图算法篇：最大流

童咏昕

北京航空航天大学
计算机学院

中国大学MOOC北航《算法设计与分析》

提纲

问题背景

算法思想

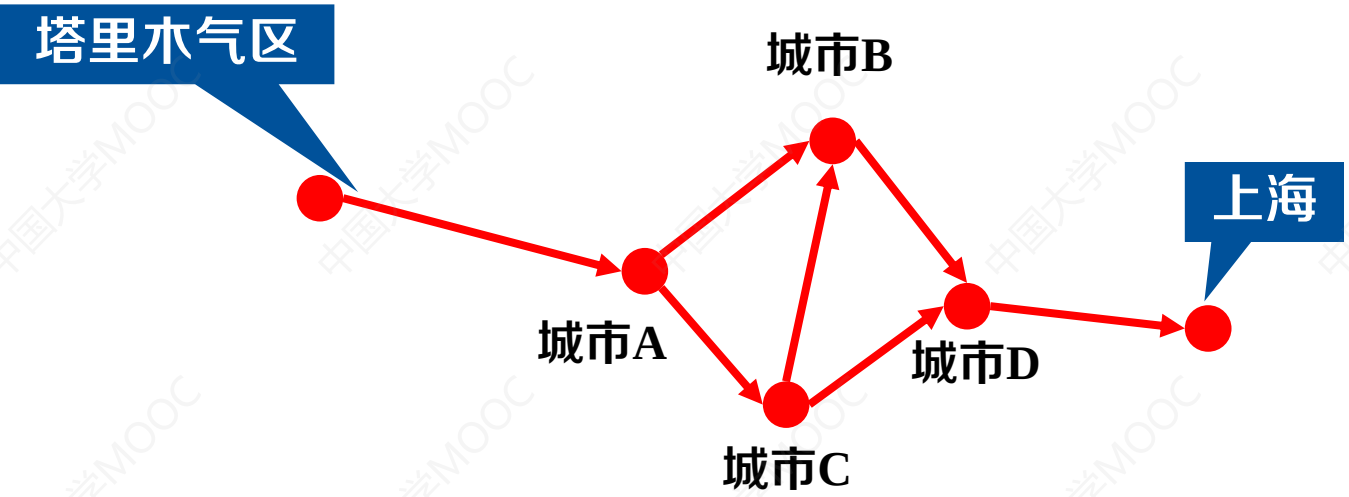
算法实例

算法分析

算法性质

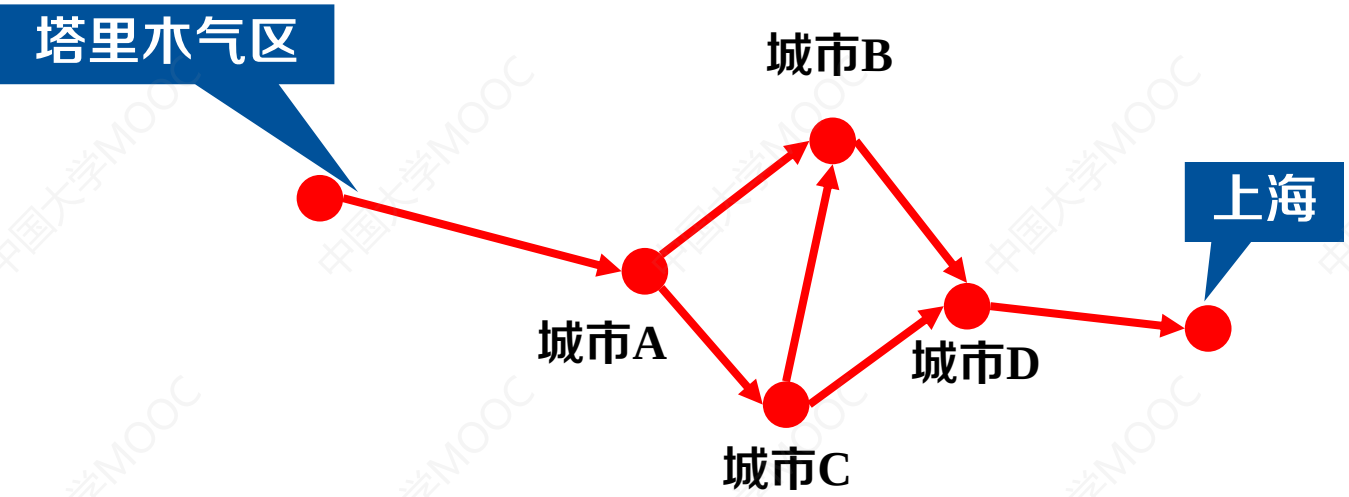
问题背景：西气东输

- 从塔里木气区到上海输送天然气，如何计算最大输送量?



问题背景：西气东输

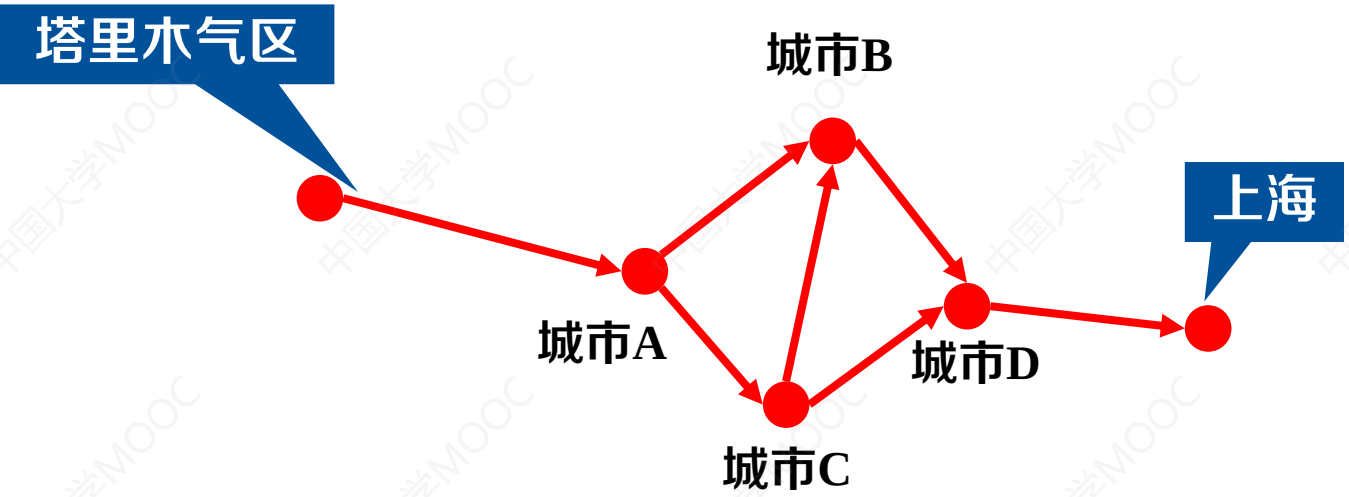
- 从塔里木气区到上海输送天然气，如何计算最大输送量?



问题：如何描述输送线路?

问题背景：西气东输

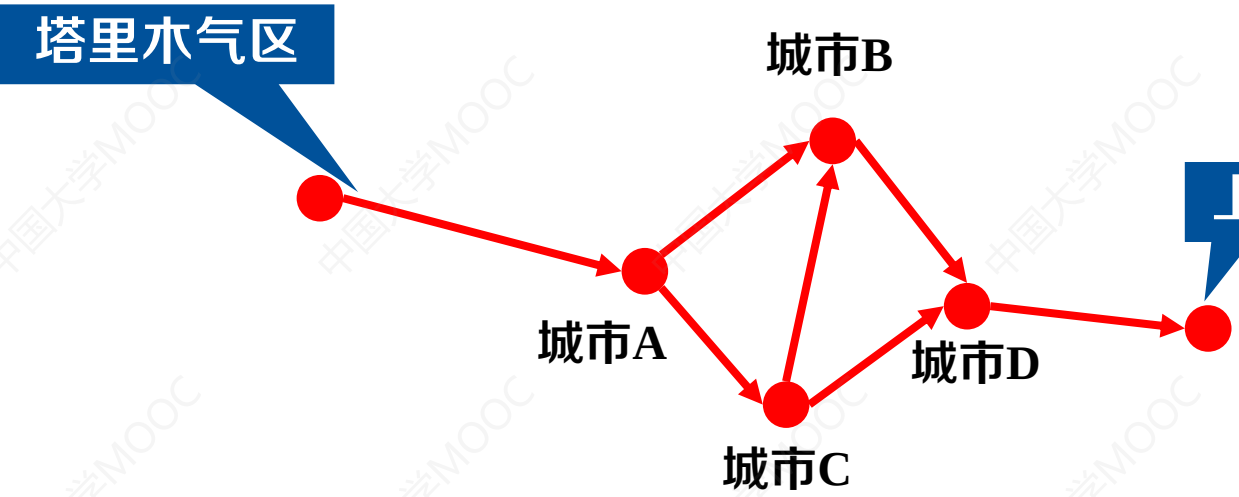
- 从塔里木气区到上海输送天然气，如何计算最大输送量?

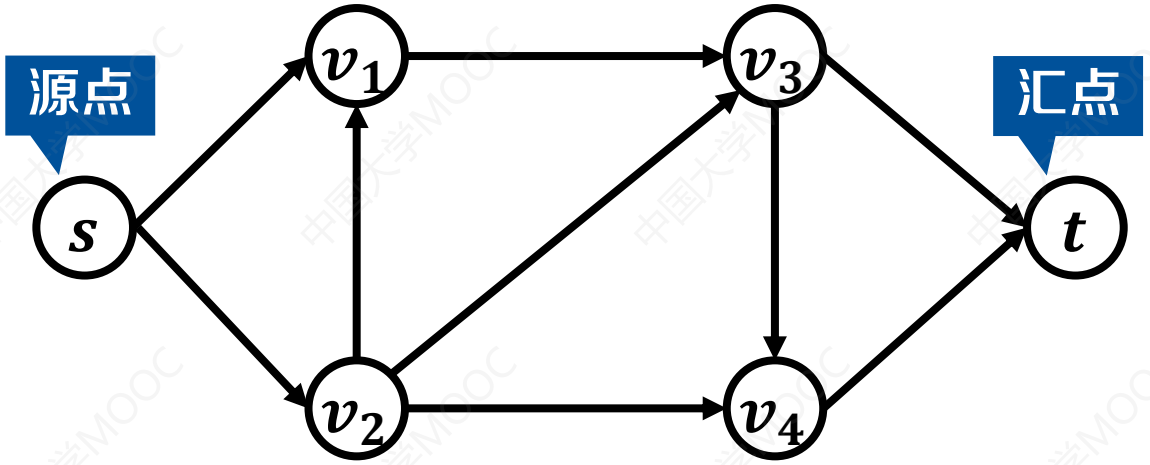


将输送线路抽象为有向图

问题背景：西气东输

- 从塔里木气区到上海输送天然气，如何计算最大输送量?



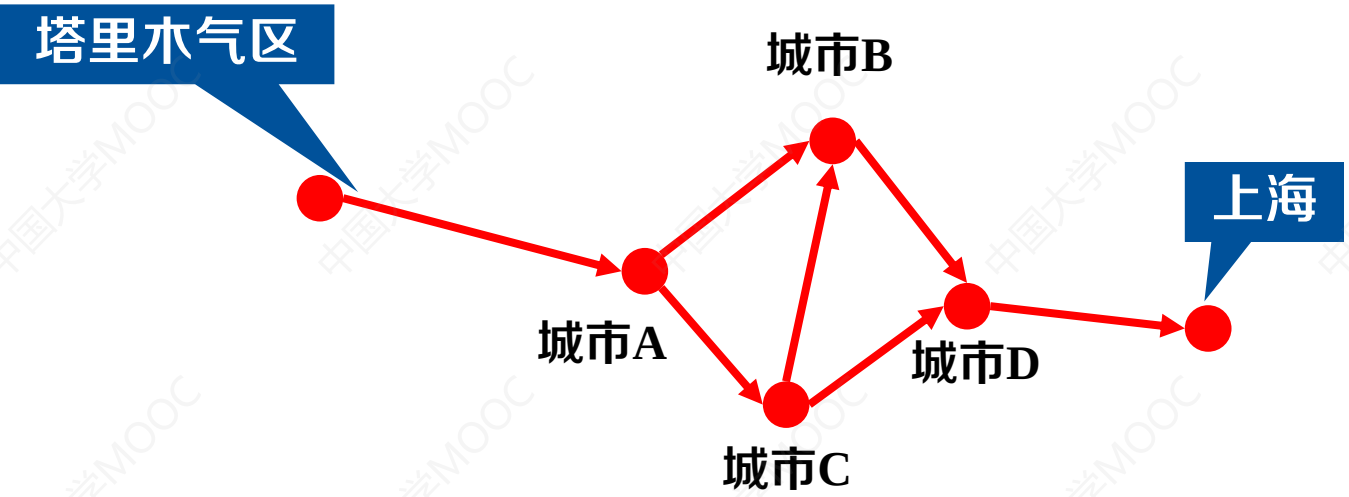


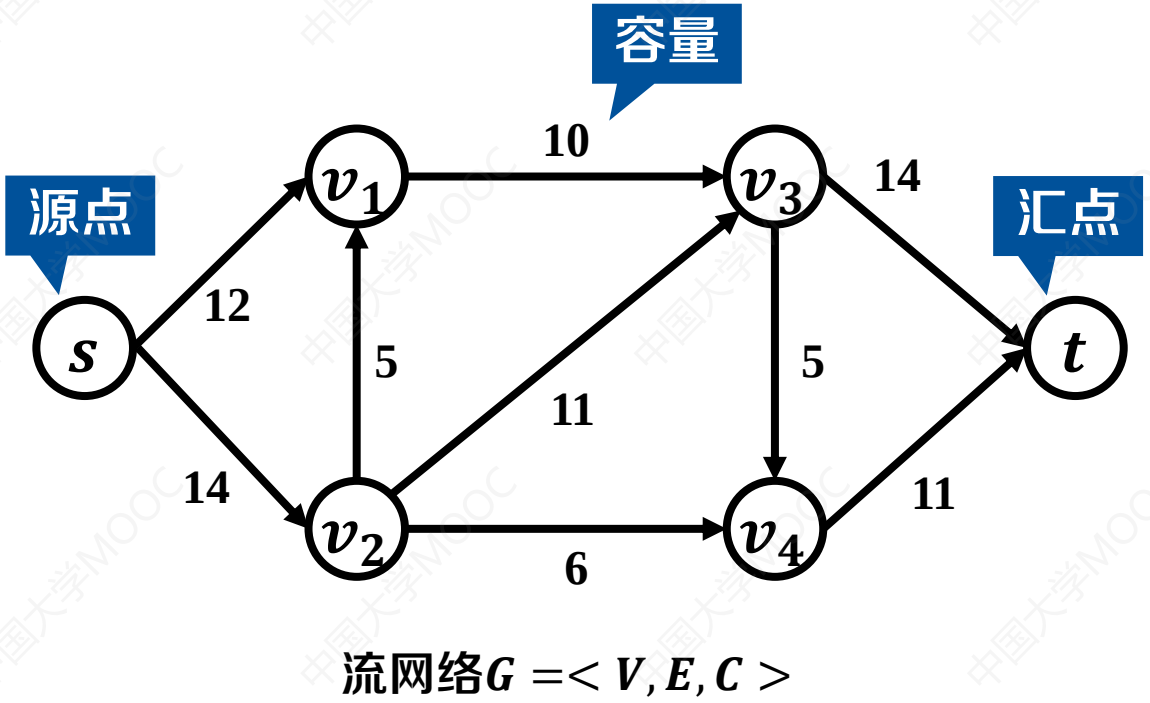
流网络$G =< V, E >$

城市视为顶点，管道视为边

问题背景：西气东输

- 从塔里木气区到上海输送天然气，如何计算最大输送量?



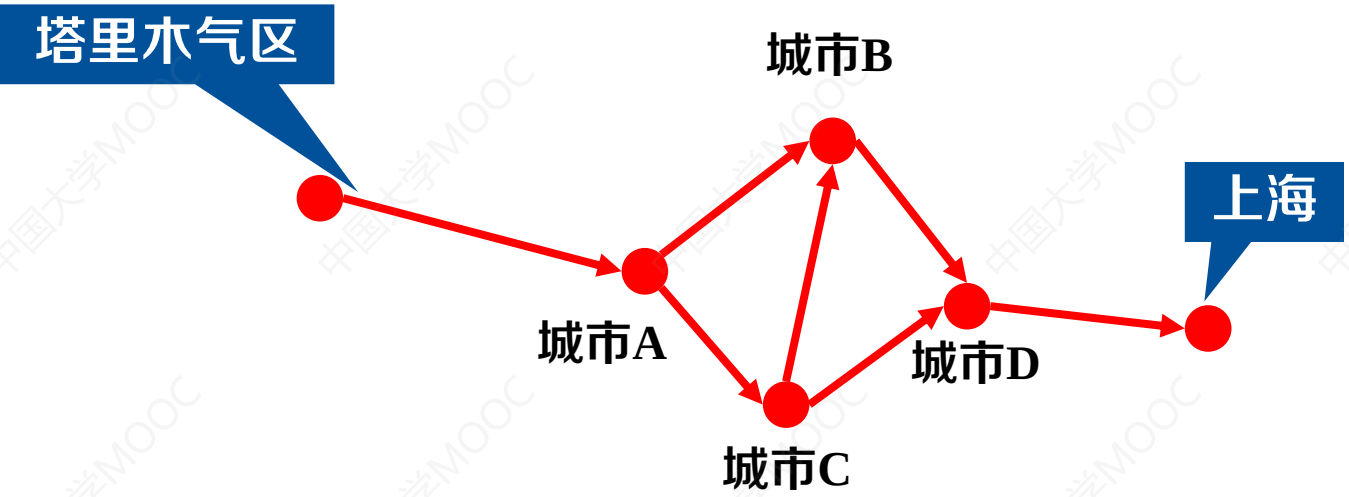


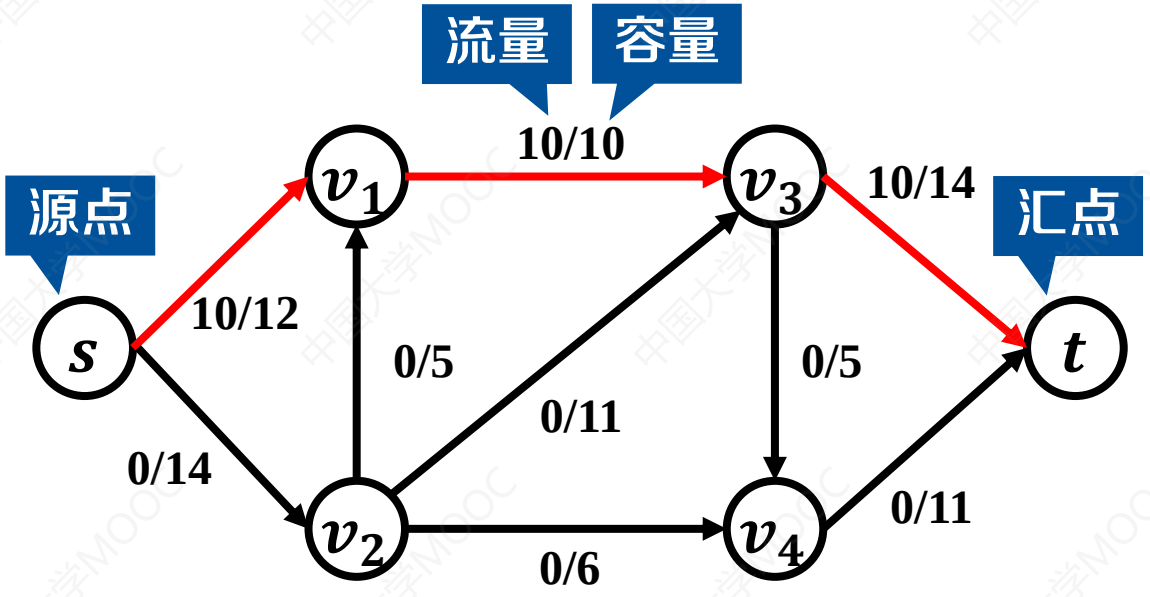
流网络$G =< V, E, C >$

用边的权值表示管道的容量

问题背景：西气东输

- 从塔里木气区到上海输送天然气，如何计算最大输送量?





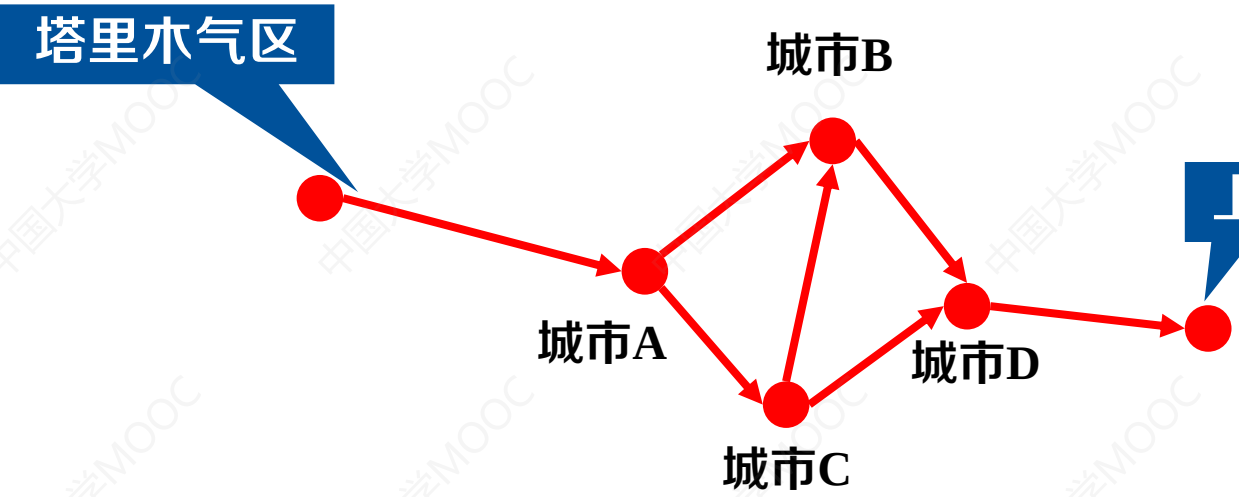
流网络$G =< V, E, C >$

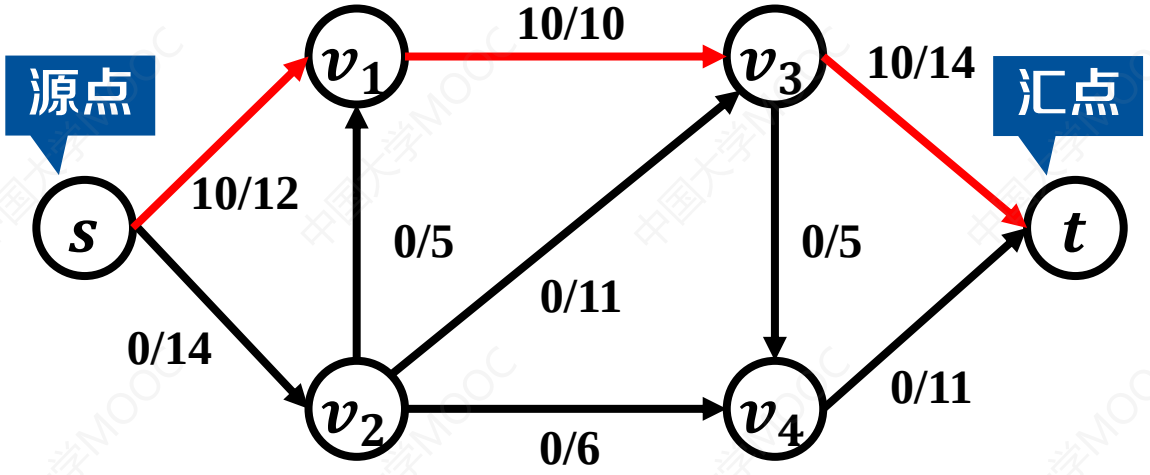
用边的权值表示管道的容量

路径上的流量不应超过边的容量

问题背景：西气东输

- 从塔里木气区到上海输送天然气，如何计算最大输送量?

