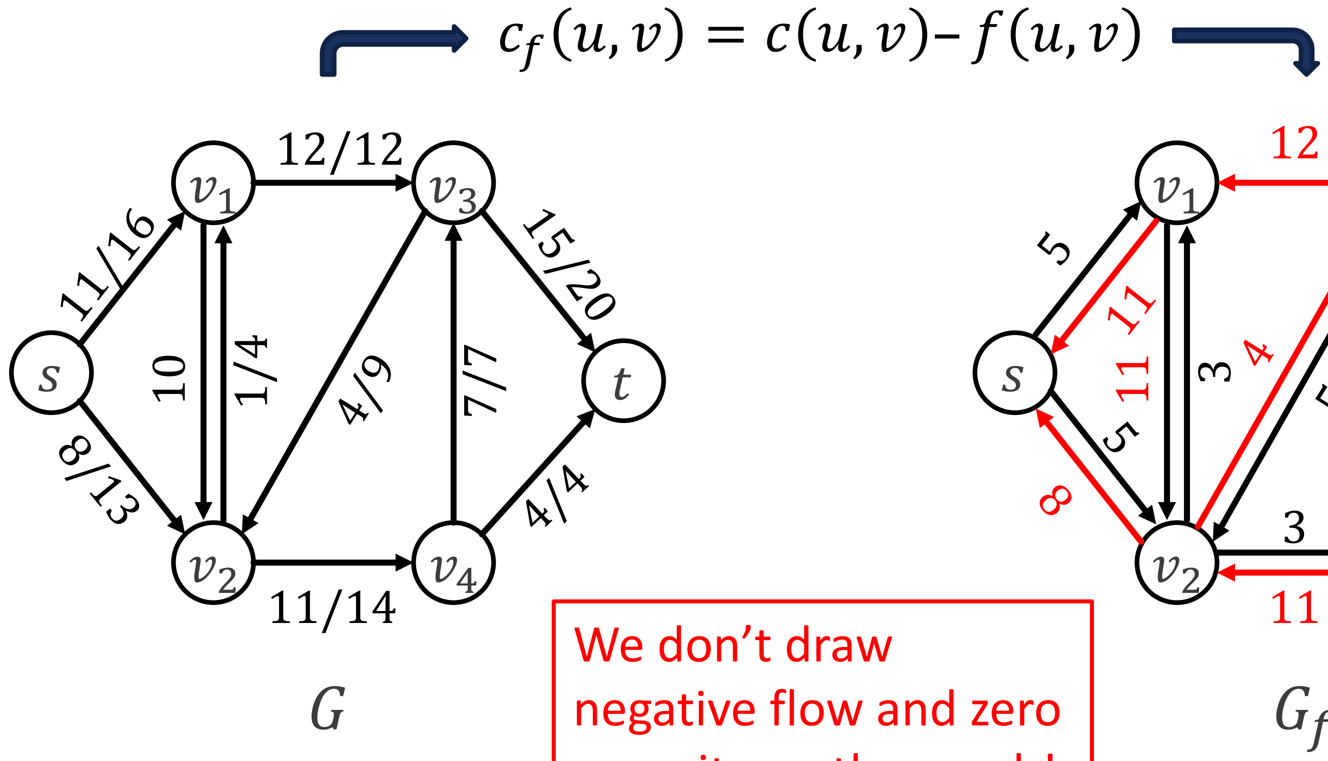
$$c_f(u, v) = c(u, v) - f(u, v)$$

for each pair of vertices $u, v \in V$.

The residual network (剩余网络) for f is the direct graph $G_f = (V, E_f)$, with capacities defined by c_f and edge set $E_f = \{(u, v) | c_f(u, v) > 0\}$.

Residual Networks

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

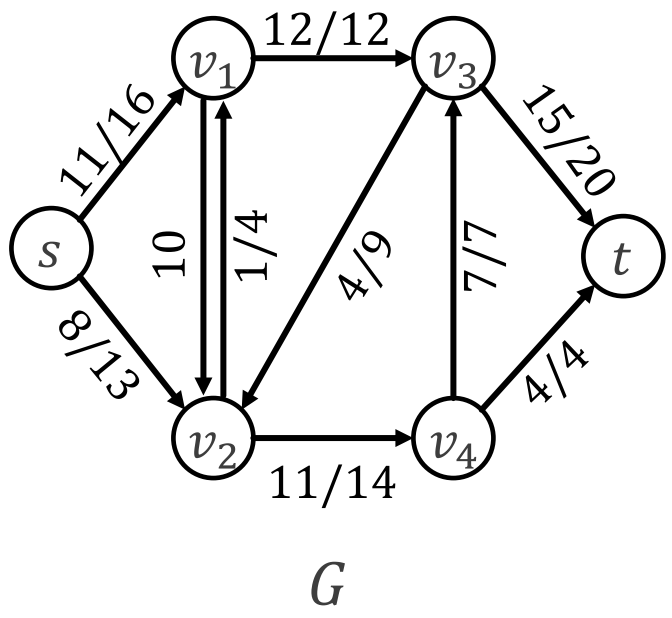


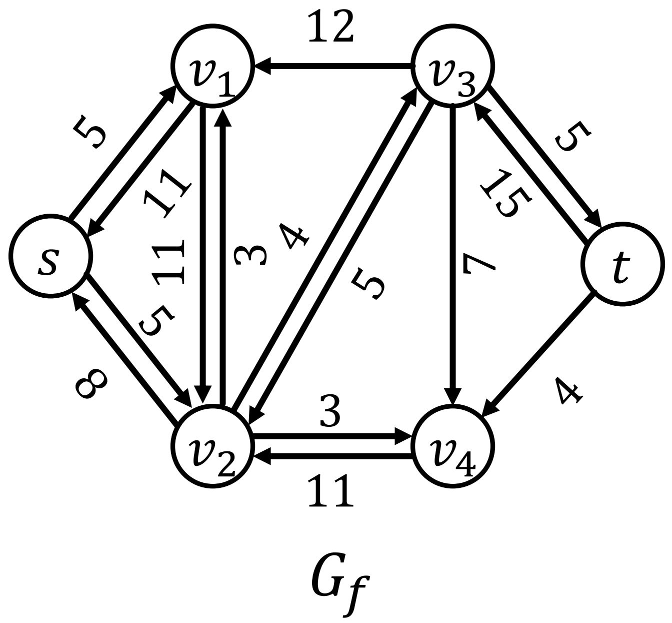
We don't draw negative flow and zero capacity on the graph!

Note: G_f is different with different flow f.

厦门大学信息学院
SCHOOL OF INFORMATICS XIAMEN UNIVERSITY
厦门大学计算机科学系
Computer Science Department of Xiamen University

Residual Networks





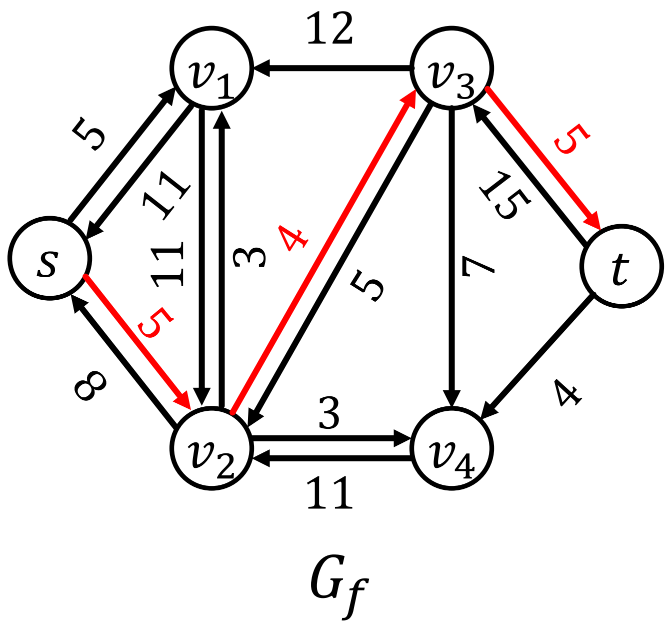
- Capacity in residual network simply means how much you can increase the flow at most.

厦门大学信息学院
SCHOOL OF INFORMATICS XIAMEN UNIVERSITY
厦门大学计算机科学系
Computer Science Department of Xiamen University

Augmenting Paths

Definition 9.5

Given a residual network $G_f = (V, E_f)$, an augmenting path (增广路径) is a direct path from s to t.



If we can find such a path in the residual network, it means we can still increase the flow!

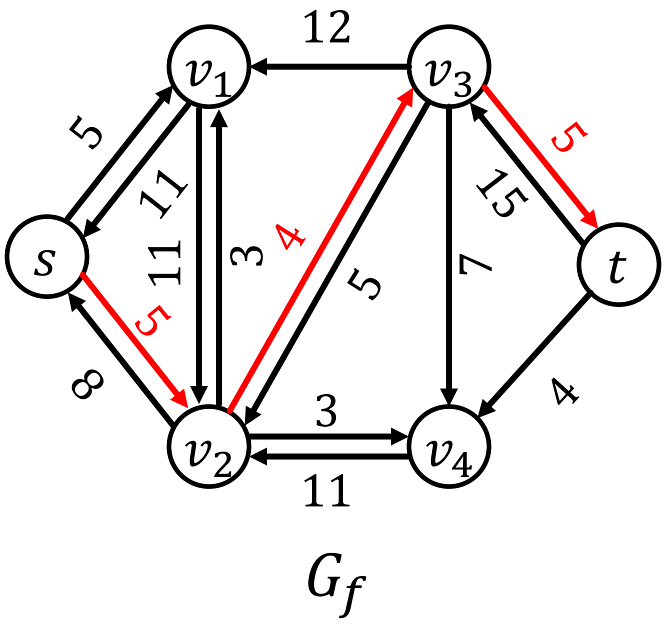
Bottleneck Capacity

Definition 9.5

Given a residual network G_f and an augmenting path p from s to t, we define $c_f(p)$ as the bottleneck capacity (瓶颈容量):

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

- $c_f(p)$ is actually the maximum amount by which we can increase the flow on each edge in an augmenting path p.



$\boxed{c_f(p) = 4}$

Augmenting Paths

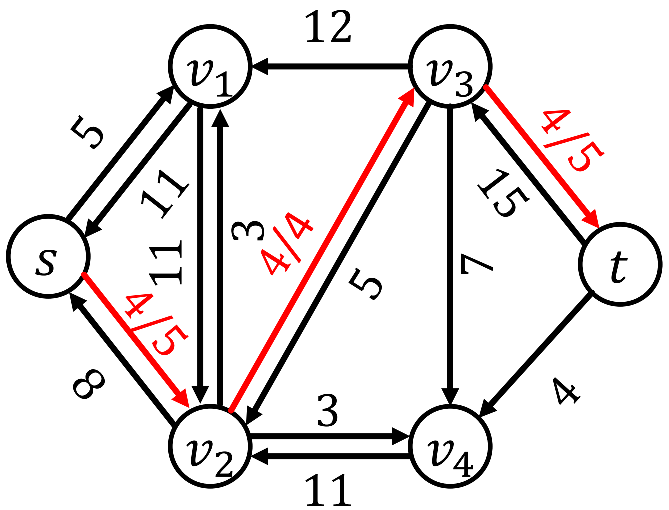
Lemma 9.2

Let p be an augmenting path in G_f.
Define a function $f_p: V \times V \rightarrow R$ by

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

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

- If we can find a augmenting path, there exists a flow of G_f.



G_f

$$f_p(s,v_2) = f_p(v_2,v_3) = f_p(v_3,t) = c_f(p) = 4$$

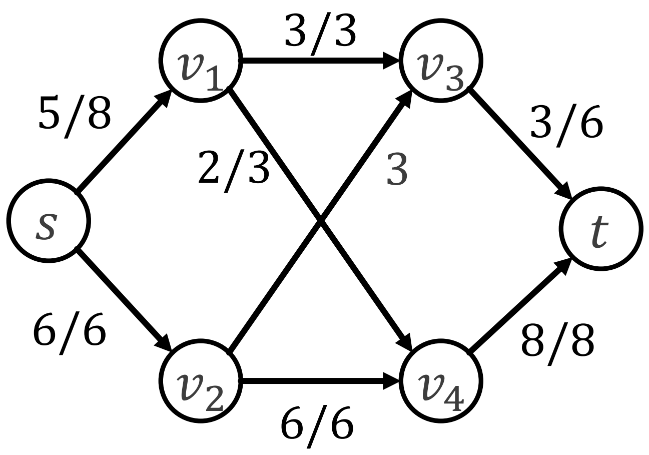
Definitions So Far...

Given a network $G = (V, E)$, now we have a bunch of definitions:

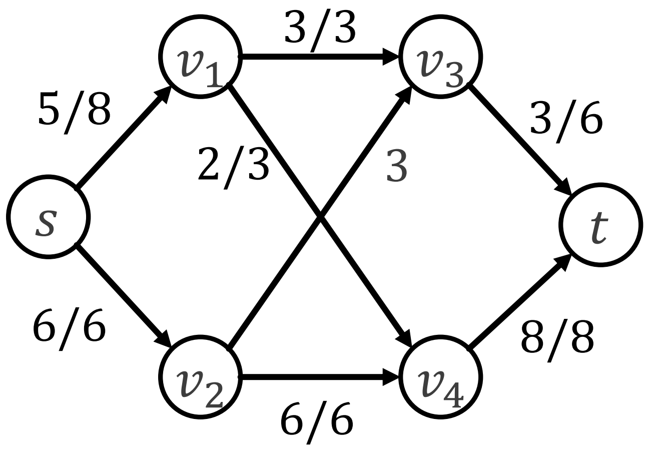
- $c(u, v)$: capacity of $(u, v) \in E$.
- $f(u, v)$: flow of $(u, v) \in E$.
- f and $|f|$: a flow of G and its value.
- $G_f = (V, E_f)$: residual network of f.
- $c_f(u, v)$: residual capacity of $(u, v) \in E_f$.
- p: an augmenting path in G_f.
- $c_f(p)$: bottleneck capacity.
- f_p: a flow of G_f with augmenting path p.

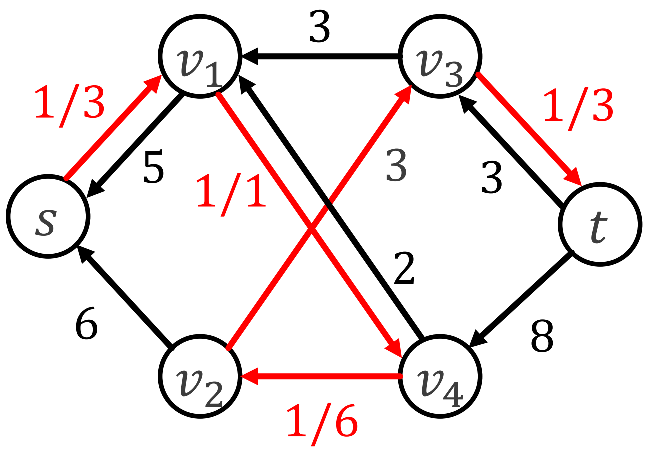
Classroom Exercise

Compute the residual network and find an augmenting path and its bottleneck capacity.



Classroom Exercise





Augmenting Paths

- Once we obtain an augmenting path, we can improve f by f_p. We first define the sum of flows.

Definition 9.4

Given two flows f_1 and f_2 on G, let $(f_1 + f_2)(u, v) = f_1(u, v) + f_2(u, v)$, $f_1 + f_2$ is called the sum of flows f_1 and f_2.

Augmenting Paths

Lemma 9.1

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 $|f + f'| = |f| + |f'|$.

- This Lemma tells you: if you find a flow f' of G_f, no worry and simply add it onto f, the resulting flow sum is also a flow in G.
- Adding f' to f will not explode. Why?

Augmenting Paths

To show that $f + f'$ is also a flow in G, we need to show that $f + f'$ satisfies the three properties of flow, and $|f + f'| = |f| + |f'|$.

Proof:

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

$$\begin{aligned}(f + f')(u,v) &= f(u,v) + f'(u,v) \\ &\leq f(u,v) + c_f(u,v) \\ &= f(u,v) + \big(c(u,v) - f(u,v)\big) \\ &= c(u,v)\end{aligned}$$

Capacity constraint of f'.

Definition of c_f.

厦门大学信息学院 SCHOOL OF INFORMATICS XIAMEN UNIVERSITY

厦门大学计算机科学系 Computer Science Department of Xiamen University

Augmenting Paths

Proof (cont'd):

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

$$\begin{aligned}(f+f')(u,v) &= f(u,v) + f'(u,v) \\ &= -f(v,u) - f'(v,u) \\ &= -\big(f(v,u) + f'(v,u)\big) \\ &= -(f+f')(v,u)\end{aligned}$$

Skew symmetry of f and f'.

Definition of flow sum

厦门大学信息学院 SCHOOL OF INFORMATICS XIAMEN UNIVERSITY 厦门大学计算机科学系 Computer Science Department of Xiamen University

Augmenting Paths

Proof (cont'd):

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

$$\begin{aligned}\sum_{v \in V} (f + f')(u, v) &= \sum_{v \in V} \big(f(u, v) + f'(u, v)\big) \\ &= \sum_{v \in V} f(u, v) + \sum_{v \in V} f'(u, v) \\ &= 0 + 0 = 0\end{aligned}$$

厦门大学信息学院
SCHOOL OF INFORMATICS XIAMEN UNIVERSITY
厦门大学计算机科学系
Computer Science Department of Xiamen University

Augmenting Paths

Proof (cont'd):

Finally, we prove $|f + f'| = |f| + |f'|$.

$$\begin{aligned}|f + f'| &= \sum_{v\in V}(f + f')(s,v)\\ &= \sum_{v\in V} f(s,v) + \sum_{v\in V} f'(s,v)\\ &= |f| + |f'|\end{aligned}$$

Augmenting Paths

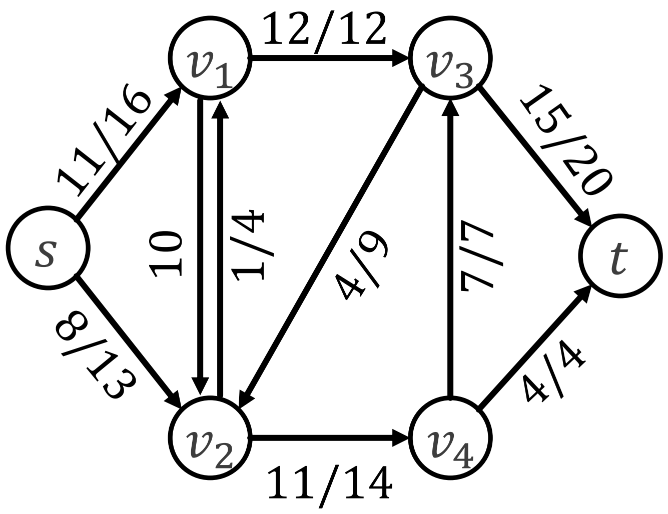
- Everything is prepared for increase the value of flow to reach max-flow.

Proposition 9.1

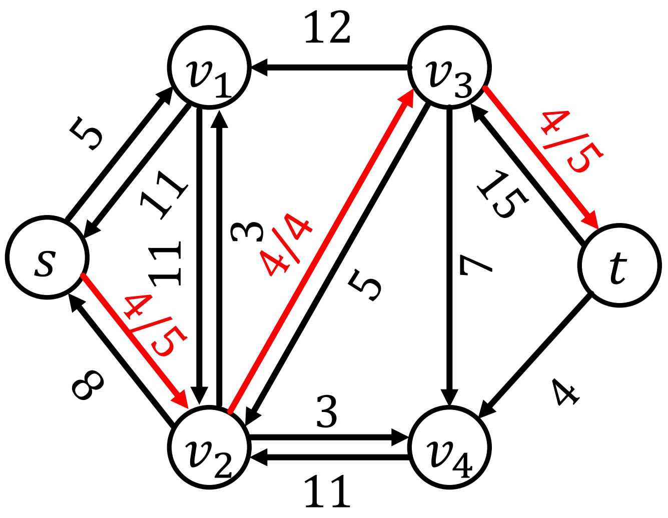
Let $G = (V, E)$ be a flow network, let f be a flow in G, p be an augmenting path in G_f, f_p be a flow of G_f with augmenting path p.

Then $f + f_p$ is a flow in G with value $|f| + |f_p| > |f|$.

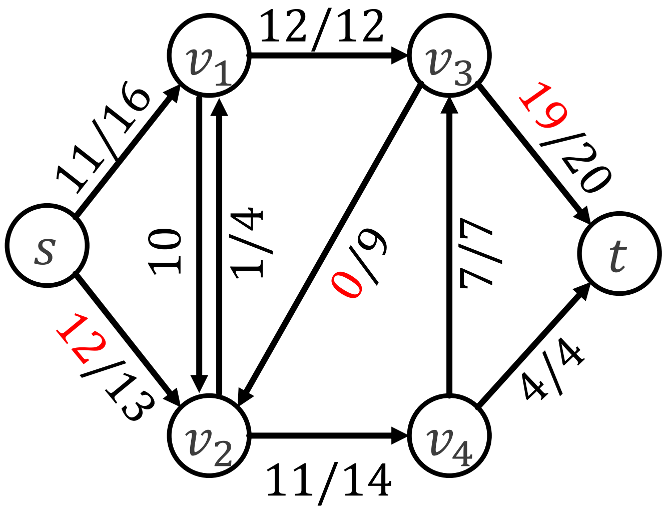
- This Proposition tells you: Adding f_p must result in larger flow value.



G and f
$|f| = 19$



G_f and f_p
$|f_p| = 4$



G and $f + f_p$
$|f + f_p| = 23$

Augmenting Paths

- Proposition 9.1 indicates:

$$\boxed{\text{Find a } f_p} \longrightarrow \boxed{\text{Can increase } |f|}$$

- By contrapositive law (假言易位), we know:

$$\boxed{\text{Can't find a } f_p} \longleftarrow \boxed{\text{Can't increase } |f|}$$

- However, we don't know:

$$\boxed{\text{Can't find a } f_p} \xrightarrow{?} \boxed{\text{Can't increase } |f|}$$

厦门大学信息学院 SCHOOL OF INFORMATICS XIAMEN UNIVERSITY 厦门大学计算机科学系 Computer Science Department of Xiamen University

Cuts of Flow Networks

Definition 9.7

A cut (割) (S, T) of a flow network $G = (V, E)$ is a partition of V into S and $T = V - S$ such that $s \in S$ and $t \in T$.

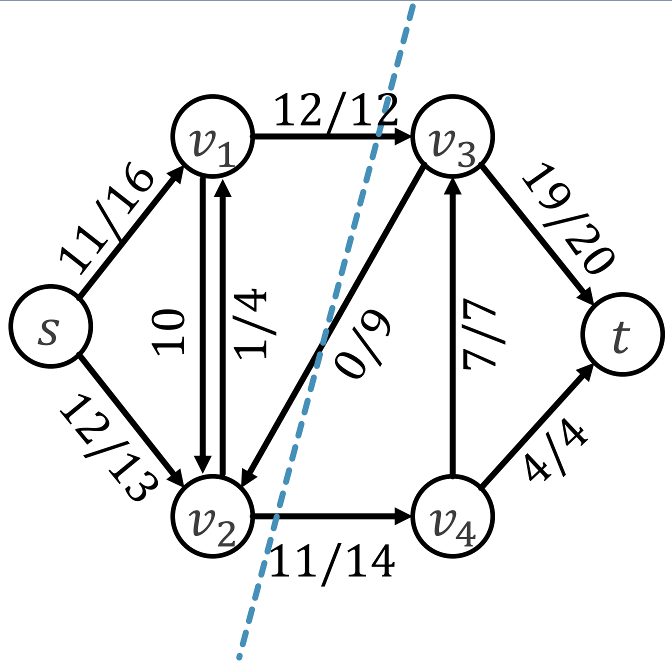
The capacity $c(S, T)$ of the cut $\{S, T\}$ is denoted as:

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

The flow $f(S, T)$ across the cut $\{S, T\}$ is denoted as:

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

Cuts of Flow Networks



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

$$c(S,T) = c(v_1, v_3) + c(v_2, v_4) = 12 + 14 = 26$$

$$f(S,T) = f(v_1, v_3) + f(v_2, v_4) + f(v_2, v_3) = 12 + 11 - 0 = 23$$

厦门大学信息学院 SCHOOL OF INFORMATICS XIAMEN UNIVERSITY 厦门大学计算机科学系 Computer Science Department of Xiamen University

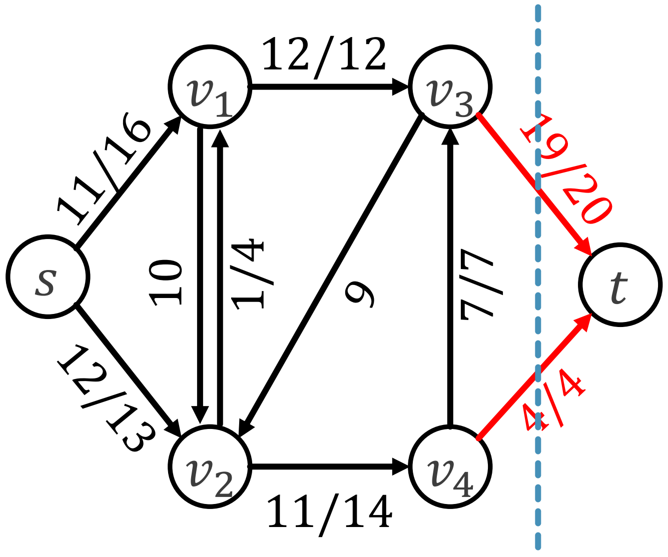
Property of Cuts

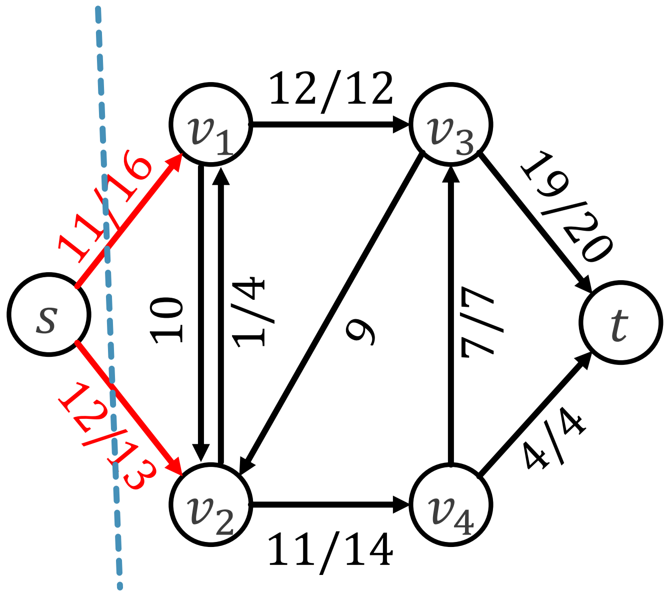
Lemma 9.3

Given a flow network $G = (V, E)$, for any cut $\{S, T\}$ and a flow f, $f(S, T) = |f|$.