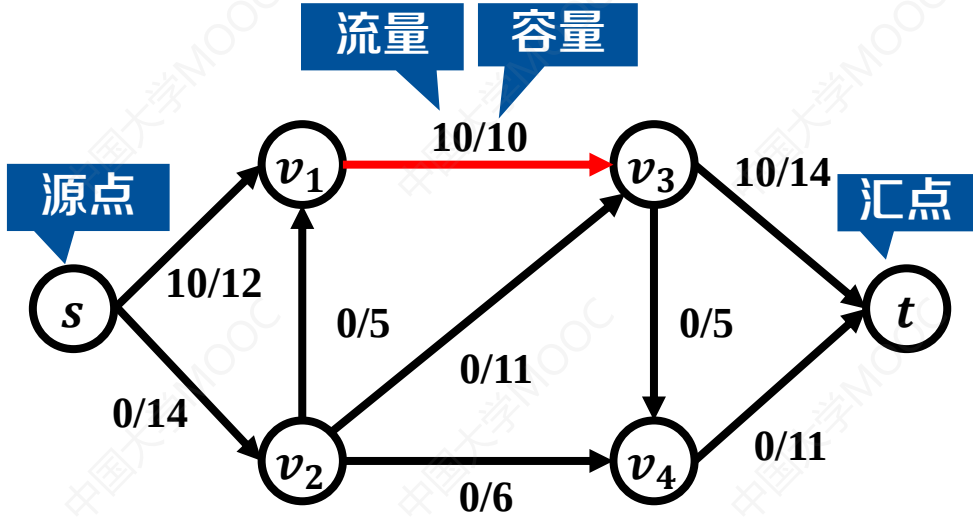




计算最大天然气输送量（最大化总流量）

流网络

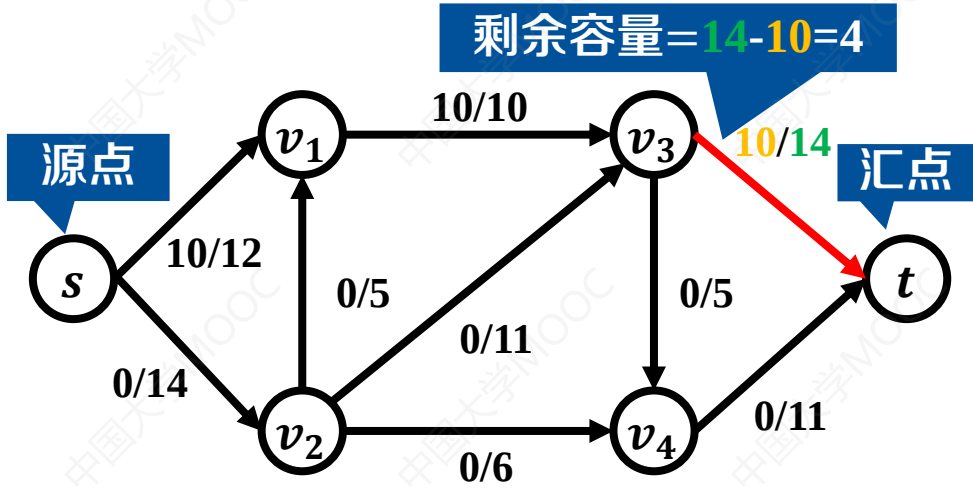
- 给定有向图$G =< V, E, C >$，其被称为流网络：
  - 容量：对于每条边$c(e) \geq 0$
  - 流量：对于每条边$c(e) \geq f(e) \geq 0$



流网络

- 给定有向图$G =< V, E, C >$，其被称为流网络：
  - 容量：对于每条边$c(e) \geq 0$
  - 流量：对于每条边$c(e) \geq f(e) \geq 0$
  - 剩余容量：对于每条边，剩余容量为$c(e) - f(e)$

剩余容量=容量−流量



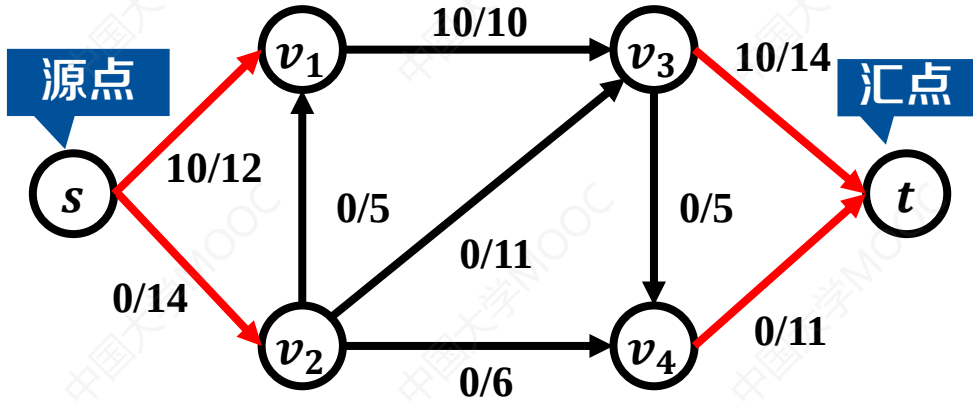
总流量：10

流网络

- 给定有向图$G =< V, E, C >$，其被称为流网络：
  - 容量：对于每条边$c(e) \geq 0$
  - 流量：对于每条边$c(e) \geq f(e) \geq 0$
  - 剩余容量：对于每条边，剩余容量为$c(e) - f(e)$
  - 总流量：

$$|f| = \sum_{e\ out\ of\ s} f(e) = \sum_{e\ in\ to\ t} f(e)$$

总流量=源点流出量=汇点流入量



总流量：10

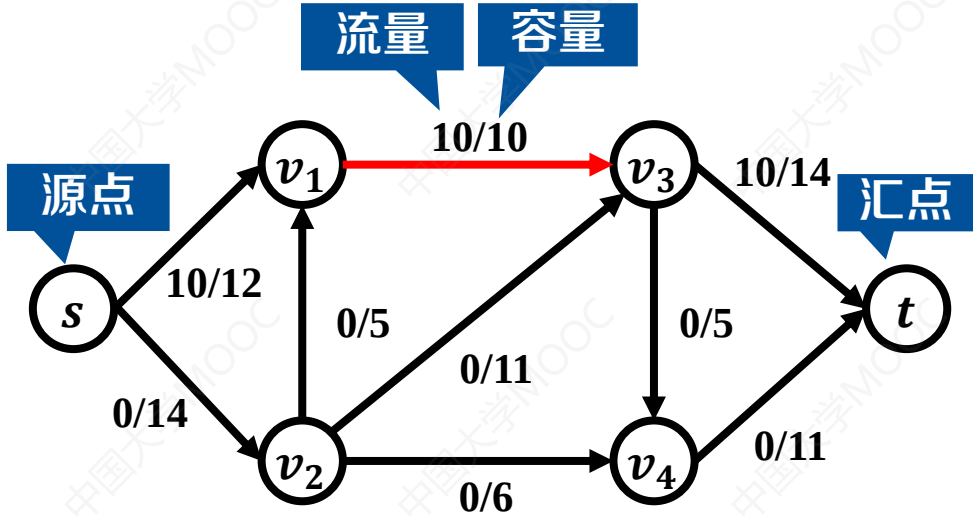
流网络

- 给定有向图$G =< V, E, C >$，其被称为流网络：
  - 容量：对于每条边$c(e) \geq 0$
  - 流量：对于每条边$c(e) \geq f(e) \geq 0$
  - 剩余容量：对于每条边，剩余容量为$c(e) - f(e)$
  - 总流量：

$$|f| = \sum_{e\ out\ of\ s} f(e) = \sum_{e\ in\ to\ t} f(e)$$

- 流量的两条性质
  - 容量限制：对边$e \in E$，有$0 \leq f(e) \leq c(e)$

边上的流量不应超过边的容量



流网络

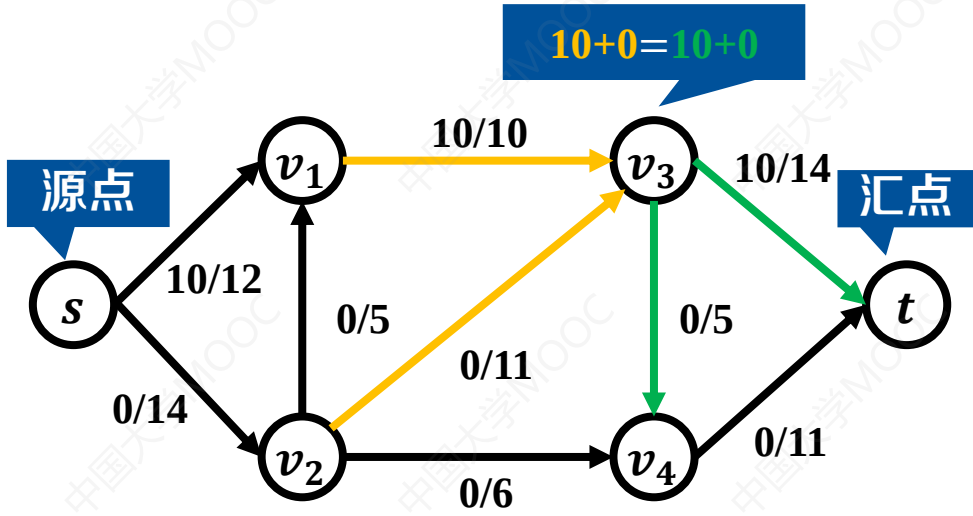
- 给定有向图$G =< V, E, C >$，其被称为流网络：
  - 容量：对于每条边$c(e) \geq 0$
  - 流量：对于每条边$c(e) \geq f(e) \geq 0$
  - 剩余容量：对于每条边，剩余容量为$c(e) - f(e)$
  - 总流量：

$$|f| = \sum_{e\ out\ of\ s} f(e) = \sum_{e\ in\ to\ t} f(e)$$

- 流量的两条性质
  - 容量限制：对边$e \in E$，有$0 \leq f(e) \leq c(e)$
  - 流量守恒：对顶点$v \in V - \{s, t\}$

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

进入某顶点v流量和
等于流出此顶点流量和



问题定义

最大流问题

Maximum Flow Problem

输入

- 有向图$G =< V, E, C >$，其中$c(e) \in C$表示边e的容量
- 源点s，汇点t

输出

- 总流量$|f|$

$$max\ |f| = max \sum_{e\ out\ of\ s} f(e)$$

优化目标

$$s.t.\ \ 0 \le f(e) \le c(e),\ \sum_{e\ in\ to\ v} f(e) = \sum_{e\ out\ of\ v} f(e)$$

约束条件

容量限制

流量守恒

提纲

问题背景

算法思想

算法实例

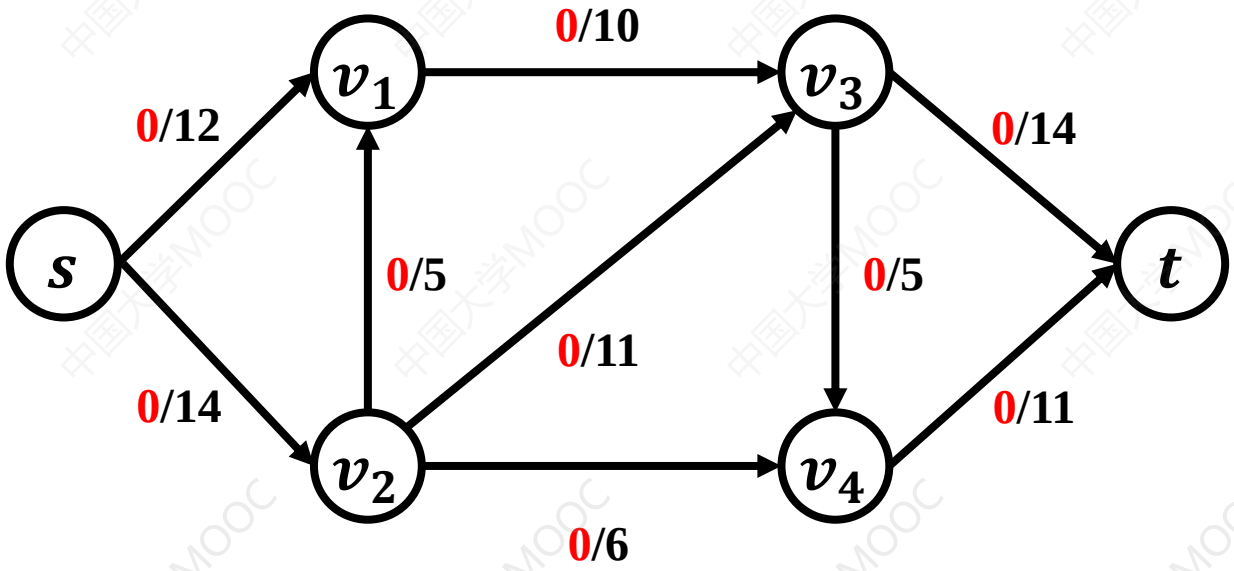
算法分析

算法性质

直观策略

- 算法思想
  - 对所有边$e \in E$，初始化流量为零$f(e) = 0$

流网络G

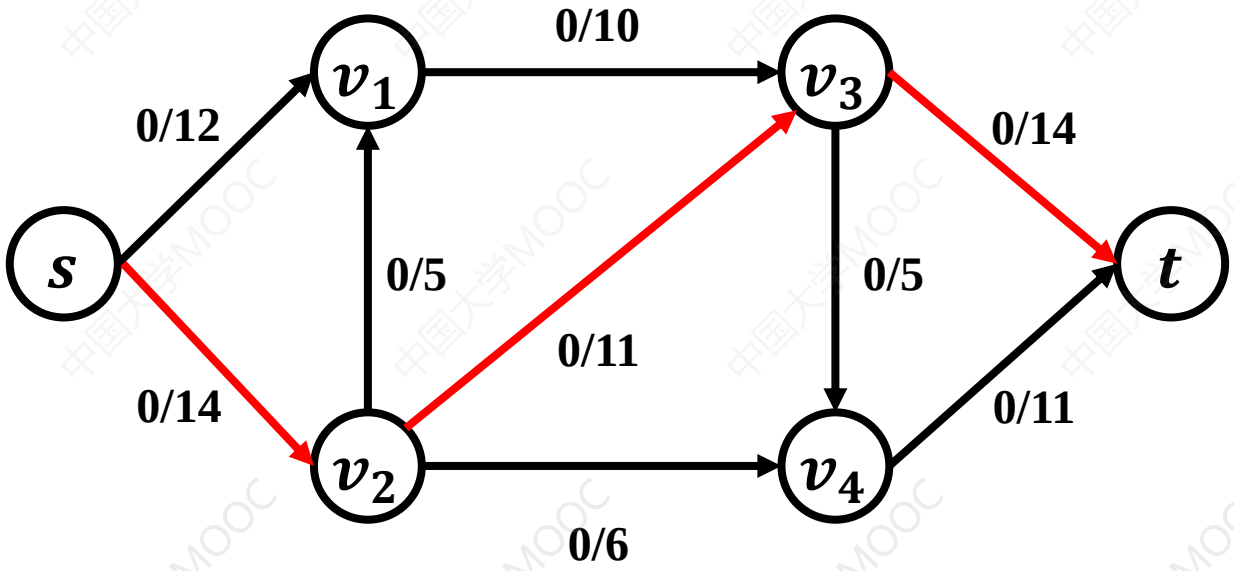


总流量：0

直观策略

- 算法思想
  - 对所有边$e \in E$，初始化流量为零$f(e) = 0$
  - 寻找一条s到t的路径P，此路径上的每条边e均满足$f(e) < c(e)$

流网络G

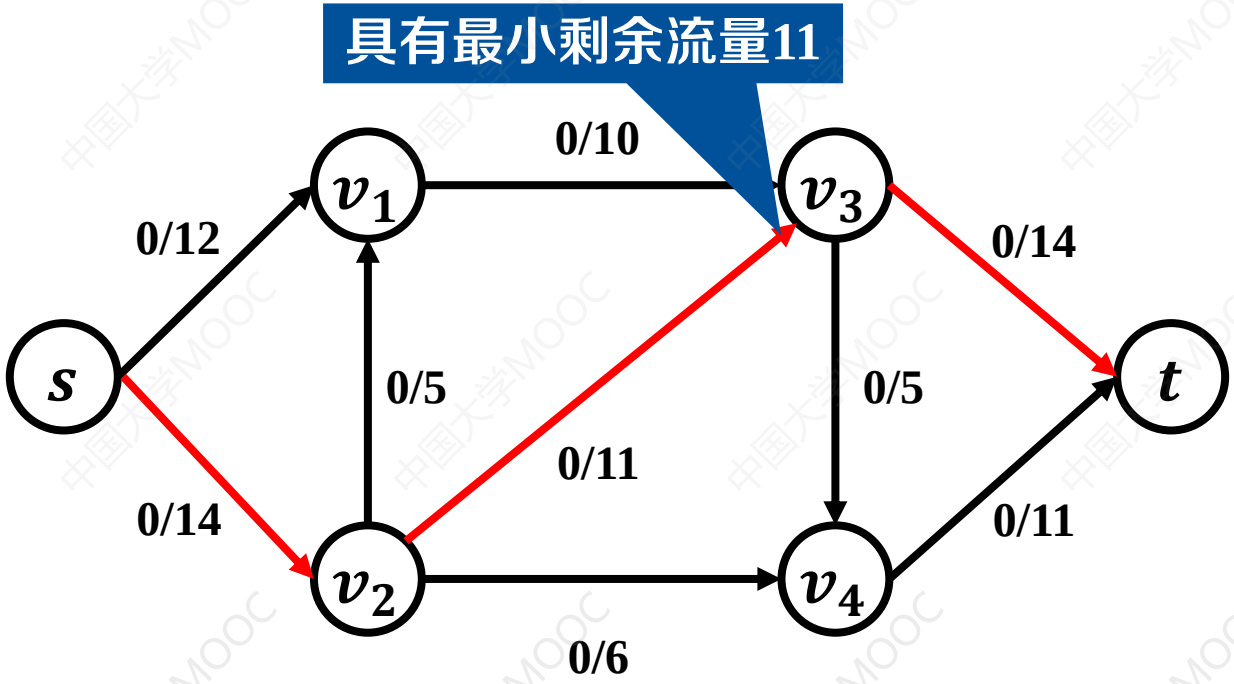


总流量：0

直观策略

- 算法思想
  - 对所有边$e \in E$，初始化流量为零$f(e) = 0$
  - 寻找一条s到t的路径P，此路径上的每条边e均满足$f(e) < c(e)$
  - 按路径P上最小剩余容量增加路径流量

流网络G

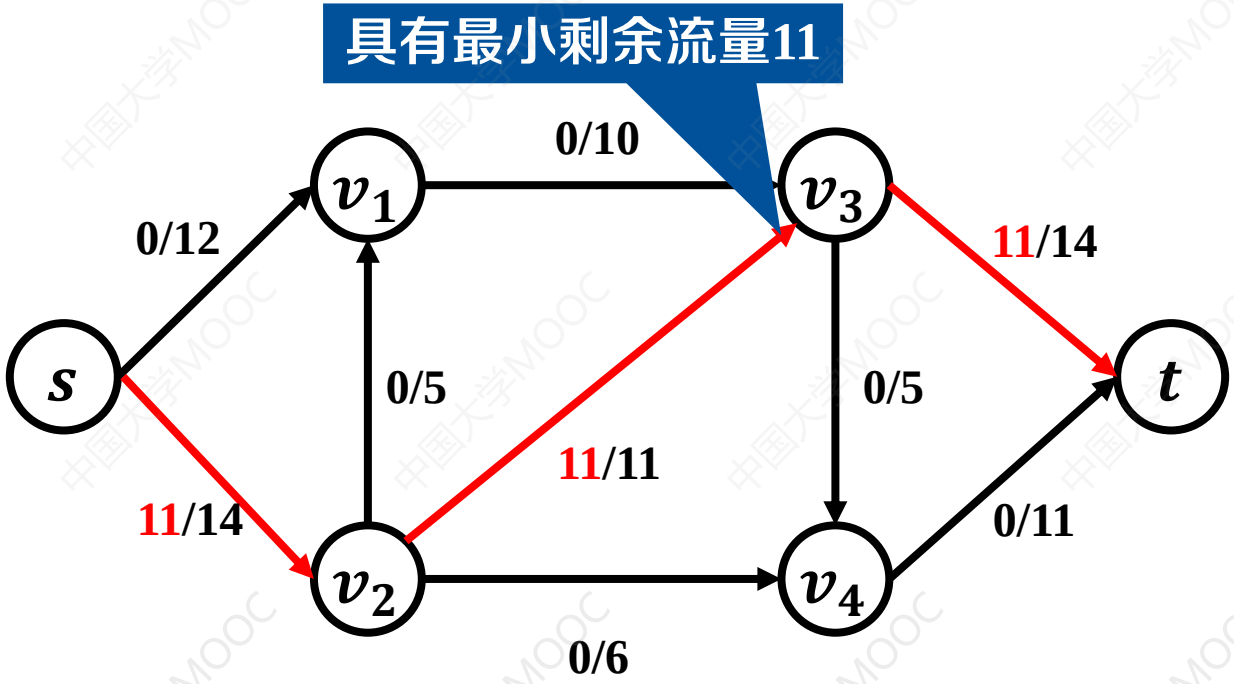


总流量：0

直观策略

- 算法思想
  - 对所有边$e \in E$，初始化流量为零$f(e) = 0$
  - 寻找一条s到t的路径P，此路径上的每条边e均满足$f(e) < c(e)$
  - 按路径P上最小剩余容量增加路径流量

流网络G

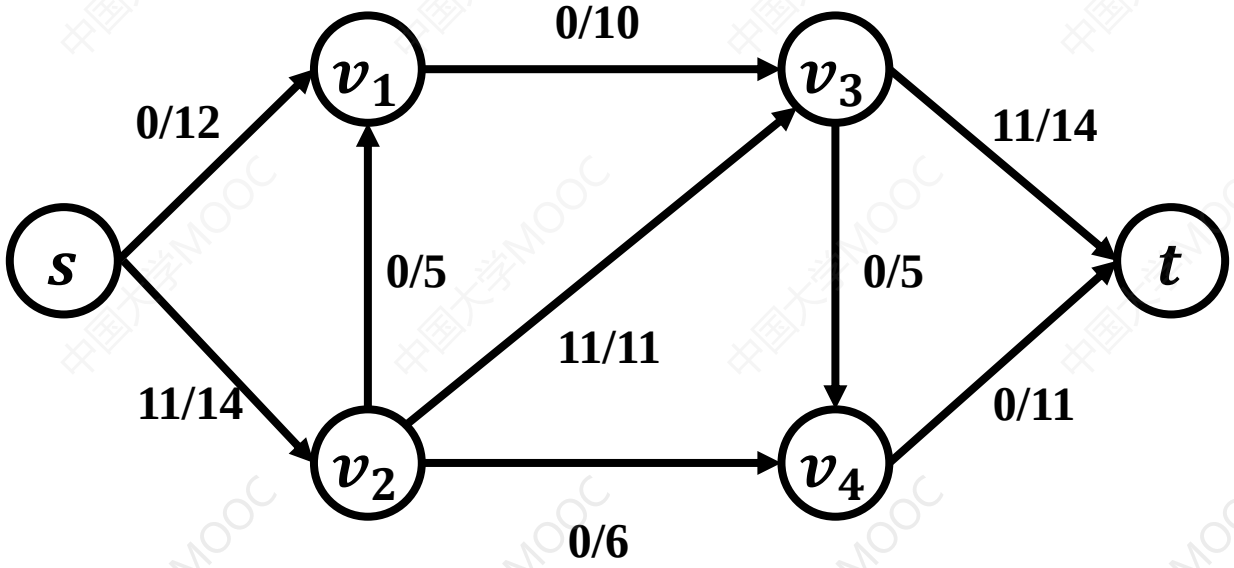


总流量：11

直观策略

- 算法思想
  - 对所有边$e \in E$，初始化流量为零$f(e) = 0$
  - 寻找一条s到t的路径P，此路径上的每条边e均满足$f(e) < c(e)$
  - 按路径P上最小剩余容量增加路径流量
  - 迭代寻找路径P直至无法增加路径流量