- This Lemma tells you: No matter how you cut, the flow across the cut is same and equals to the value of the flow.

Property of Cuts

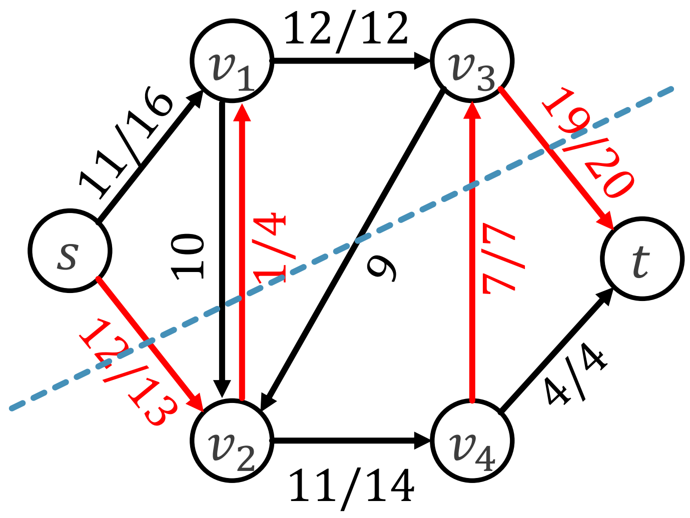


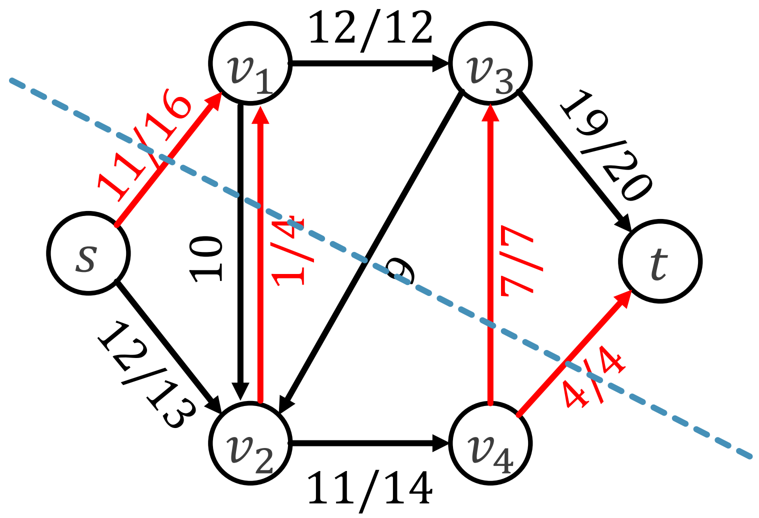


$$f(S,T) = f(v_3,t) + f(v_4,t) = 19 + 4 = 23$$

$$f(S,T) = f(s,v_1) + f(s,v_2) = 11 + 12 = 23$$

Property of Cuts



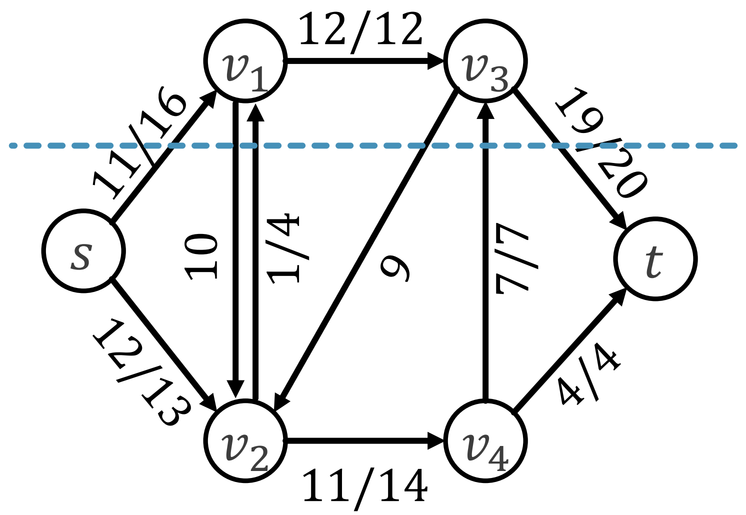


$f(S,T)$
$= f(s, v_2) + f(v_1, v_2) + f(v_3, v_4)$
$+ f(v_3, t)$
$= 12 - 1 - 7 + 19 = 23$

$f(S,T)$
$= f(s, v_1) + f(v_2, v_1) + f(v_4, v_3)$
$+ f(v_4, t)$
$= 11 + 1 + 7 + 4 = 23$

厦门大学信息学院
SCHOOL OF INFORMATICS XIAMEN UNIVERSITY
厦门大学 计算机科学系
Computer Science Department of Xiamen University

Property of Cuts



Not a valid cut! s and t are not separated.

厦门大学信息学院
SCHOOL OF INFORMATICS XIAMEN UNIVERSITY
厦门大学计算机科学系
Computer Science Department of Xiamen University

Property of Cuts

Proof:

By induction on the number of vertices in S.

If $S = \{s\}$, it is true by the definition of flow: $|f| = \sum_{v \in V} f(s, v)$.

Assume it is true for the cut $\{S, T\}$, namely $f(S, T) = |f|$. We show that it also holds for the cut $\{S \cup \{w\}, T - \{w\}\}$ for $w \in T - \{s, t\}$:

$$\begin{aligned} f(S \cup \{w\}, T - \{w\}) &= f(S, T - \{w\}) + f(w, T - \{w\}) \\ &= f(S, T) - f(S, w) + f(w, T) \\ &= f(S, T) + f(w, S) + f(w, T) \\ &= f(S, T) + f(w, V) \\ &= f(S, T) + 0 = |f| \end{aligned}$$

By definition of cut

Flow conservation property

Property of Cuts

Corollary 9.2

The value of any flow f in a flow network G is bounded from above by the capacity of any cut of G, namely $|f| \leq c(S,T)$.

Proof:

$$|f| = f(S,T)$$

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

- It means that max-flow can't exceed min-cut.

Max-Flow Min-Cut Theorem

Theorem 9.1 (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.

- This Theorem tells you: When you can’t find an augmenting path, you have obtained the max-flow.
- We prove (1)->(2), (2)->(3), and (3)->(1).

Max-Flow Min-Cut Theorem

(1) f is a maximum flow in G.

(2) The residual network G_f contains no augmenting paths.

Proof of (1)->(2):

- We prove by contradiction that f is a maximum flow in G but there still exists an augmenting path p in G_f.
- Then by Corollary 9.1, we can augment the flow in G:

$$|f'| = \left|f + f_p\right| > |f|$$

- The flow f' is strictly greater than f which is in contradiction to our assumption that f is a maximum flow.

Max-Flow Min-Cut Theorem

(2) The residual network G_f contains no augmenting paths.

(3) $|f| = c(S, T)$ for some cut (S, T) of G.

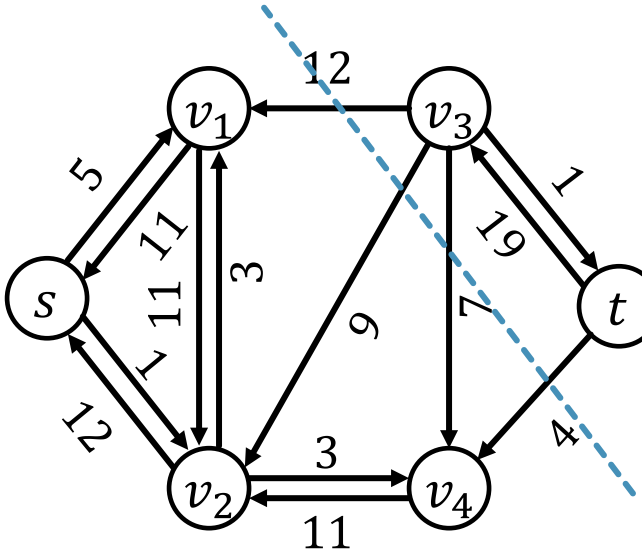
Proof of (2)->(3):

- Define

$S = \{v \in V \mid \exists \text{path } p \text{ from } s \text{ to } v \text{ in } G_f\}$

$T = V - S$

- We have $s \in S$ and $t \notin S$, otherwise there exists an augmenting path from s to t.
- Therefore, (S, T) is a cut of G by definition.



G_f

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

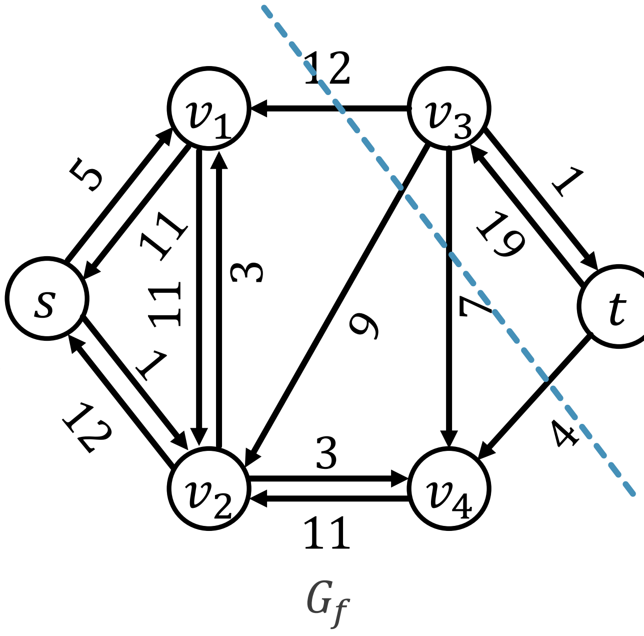
Max-Flow Min-Cut Theorem

(2) The residual network G_f contains no augmenting paths.

(3) $|f| = c(S,T)$ for some cut (S,T) of G.

Proof of (2)->(3) (cont'd):

- All edges (u,v) from S to T are saturated, i.e. $f(u,v) = c(u,v)$.
- Otherwise, there exists a path from u to v, i.e. $(u,v) \in E_f$, which means $v \in S$ contradicting to $v \in T$.
- Therefore, $f(S,T) = c(S,T)$ and $|f| = f(S,T)$ by Lemma 9.3, we have $|f| = c(S,T)$.



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

Max-Flow Min-Cut Theorem

(3) $|f| = c(S, T)$ for some cut (S, T) of G.

(1) f is a maximum flow in G.

Proof of (3)->(1):

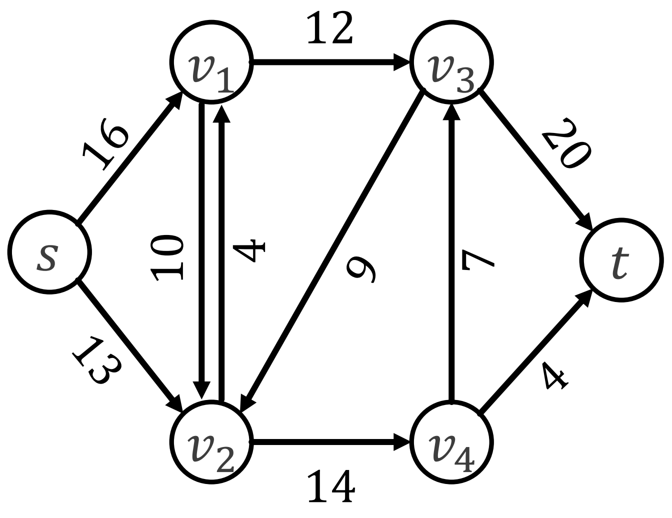
- By Lemma 9.2, $|f| \leq c(S, T)$ for any cut (S, T).
- Now we find a cut (S, T) such that $|f| = c(S, T)$. It implies that f is a maximum flow.
  - Cut (S, T) must be the minimum cut, otherwise $|f|$ will be greater than some other $c(S, T)$.
  - $|f|$ must be maximum, otherwise some other $|f|$ will be greater than $c(S, T)$.

Ford-Fulkerson Method

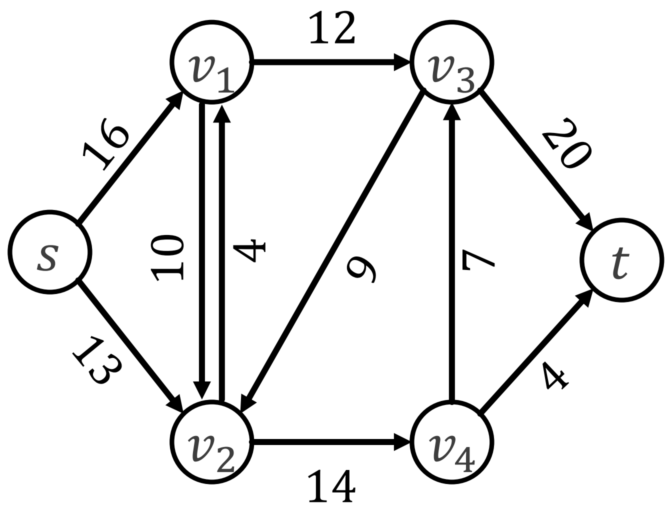
```
FordFulkerson($G, s, t, c$)
1  for each edge $(u,v) \in E$ do
2      $f(u,v) \leftarrow 0$
3  $G_f \leftarrow G$
4  while there is an augmenting path $p$ in $G_f$ do
5      Let $c_f(p)$ be the bottleneck capacity of $p$
6      for each edge $(u,v)$ in $p$ do
7          $f(u,v) \leftarrow f(u,v) + c_f(p)$
8      Update the residual graph $G_f$
```