流网络G

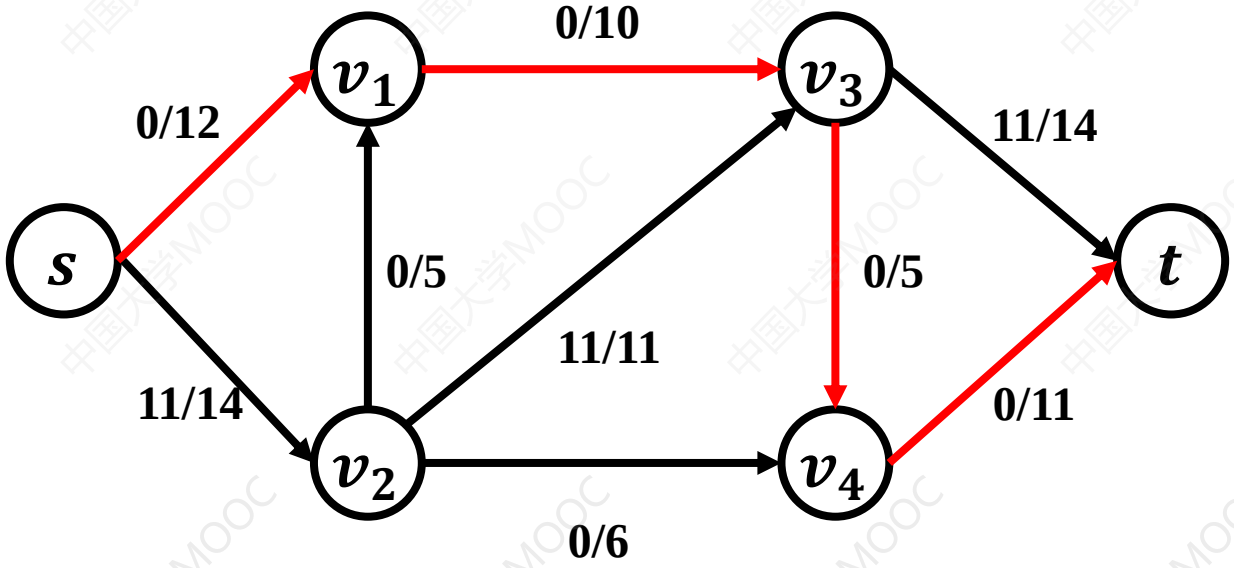


总流量：11

直观策略

- 算法思想
  - 对所有边$e \in E$，初始化流量为零$f(e) = 0$
  - 寻找一条s到t的路径P，此路径上的每条边e均满足$f(e) < c(e)$
  - 按路径P上最小剩余容量增加路径流量
  - 迭代寻找路径P直至无法增加路径流量

流网络G

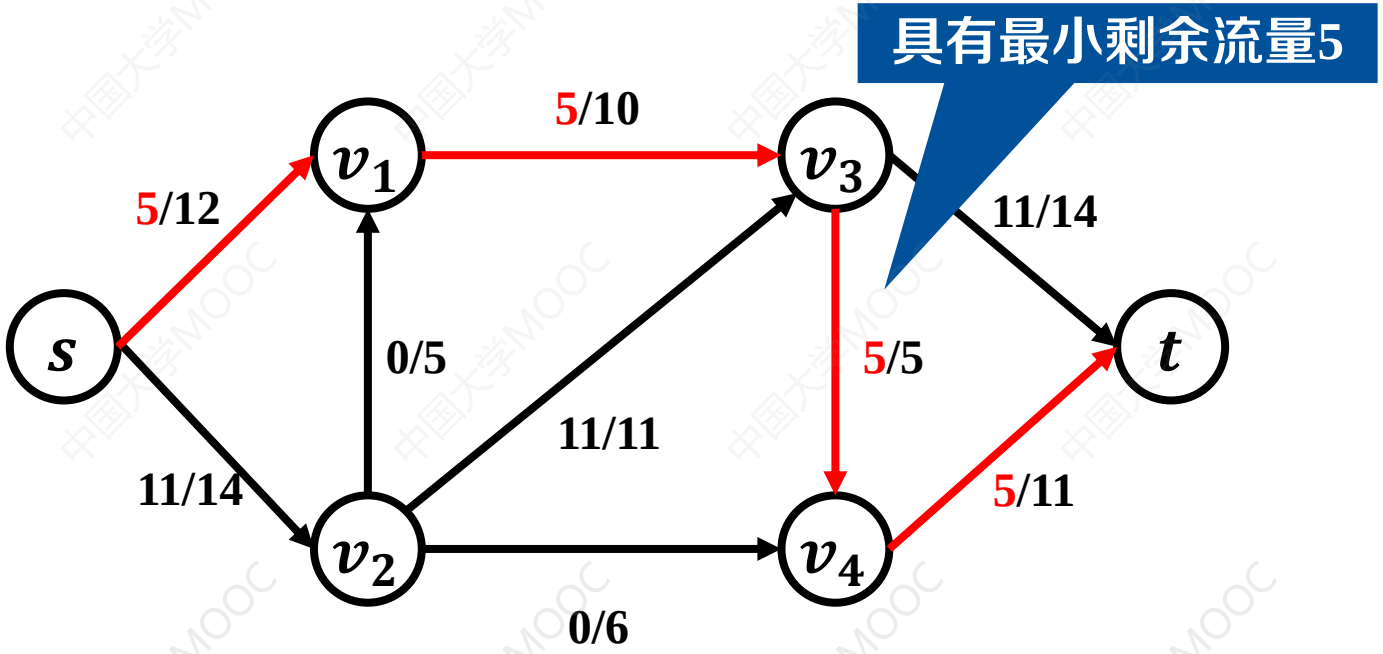


总流量：11

直观策略

- 算法思想
  - 对所有边$e \in E$，初始化流量为零$f(e) = 0$
  - 寻找一条s到t的路径P，此路径上的每条边e均满足$f(e) < c(e)$
  - 按路径P上最小剩余容量增加路径流量
  - 迭代寻找路径P直至无法增加路径流量

流网络G

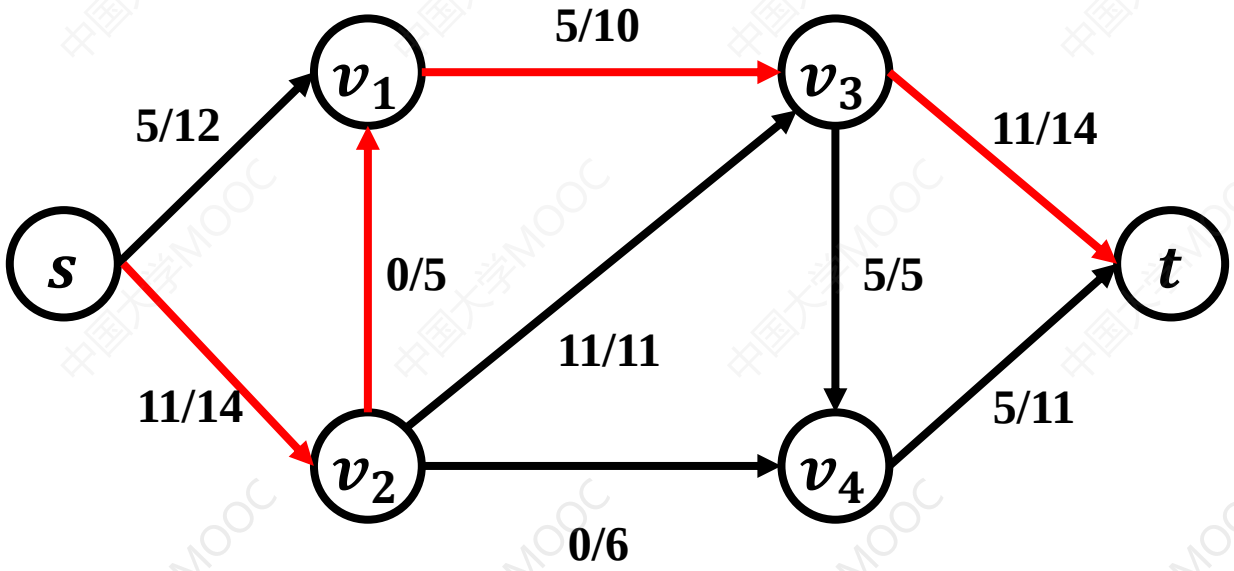


总流量：16

直观策略

- 算法思想
  - 对所有边$e \in E$，初始化流量为零$f(e) = 0$
  - 寻找一条s到t的路径P，此路径上的每条边e均满足$f(e) < c(e)$
  - 按路径P上最小剩余容量增加路径流量
  - 迭代寻找路径P直至无法增加路径流量

流网络G

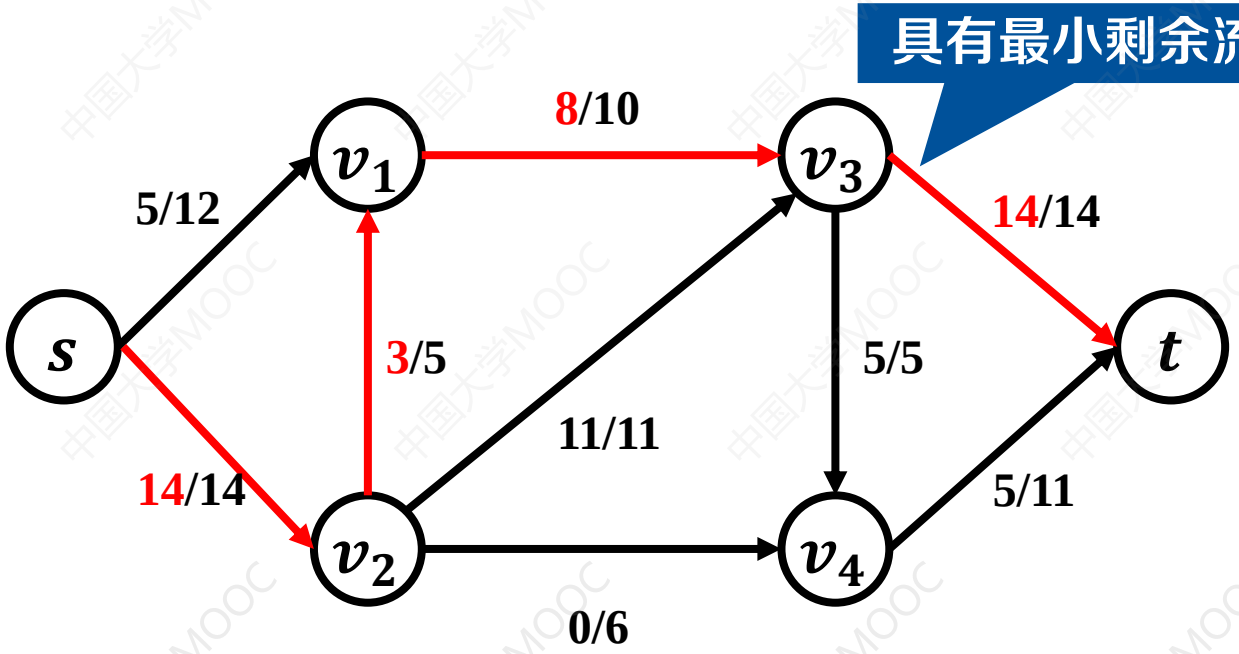


总流量：16

直观策略

- 算法思想
  - 对所有边$e \in E$，初始化流量为零$f(e) = 0$
  - 寻找一条s到t的路径P，此路径上的每条边e均满足$f(e) < c(e)$
  - 按路径P上最小剩余容量增加路径流量
  - 迭代寻找路径P直至无法增加路径流量

流网络G

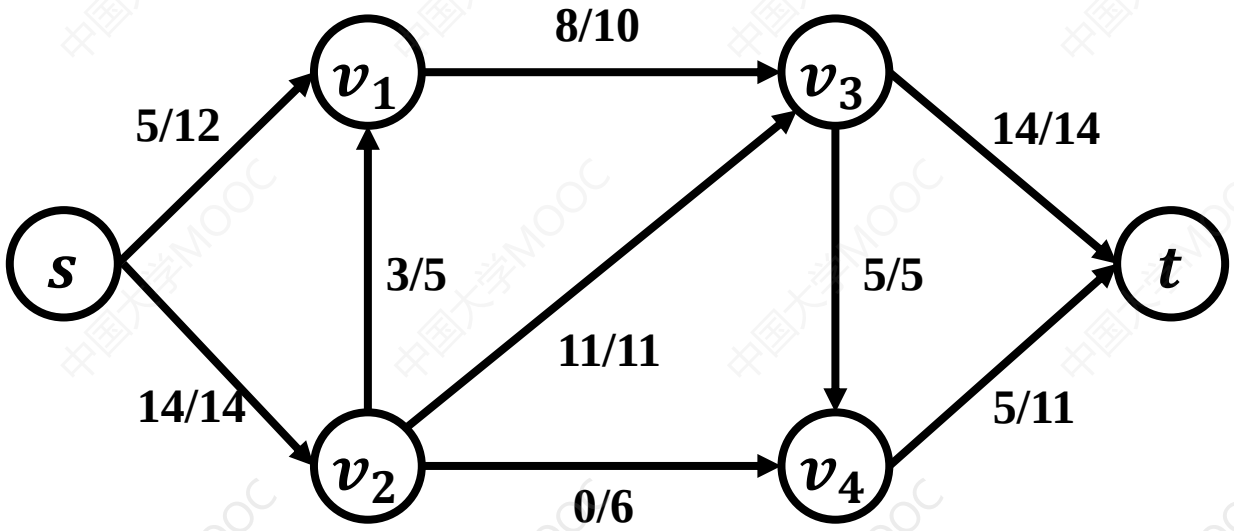


总流量：19

直观策略

- 算法思想
  - 对所有边$e \in E$，初始化流量为零$f(e) = 0$
  - 寻找一条s到t的路径P，此路径上的每条边e均满足$f(e) < c(e)$
  - 按路径P上最小剩余容量增加路径流量
  - 迭代寻找路径P直至无法增加路径流量