Example

Initialization



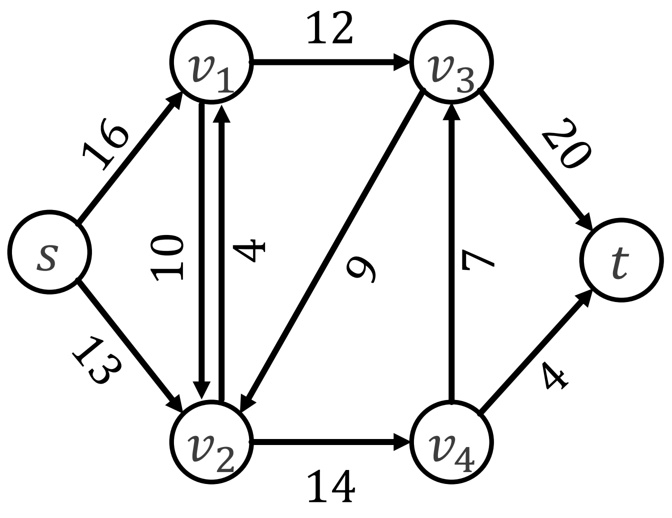
G



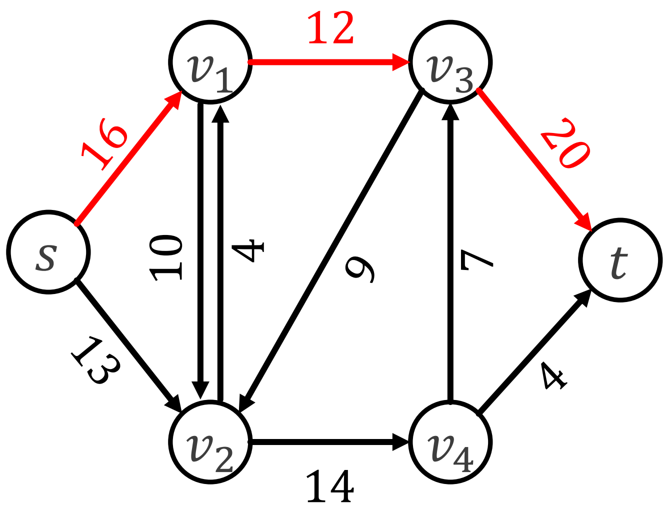
G_f

Example

Find an augmenting path p



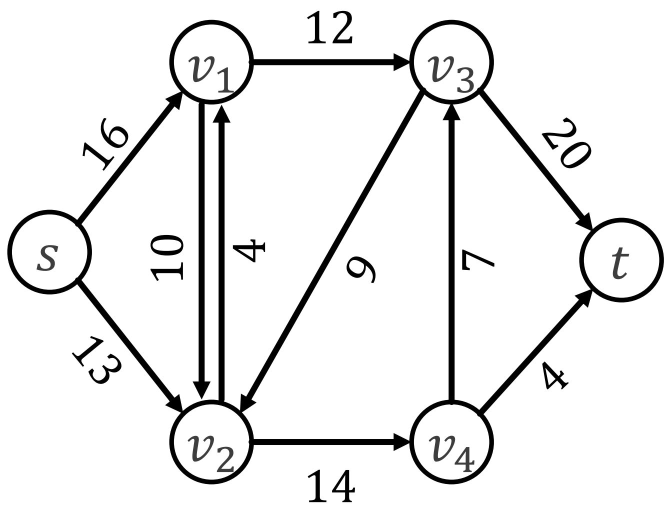
G



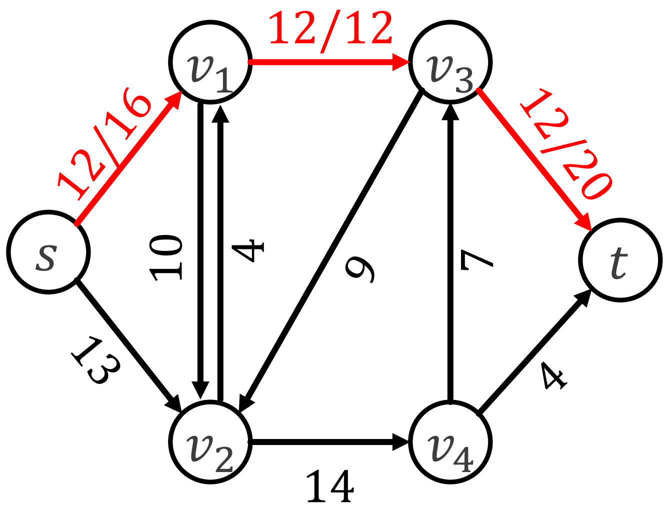
G_f

Example

Determine the bottleneck capacity $c_f(p) = 12$



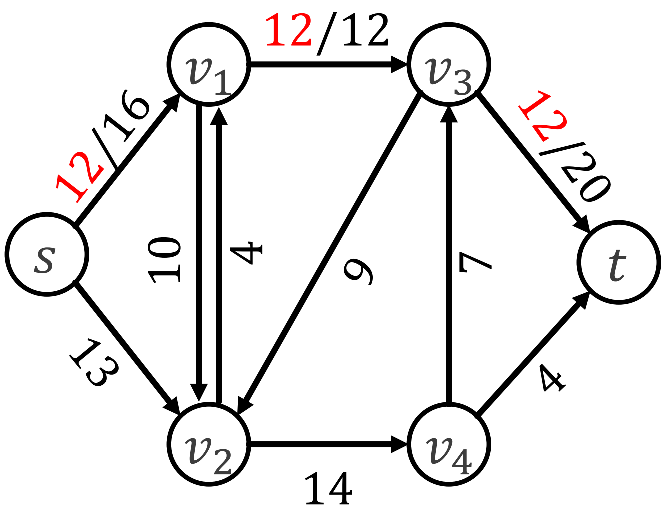
G



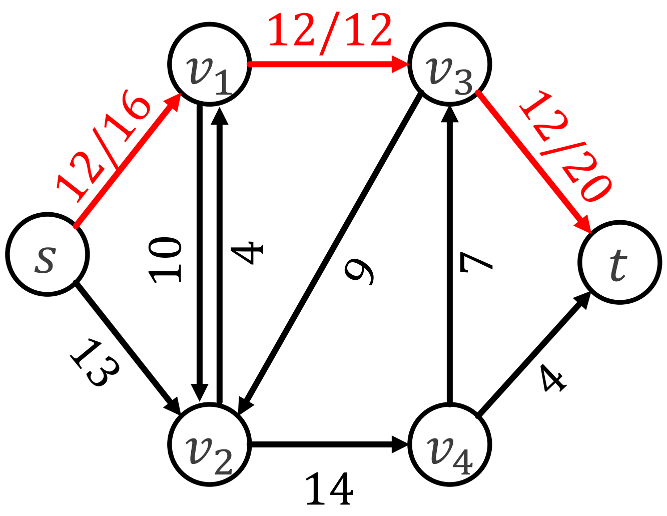
G_f

Example

Add the flow f_p to f



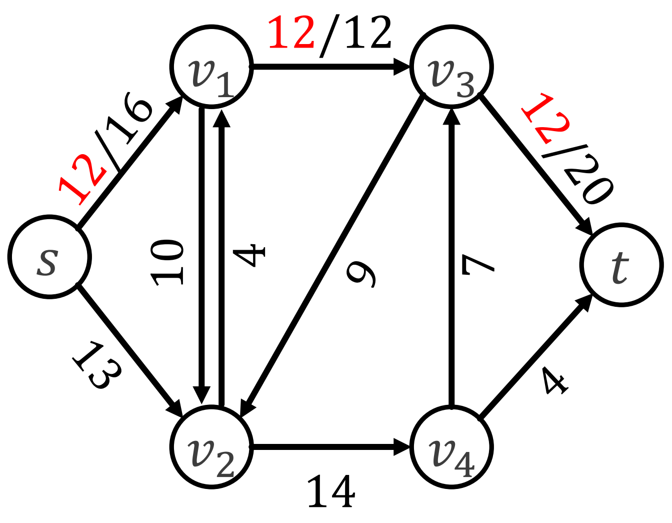
G



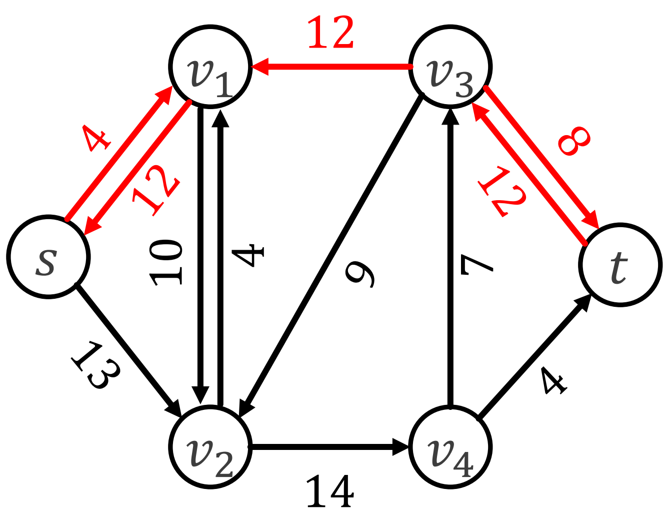
G_f

Example

Update G_f



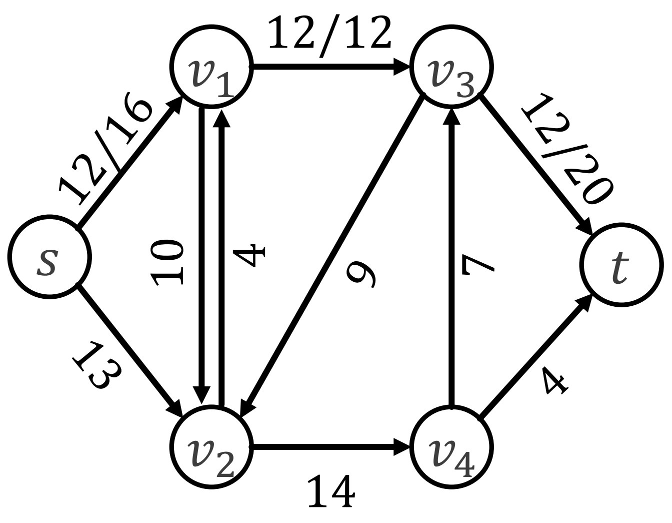
G



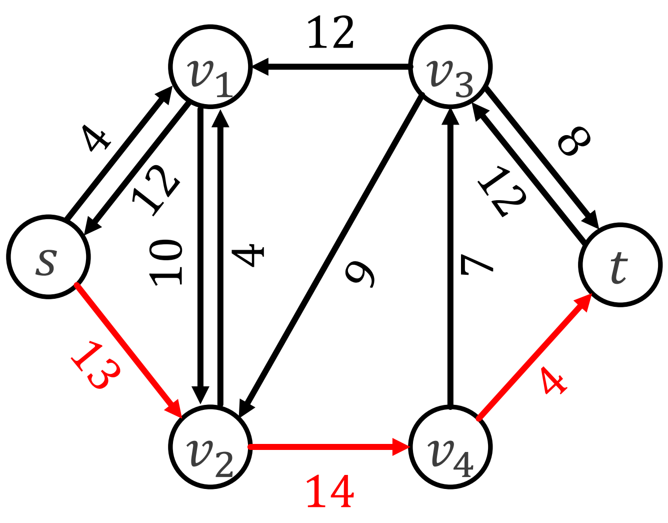
G_f

Example

Find an augmenting path p



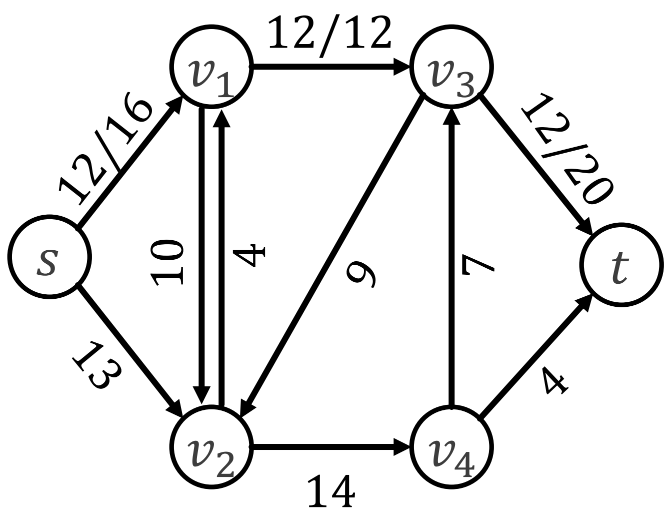
G



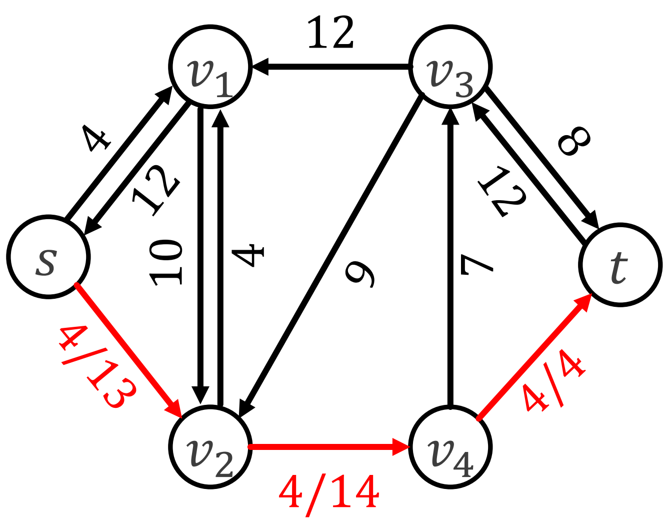
G_f

Example

Determine the bottleneck capacity $c_f(p) = 4$



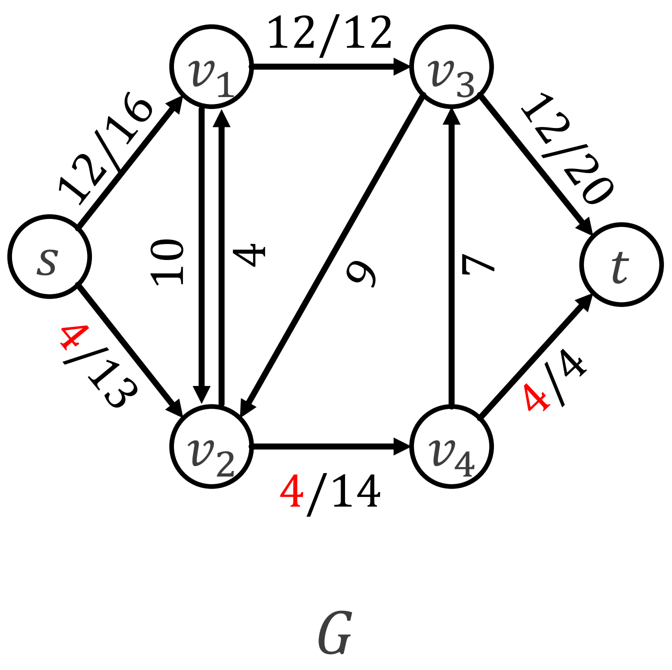
G

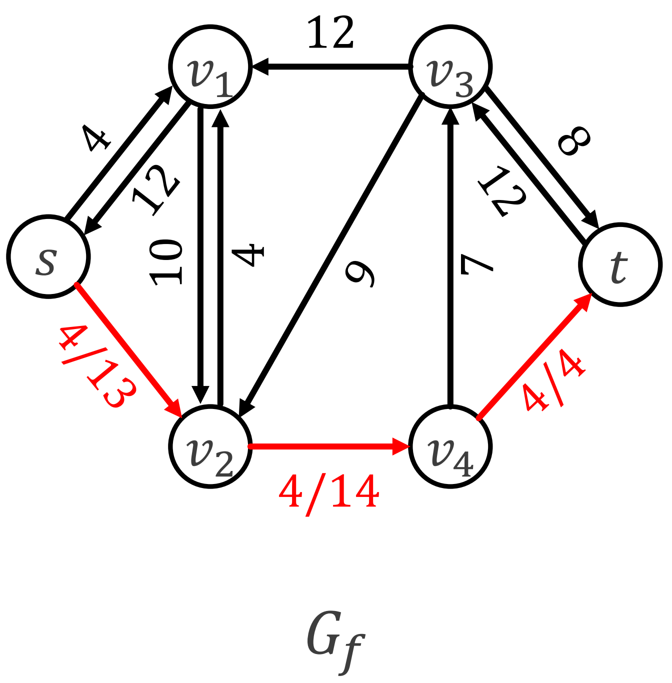


G_f

Example

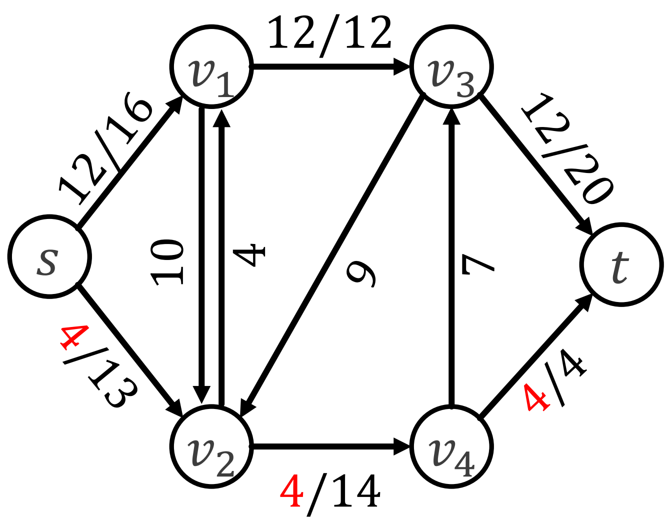
Add the flow f_p to f



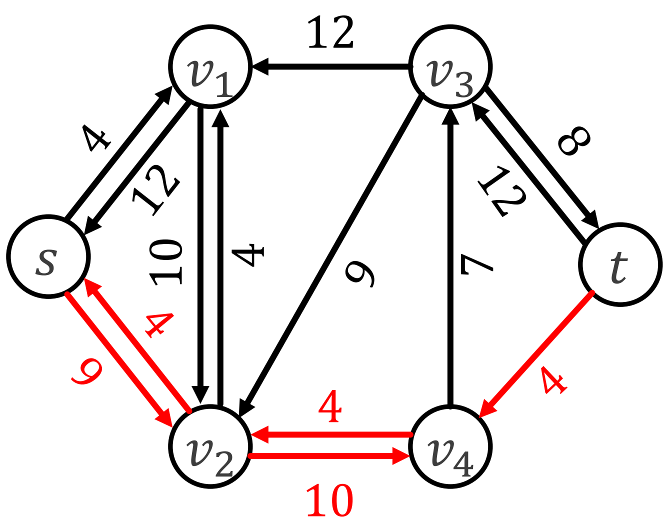


Example

Update G_f



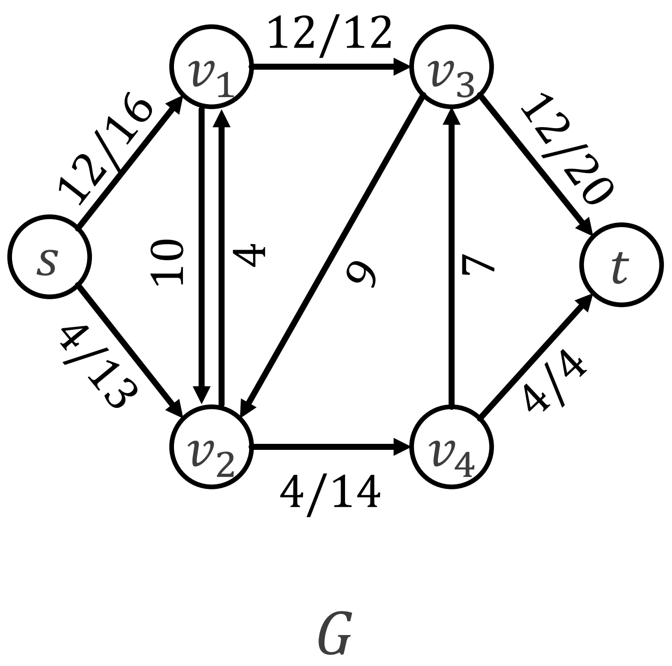
G

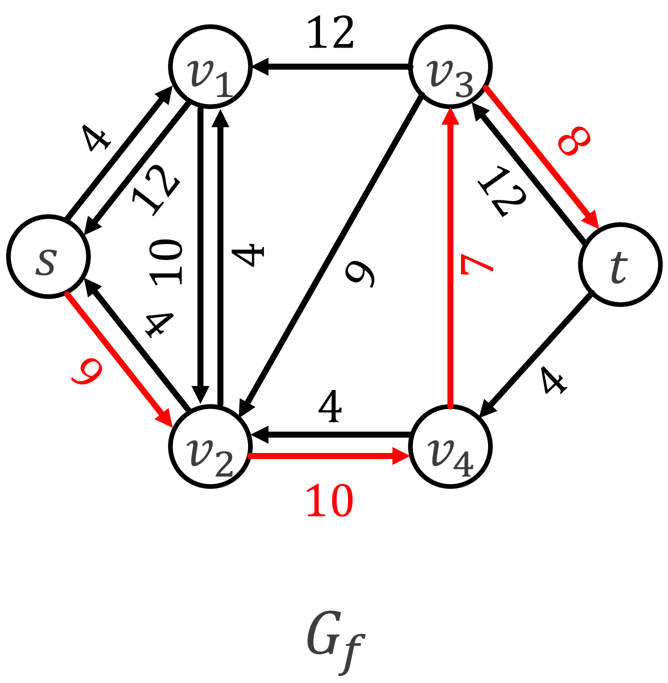


G_f

Example

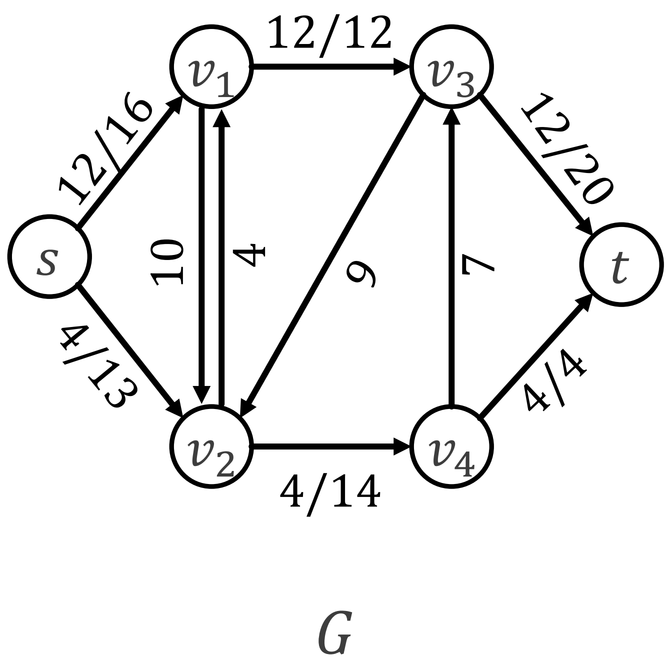
Find an augmenting path p

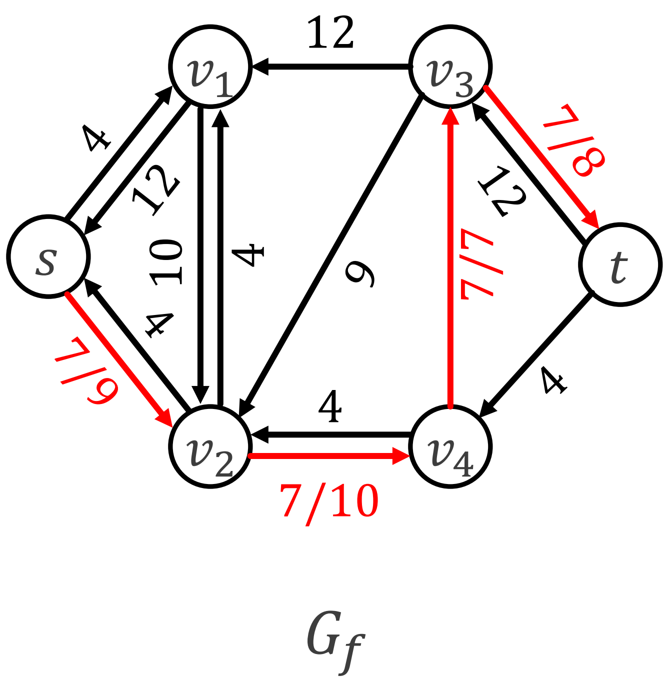




Example

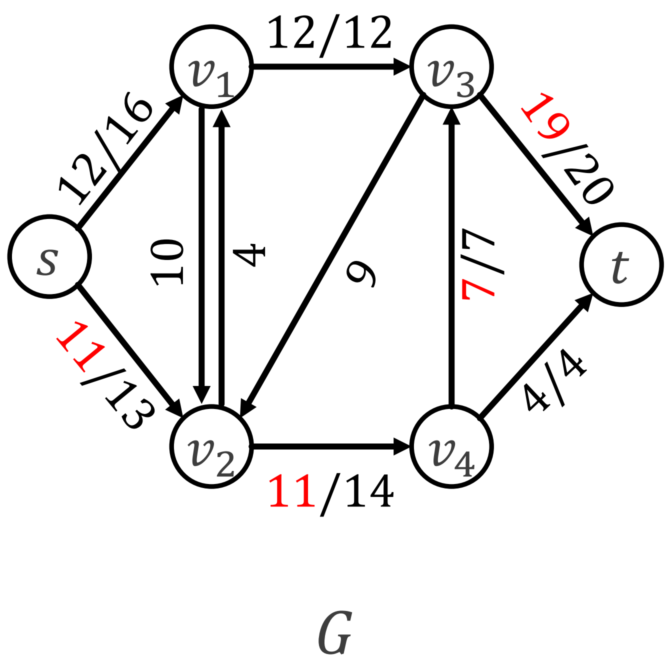
Determine the bottleneck capacity $c_f(p) = 7$

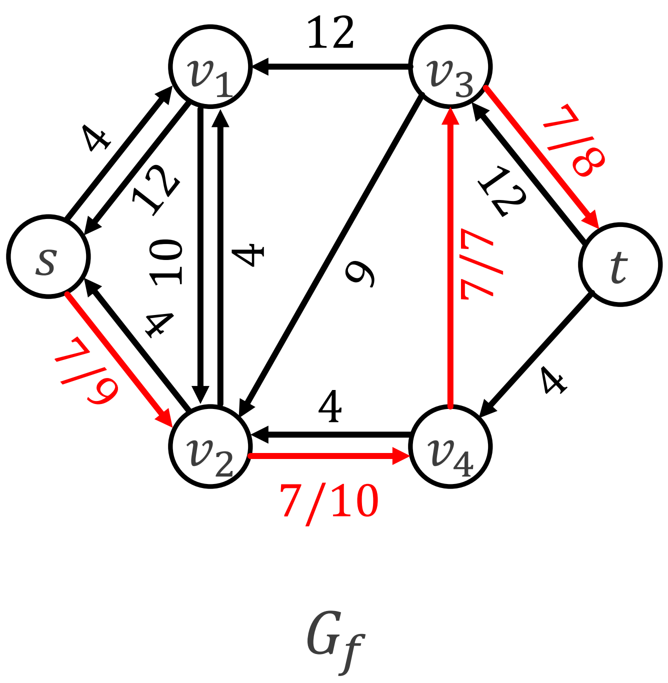




Example

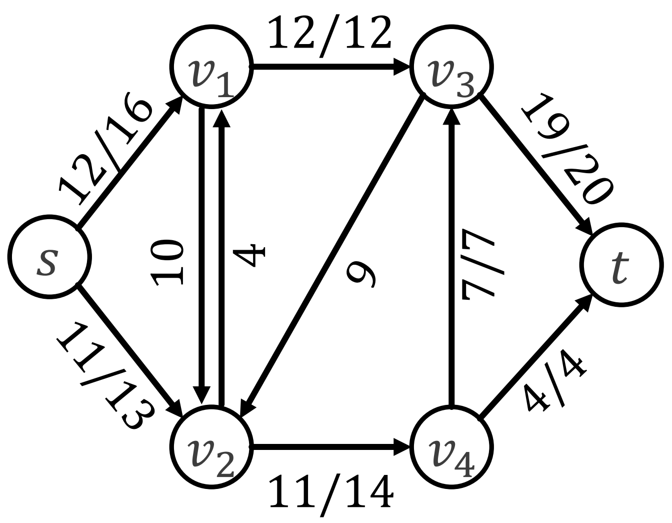
Add the flow f_p to f



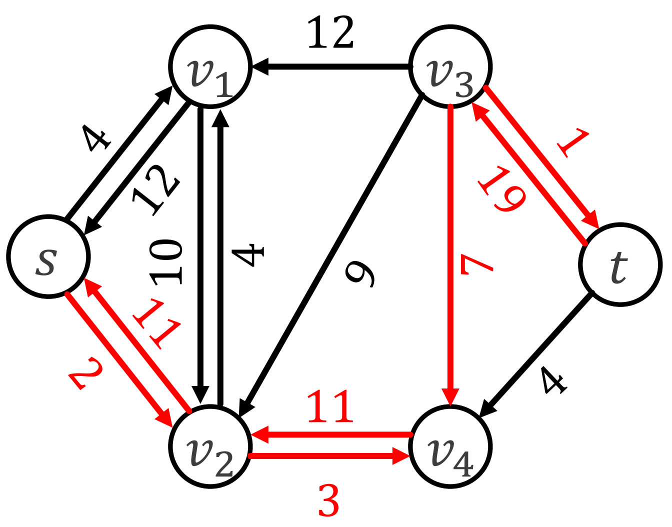


Example

Update G_f



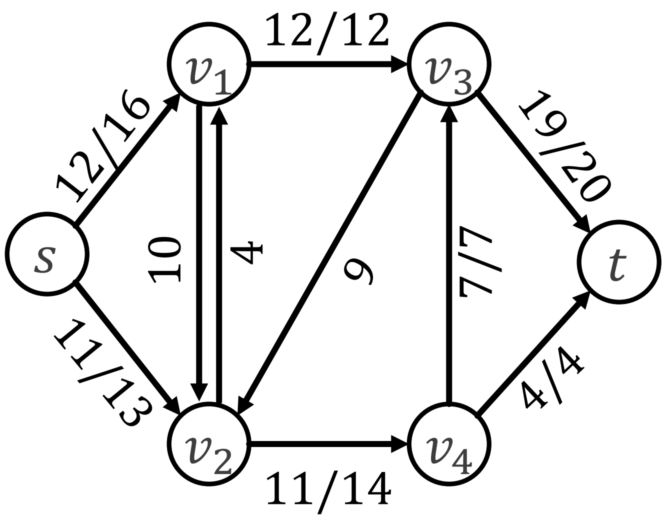
G



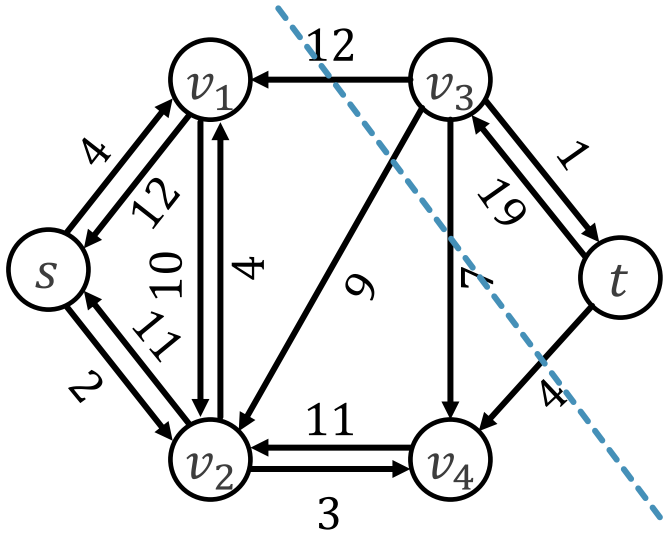
G_f

Example

Can't find an augmenting path and $|f| = c(S,T) = 23$ for some cut (S,T).



G



G_f

Ford-Fulkerson Method

```
FordFulkerson($G, s, t, c$)
1  for each edge $(u, v) \in E$ do
2      $f(u, v) \leftarrow 0$
3  $G_f \leftarrow G$
4  while there is an augmenting path $p$ in $G_f$ do
5      Let $c_f(p)$ be the bottleneck capacity of $p$
6      for each edge $(u, v)$ in $p$ do
7          $f(u, v) \leftarrow f(u, v) + c_f(p)$
8      Update the residual graph $G_f$
```

$O(|E|)$

$O(|f^*|)$

$O(|E|)$

$O(|E|)$

$O(|E|)$

Total: $O(|E||f^*|)$

- Each time we can find a augmenting path which can at least increase the flow value by 1. Therefore, at most f^* iterations in while loop.
  - f^* is the maximum flow value that the method can find.

Method vs. Algorithm

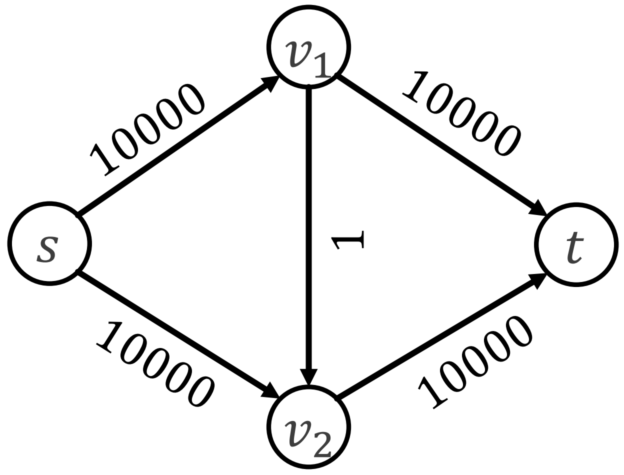
- We call it a "method" rather than an "algorithm" because the way to find the augmented path is not identified.
  - Different implementations have differing running times.

```
FordFulkerson($G$, $s$, $t$, $c$)
1  for each edge $(u, v) \in E$ do
2      $f(u, v) \leftarrow 0$
3  $G_f \leftarrow G$
4  while there is an augmenting path $p$ in $G_f$ do
5      Let $c_f(p)$ be the bottleneck capacity of $p$
6      for each edge $(u, v)$ in $p$ do
7          $f(u, v) \leftarrow f(u, v) + c_f(p)$
8      Update the residual graph $G_f$
```

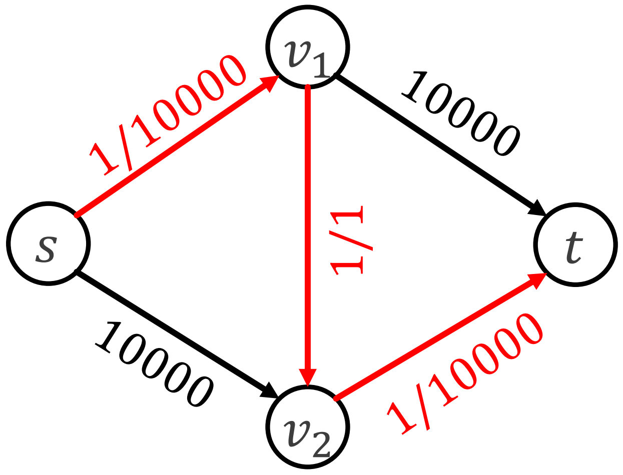
厦门大学信息学院 SCHOOL OF INFORMATICS XIAMEN UNIVERSITY 厦门大学计算机科学系 Computer Science Department of Xiamen University

Example

Find an augmenting path p and bottleneck capacity



G

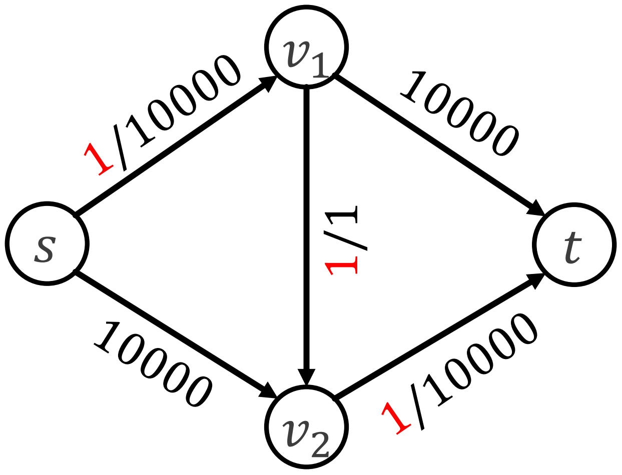


G_f

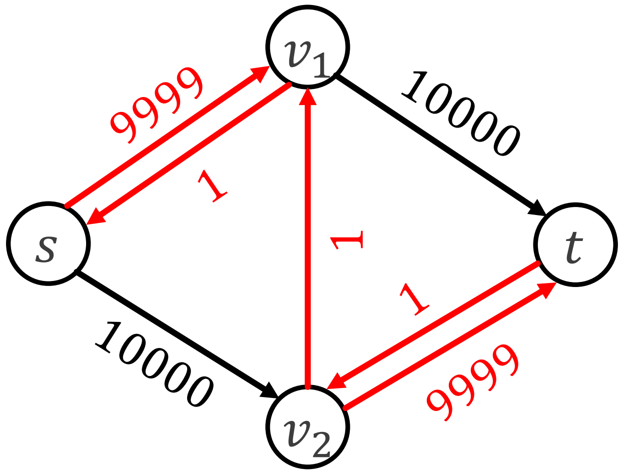
厦门大学信息学院
SCHOOL OF INFORMATICS XIAMEN UNIVERSITY
厦门大学计算机科学系
Computer Science Department of Xiamen University