流网络G

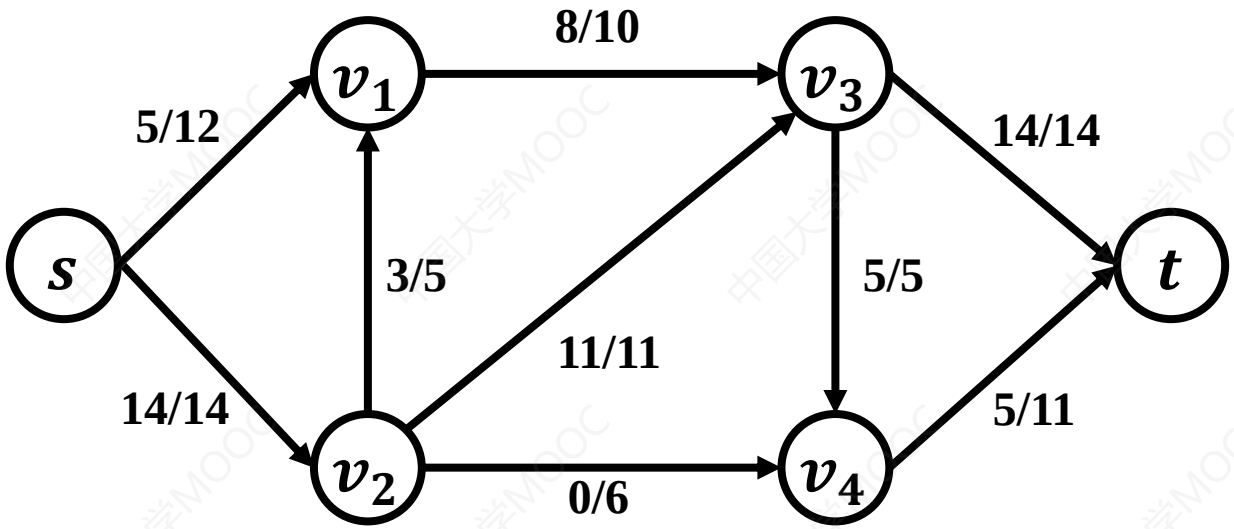


总流量：19

无法寻找到可增加流量的路径

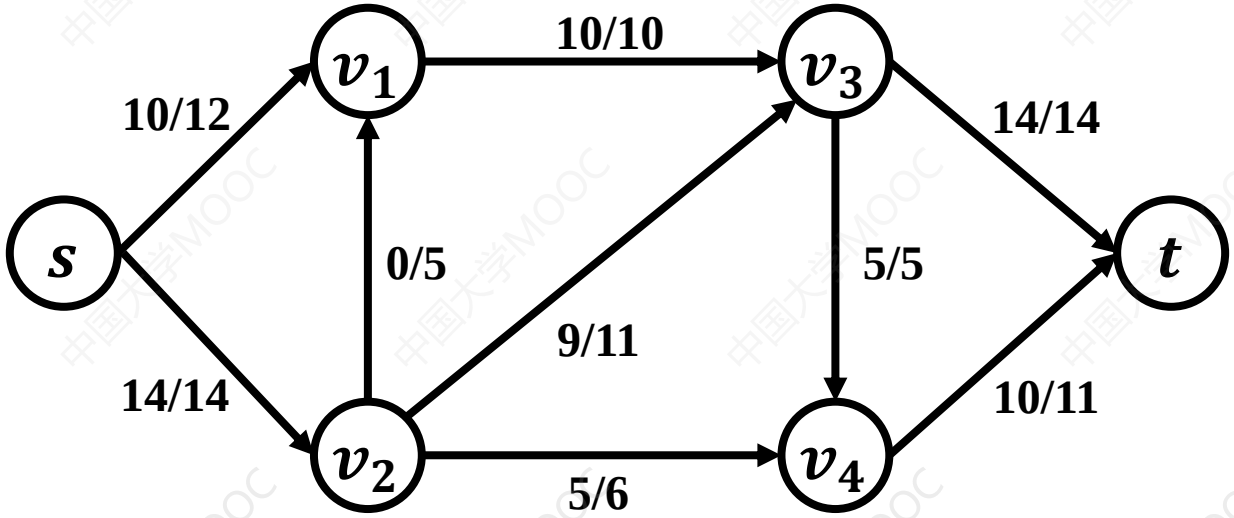
直观策略：比较观察

- 直观策略



总流量：19

- 最优方案



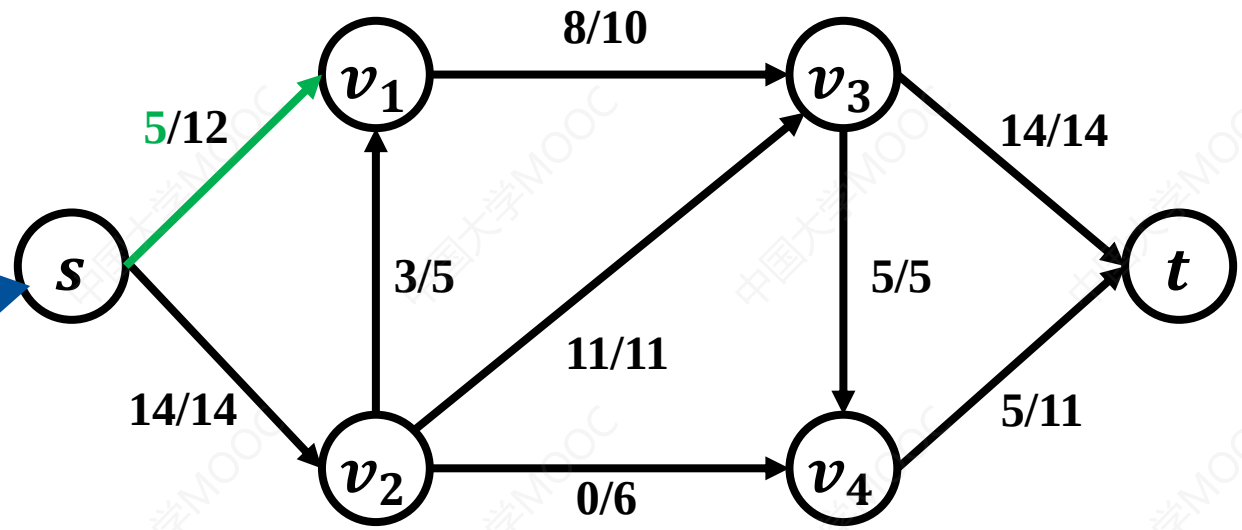
最大总流量：24

直观策略可能达不到最优方案，观察解的区别

直观策略：比较观察

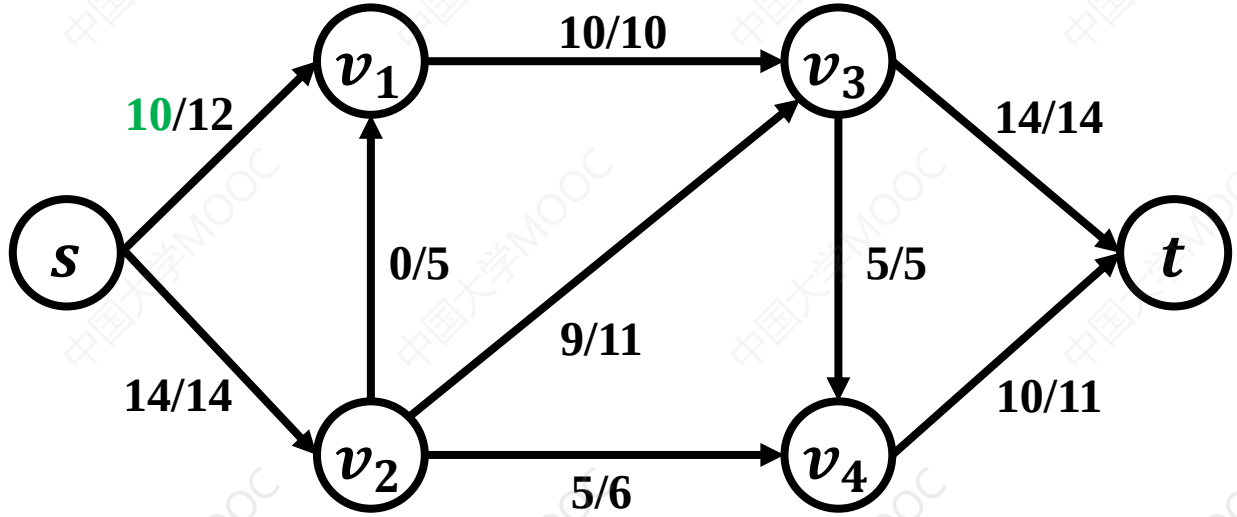
- 直观策略

$$|f| = \sum_{e \text{ out of } s} f(e)$$



总流量：19

- 最优方案

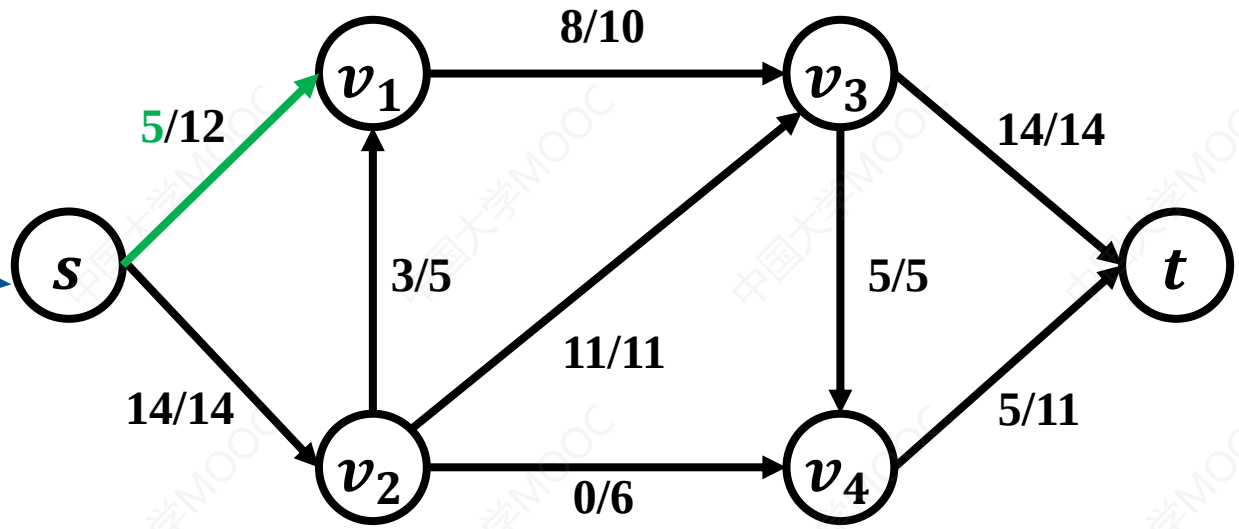


最大总流量：24

直观策略：比较观察

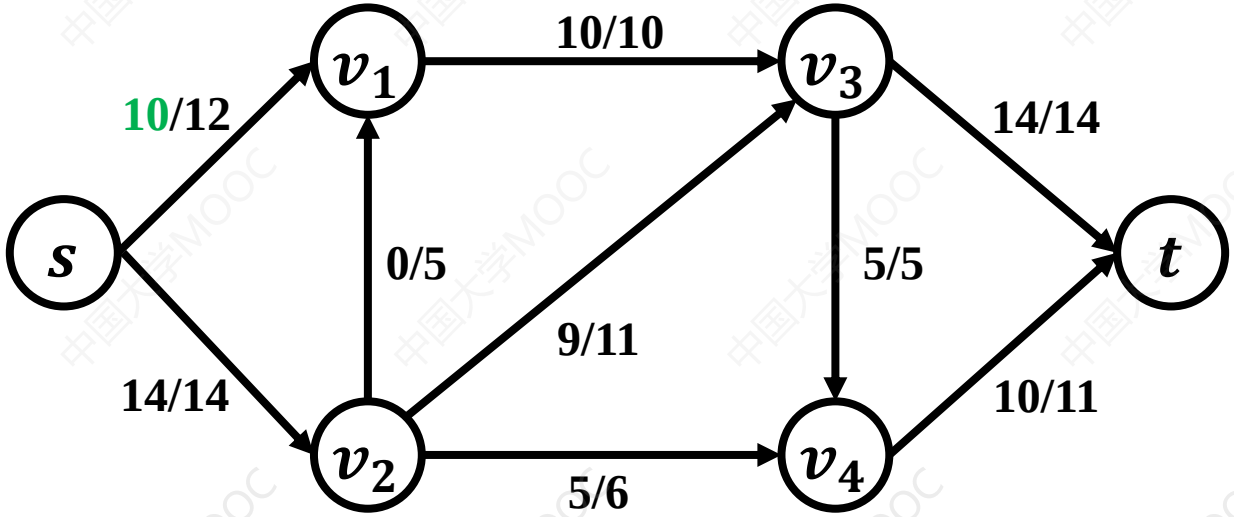
- 直观策略

容量未充分利用



总流量：19

- 最优方案

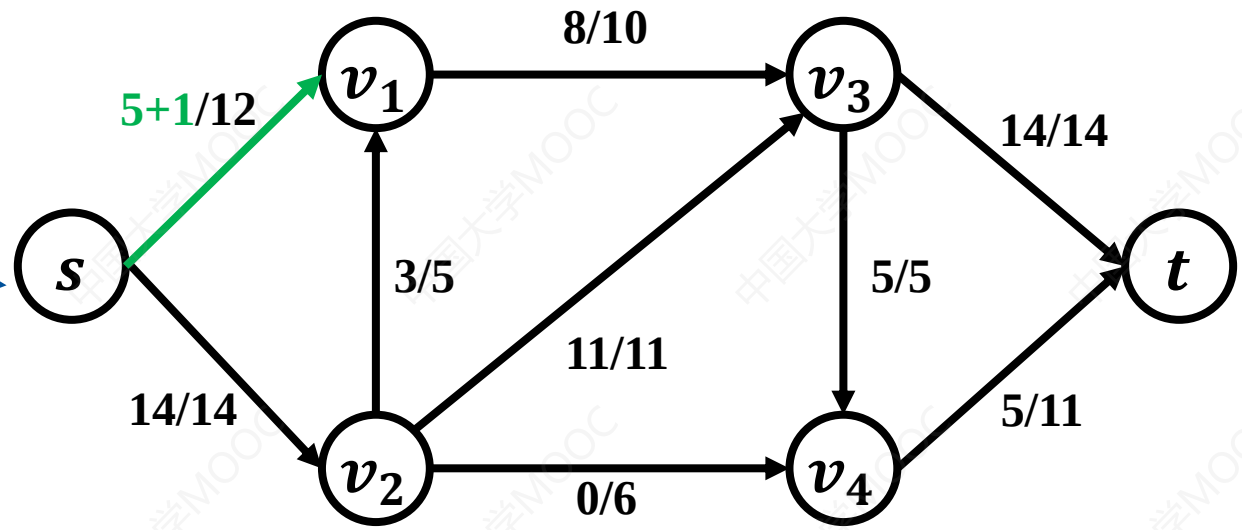


最大总流量：24

直观策略：比较观察

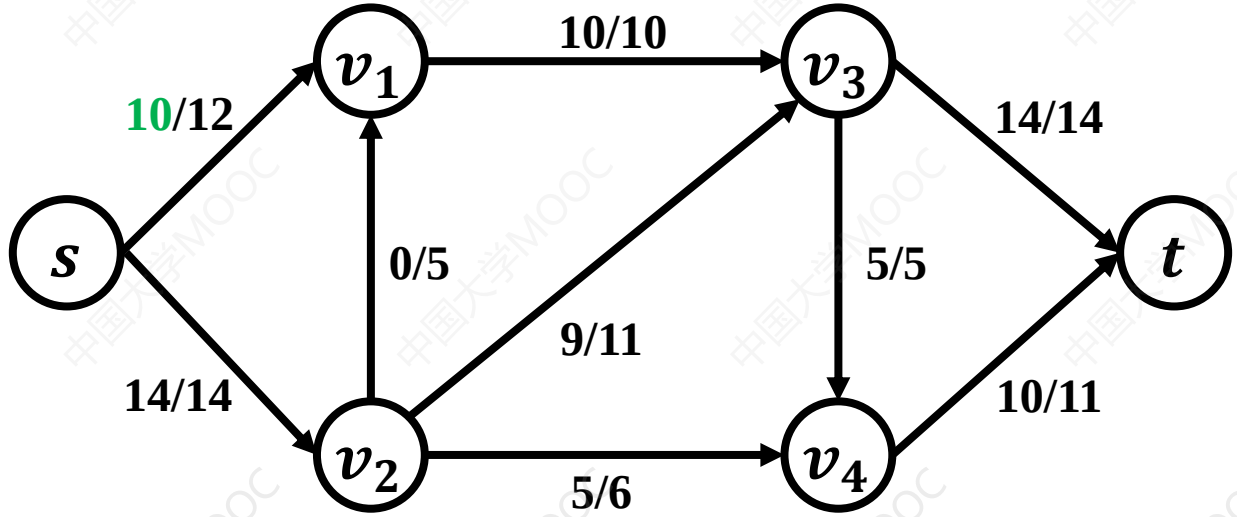
- 直观策略

尝试扩充
一个单位流量



总流量：19

- 最优方案

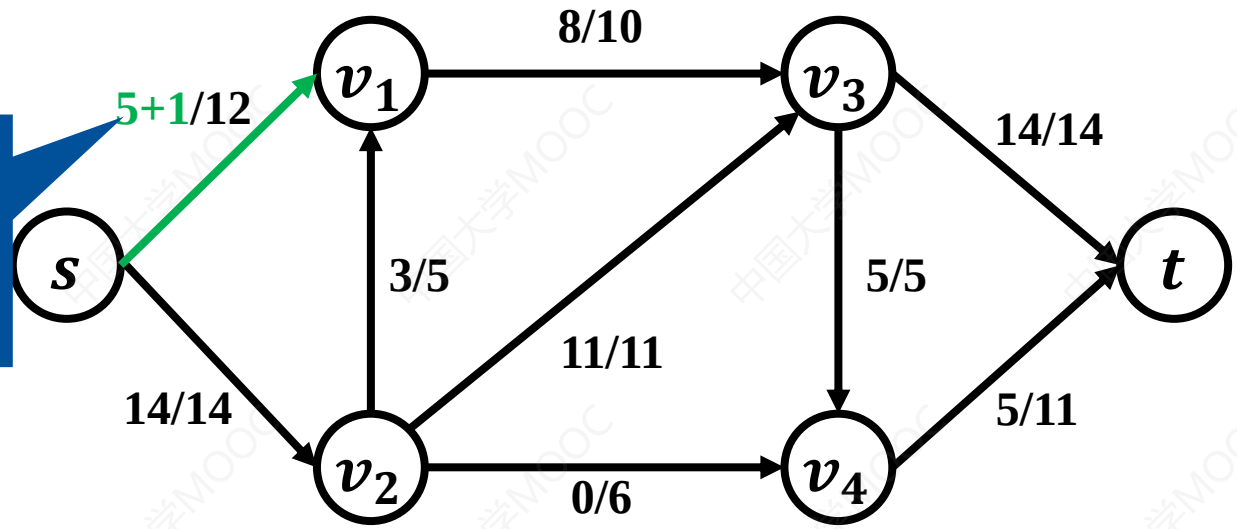


最大总流量：24

直观策略：比较观察

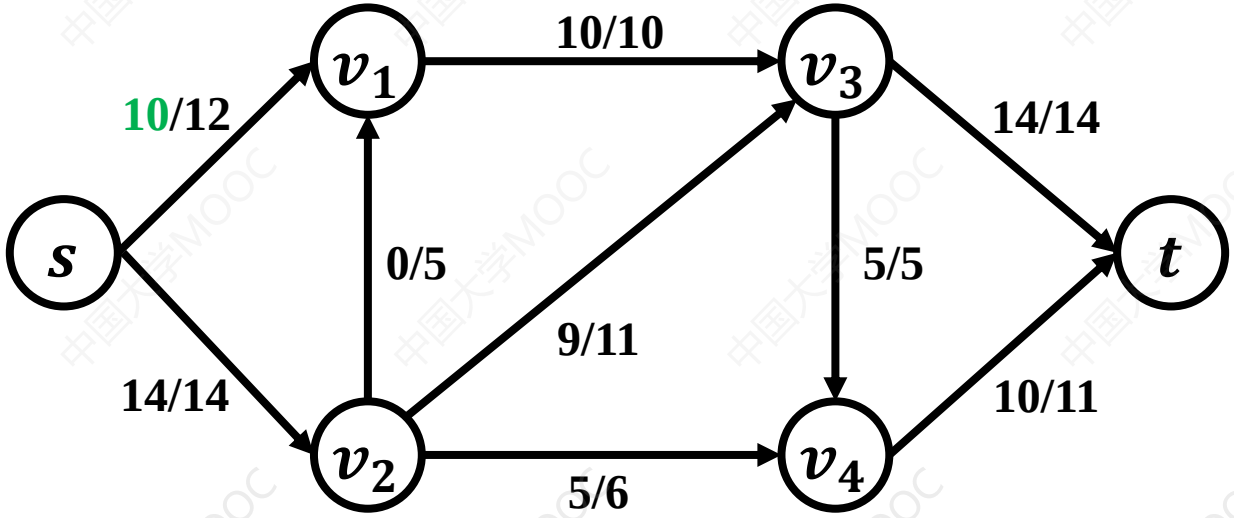
- 直观策略

为满足 v_1 流量守恒，扩充从其它边流出的流量



总流量：19

- 最优方案

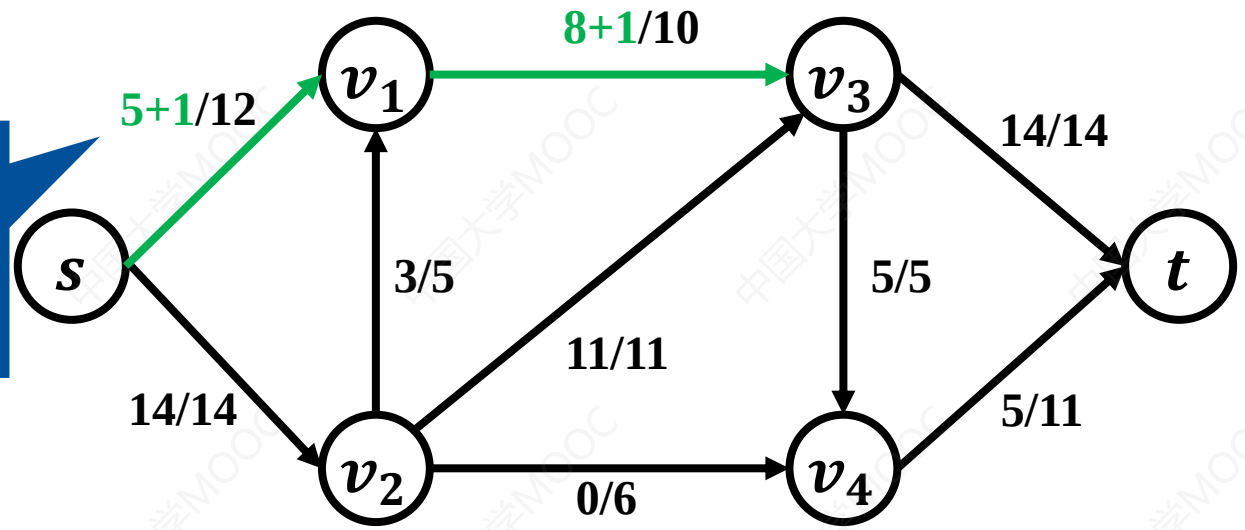


最大总流量：24

直观策略：比较观察

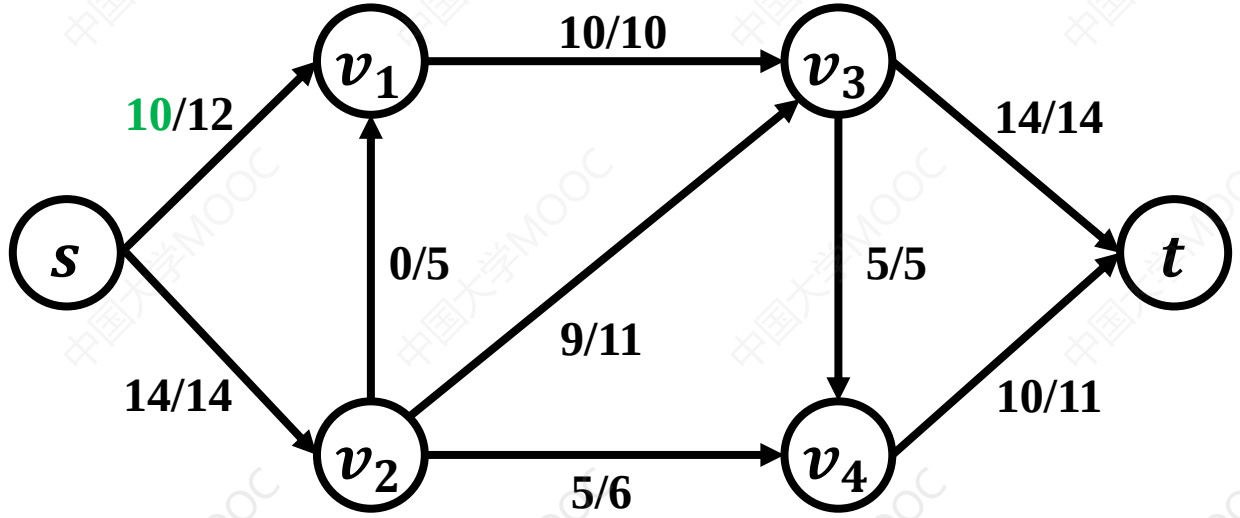
- 直观策略

重新分配经过v_1的流量



总流量：19

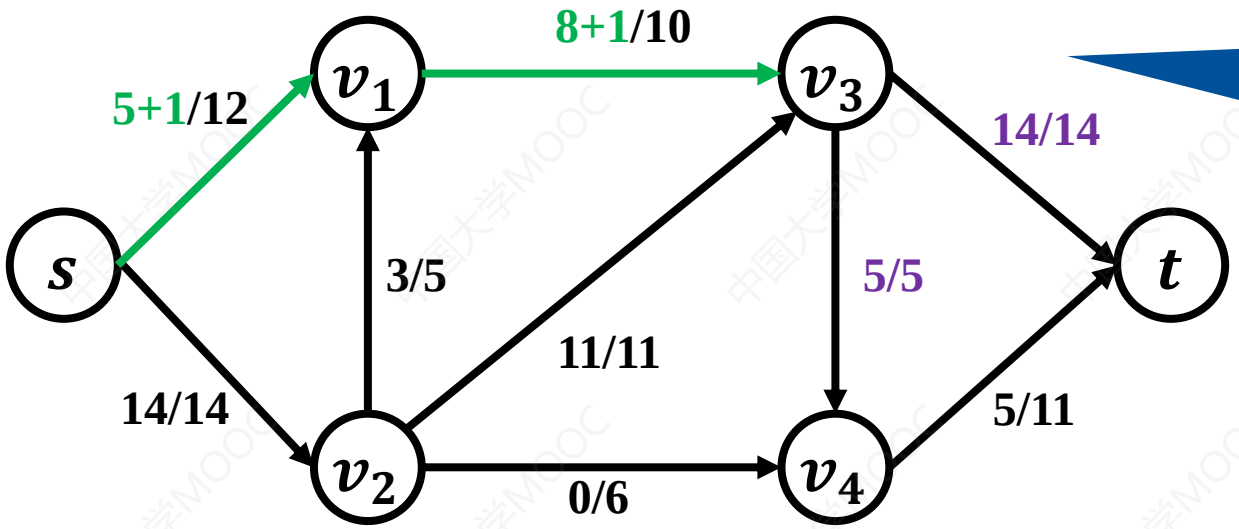
- 最优方案



最大总流量：24

直观策略：比较观察

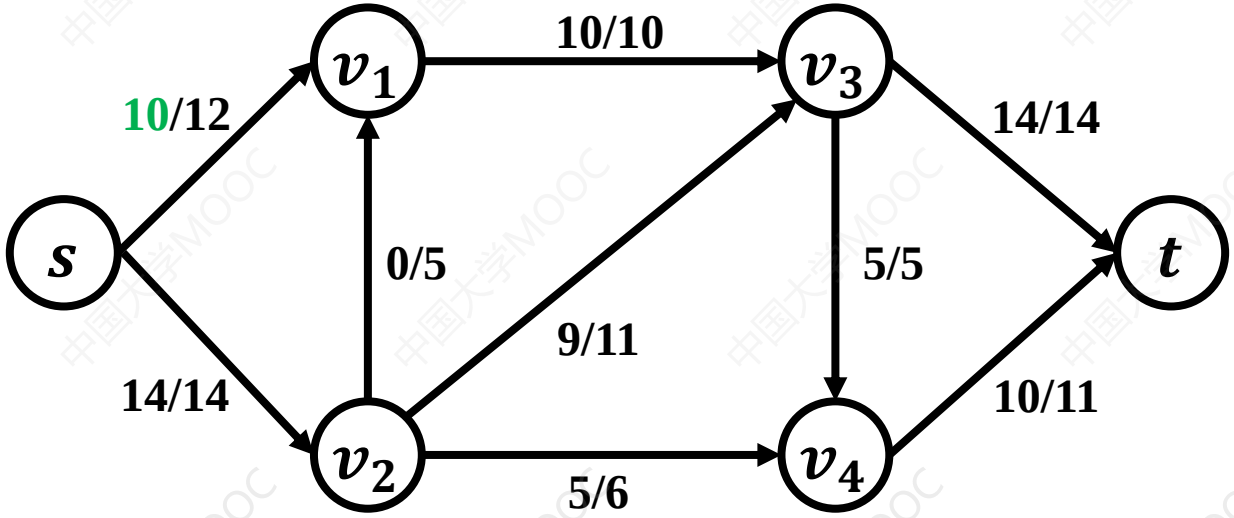
- 直观策略



无法扩充v_3从其它边流出的流量

总流量：19

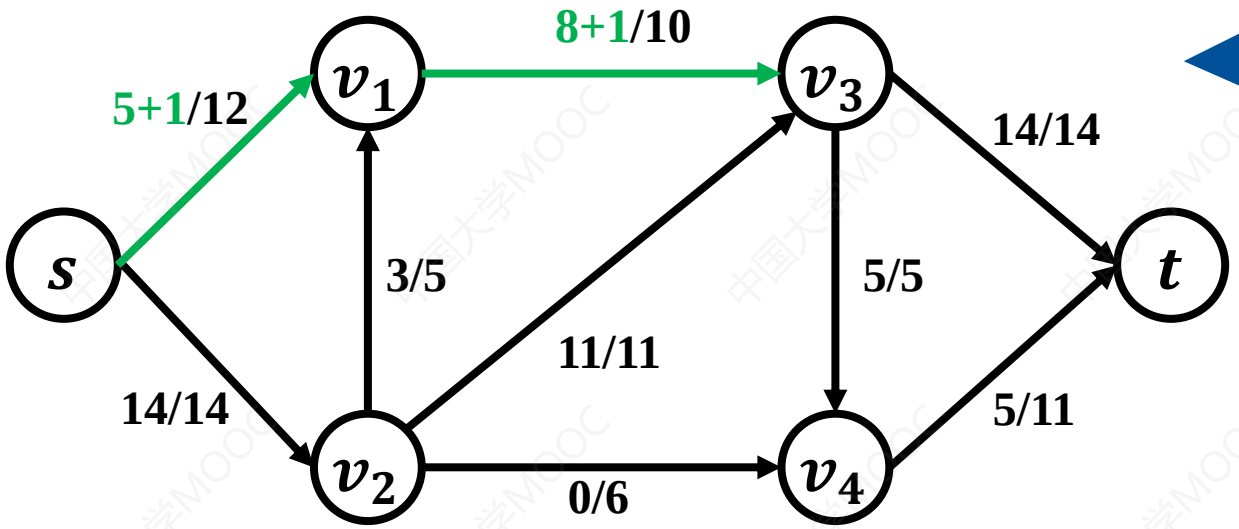
- 最优方案



最大总流量：24

直观策略：比较观察

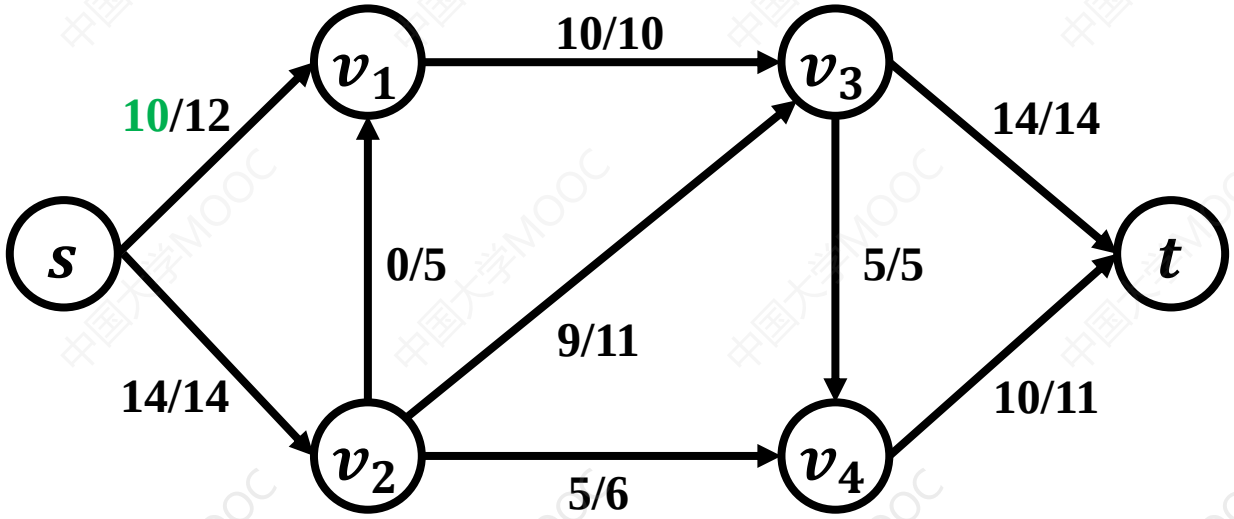
- 直观策略



为满足 v_3 流量守恒，缩减其它边的进入流量

总流量：19

- 最优方案

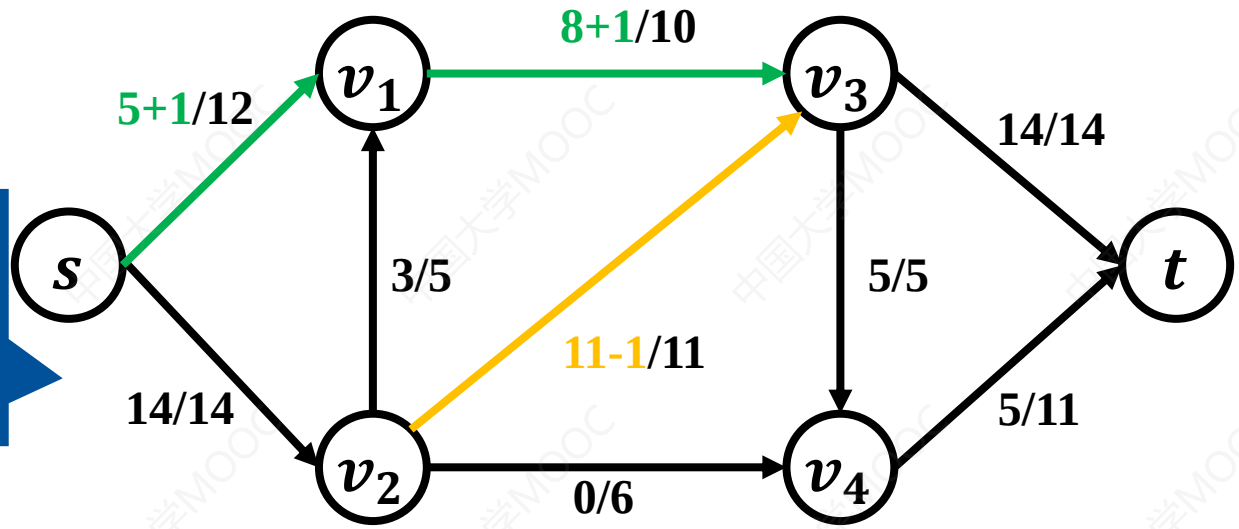


最大总流量：24

直观策略：比较观察

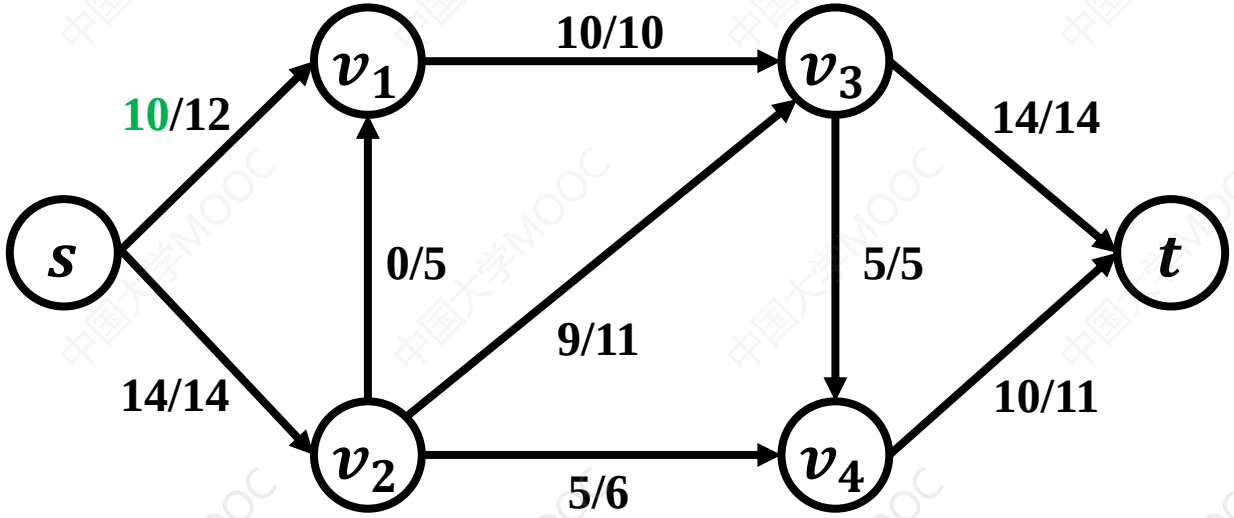
- 直观策略

重新分配经过v_3的流量



总流量：19

- 最优方案

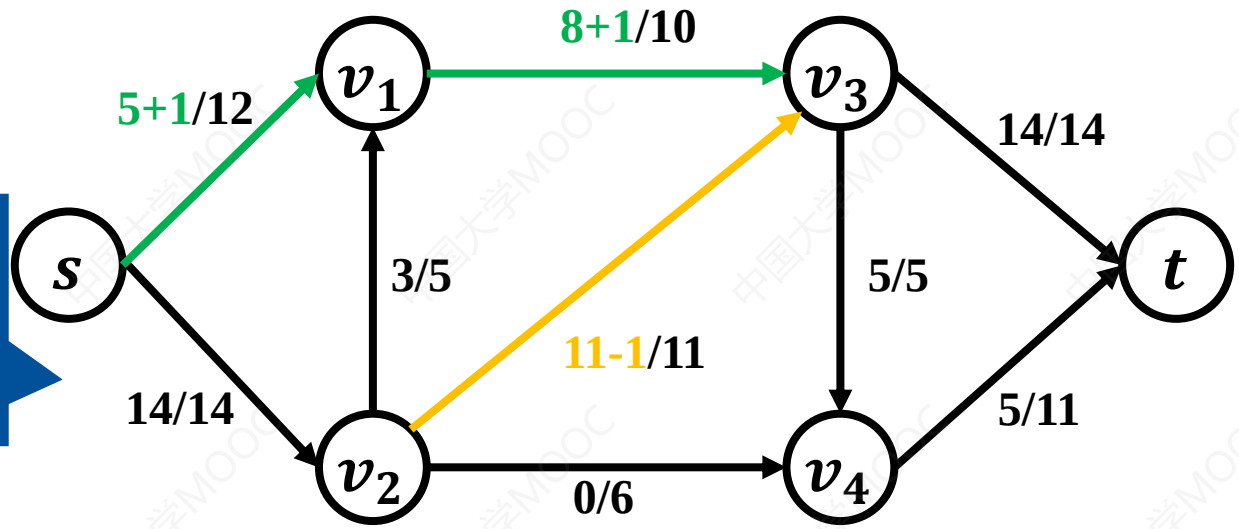


最大总流量：24

直观策略：比较观察

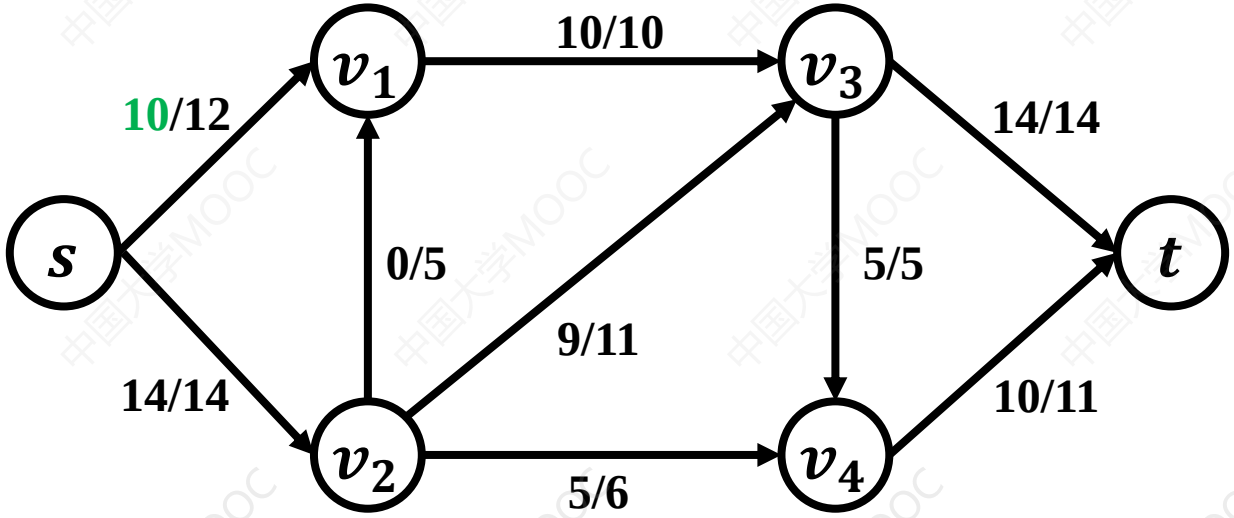
- 直观策略

为满足 v_2 流量守恒，扩充从其它边流出的流量



总流量：19

- 最优方案

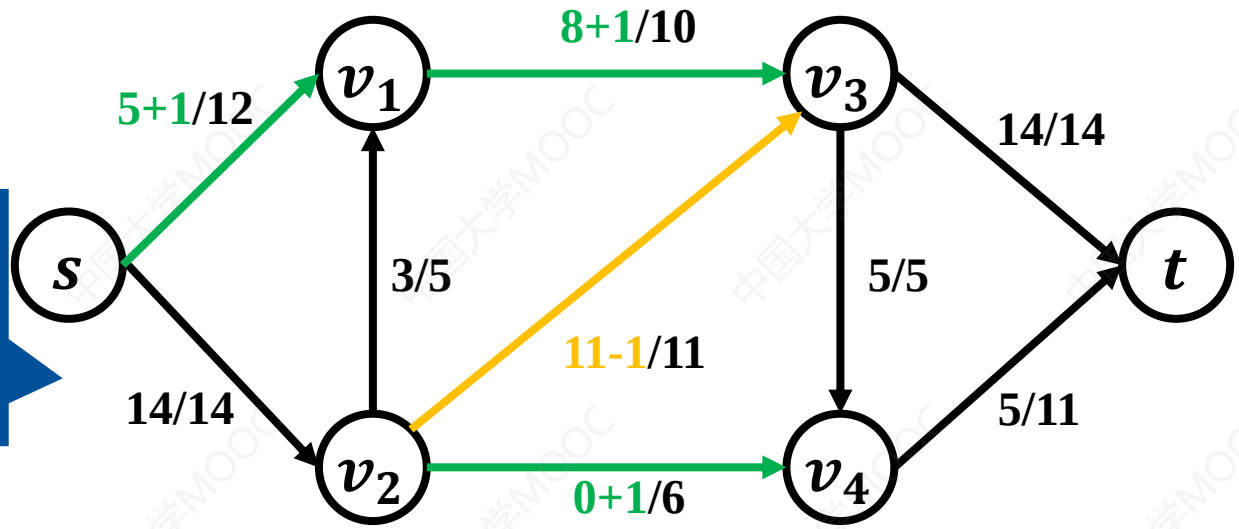


最大总流量：24

直观策略：比较观察

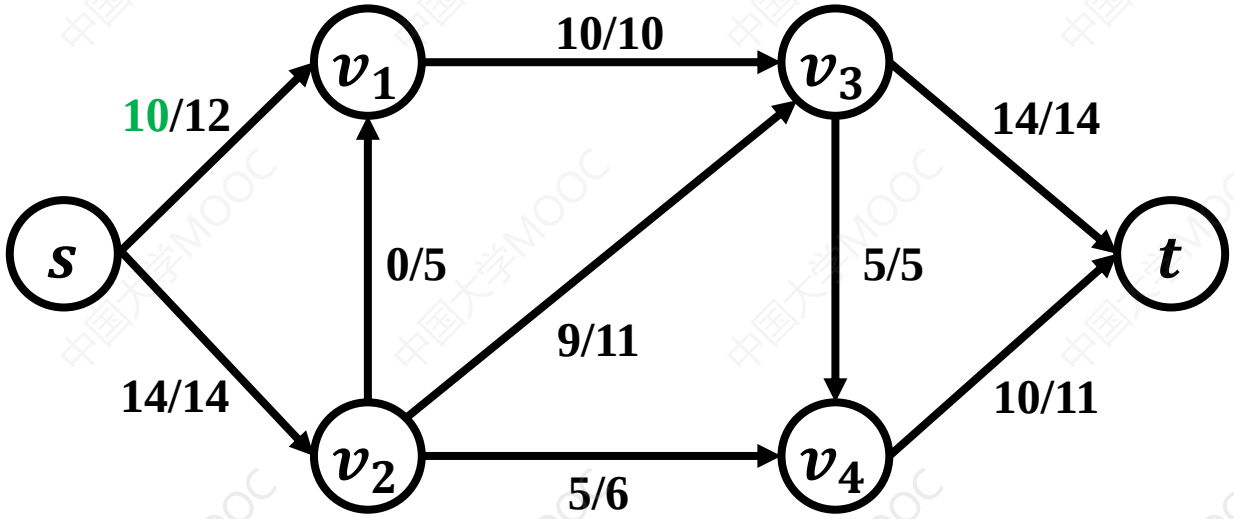
- 直观策略

重新分配经过v_2的流量



总流量：19

- 最优方案

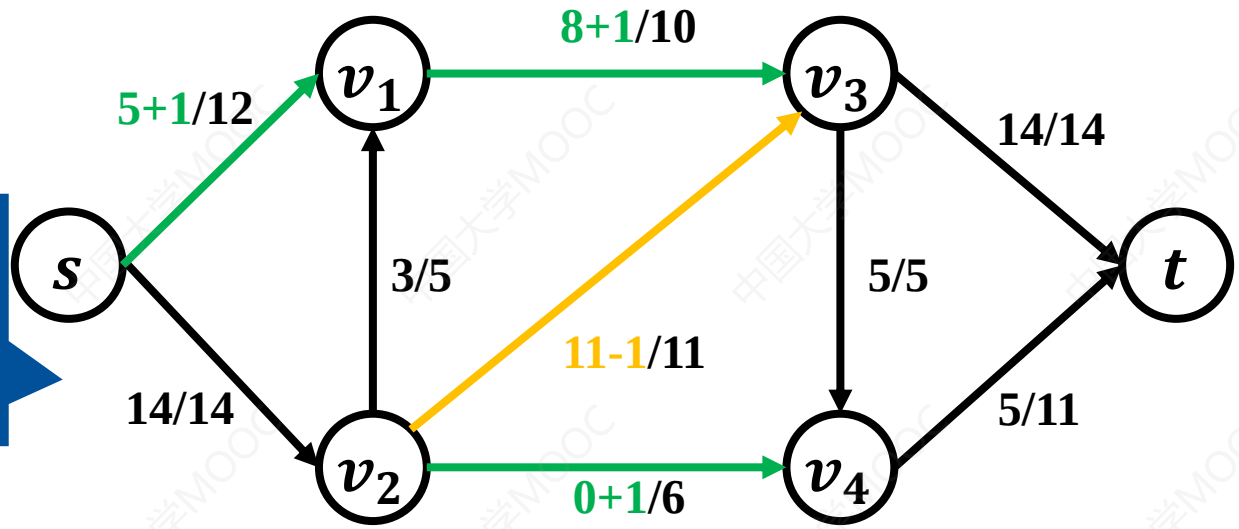


最大总流量：24

直观策略：比较观察

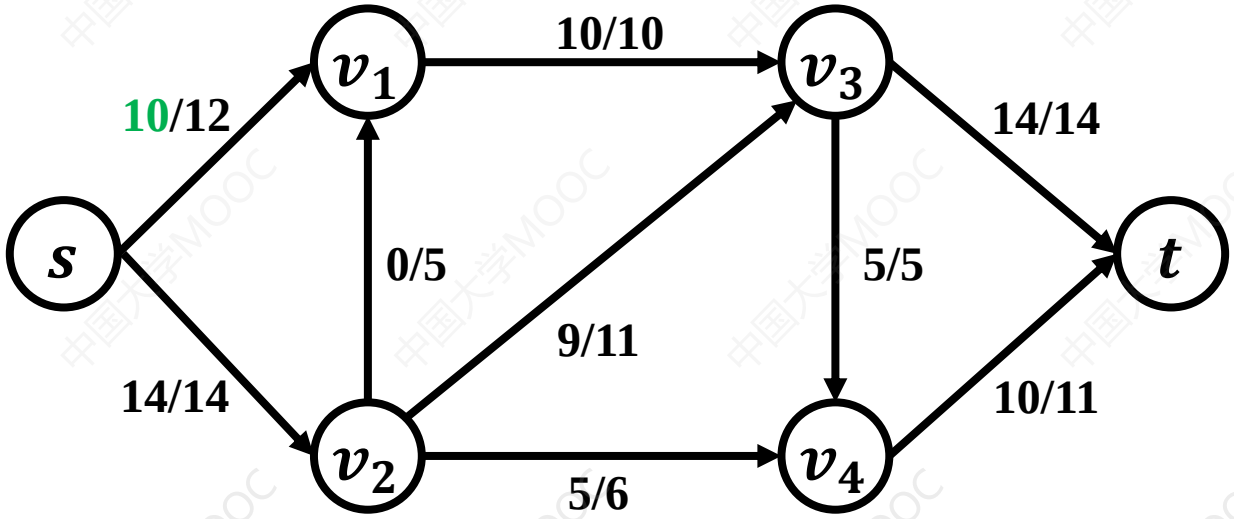
- 直观策略

为满足 v_4 流量守恒，扩充从其它边流出的流量



总流量：19

- 最优方案

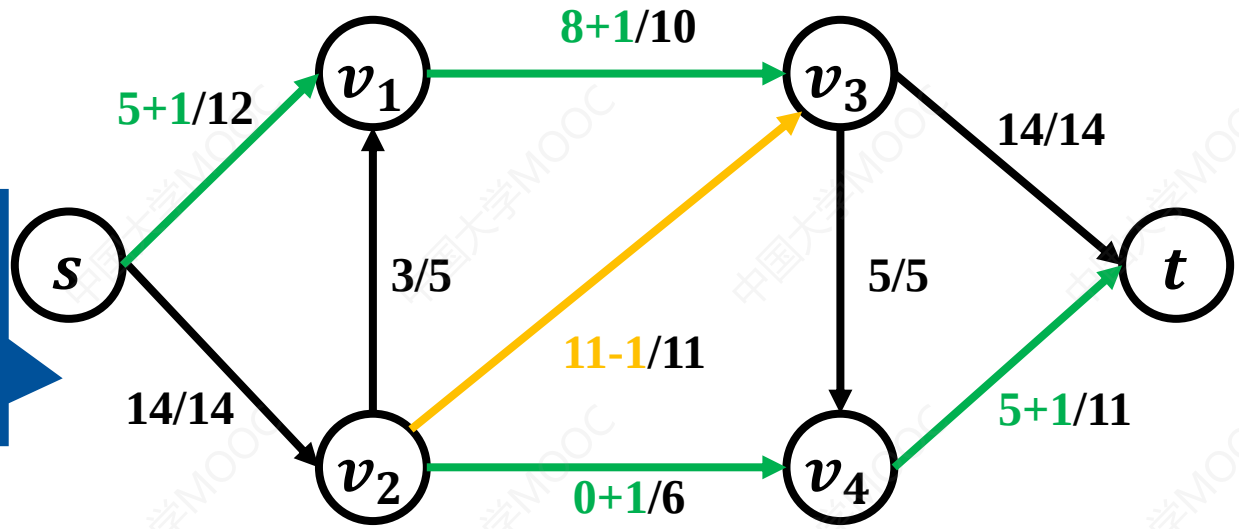


最大总流量：24

直观策略：比较观察

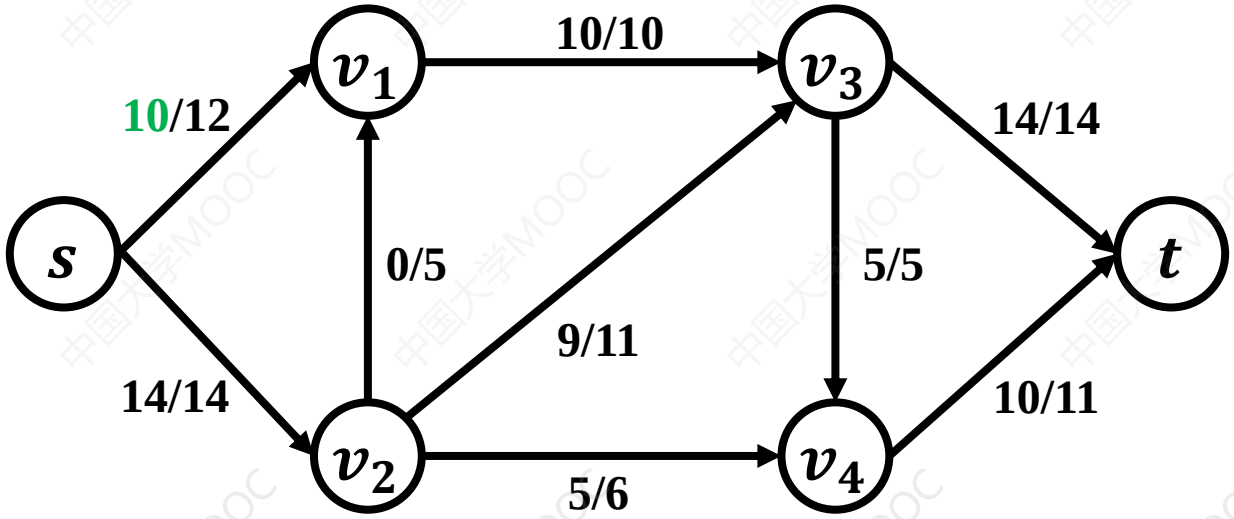
- 直观策略

重新分配经过v_4的流量



总流量：19+1=20

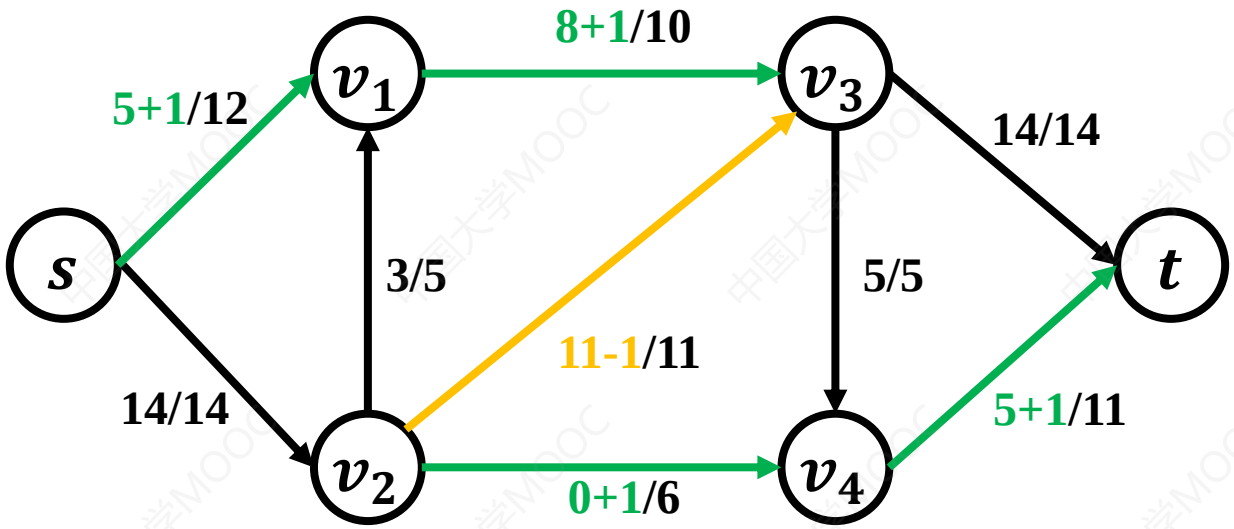
- 最优方案



最大总流量：24

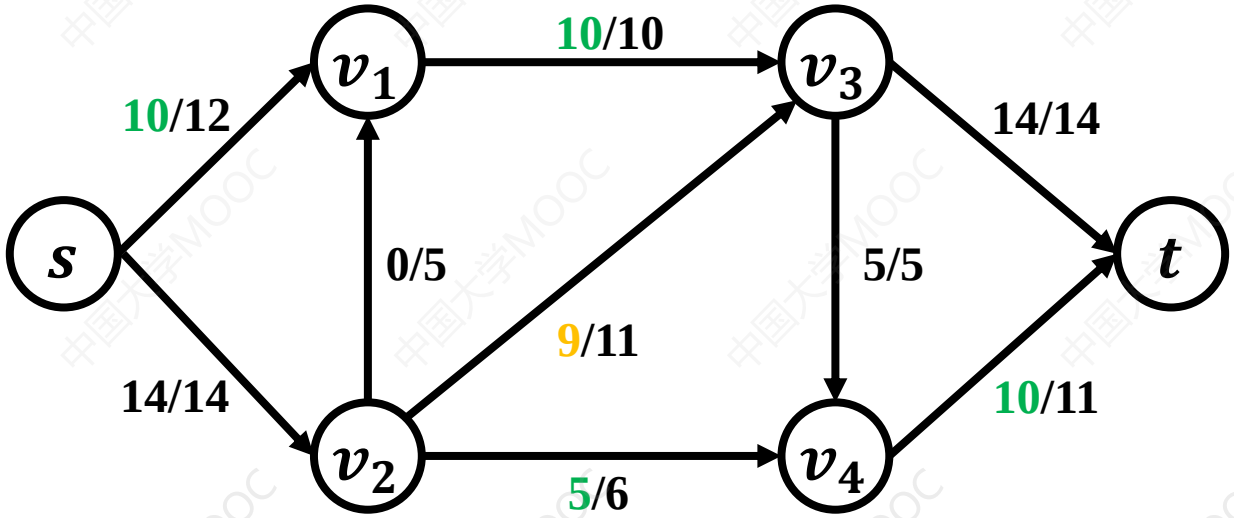
直观策略：比较观察

- 直观策略



总流量：20

- 最优方案



最大总流量：24

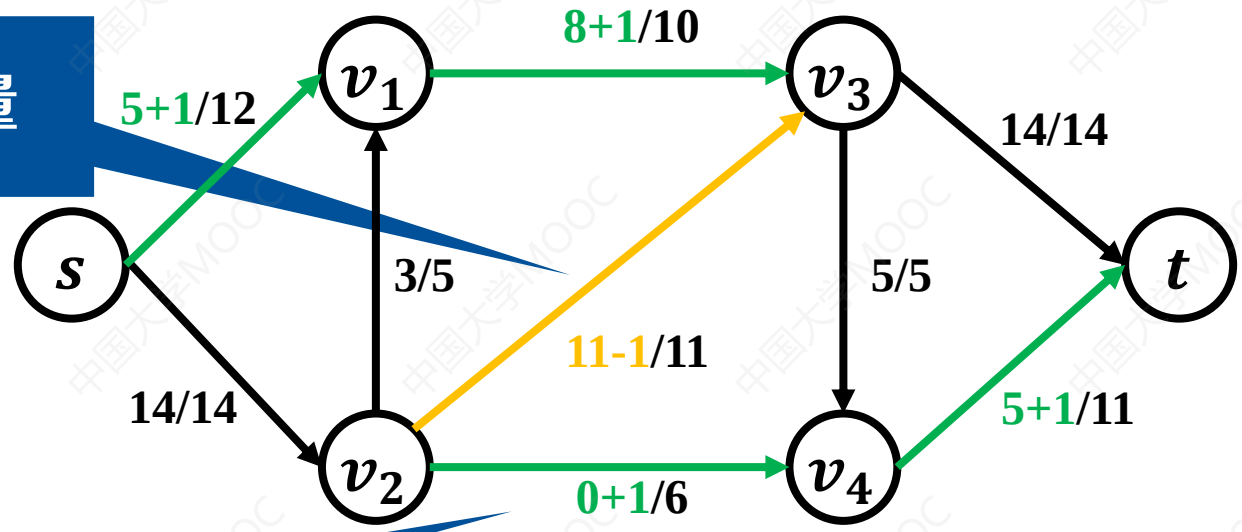
如果允许缩减边上的容量，可进一步增大总流量

直观策略：比较观察

- 调整流量的两种方式

缩减边上的流量

扩充边上的流量



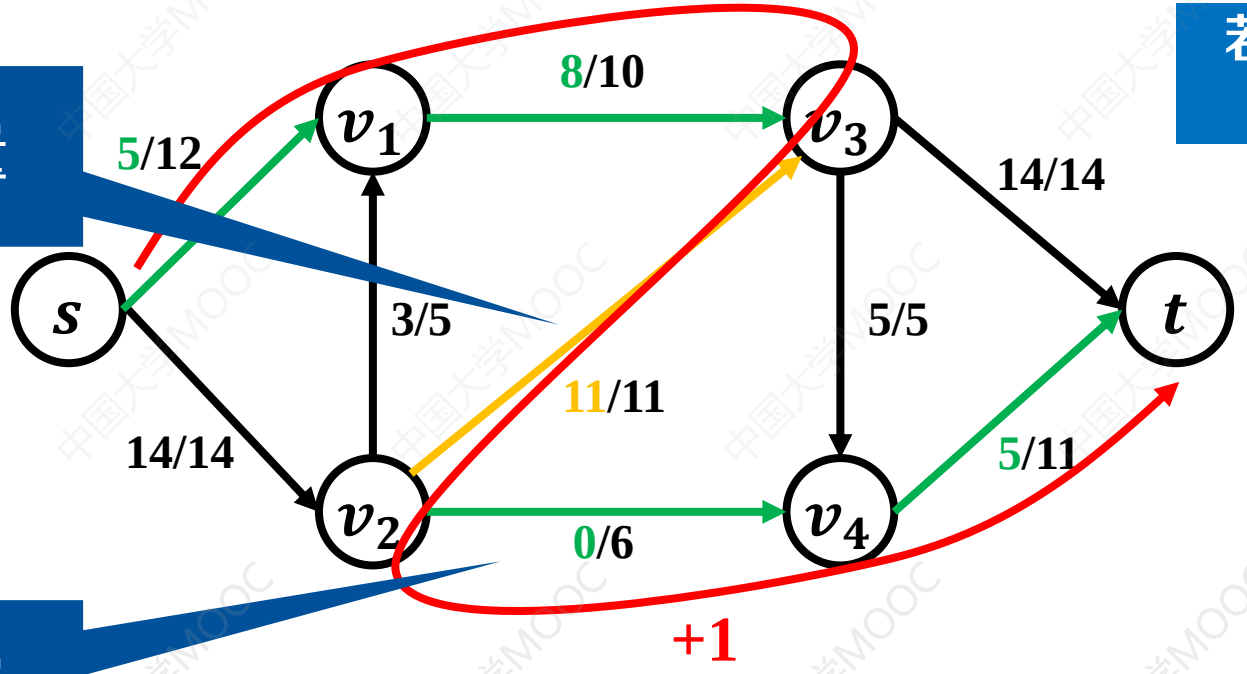