Example

Add the flow f_p to f and update G_f



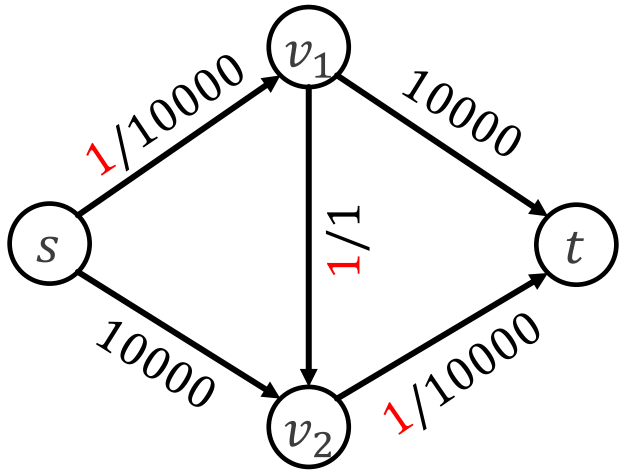
G



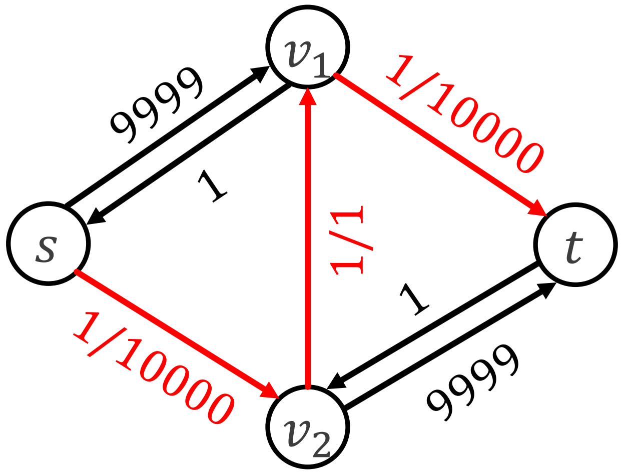
G_f

Example

Find an augmenting path p and bottleneck capacity



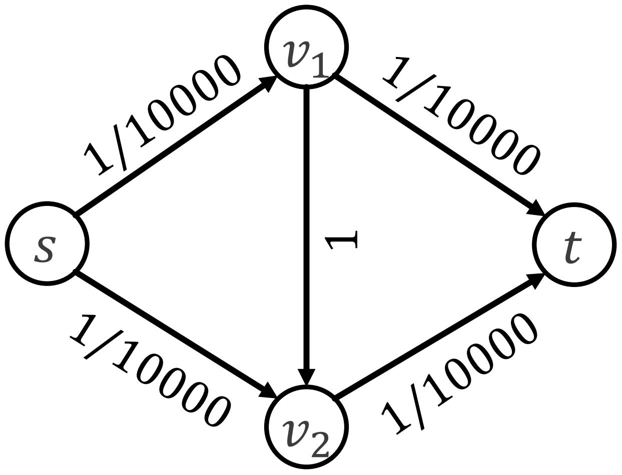
G

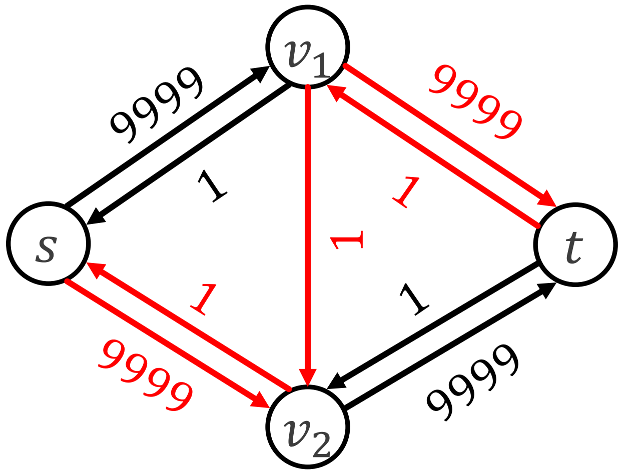


G_f

Example

Add the flow f_p to f and update G_f





G

G_f

Running time: Execute 20000 times…

MAX-FLOW PROBLEM

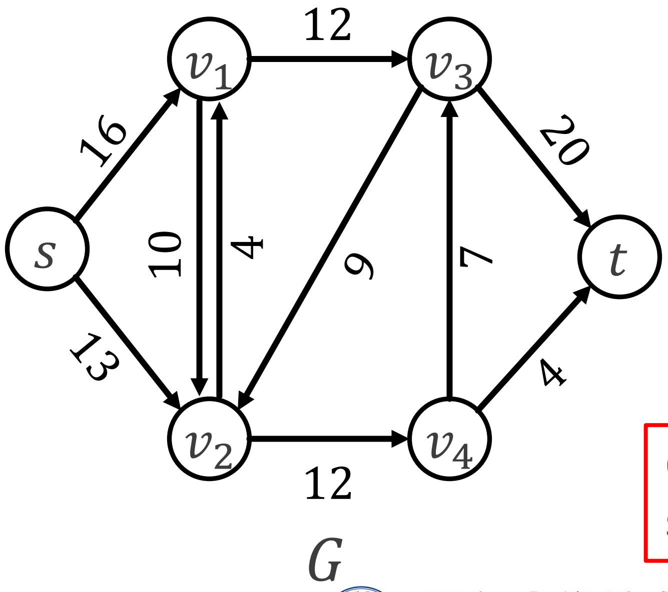
SHORTEST PATH AUGMENTATION ALGORITHM

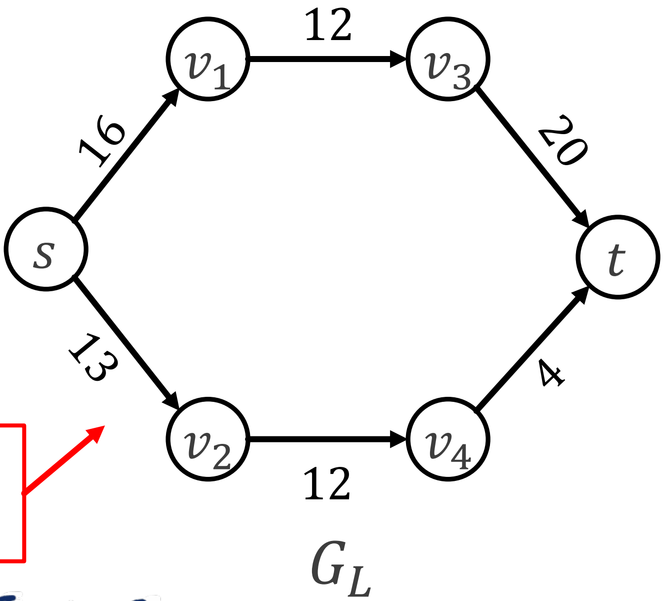
Shortest Path Augmentation Algorithm

Definition 9.8

The level (层次) of a vertex v, denoted by $d(v)$, is the least number of edges in a path from s to v.

Given a directed graph $G = (V, E)$, the level graph (层次图) is defined as $G_L = (V, E')$, where $E' = \{(u, v)\colon\ d(v) = d(u) + 1\}$.



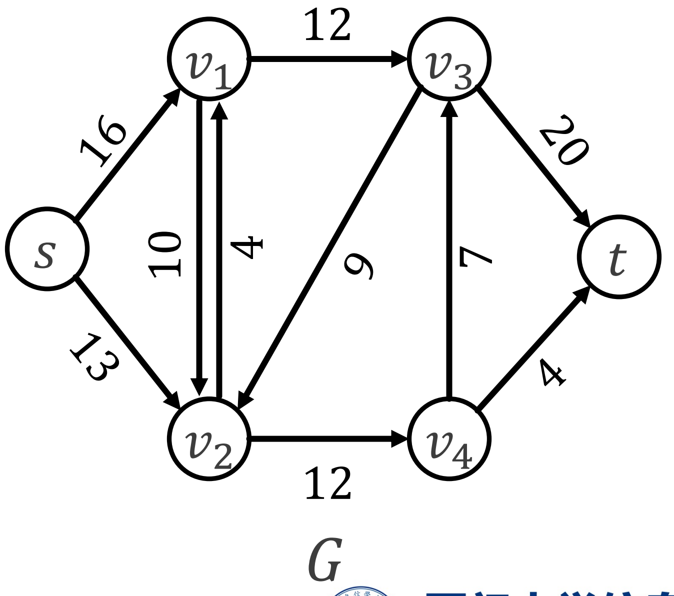


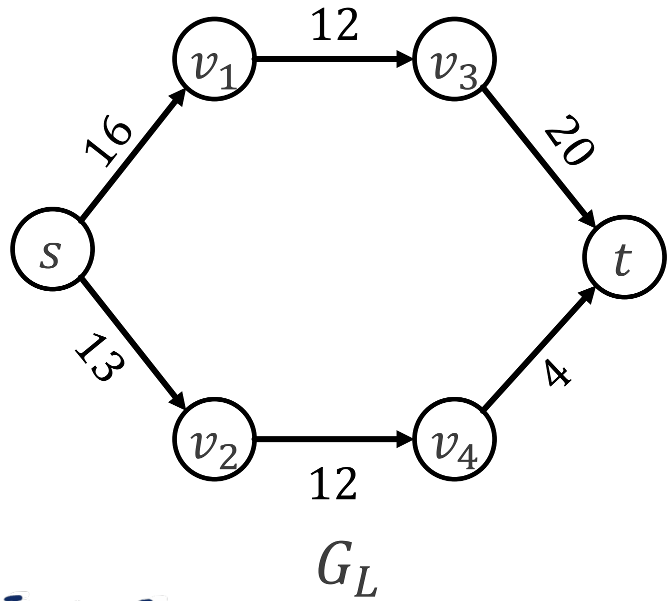
Only contain shortest paths

厦门大学信息学院 SCHOOL OF INFORMATICS XIAMEN UNIVERSITY 厦门大学 计算机科学系 Computer Science Department of Xiamen University

Shortest Path Augmentation Algorithm

- The level graph can be easily obtained by BFS.
- Can DFS do this?
  - No, DFS doesn't give the shortest unless each time we compare $d[v]$.





Shortest Path Augmentation Algorithm

- This algorithm is also called Edmonds-Karp algorithm.
- Idea: Select an augmenting path of minimum length and increases the current flow by its bottleneck capacity.
- Generally there are two steps.

1. Compute the level graph G_L from the residual graph G_f. If t is not in G_L, then halt: otherwise continue.
    - t not in G_L means there's no path from s to t in G_f.
2. As long as there is a path p form s to t in G_L, augment the current flow by p, and update G_L and G_f accordingly.

Shortest Path Augmentation Algorithm

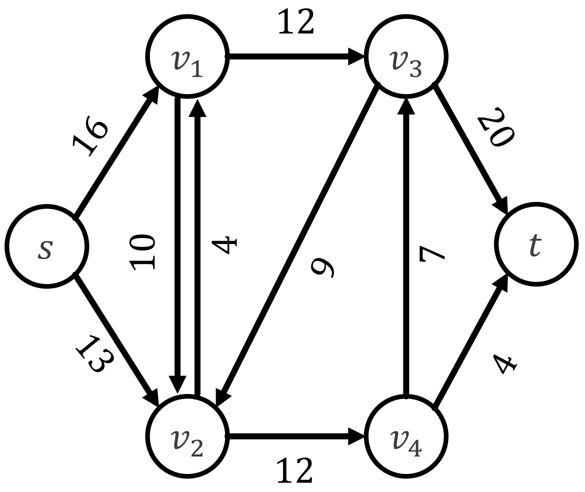
```
ShortestPathAugmentation($G, s, t$)
1  **for** each edge $(u, v) \in E$ **do**
2      $f(u, v) \leftarrow 0$
3  $G_f \leftarrow G$
4  find the level graph $G_L$ of $G_f$
5  **while** $t$ is a vertex in $G_L$ **do**
6      **while** there is a path $p$ from $s$ to $t$ in $G_L$ **do**
7          let $c_f(p)$ be the bottleneck capacity on $p$
8          augment the current flow $f$ by $c_f(p)$
9          update $G_L$ and $G_f$ along the path $p$
10     use $G_f$ to compute a new level graph $G_L$
```

There may be multiple shortest paths in G_L from s to t.

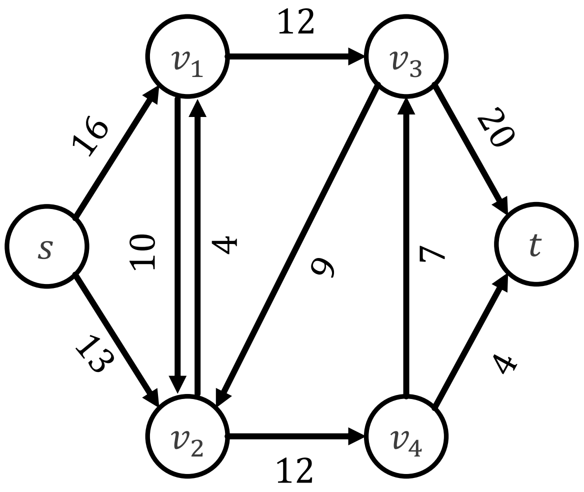
厦门大学信息学院
SCHOOL OF INFORMATICS XIAMEN UNIVERSITY
厦门大学计算机科学系
Computer Science Department of Xiamen University

Example

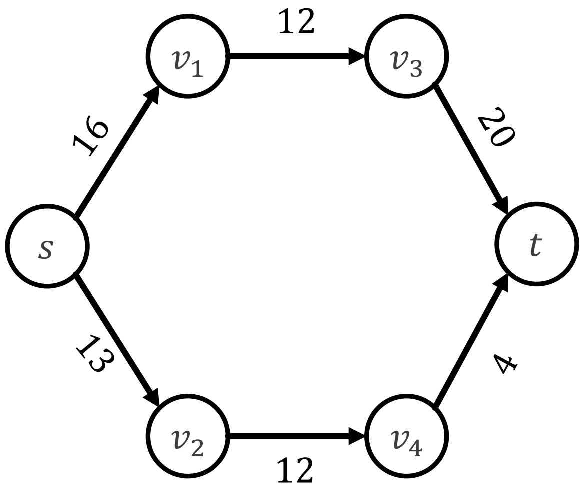
Initialization



G



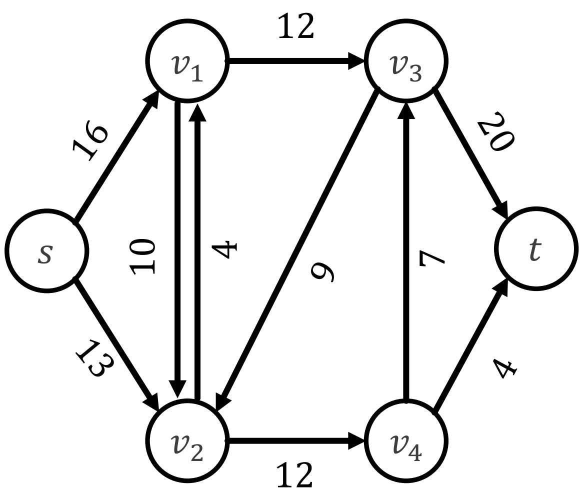
G_f



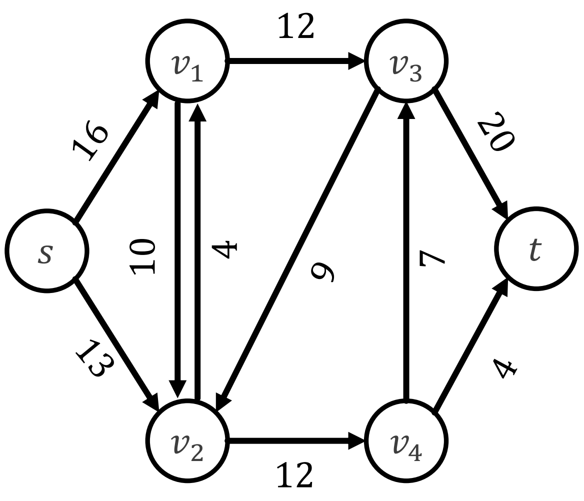
G_L

Example

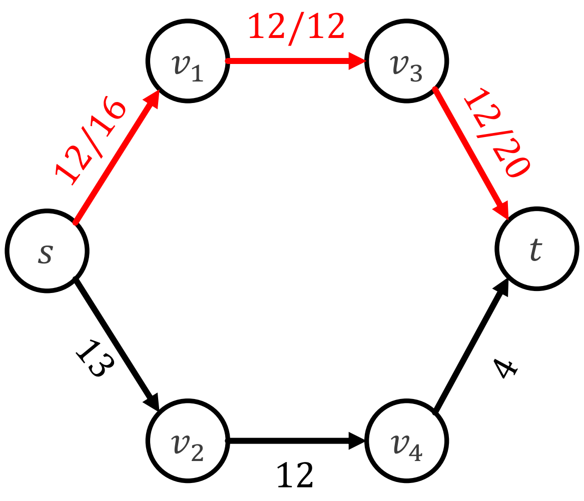
Find an augmenting path p and bottleneck capacity in G_L



G



G_f

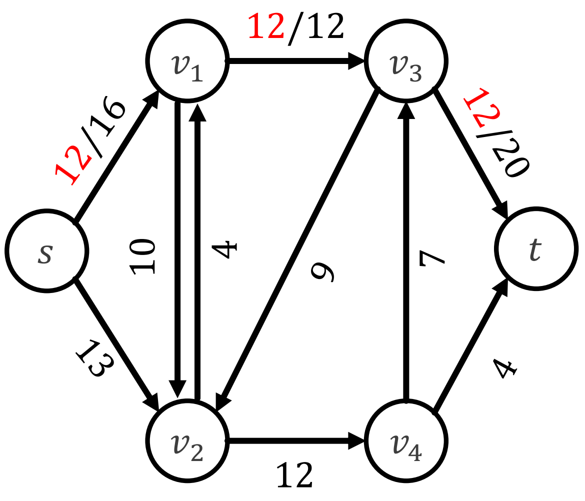


G_L

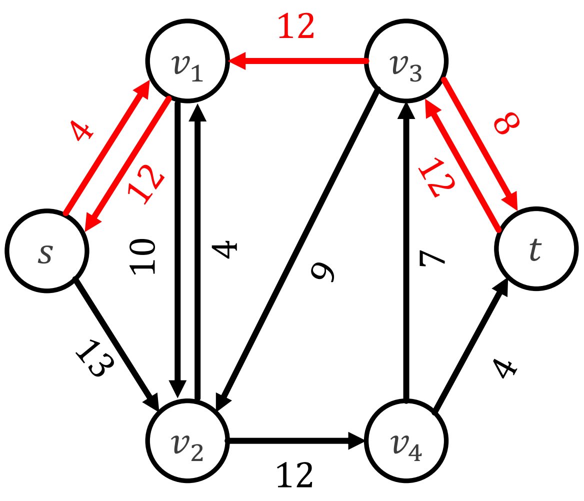
厦门大学信息学院 SCHOOL OF INFORMATICS XIAMEN UNIVERSITY 厦门大学计算机科学系 Computer Science Department of Xiamen University

Example

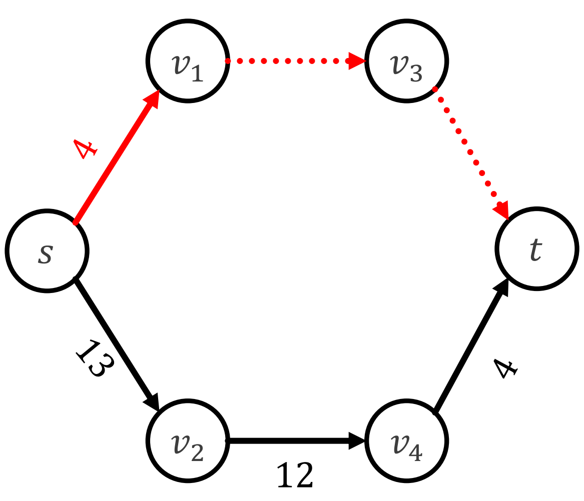
Add the flow f_p to f and update G_f and G_L



G



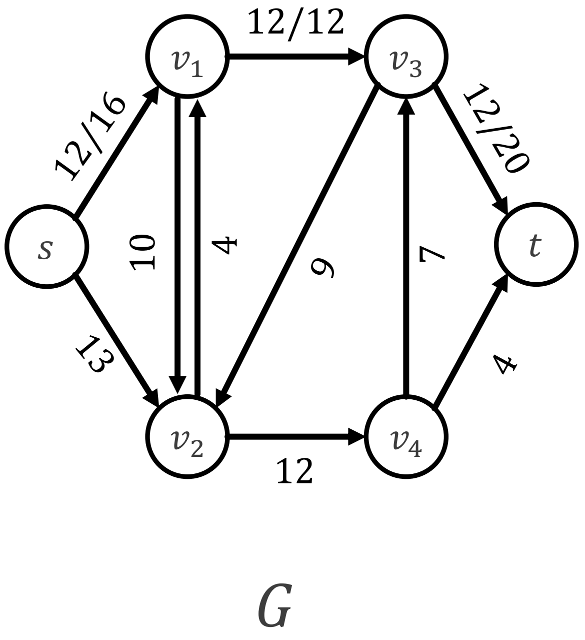
G_f

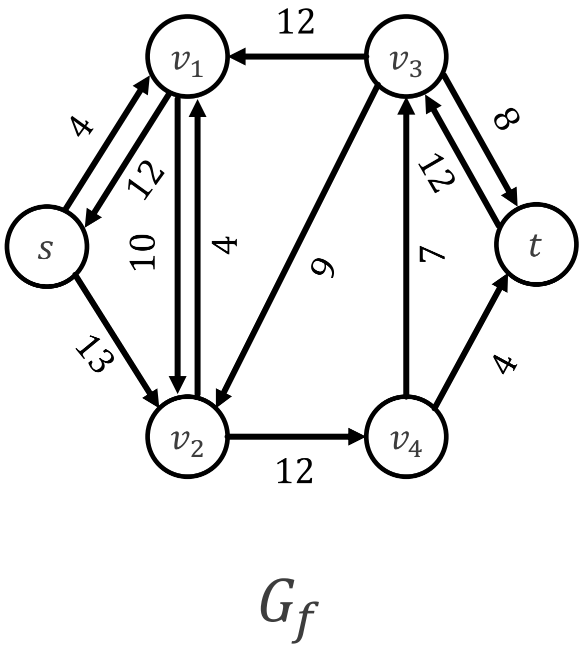


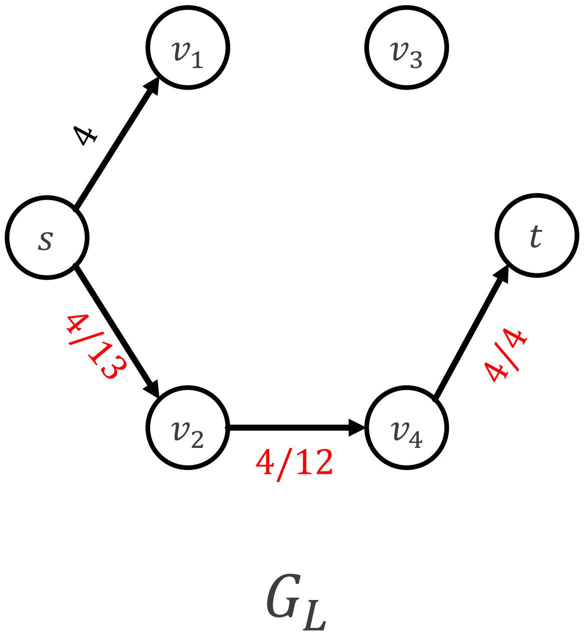
G_L

Example

Find an augmenting path p and bottleneck capacity in G_L

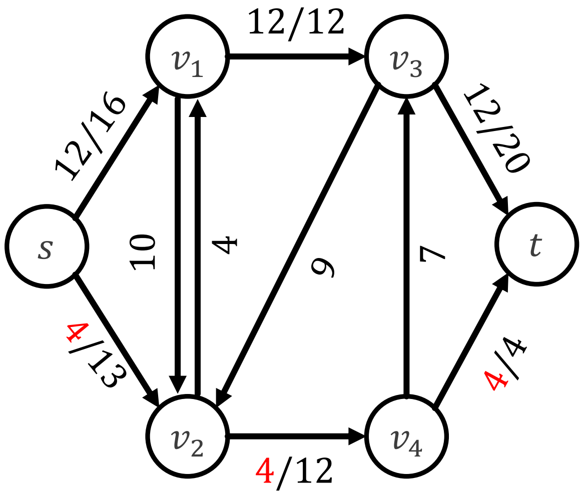




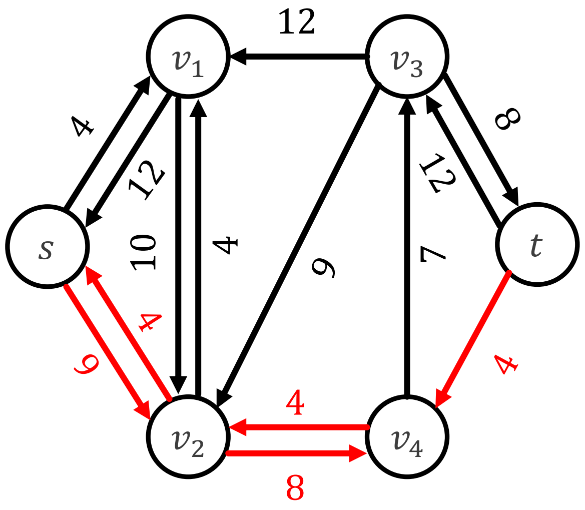


Example

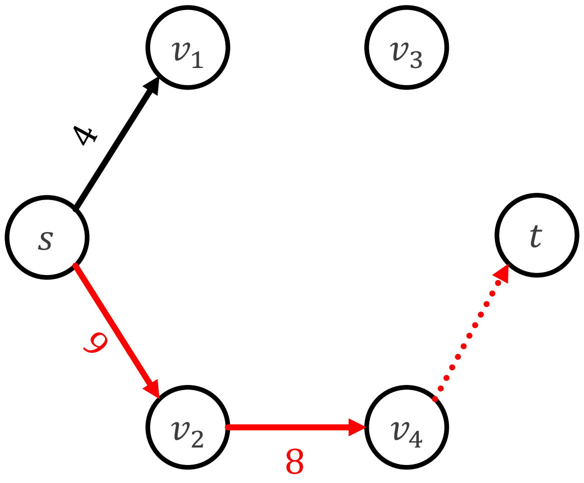
Add the flow f_p to f and update G_f and G_L