总流量：20

问题：如何改进直观策略？

直观策略：调整

- 算法思想
  - 对所有边$e \in E$，初始化流量为零$f(e) = 0$
  - 寻找一条s到t的路径P，此路径上的每条边e均满足$f(e) < c(e)$
  - 按路径P上最小剩余容量增加路径流量
  - 迭代寻找路径P直至无法增加路径流量

缩减边上的流量



扩充边上的流量

若寻找路径时允许逆向搜索
可以增大总流量

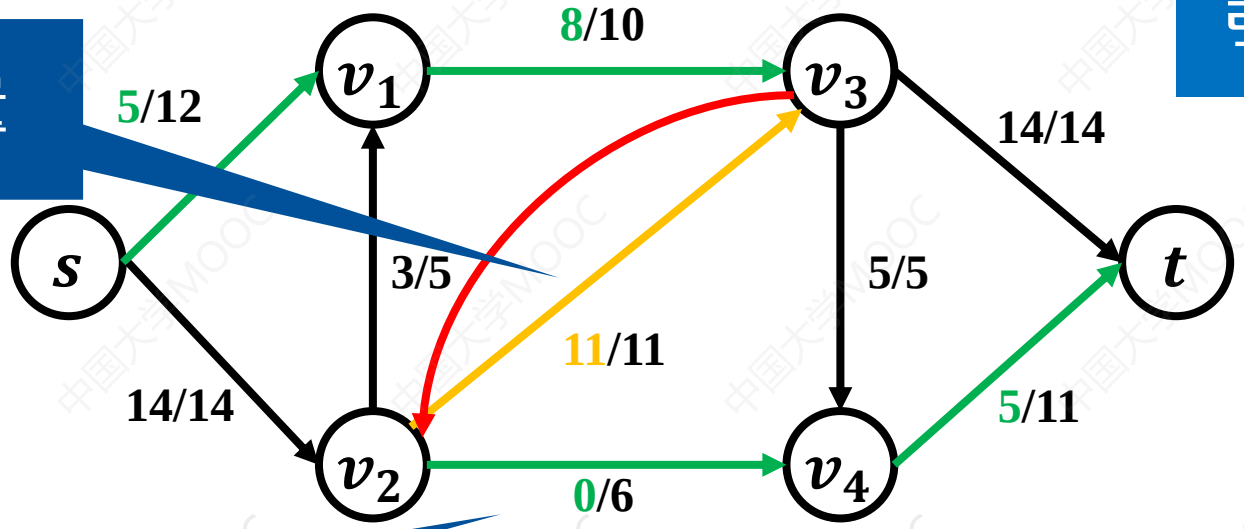
直观策略：调整

- 算法思想
  - 对所有边$e \in E$，初始化流量为零$f(e) = 0$
  - 寻找一条s到t的路径P，此路径上的每条边e均满足$f(e) < c(e)$
  - 按路径P上最小剩余容量增加路径流量
  - 迭代寻找路径P直至无法增加路径流量

缩减边上的流量

扩充边上的流量

引入反向边，实现逆向搜索



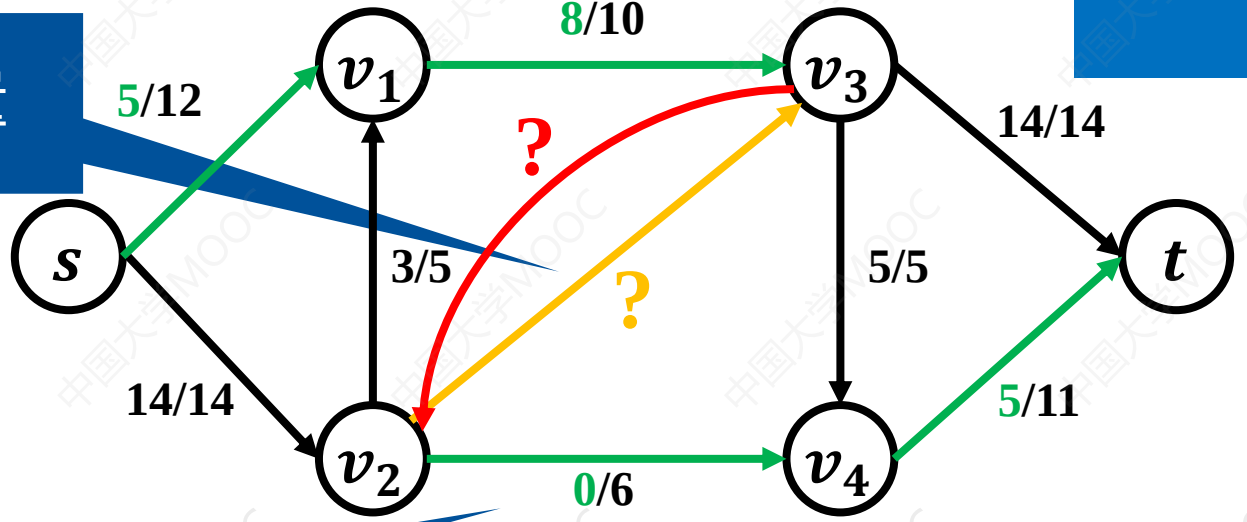
直观策略：调整

- 算法思想
  - 对所有边$e \in E$，初始化流量为零$f(e) = 0$
  - 寻找一条s到t的路径P，此路径上的每条边e均满足$f(e) < c(e)$
  - 按路径P上最小剩余容量增加路径流量
  - 迭代寻找路径P直至无法增加路径流量

缩减边上的流量

扩充边上的流量

边权如何定义?

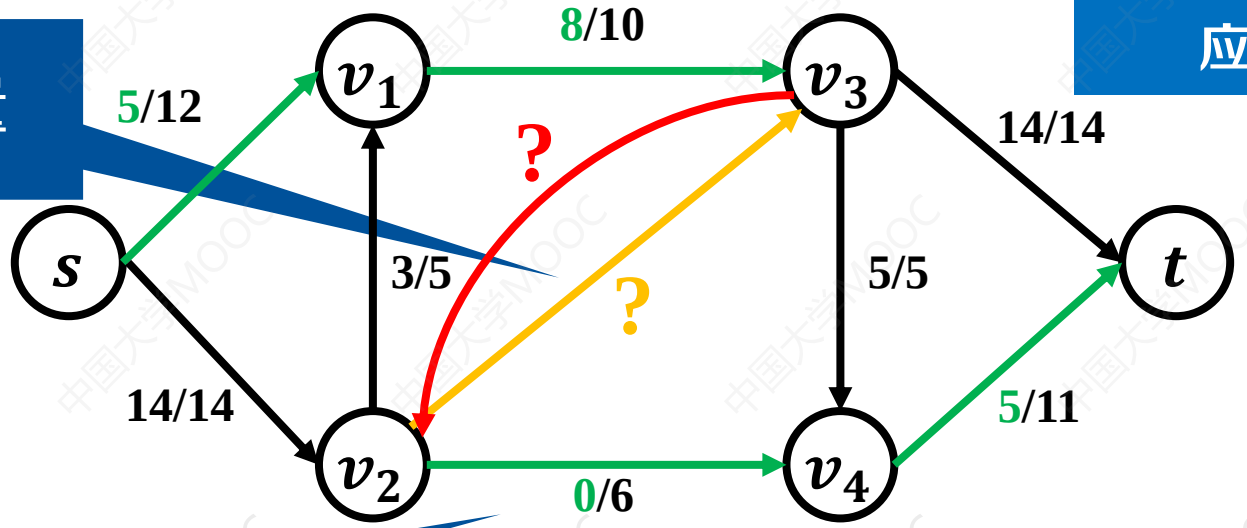


直观策略：调整

- 算法思想
  - 对所有边$e \in E$，初始化流量为零$f(e) = 0$
  - 寻找一条s到t的路径P，此路径上的每条边e均满足$f(e) < c(e)$
  - 按路径P上最小剩余容量增加路径流量
  - 迭代寻找路径P直至无法增加路径流量

缩减边上的流量

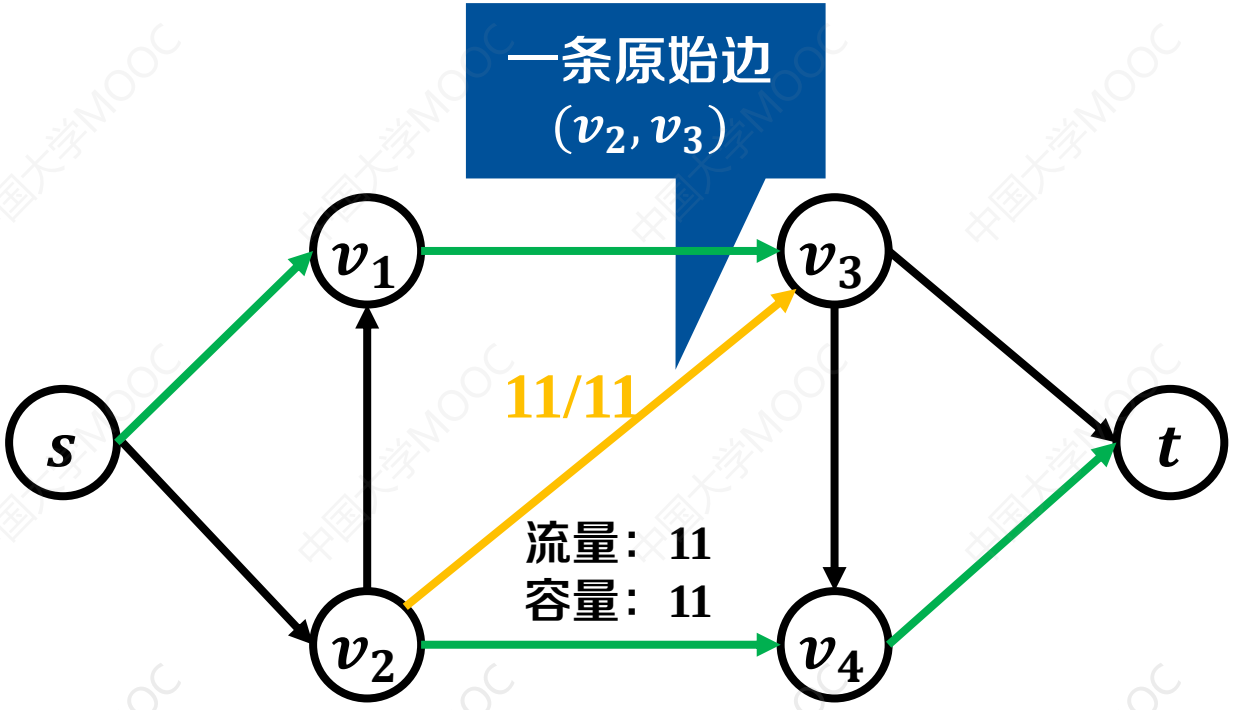
扩充边上的流量

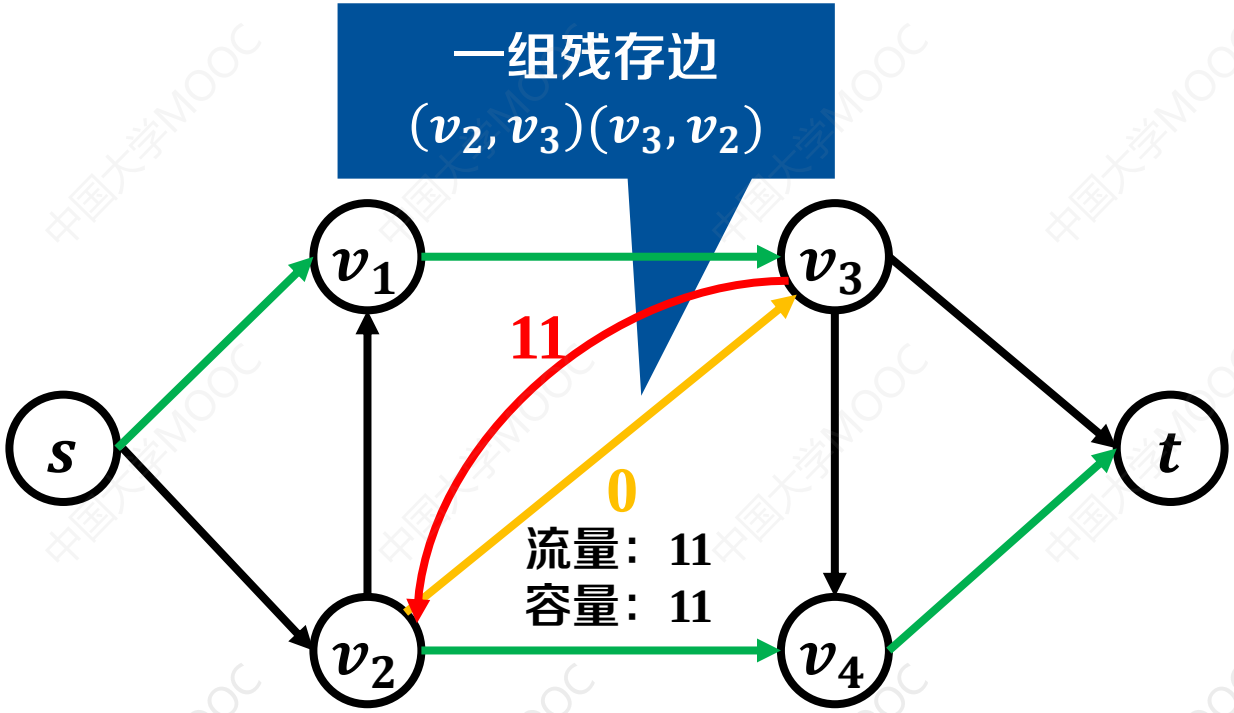


边权如何定义：
应保证流量调整的合理性

直观策略：调整

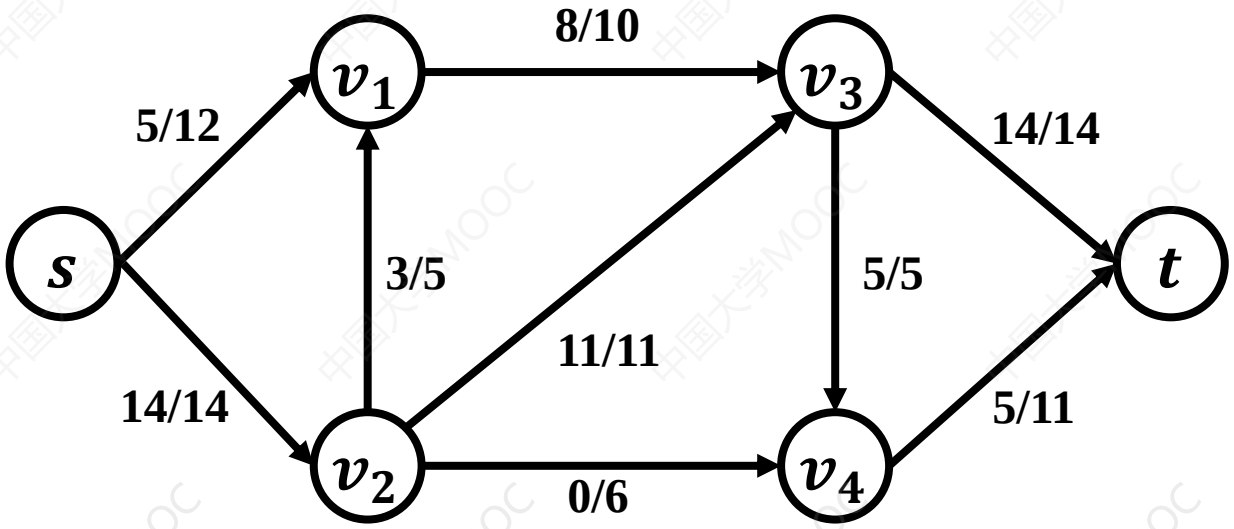
- 如何保证流量调整合理
  - 反向边权重：可缩减流量的上限，即原始边上的流量$f(e)$
  - 正向边权重：可扩充流量的上限，即原始边上的剩余容量$c(e)-f(e)$



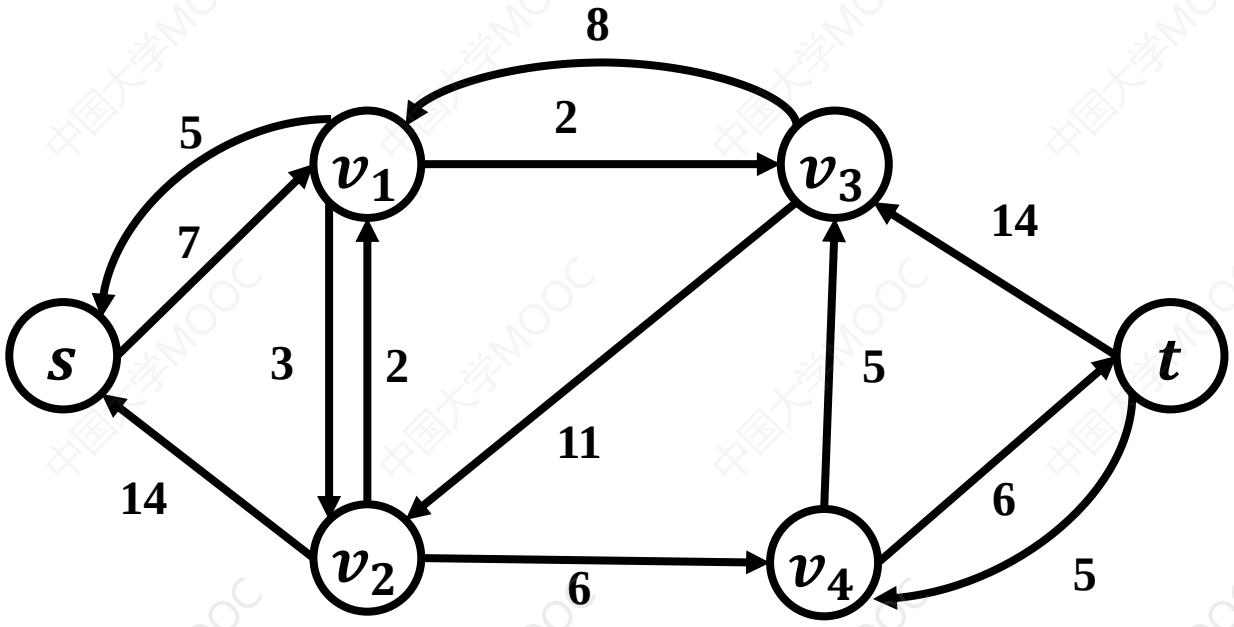


残存网络(Residual Network)

- 给定流网络$G=<V,E,C>$和流量f，可得残存网络$G_f=<V,E_f>$，其中每条边的残存容量：
  - $c_f(e)=\begin{cases}c(e)-f(e), & e\text{为正向边}\\ f(e), & e\text{为反向边}\end{cases}$



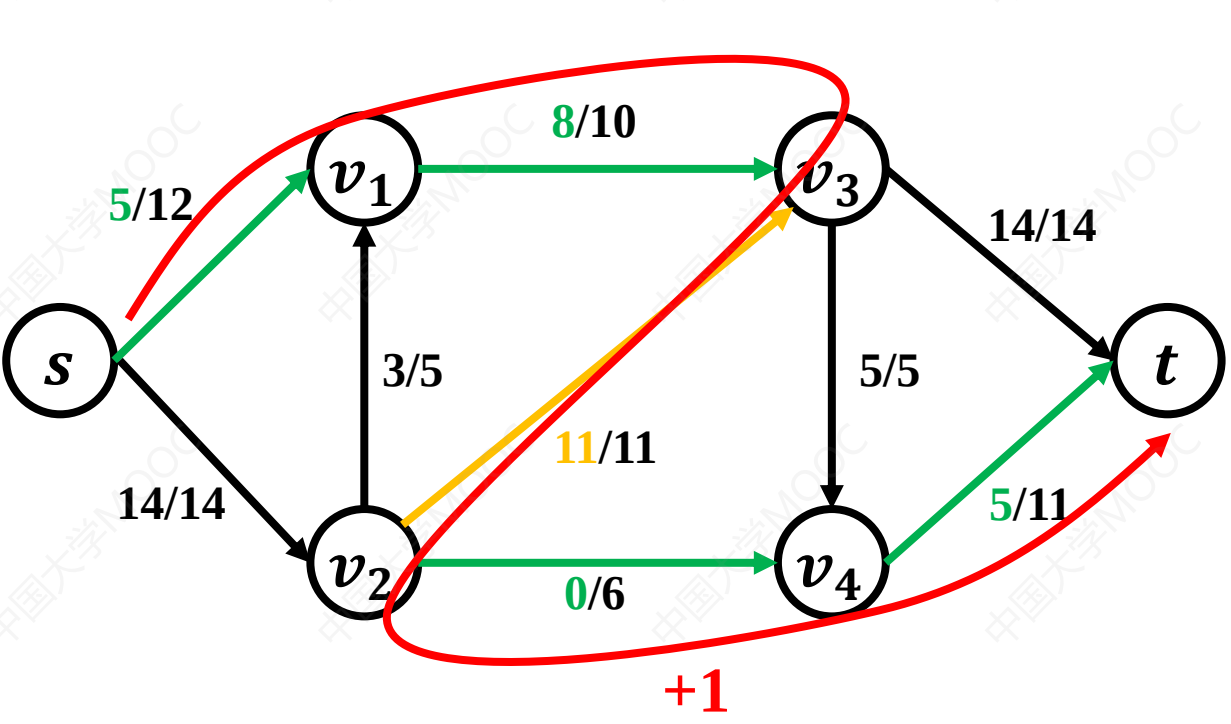
流网络

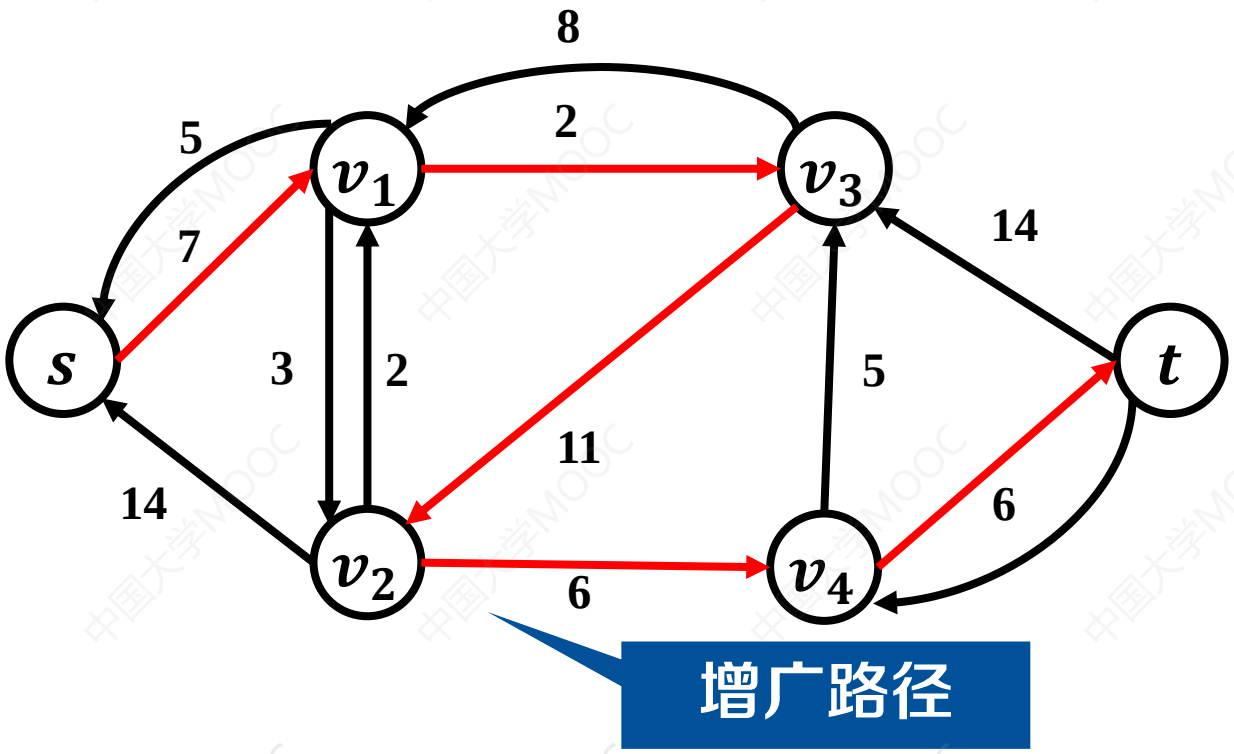


残存网络

残存网络(Residual Network)

- 增广路径(Augmenting Path)
  - 给定流网络$G =< V, E >$和流f，增广路径p是残存网络G_f中一条从源顶点s到汇点t的简单路径（路径上的各顶点均不互相重复）



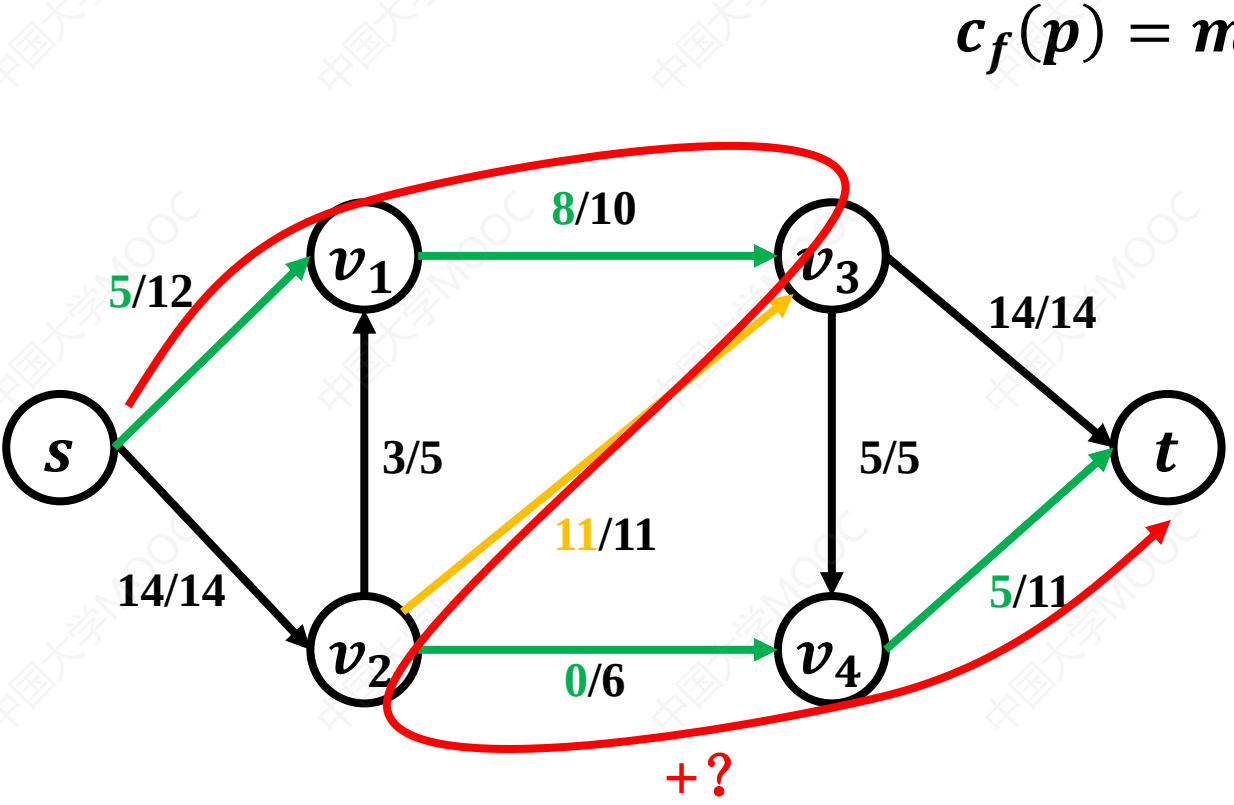


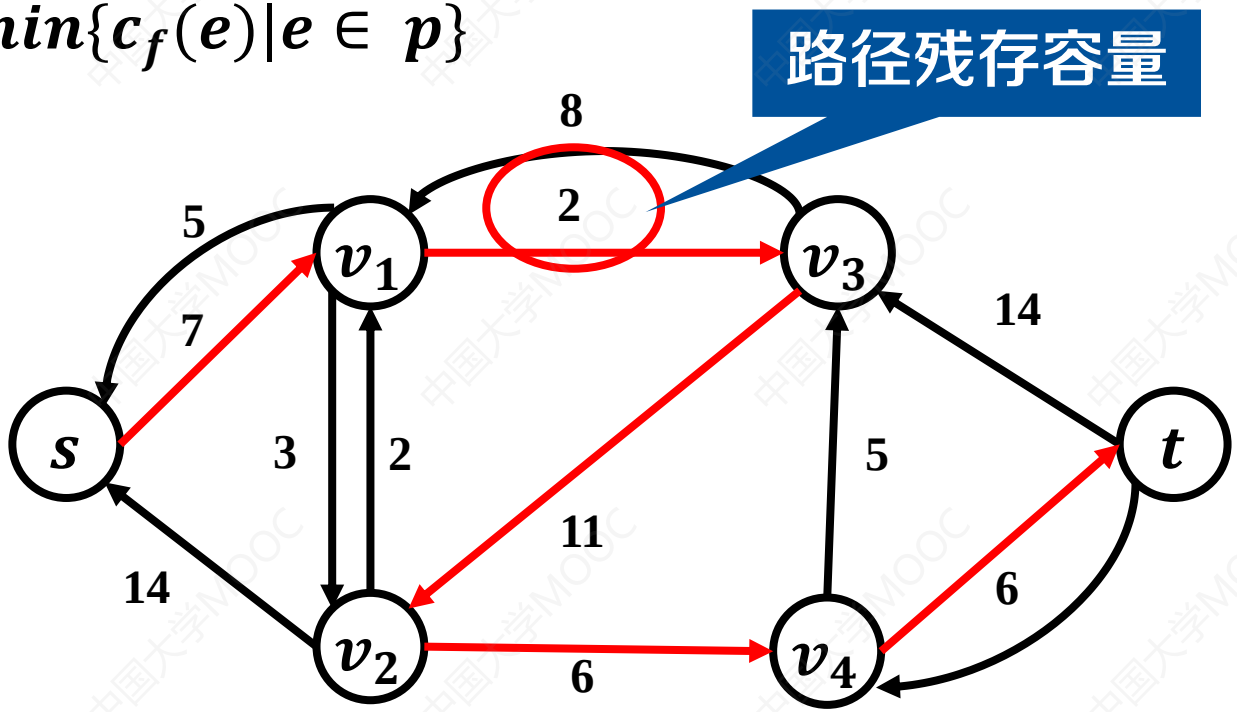
在残存网络上搜索增广路径，整合流量调整方式

残存网络(Residual Network)

- 增广路径的残存容量
  - 一条增广路径p上各边残存容量的最小值

$$c_f(p) = min\{c_f(e)|e \in p\}$$





流量扩充的最大值为增广路径的残存容量

从直观策略到Ford-Fulkerson算法

- 算法思想
  - 对所有边$e \in E$，初始化流量为零$f(e) = 0$
  - 寻找一条s到t的路径P，此路径上的每条边e均满足$f(e) < c(e)$
  - 按路径P上最小剩余容量增加路径流量
  - 迭代寻找路径P直至无法增加路径流量

从直观策略到Ford-Fulkerson算法

- 算法思想
  - 对所有边$e \in E$，初始化流量为零$f(e) = 0$
  - 构造残存网络G_f，寻找s到t的增广路径P
  - 按路径P上最小剩余容量增加路径流量
  - 迭代寻找路径P直至无法增加路径流量

从直观策略到Ford-Fulkerson算法

- 算法思想
  - 对所有边$e \in E$，初始化流量为零$f(e) = 0$
  - 构造残存网络G_f，寻找s到t的增广路径P
  - 按路径P的残存容量增加流量
  - 迭代寻找路径P直至无法增加路径流量

从直观策略到Ford-Fulkerson算法

- 算法思想
  - 对所有边$e \in E$，初始化流量为零$f(e) = 0$
  - 构造残存网络G_f，寻找s到t的增广路径P
  - 按路径P的残存容量增加流量
  - 迭代寻找路径P直至无法增加路径流量

- 设计思路小结

问题1：直观策略有何不足之处？ → 只能扩充边的流量，不能缩减边的流量

从直观策略到Ford-Fulkerson算法

- 算法思想
  - 对所有边$e \in E$，初始化流量为零$f(e) = 0$
  - 构造残存网络G_f，寻找s到t的增广路径P
  - 按路径P的残存容量增加流量
  - 迭代寻找路径P直至无法增加路径流量

- 设计思路小结

问题1：直观策略有何不足之处？ → 只能扩充边的流量，不能缩减边的流量

问题2：如何整合流量调整方式？ → 引入残存网络，寻找增广路径

从直观策略到Ford-Fulkerson算法

- 算法思想
  - 对所有边$e \in E$，初始化流量为零$f(e) = 0$
  - 构造残存网络G_f，寻找s到t的增广路径P
  - 按路径P的残存容量增加流量
  - 迭代寻找路径P直至无法增加路径流量

- 设计思路小结

问题1：直观策略有何不足之处？	只能扩充边的流量，不能缩减边的流量
问题2：如何整合流量调整方式？	引入残存网络，寻找增广路径
问题3：如何确定流量扩充数值？	按照路径残存容量增加流量