G



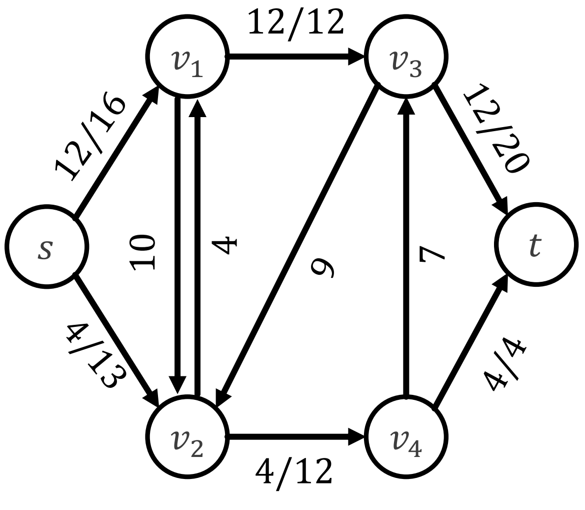
G_f



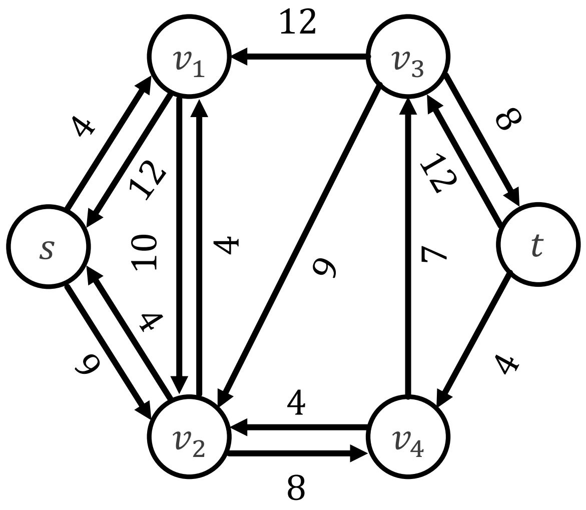
G_L

Example

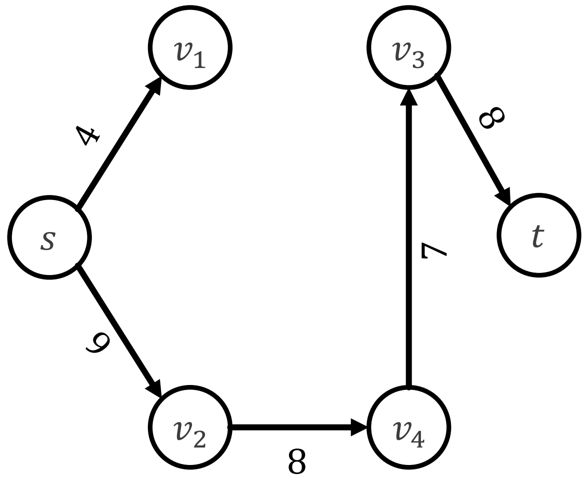
No path from s to t in G_L, use G_f to compute a new G_L



G



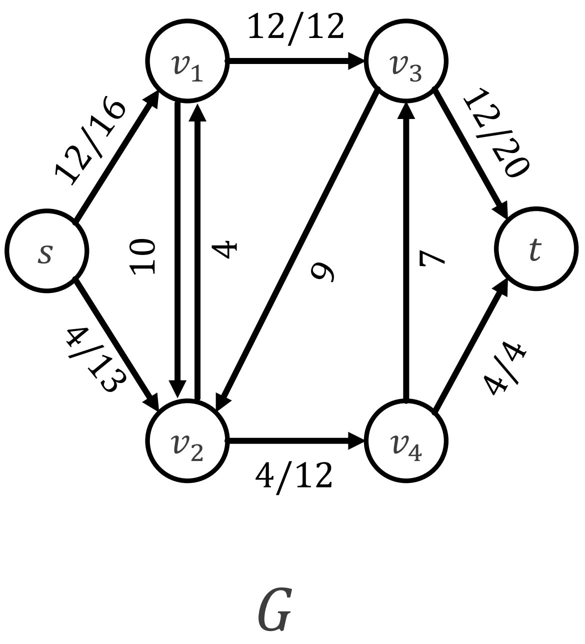
G_f

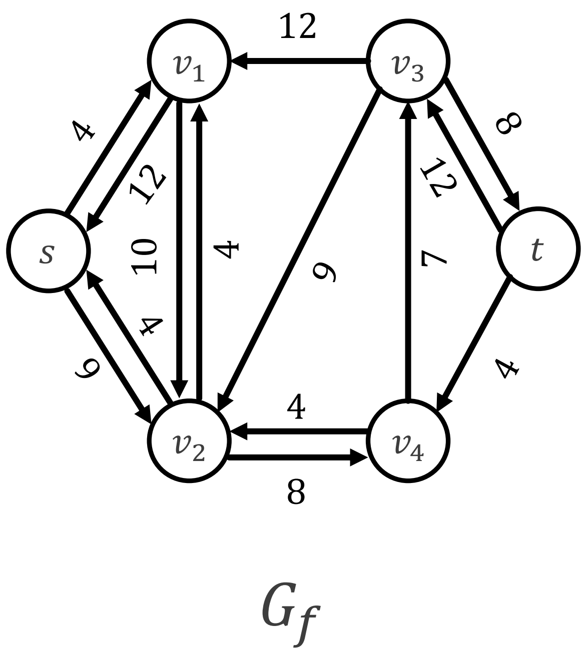


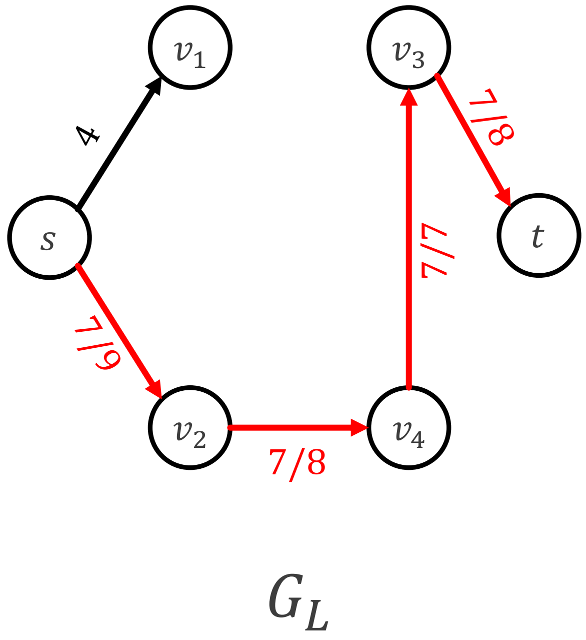
G_L

Example

Find an augmenting path p and bottleneck capacity in G_L

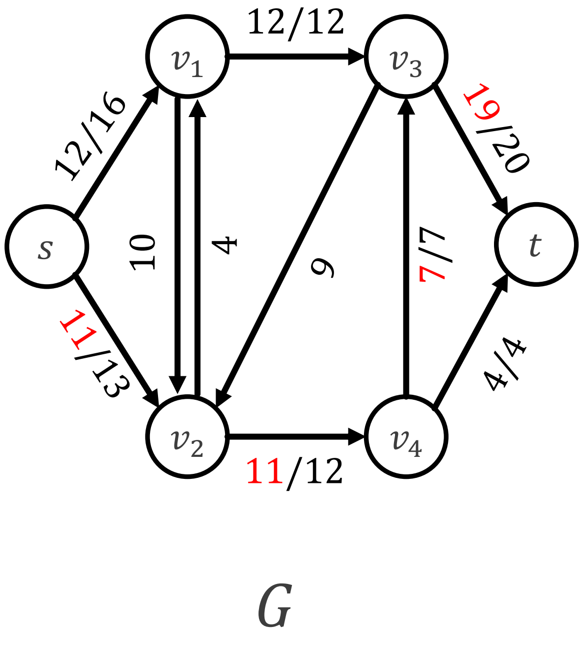


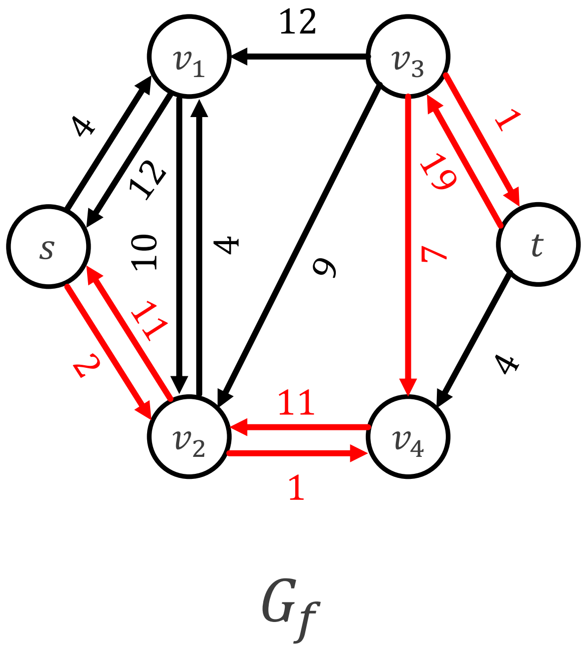


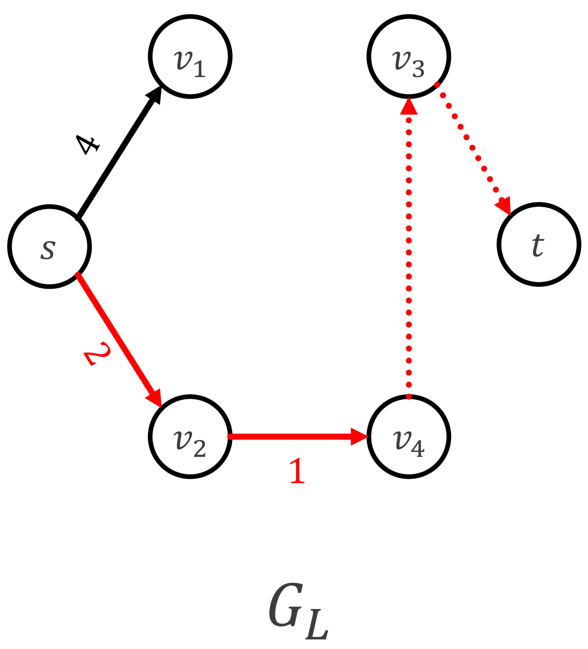


Example

Add the flow f_p to f and update G_f and G_L



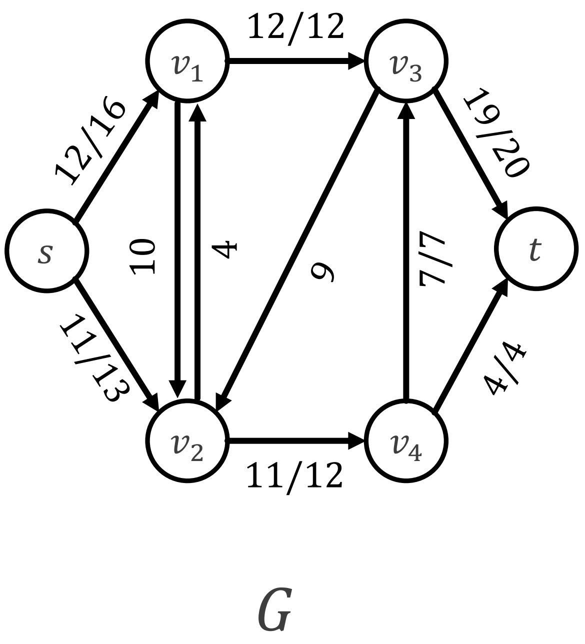


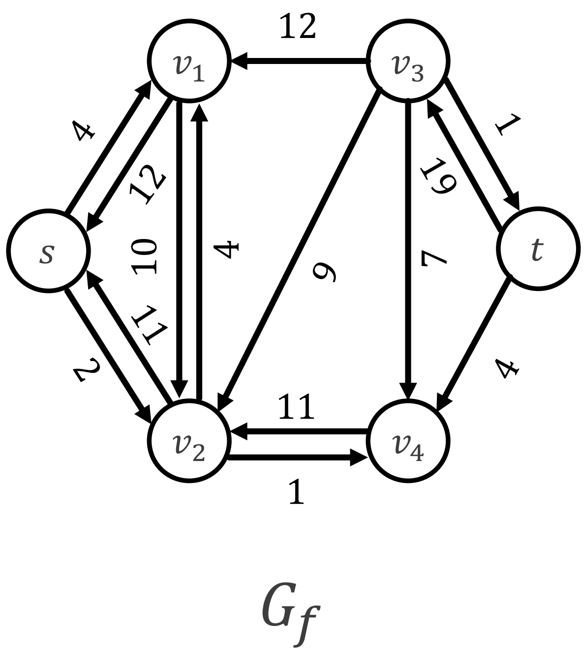


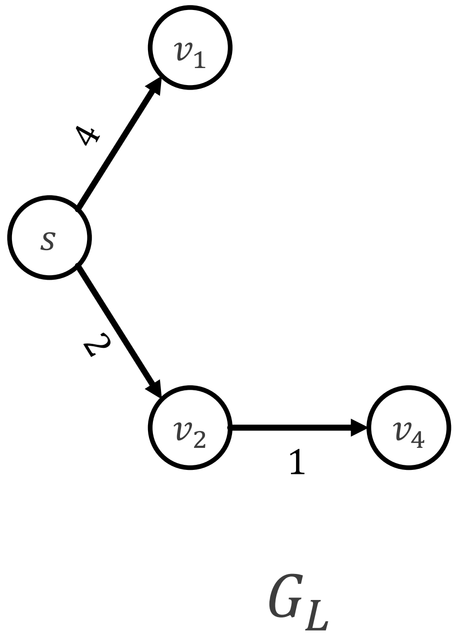
厦门大学信息学院 SCHOOL OF INFORMATICS XIAMEN UNIVERSITY 厦门大学计算机科学系 Computer Science Department of Xiamen University

Example

No path from s to t in G_L, use G_f to compute a new G_L



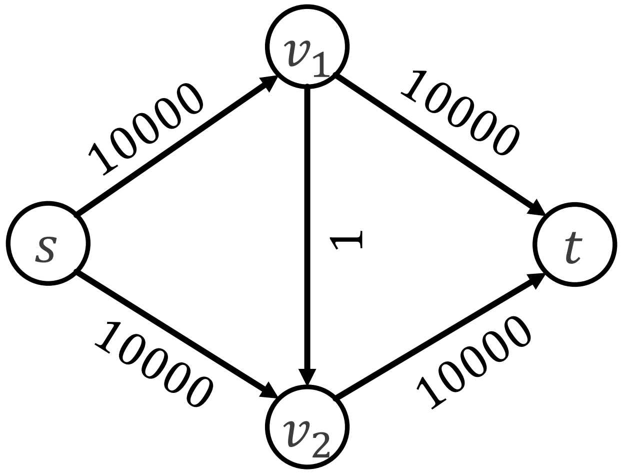




t is not a vertex in G_L. Algorithm terminates.

Example

- Now, look back at this example again…



G

Analysis of Shortest Path Augmentation Algorithm

Lemma 9.4

The while loop in Line 5 in ShortestPathAugmentation is executed at most $|V|$ times.

Proof:

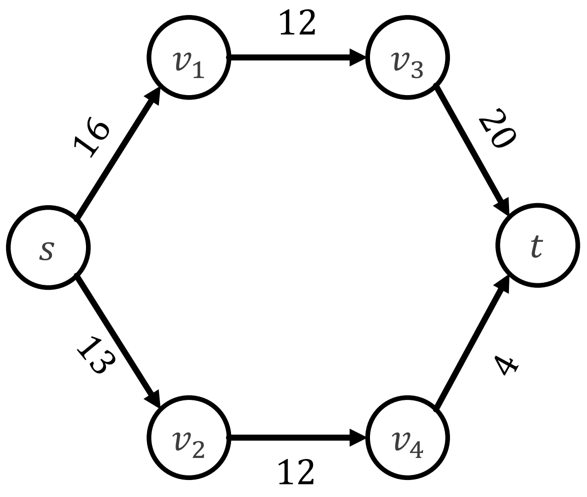
- We show that the number of level graph computed using the algorithm is at most $|V|$.

```
5  while $t$ is a vertex in $G_L$ do
6      while there is a path $p$ from $s$ to $t$ in $G_L$ do
7          let $c_f(p)$ be the bottleneck capacity on $p$
8          augment the current flow $f$ by $c_f(p)$
9          update $G_L$ and $G_f$ along the path $p$
10     use $G_f$ to compute a new level graph $G_L$
```

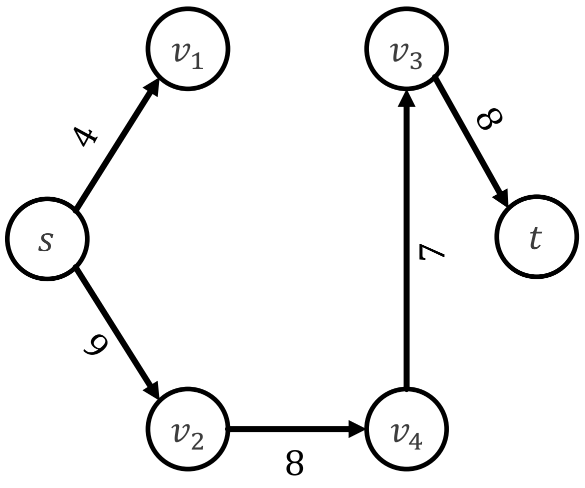
Analysis of Shortest Path Augmentation Algorithm

Proof (cont'd):

- First, we show that the sequence of lengths of augmenting paths using ShortestPathAugmentation is strictly increasing.



Length of augmenting path of the first G_L: 3

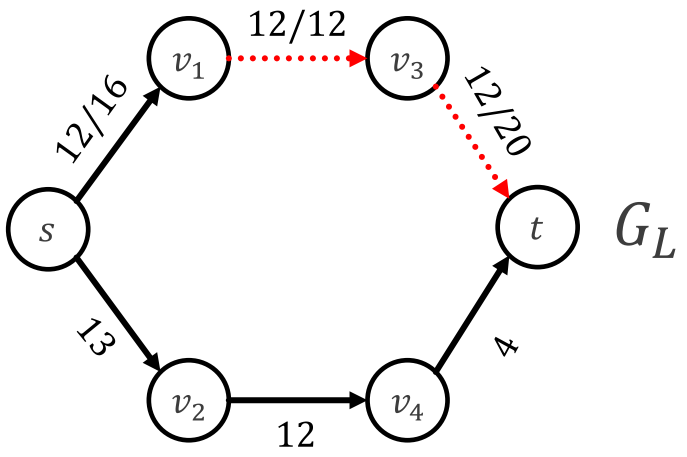


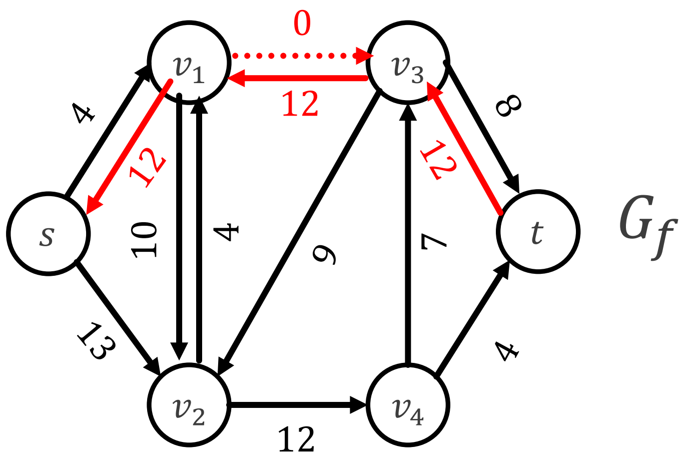
Length of augmenting path of the second G_L: 4

Analysis of Shortest Path Augmentation Algorithm

Proof (cont'd):

- Let p be any augmenting path in the current G_L.
- After augmenting using p, at least one edge will be saturated and will disappear in G_f.
- At most $|p|$ new edges will appear in G_f, but they are back edges, and hence will not contribute to a shortest path from s to t.
- When all shortest paths are cut in the current G_L, BFS will find strict longer shortest paths for the next G_L.





Analysis of Shortest Path Augmentation Algorithm

Proof (cont’d):

- When t is no longer in G_L, we can’t find an augmenting path any more.
- Since the length of any augmenting path is between 1 and $|V| - 1$, the number of level graphs use for augmentations is at most $|V| - 1$.
- Since one more level graph is computed in which t does not appear, the total number of level graphs computed is at most $|V|$.

Analysis of Shortest Path Augmentation Algorithm

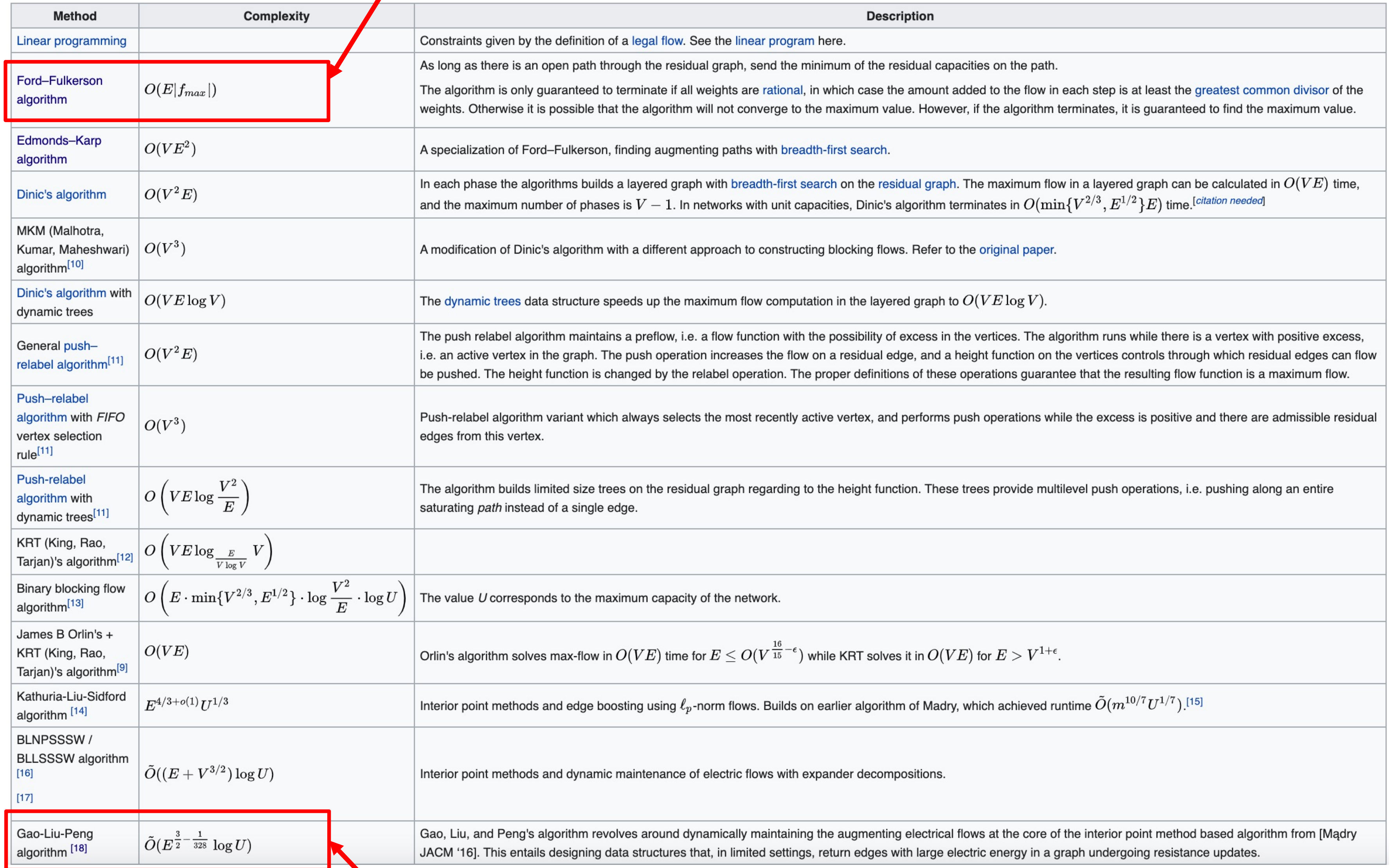
Theorem 9.2

Given a flow network $G = (V, E)$, $\mathrm{ShortestPathAugmentation}$ finds a maximum flow in $O(|V||E|^2)$.

Proof:

- Given a G_L, there are at most $|E|$ augmenting paths of the same length.
- By Lemma 9.4, the number of augmenting steps is at most $|V||E|$.
- Computing each level graph takes $O(|E|)$ using BFS.
- Therefore, the total time required to compute all level graphs is $O((|V||E|^2)$.

Published in 1956



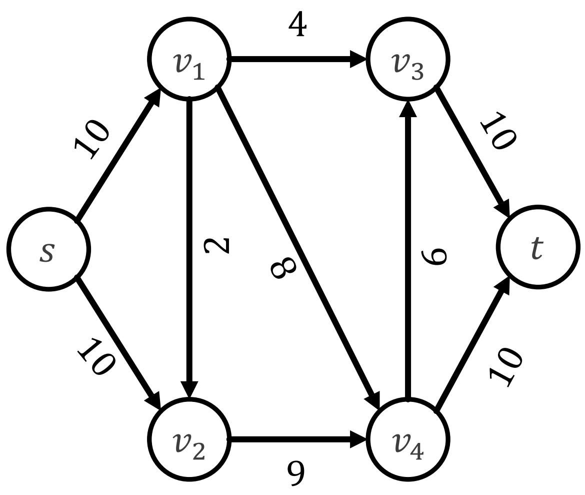
Method	Complexity	Description
Linear programming		Constraints given by the definition of a legal flow. See the linear program here.
Ford–Fulkerson algorithm	$O(E\|f_{max}\|)$	As long as there is an open path through the residual graph, send the minimum of the residual capacities on the path. The algorithm is only guaranteed to terminate if all weights are rational, in which case the amount added to the flow in each step is at least the greatest common divisor of the weights. Otherwise it is possible that the algorithm will not converge to the maximum value. However, if the algorithm terminates, it is guaranteed to find the maximum value.
Edmonds–Karp algorithm	$O(VE^2)$	A specialization of Ford–Fulkerson, finding augmenting paths with breadth-first search.
Dinic's algorithm	$O(V^2E)$	In each phase the algorithms builds a layered graph with breadth-first search on the residual graph. The maximum flow in a layered graph can be calculated in $O(VE)$ time, and the maximum number of phases is $V-1$. In networks with unit capacities, Dinic's algorithm terminates in $O(\min\{V^{2/3}, E^{1/2}\}E)$ time.[citation needed]
MKM (Malhotra, Kumar, Maheshwari) algorithm[10]	$O(V^3)$	A modification of Dinic's algorithm with a different approach to constructing blocking flows. Refer to the original paper.
Dinic's algorithm with dynamic trees	$O(VE\log V)$	The dynamic trees data structure speeds up the maximum flow computation in the layered graph to $O(VE\log V)$.
General push–relabel algorithm[11]	$O(V^2E)$	The push relabel algorithm maintains a preflow, i.e. a flow function with the possibility of excess in the vertices. The algorithm runs while there is a vertex with positive excess, i.e. an active vertex in the graph. The push operation increases the flow on a residual edge, and a height function on the vertices controls through which residual edges can flow be pushed. The height function is changed by the relabel operation. The proper definitions of these operations guarantee that the resulting flow function is a maximum flow.
Push–relabel algorithm with *FIFO* vertex selection rule[11]	$O(V^3)$	Push-relabel algorithm variant which always selects the most recently active vertex, and performs push operations while the excess is positive and there are admissible residual edges from this vertex.
Push–relabel algorithm with dynamic trees[11]	$O\left(VE\log\frac{V^2}{E}\right)$	The algorithm builds limited size trees on the residual graph regarding to the height function. These trees provide multilevel push operations, i.e. pushing along an entire saturating *path* instead of a single edge.
KRT (King, Rao, Tarjan)'s algorithm[12]	$O\left(VE\log_{\frac{E}{V\log V}}V\right)$	
Binary blocking flow algorithm[13]	$O\left(E\cdot\min\{V^{2/3},E^{1/2}\}\cdot\log\frac{V^2}{E}\cdot\log U\right)$	The value *U* corresponds to the maximum capacity of the network.
James B Orlin's + KRT (King, Rao, Tarjan)'s algorithm[9]	$O(VE)$	Orlin's algorithm solves max-flow in $O(VE)$ time for $E\leq O(V^{\frac{16}{15}-\epsilon})$ while KRT solves it in $O(VE)$ for $E>V^{1+\epsilon}$.
Kathuria-Liu-Sidford algorithm [14]	$E^{4/3+o(1)}U^{1/3}$	Interior point methods and edge boosting using ℓ_p-norm flows. Builds on earlier algorithm of Madry, which achieved runtime $\tilde{O}(m^{10/7}U^{1/7})$.[15]
BLNPSSSW / BLLSSSW algorithm [16] [17]	$\tilde{O}((E+V^{3/2})\log U)$	Interior point methods and dynamic maintenance of electric flows with expander decompositions.
Gao-Liu-Peng algorithm [18]	$\tilde{O}(E^{\frac{3}{2}-\frac{1}{328}}\log U)$	Gao, Liu, and Peng's algorithm revolves around dynamically maintaining the augmenting electrical flows at the core of the interior point method based algorithm from [Mądry JACM '16]. This entails designing data structures that, in limited settings, return edges with large electric energy in a graph undergoing resistance updates.

Submitted to arXiv on 18 Jan 2021

Image source: https://en.wikipedia.org/wiki/Maximum_flow_problem#cite_note-18

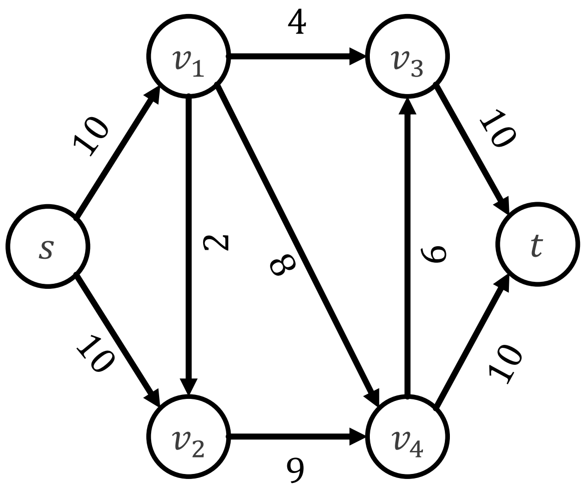
Classroom Exercise

- Use ShortestPathAugmentation to find the max-flow of the following graph.

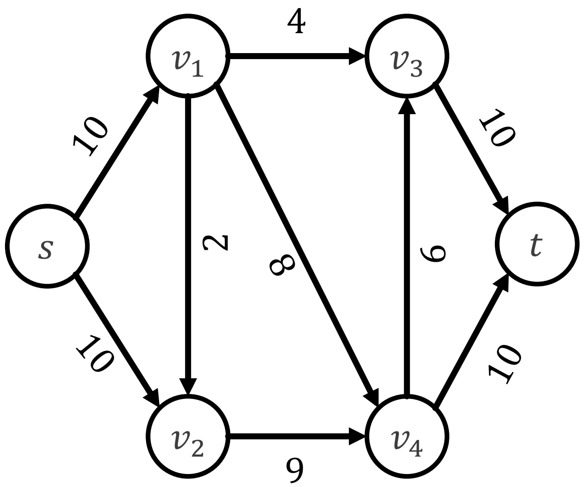


Classroom Exercise

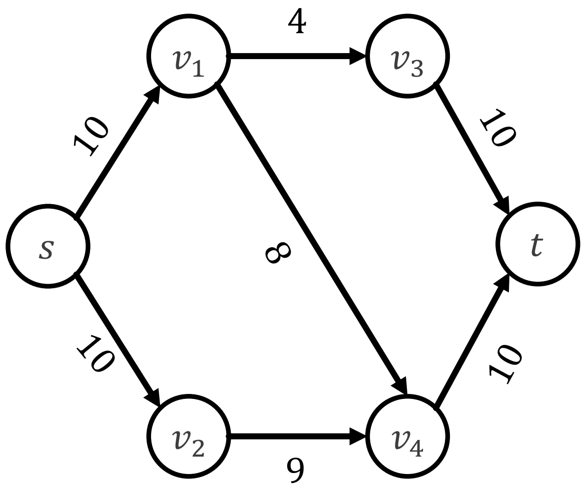
$$|f| = 0$$



G



G_f

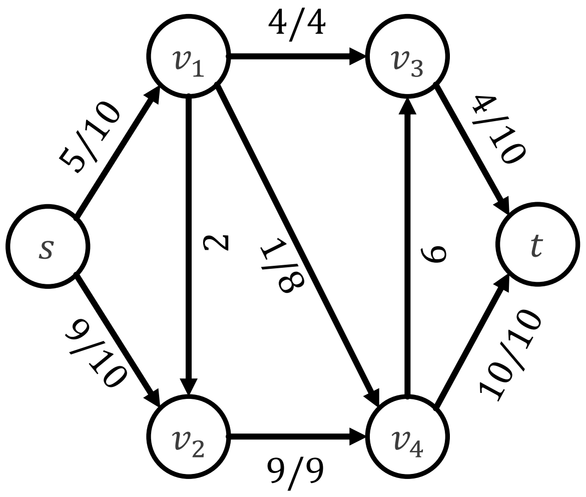


G_L

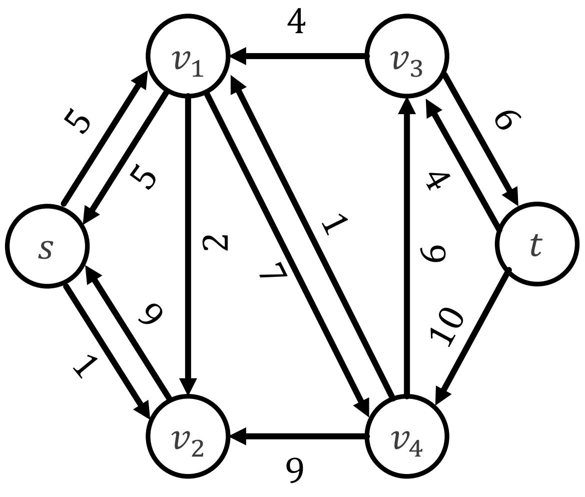
厦门大学信息学院 SCHOOL OF INFORMATICS XIAMEN UNIVERSITY 厦门大学计算机科学系 Computer Science Department of Xiamen University

Classroom Exercise

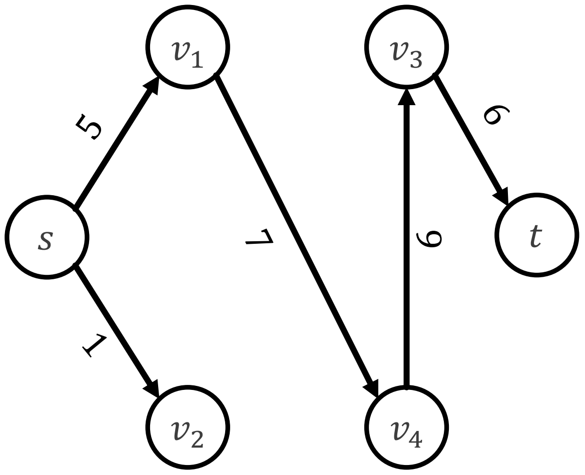
$$|f| = 14$$



G



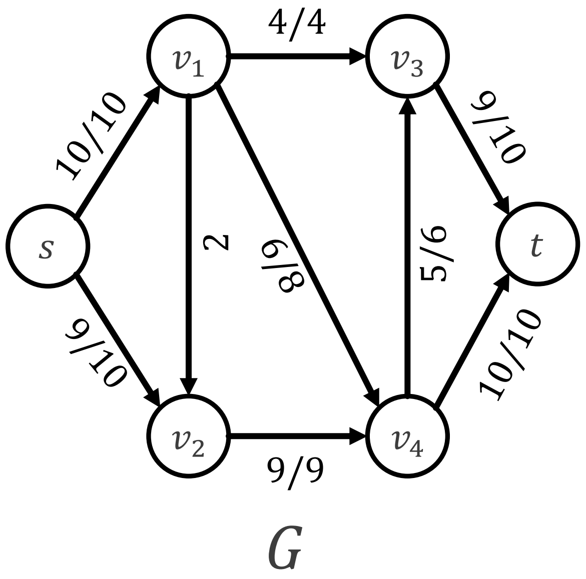
G_f

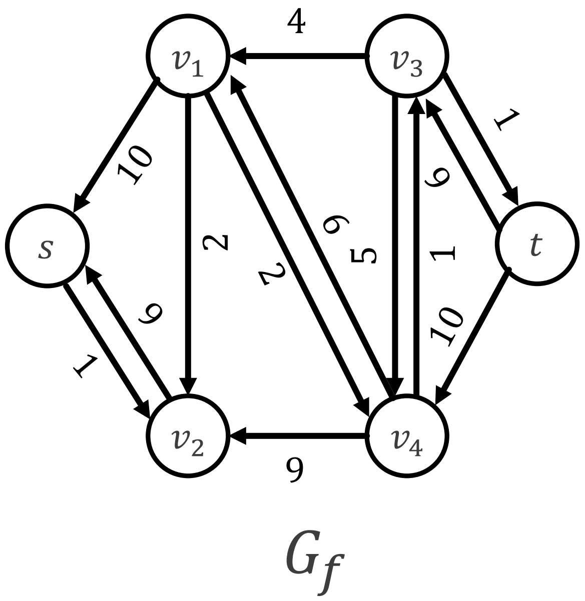


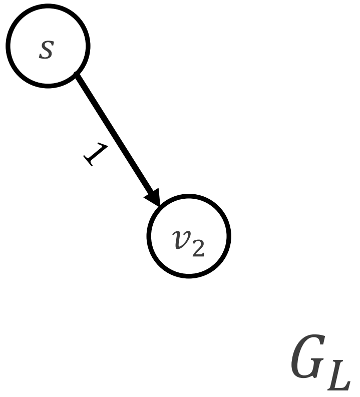
G_L

Classroom Exercise

$$|f| = 19$$

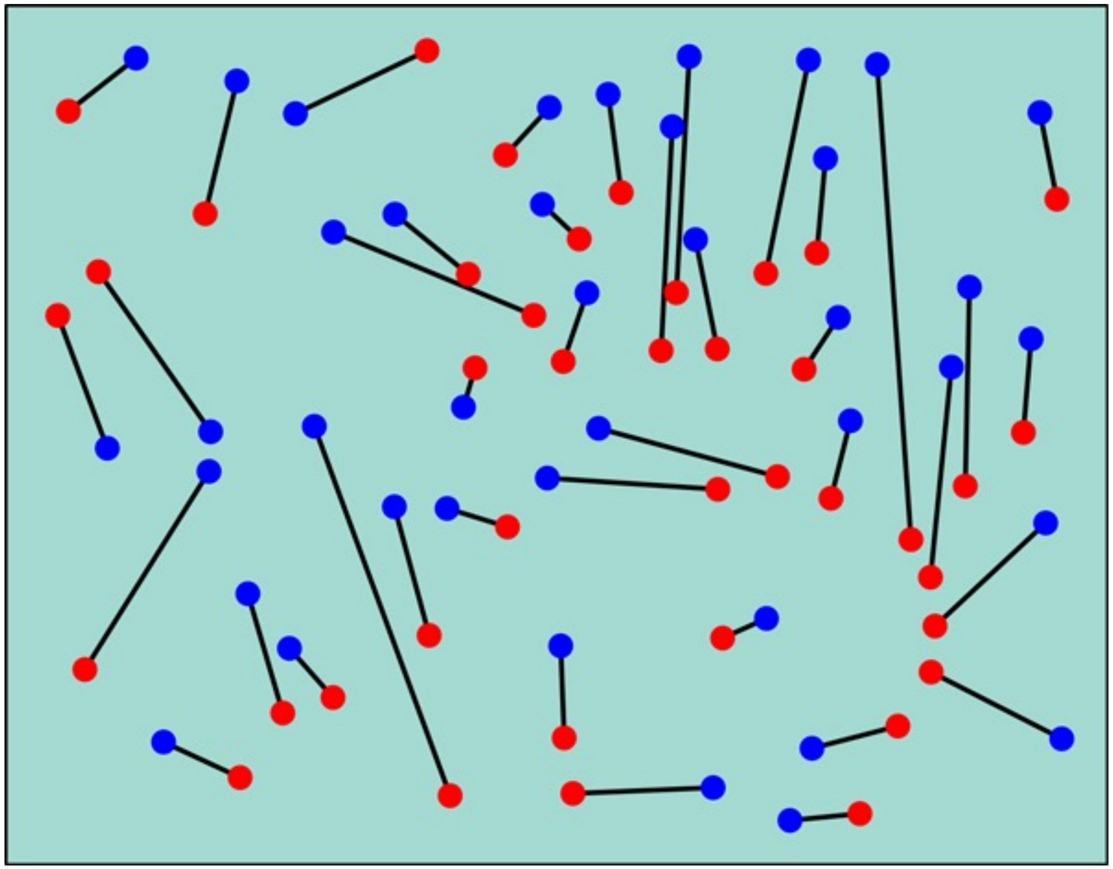






MATCHING PROBLEM

Matching Problem



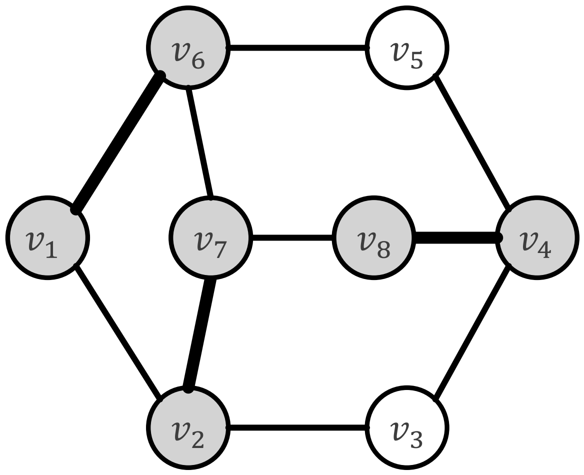
Matching Problem

Definition 9.14

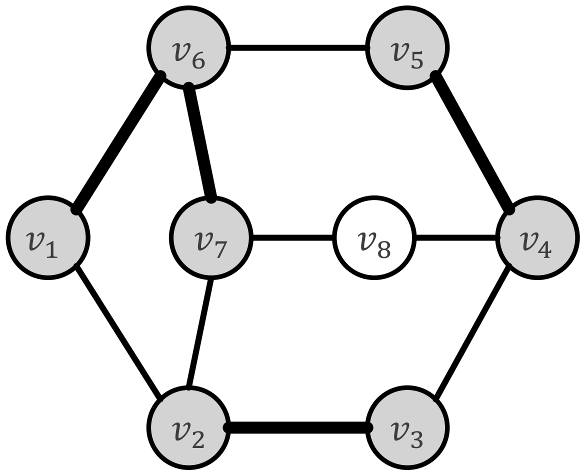
Given an undirected graph $G = (V, E)$, a matching (匹配) is a subset of edges $M \subseteq E$ such that for all vertices $v \in V$, at most one edge of M is connected on v.

We say that a vertex $v \in V$ is matched by matching M if some edge in M is connected on v; otherwise, v is unmatched.

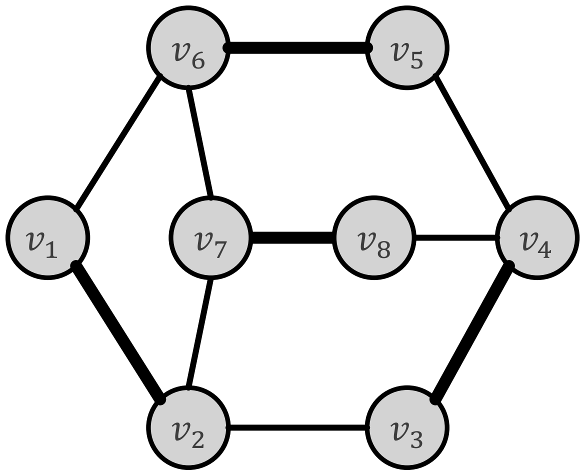
A maximum matching (最大匹配) is a matching of maximum cardinality.



A match



Not a match



A maximum match

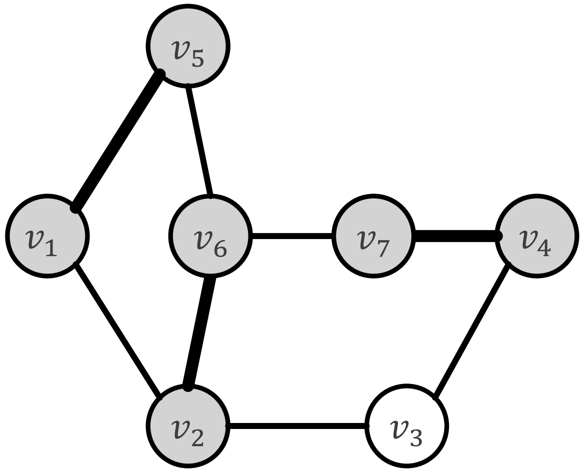
Dark vertex means matched

Matching Problem

Definition 9.15

Perfect matching (完美匹配) is one in which every vertex in V is matched.

- Not every undirected graph has perfect matching, e.g. graphs with odd number of vertices.
- The maximum matching may not be perfect, but perfect matching must be maximum.



A maximum but not perfect matching

Matching Problem

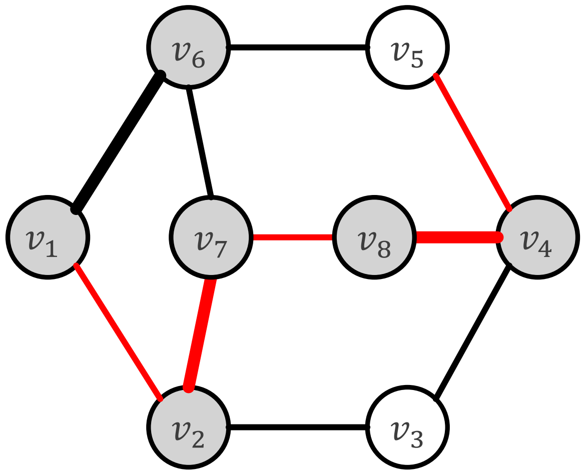
Definition 9.15

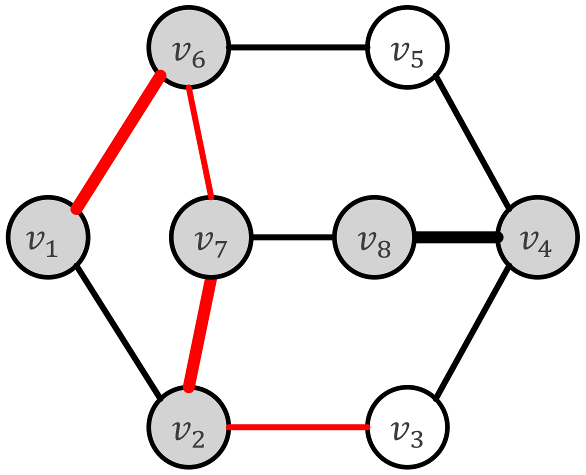
Given an undirected graph $G = (V, E)$ and M is a matching of G. We define:

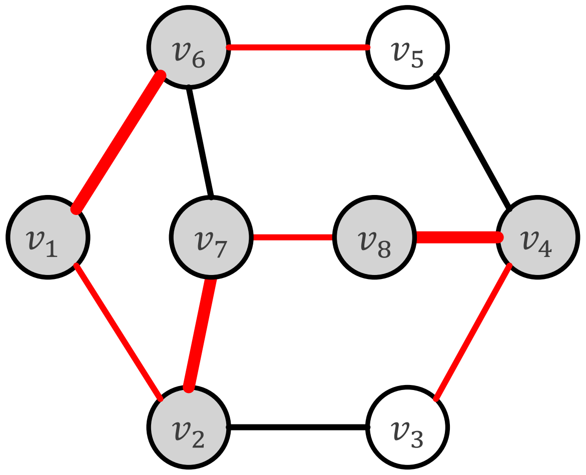
Alternating path (交错路径): A simple path alternating between matching and non-matching edges.

Augmenting path (增广路径): A nontrivial alternating path that begins and ends with unmatched vertices.

Matching Problem

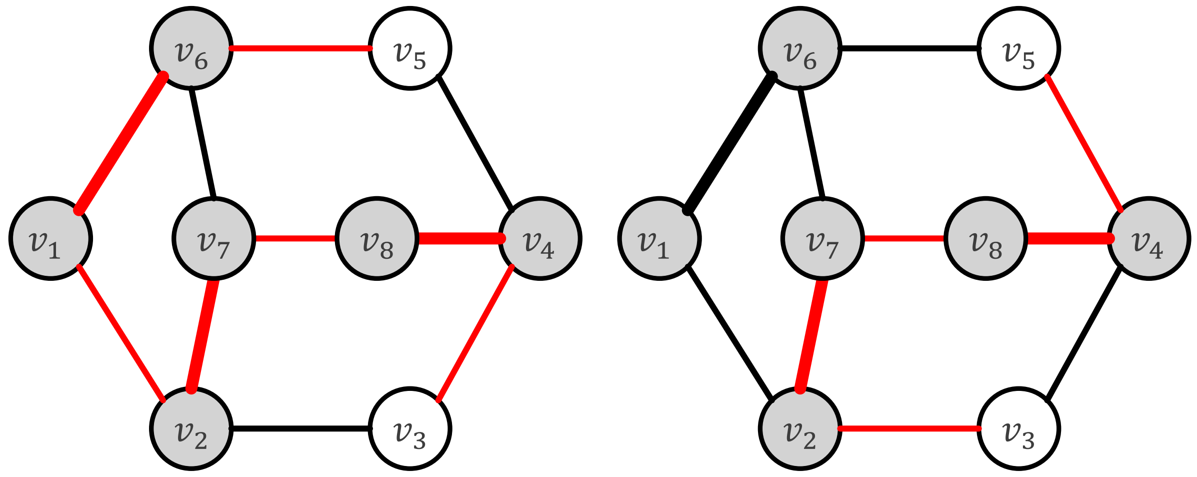






Alternating path

Matching Problem



Augmenting path

Matching Problem

Definition 9.17

Let M_1 and M_2 be two matchings in a graph G, define

$$M_1 \oplus M_2 = (M_1 \cup M_2) - (M_1 \cap M_2)$$

Lemma 9.6

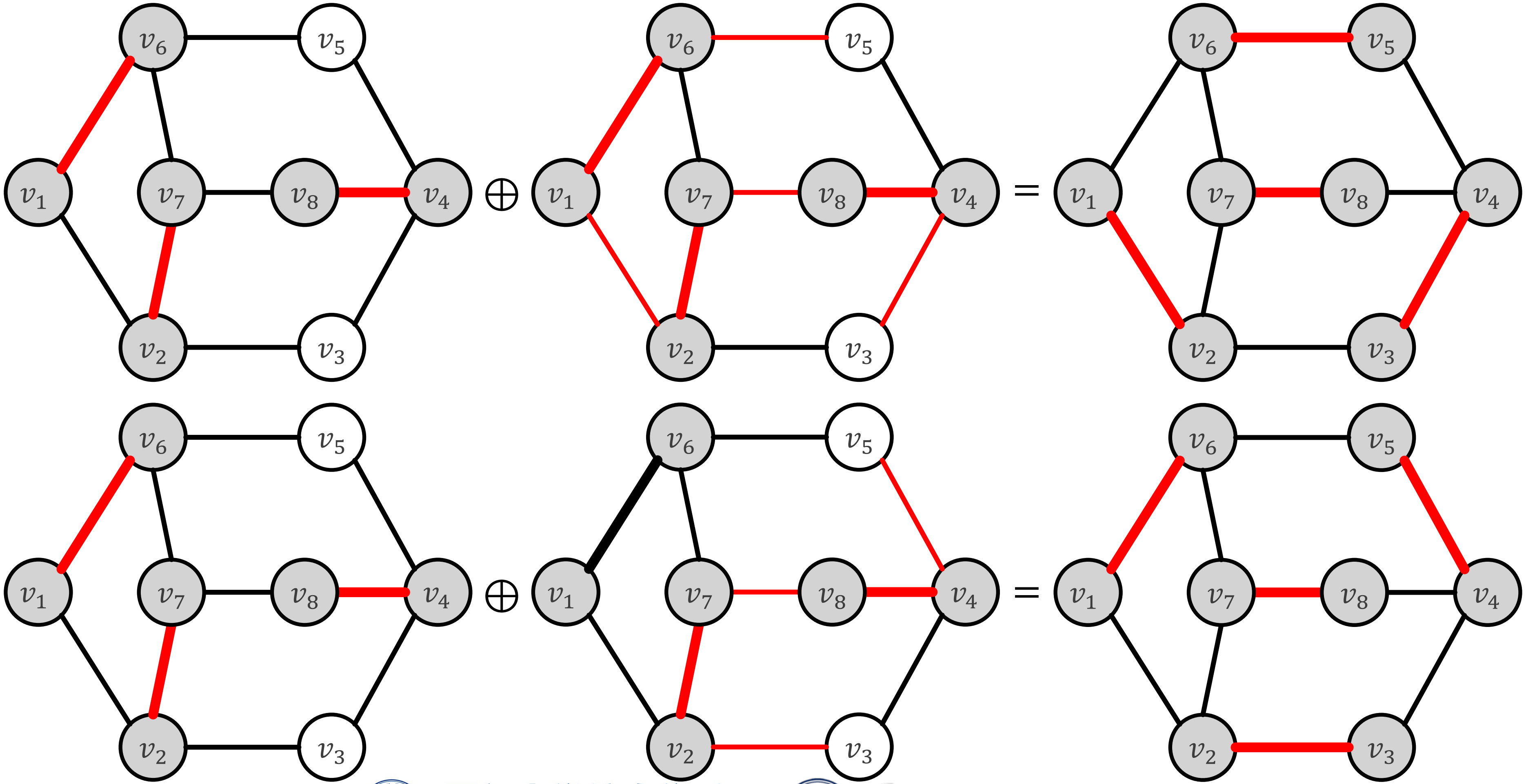
Suppose M is a matching and p is an augmenting path, Then $M \oplus p$ is also a matching, and $|M \oplus p| = |M| + 1$.

Theorem 9.9

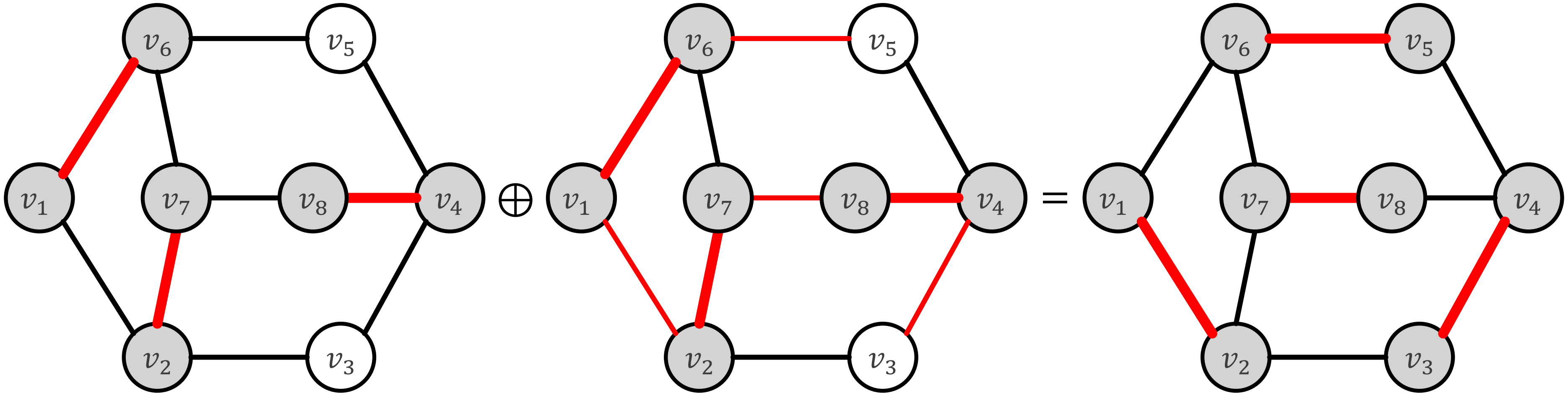
A matching is maximum if and only if there is no augmenting path.

- All these things tell you: once you can find a augmenting path, adding it to the current matching results in a larger matching.

Example



Example



- What $\oplus$ does is actually flipping matching and non-matching edges in p.
- There are always one more non-matching edge in p, by its definition.
  - Begins and ends with unmatched vertices

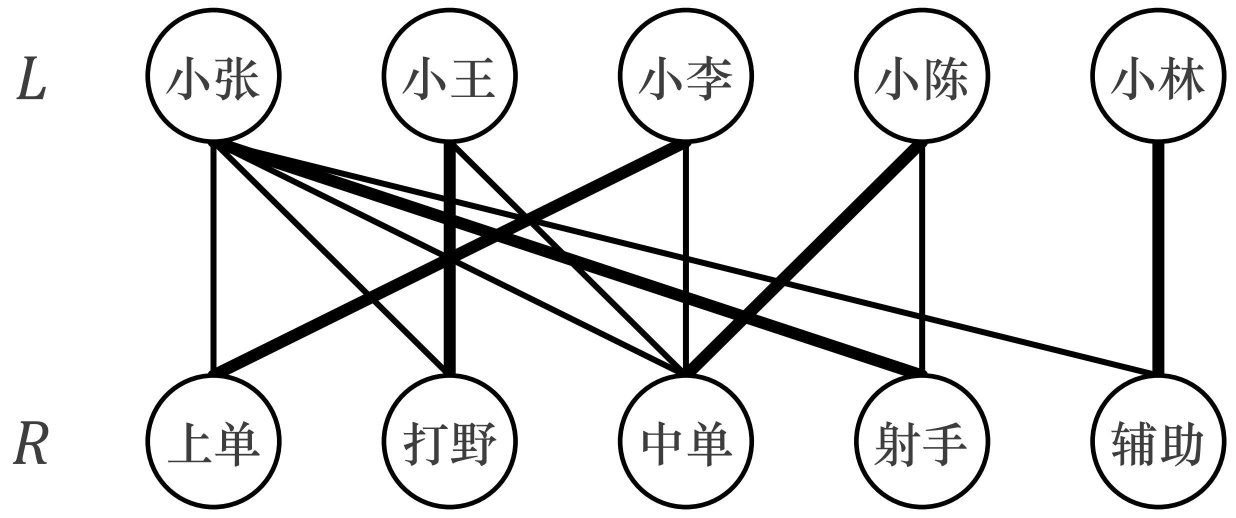
Max Matching Algorithm

```
MaximumMatching($G$)
1  $M \leftarrow 0$
2  **repeat**
3      find an augmenting path $p$
4      $M \leftarrow M \oplus p$
5  **until** no augmenting path in $G$
6  **return** $M$
```

Bipartite Graph

Definition

A bipartite graph (二分图, 二部图) $G = (V, E)$ is defined if V can be partitioned into two subsets L and R with $V = L \cup R$ and $L \cap R = \emptyset$, and $\forall e \in E$ has one endpoint in L and the other endpoint in R.



Maximum Matching Algorithms for Bipartite Graph

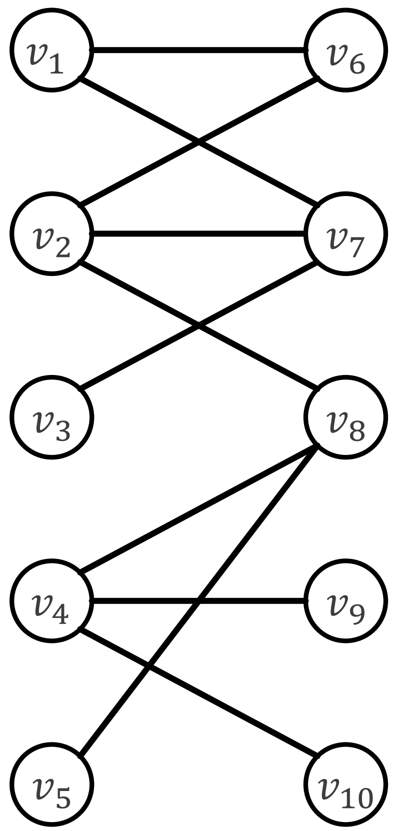
Solution 1: Use the max-flow algorithm.

- We define the corresponding flow network $G' = (V', E')$ for the bipartite graph G as follows.
  - G' is directed but G is undirected.
- We let the source s and sink t be new vertices: $V' = V \cup \{s, t\}$. The directed edges of G' contain three parts: edges from s to L, from L to R, and from R to t:

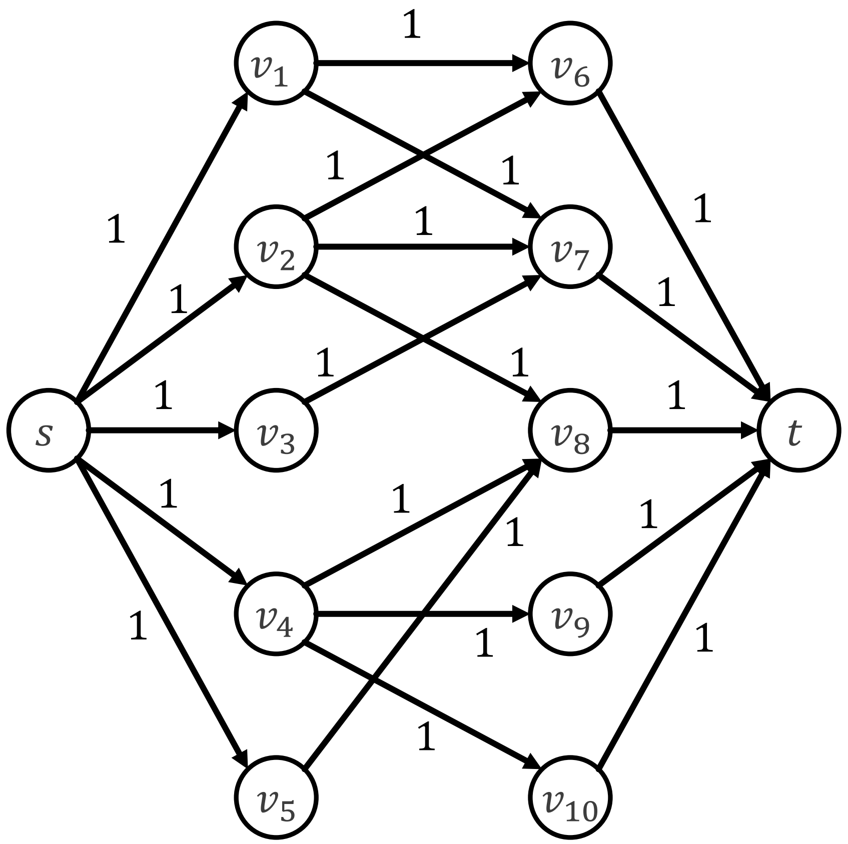
$$\begin{aligned} E' &= \{(s,u): u \in L\} \\ &\cup \{(u,v): u \in L, v \in R, (u,v) \in E\} \\ &\cup \{(v,t): v \in R\}. \end{aligned}$$

- All edges in E' have capacity 1.

Example



$G = (V, E)$



$G' = (V', E')$

Maximum Matching Algorithms for Bipartite Graph

Theorem 9.11

Let $G = (V, E)$ be a bipartite graph with vertex partition $V = L \cup R$, and let $G' = (V', E')$ be its corresponding flow network.

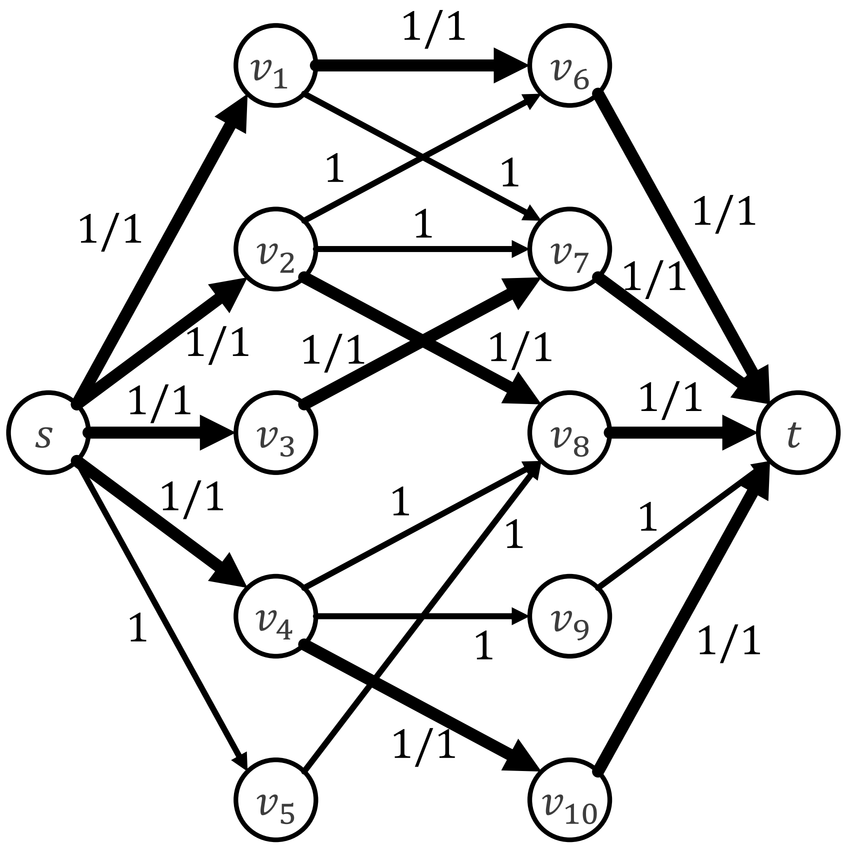
If M is a matching in G, then there is an integer-valued flow f in G' with value $|f| = |M|$.

Conversely, if f is an integer-valued flow in G', then there is a matching M in G with cardinality $|M| = |f|$.

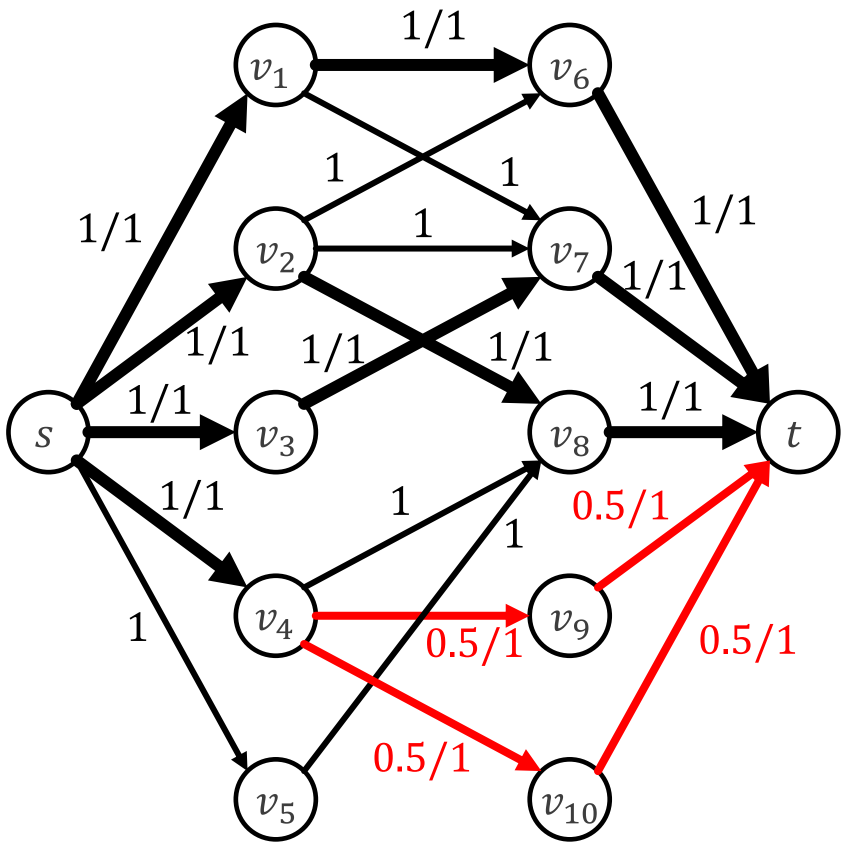
- This Theorem tells you: Just run max-flow algorithm and you will obtain maximum matching.
- The value of flow needs to be integer because floating point flow does not correspond to a match in G.

厦门大学信息学院
SCHOOL OF INFORMATICS XIAMEN UNIVERSITY
厦门大学计算机科学系
Computer Science Department of Xiamen University

Example



Integer-valued flow

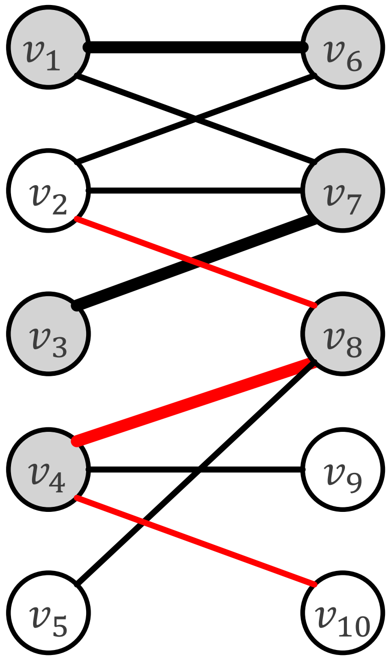


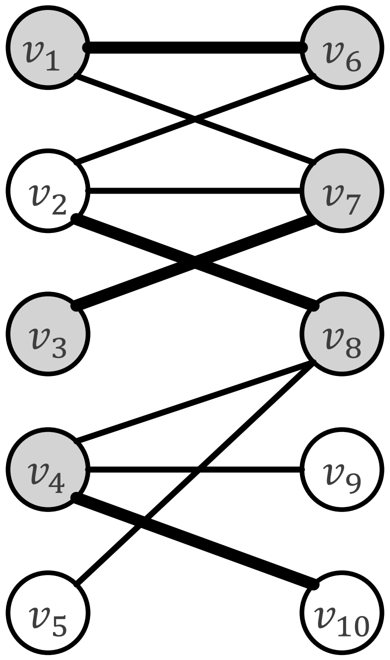
Non integer-valued flow

Maximum Matching Algorithms for Bipartite Graph

Solution 2: Hungarian tree algorithm (匈牙利树算法).

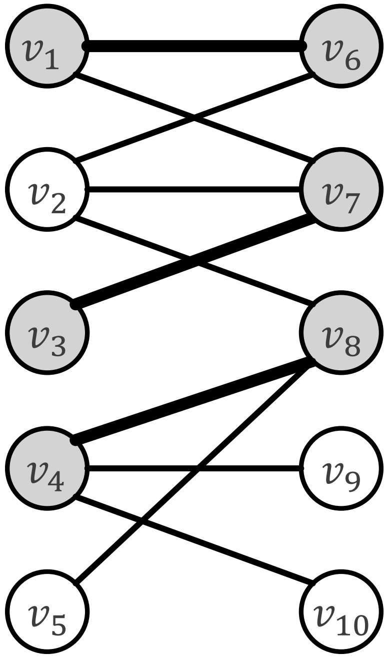
- Idea: Find augmenting path and add the path to increase matching by MaximumMatching.



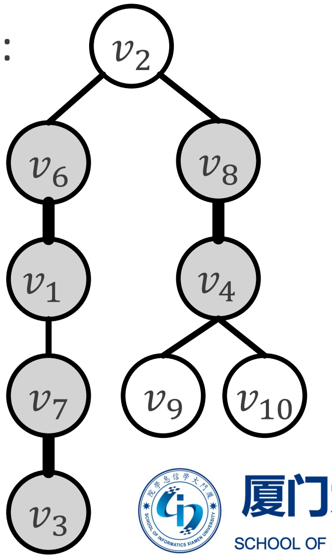


Hungry Tree Algorithm

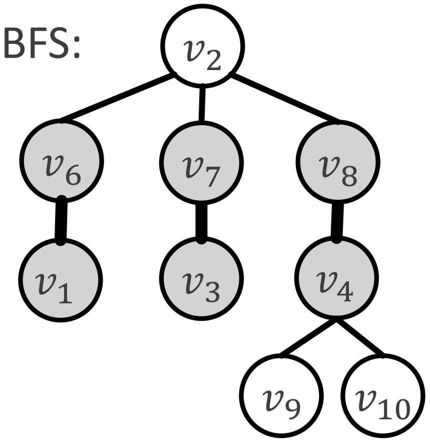
- But how to find an augmenting path?
- We can start from an unmatched vertex and use BFS or DFS to search all alternating paths. The generated tree is called alternating path tree (交错路径树).
- If all leaves in the alternating path tree are matched, this tree is called Hungarian tree (匈牙利树).



DFS:



BFS:

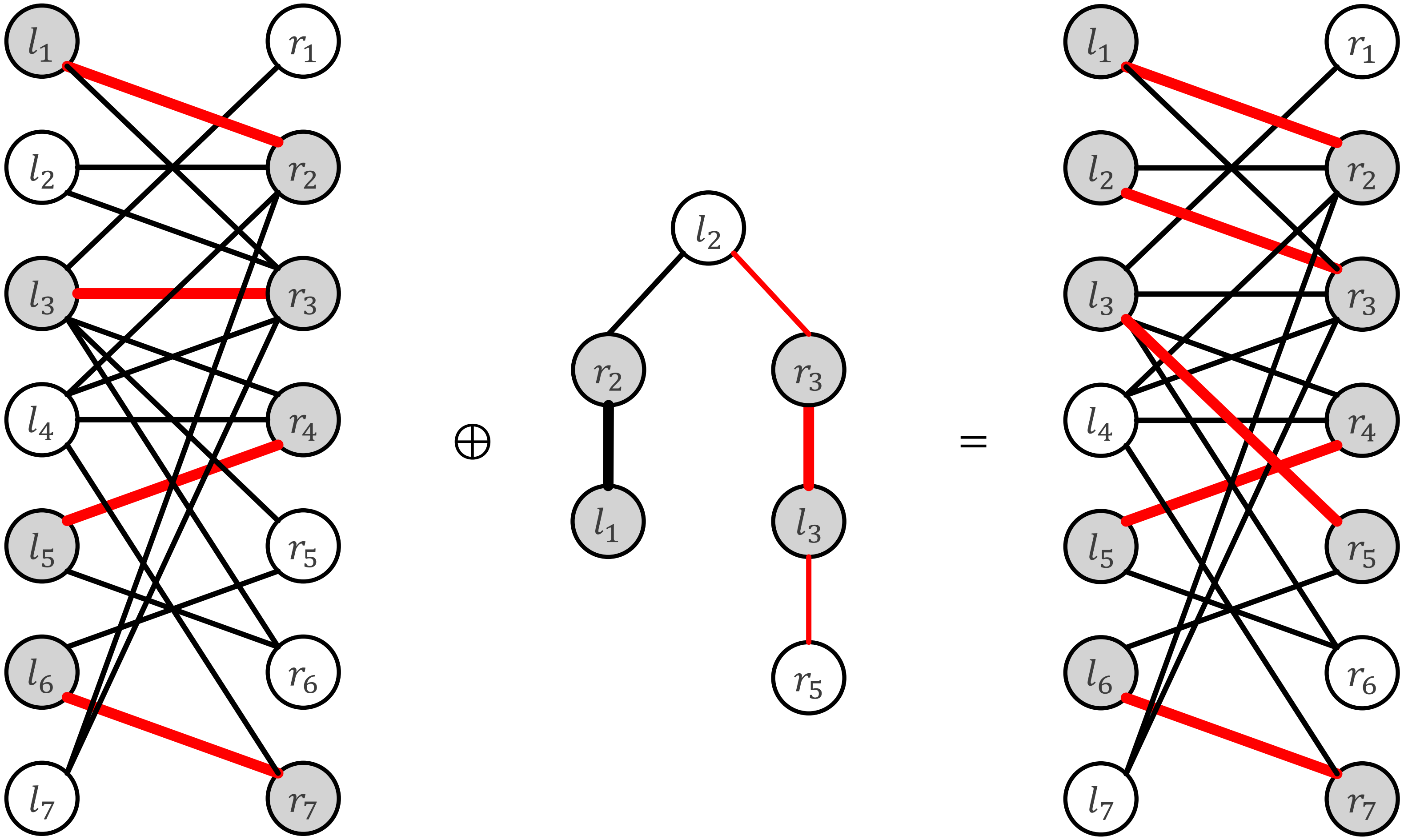


厦门大学信息学院 SCHOOL OF INFORMATICS XIAMEN UNIVERSITY 厦门大学计算机科学系 Computer Science Department of Xiamen University

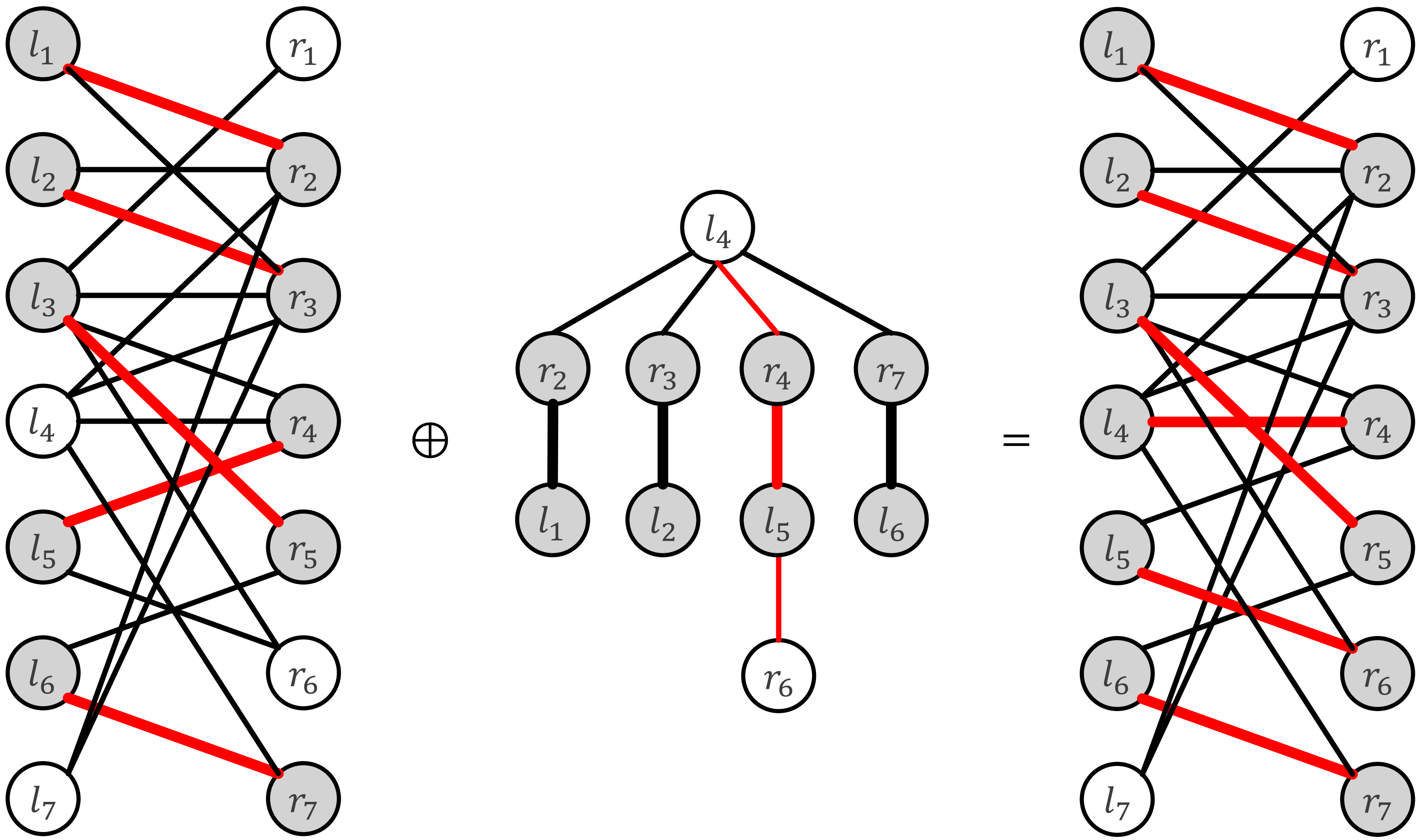
Hungry Tree Algorithm

```
HungryBipartiteGraph($G$)
1  begin with any matching $M$
2  **while** there exists an unmatched vertex in $L$ and $R$ respectively **do**
3      let $r$ is an unmatched vertex in $L$
4      grow an alternating path tree $T$ with root $r$ using breadth first search
5      **if** $T$ is a Hungarian tree **then**
6          $G \leftarrow G - T$
7      **else**
8          find an augmenting path $p$ in $T$
9          $M \leftarrow M \oplus p$
10 **return** $M$
```

Example

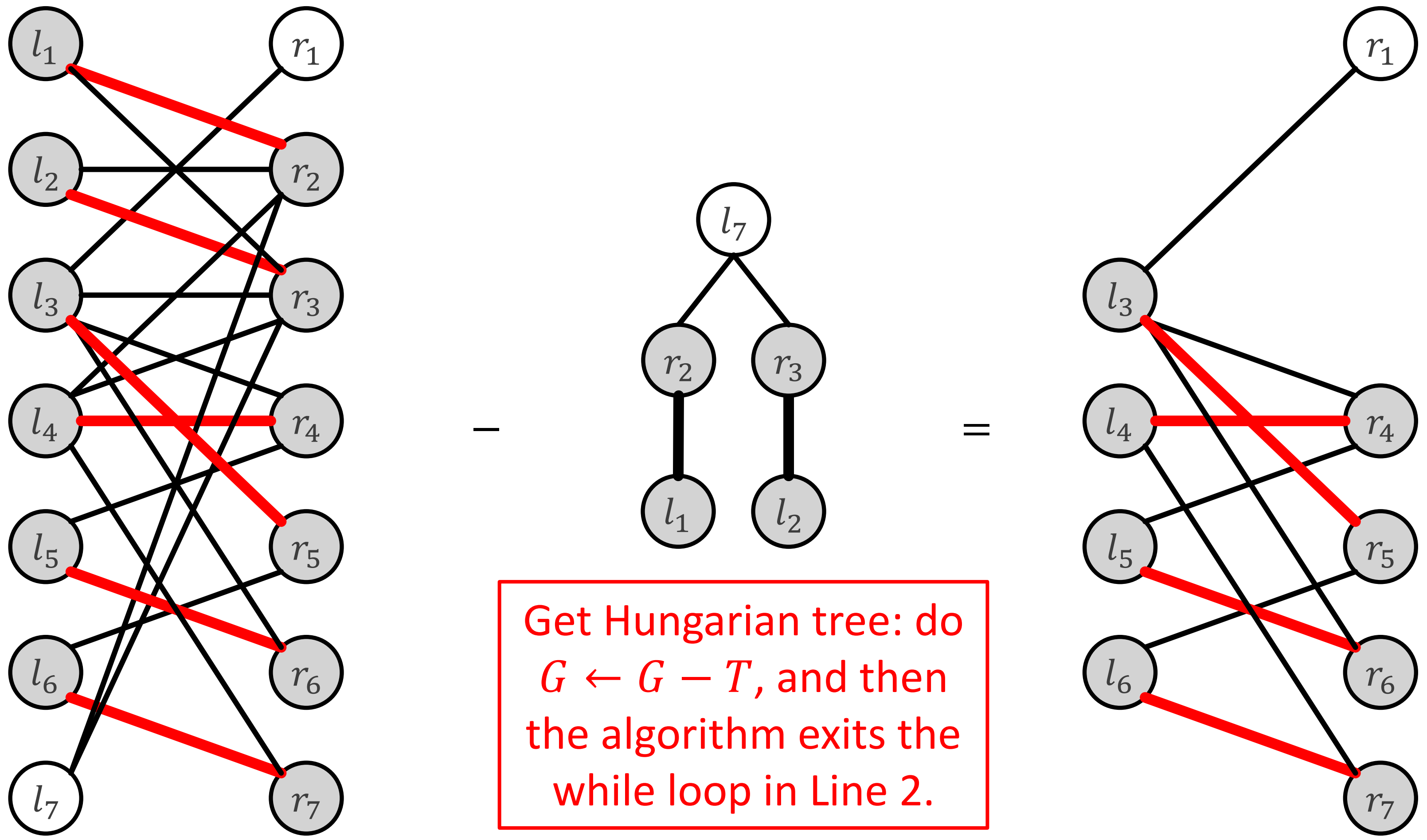

l_1
l_2
l_3
l_4
l_5
l_6
l_7
r_1
r_2
r_3
r_4
r_5
r_6
r_7
$\oplus$
l_2
r_2
r_3
l_1
l_3
r_5
$=$
l_1
l_2
l_3
l_4
l_5
l_6
l_7
r_1
r_2
r_3
r_4
r_5
r_6
r_7

Example



l_1
l_2
l_3
l_4
l_5
l_6
l_7
r_1
r_2
r_3
r_4
r_5
r_6
r_7
$\oplus$
l_4
r_2
r_3
r_4
r_7
l_1
l_2
l_5
l_6
r_6
$=$

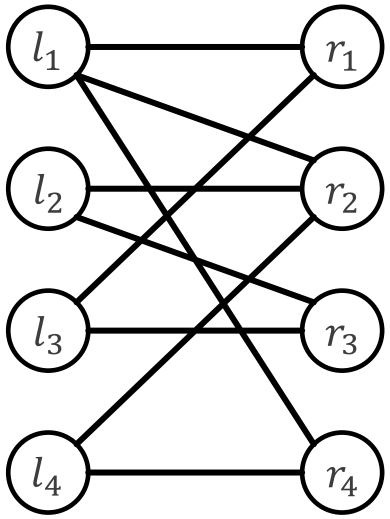
Example



Get Hungarian tree: do $G \leftarrow G - T$, and then the algorithm exits the while loop in Line 2.

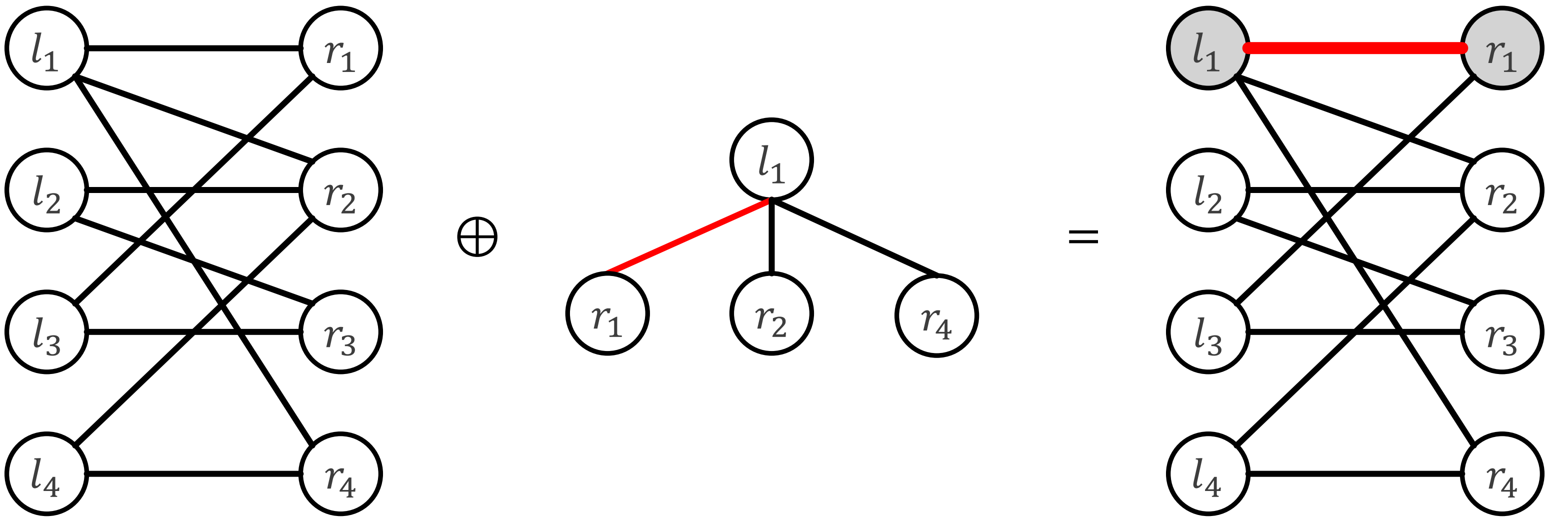
Classroom Exercise

Use Hungarian tree algorithm to find the maximum matching of the following bipartite graph.



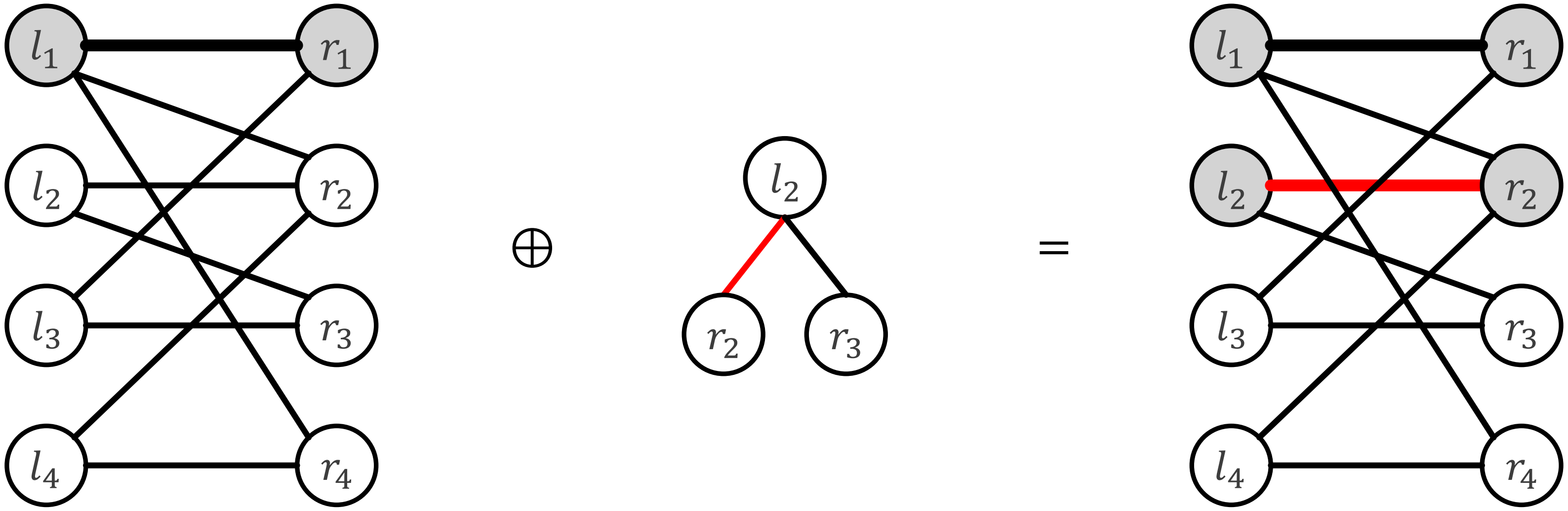
Classroom Exercise

Pick l_1 to generate alternating path tree, and
Select an augmenting path and add into G



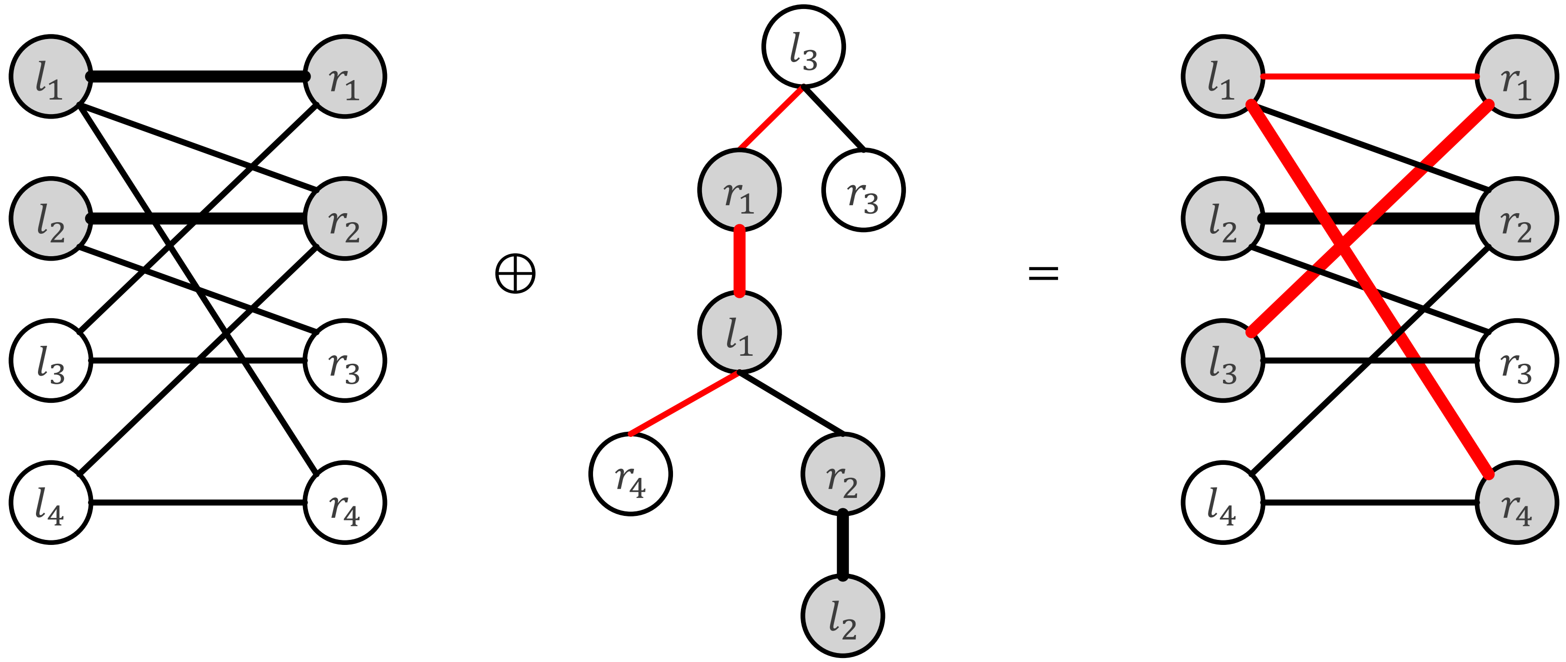
Classroom Exercise

Pick l_2 to generate alternating path tree, and
Select an augmenting path and add into G



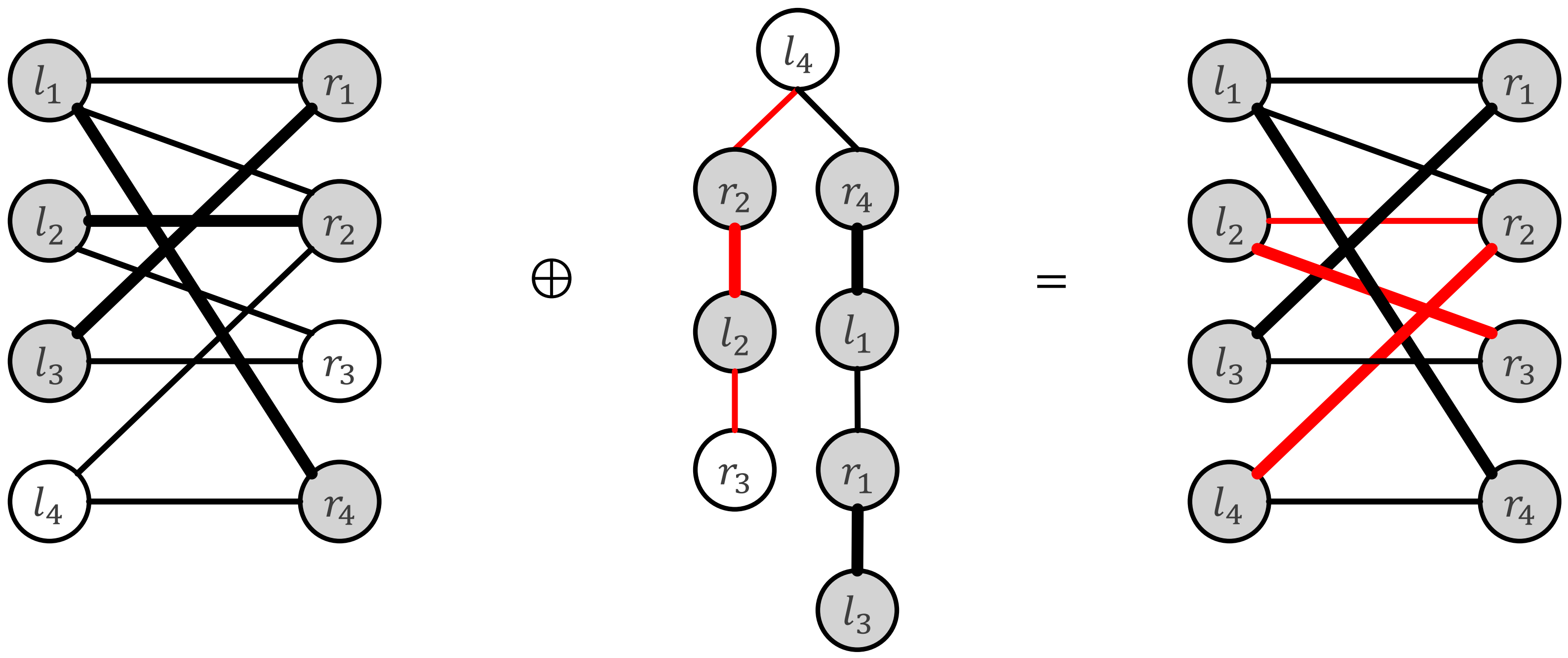
Classroom Exercise

Pick l_3 to generate alternating path tree, and Select an augmenting path and add into G



Classroom Exercise

Pick l_4 to generate alternating path tree, and
Select an augmenting path and add into G



厦门大学信息学院 SCHOOL OF INFORMATICS XIAMEN UNIVERSITY 厦门大学计算机科学系 Computer Science Department of Xiamen University

Conclusion

After this lecture, you should know:

- What is network, capacity and flow in terms of graph.
- What properties does flow have.
- How to obtain the residual network.
- How to find an augmenting path and its bottleneck capacity.
- How to use the augmenting path to increase value of flow.
- How to use shortest path to improve max-flow problem.
- How to solve matching problem.

Homework

Page 172-174

9.1

9.3

9.4

9.6

9.10

9.12

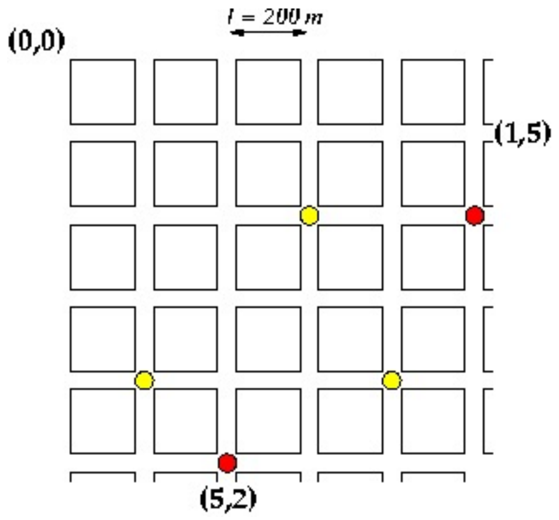
9.19

Experiment

滴滴出行
—滴滴一下 美好出行—

DiDi scheduler problem

- In Beijing all streets are built as straight lines intersecting at right angles at fixed distances, with the distance between intersections being a fixed 200 meters.
- Now, three passengers (marked yellow) want to call available cars (marked red) by DiDi.
- If you are a DiDi scheduler, how many passengers can you transported?



Experiment

滴滴出行
—滴滴一下 美好出行—

You have to adhere to following constraints:

- Each car can only take one passenger (no 顺风车).
- Passengers and cars always wait at intersections of roads.
- The car has to reach the passenger within a given time limit (otherwise the passenger will cancel DiDi and use CaoCao)

Experiment

滴滴出行
—滴滴一下 美好出行—

Input:

- The first line contains:
  - The number of passengers p $(1 <= p <= 400)$.
  - The number of available cars t $(1 <= t <= 200)$.
  - The speed s $(1 <= s <= 2000)$ of the cars in meters per seconds.
  - The time c to collect a passenger in seconds $(1 <= c <= 1000000)$.
- The next p lines contains the position of the passengers.
- The next t lines contain the position of the available cars.

Output:

- The maximal number of passengers that can be picked up.

```
Input:
2 3 10 40
2 5
5 2
2 3
4 1
4 4
```

```
Output:
2
```

厦门大学信息学院
SCHOOL OF INFORMATICS XIAMEN UNIVERSITY
厦门大学计算机科学系
Computer Science Department of Xiamen University

谢谢

有问题欢迎随时跟我讨论