提纲

问题背景

算法思路

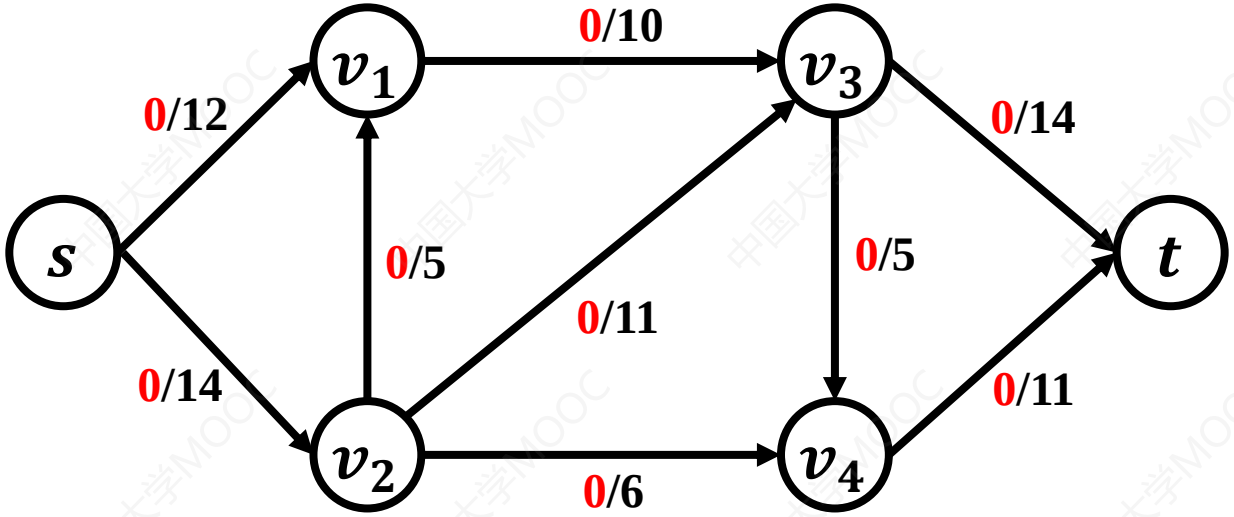
算法实例

算法分析

算法性质

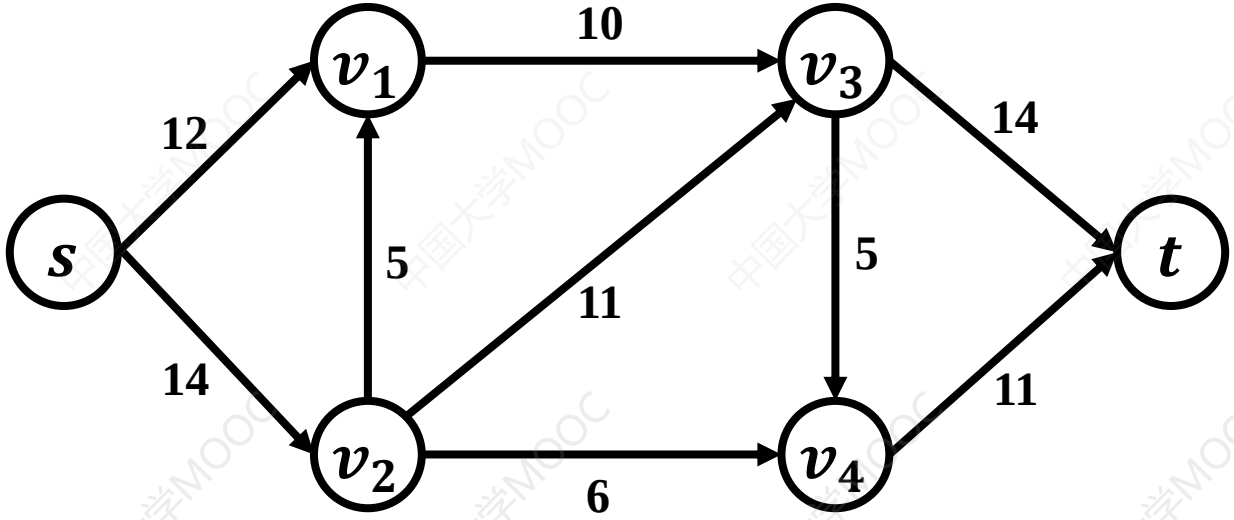
算法实例

流网络G



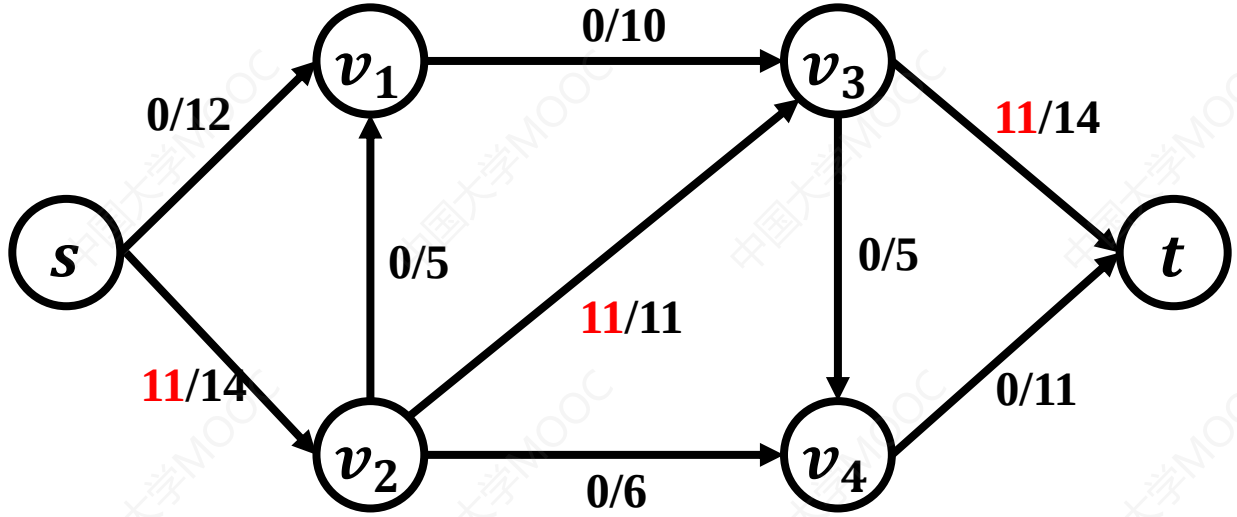
总流量：0

残存网络G_f



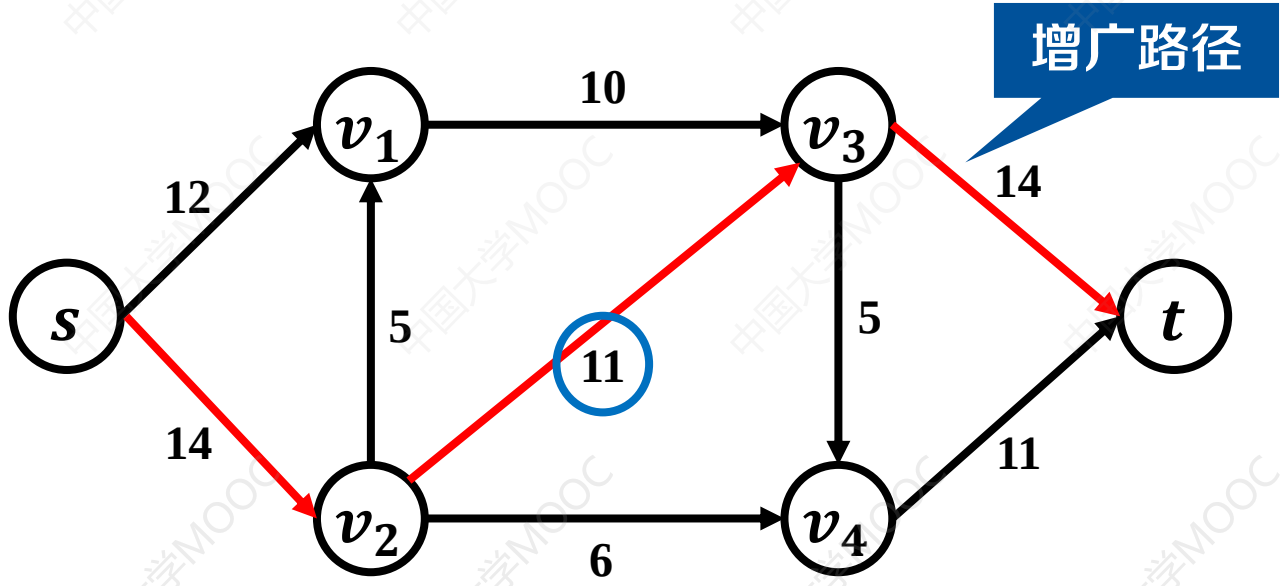
算法实例

流网络G



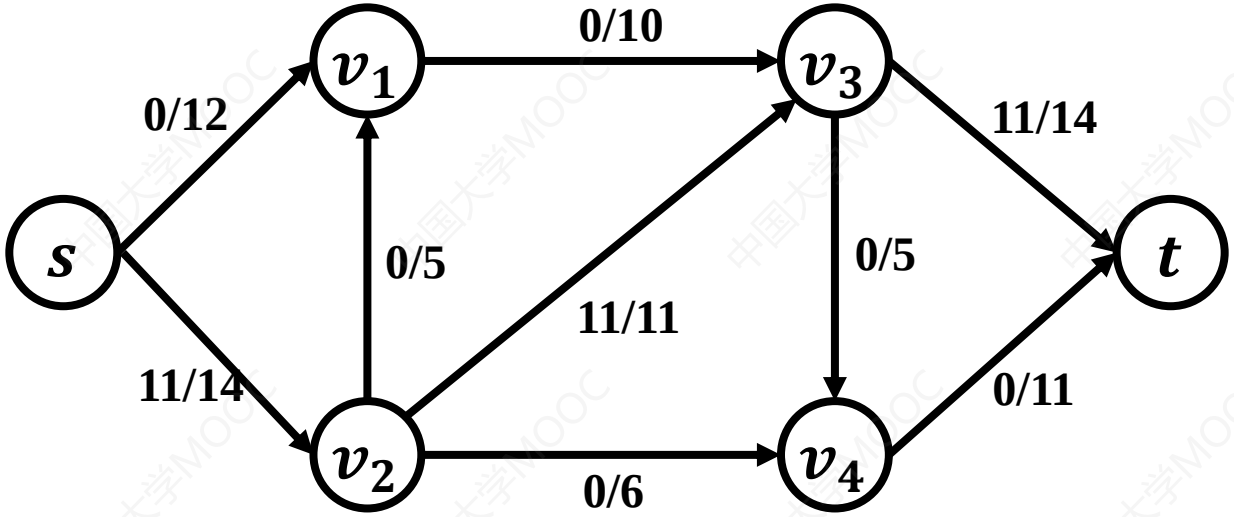
总流量： 0+11=11

残存网络G_f



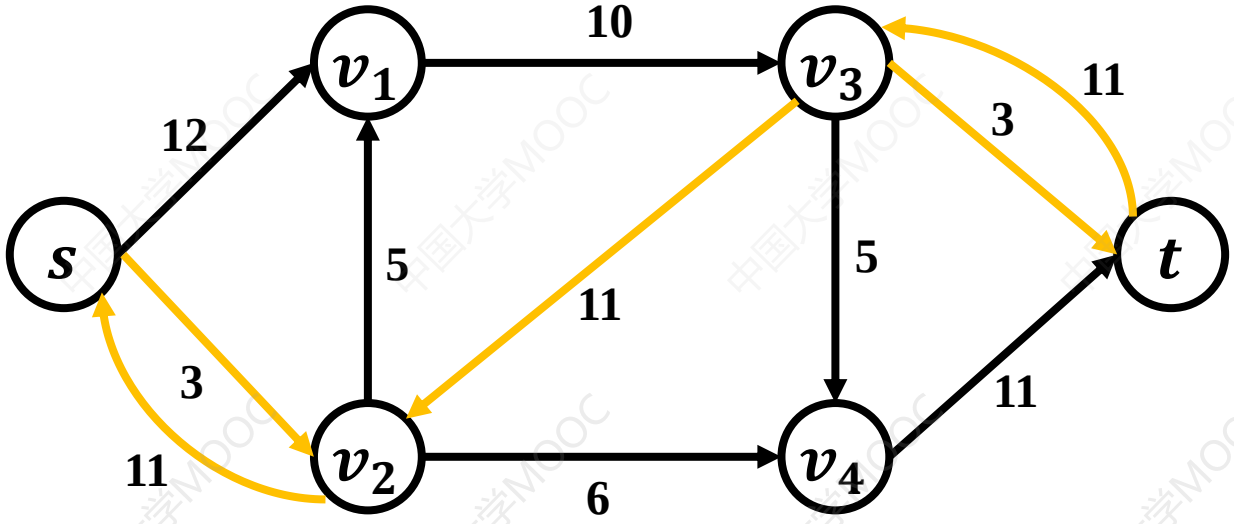
算法实例

流网络G



总流量：11

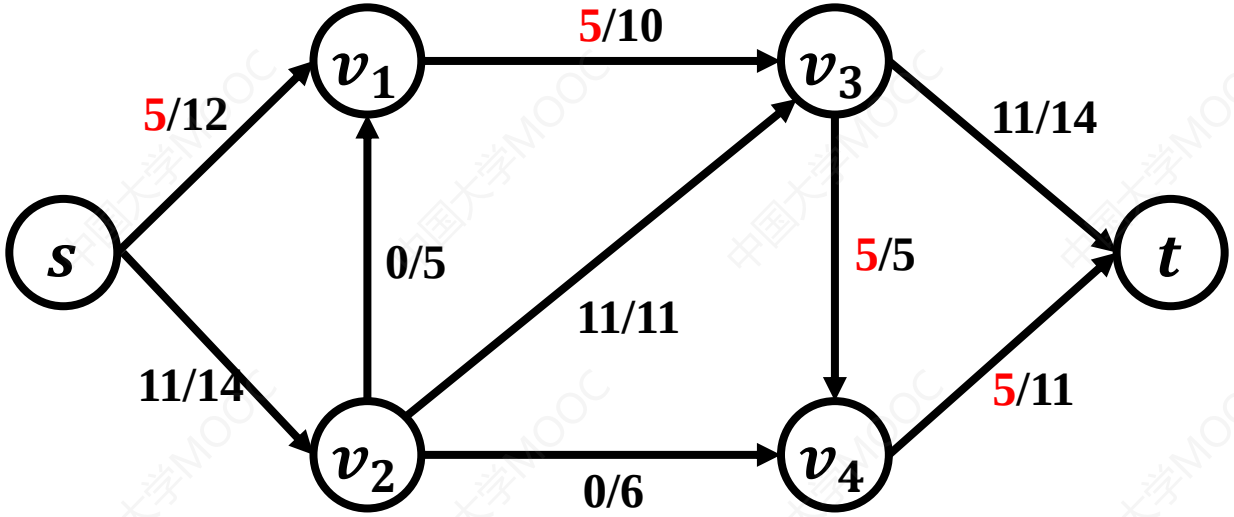
残存网络G_f



更新残存网络

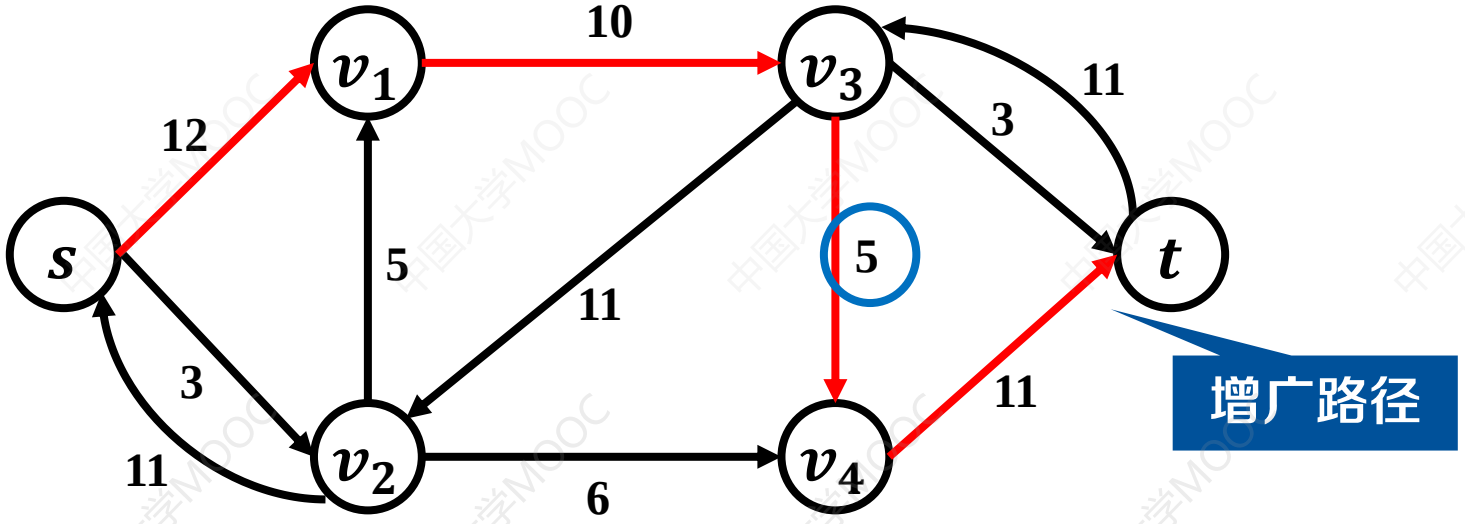
算法实例

流网络G



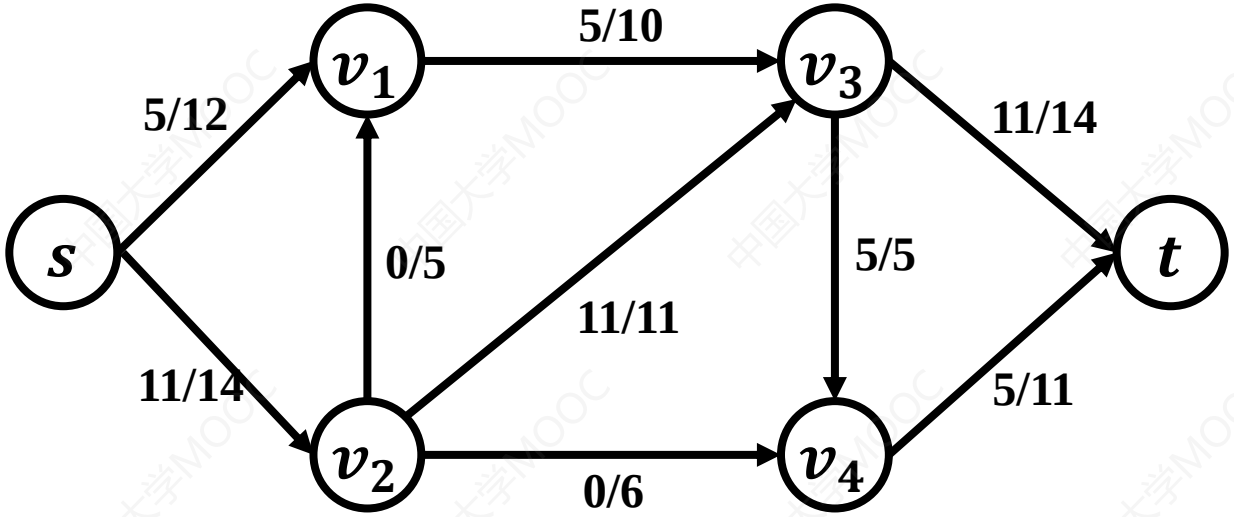
总流量：11+5=16

残存网络G_f



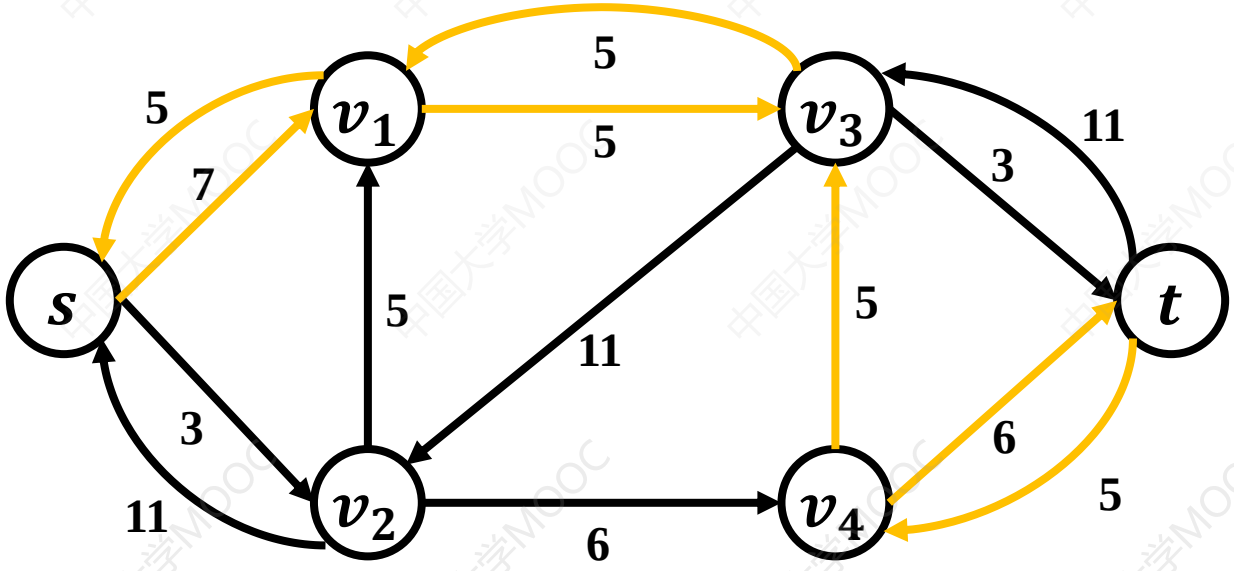
算法实例

流网络G



总流量：16

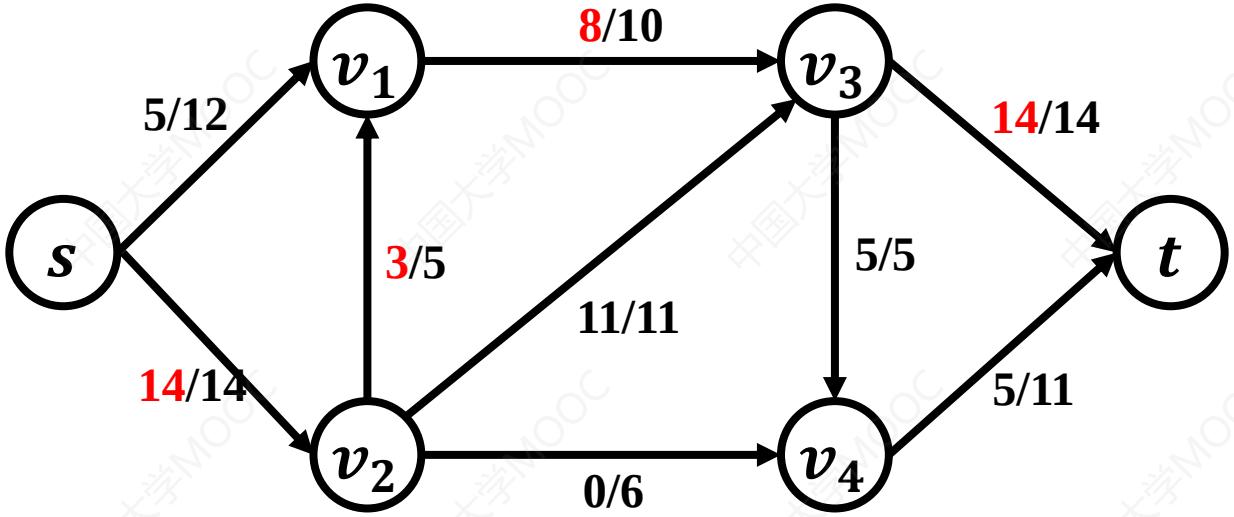
残存网络G_f



更新残存网络

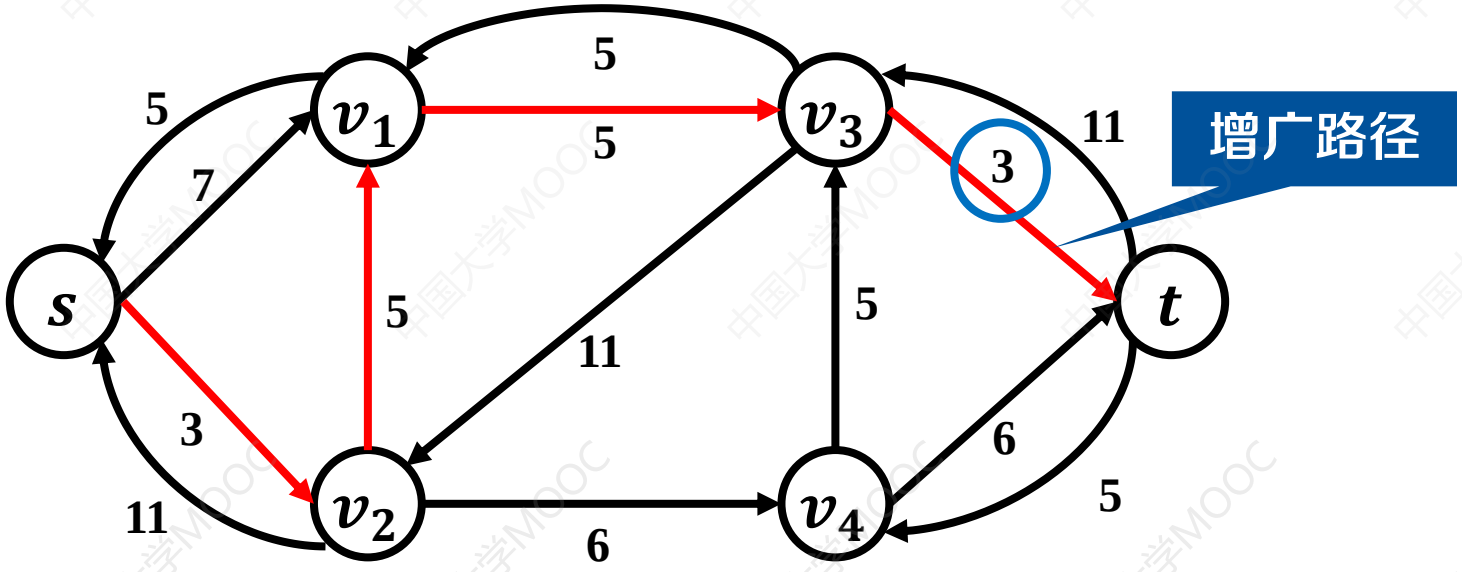
算法实例

流网络G



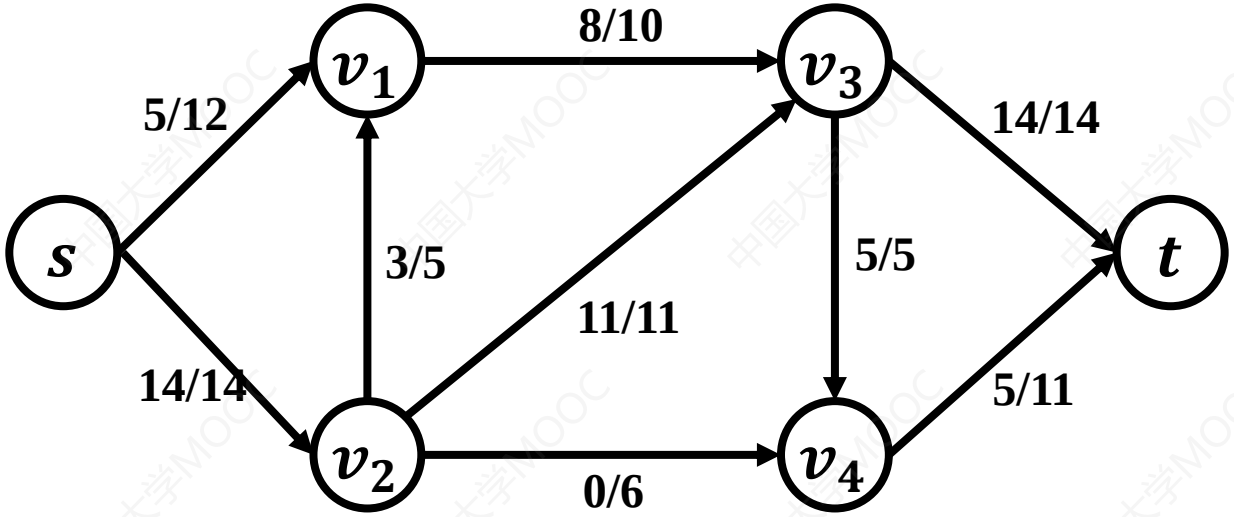
总流量：16+3=19

残存网络G_f



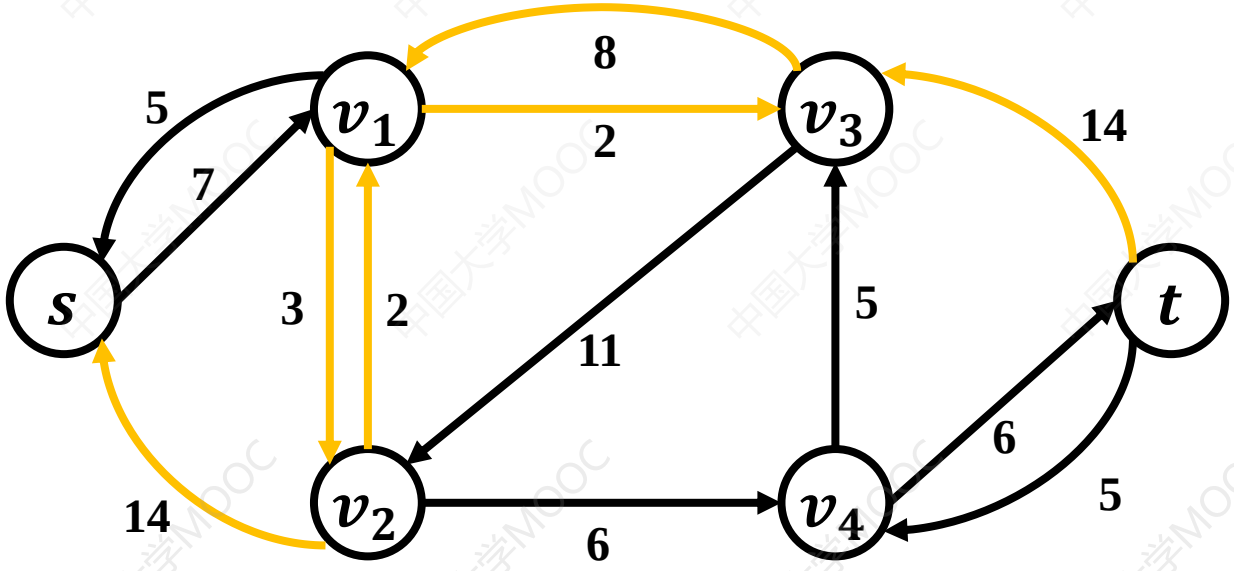
算法实例

流网络G



总流量：19

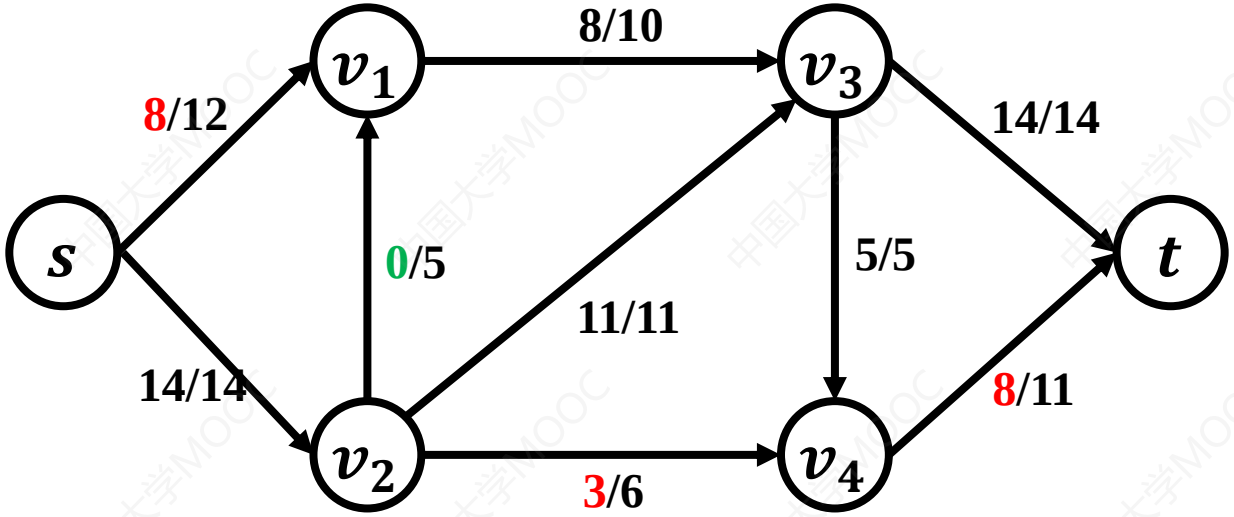
残存网络G_f



更新残存网络

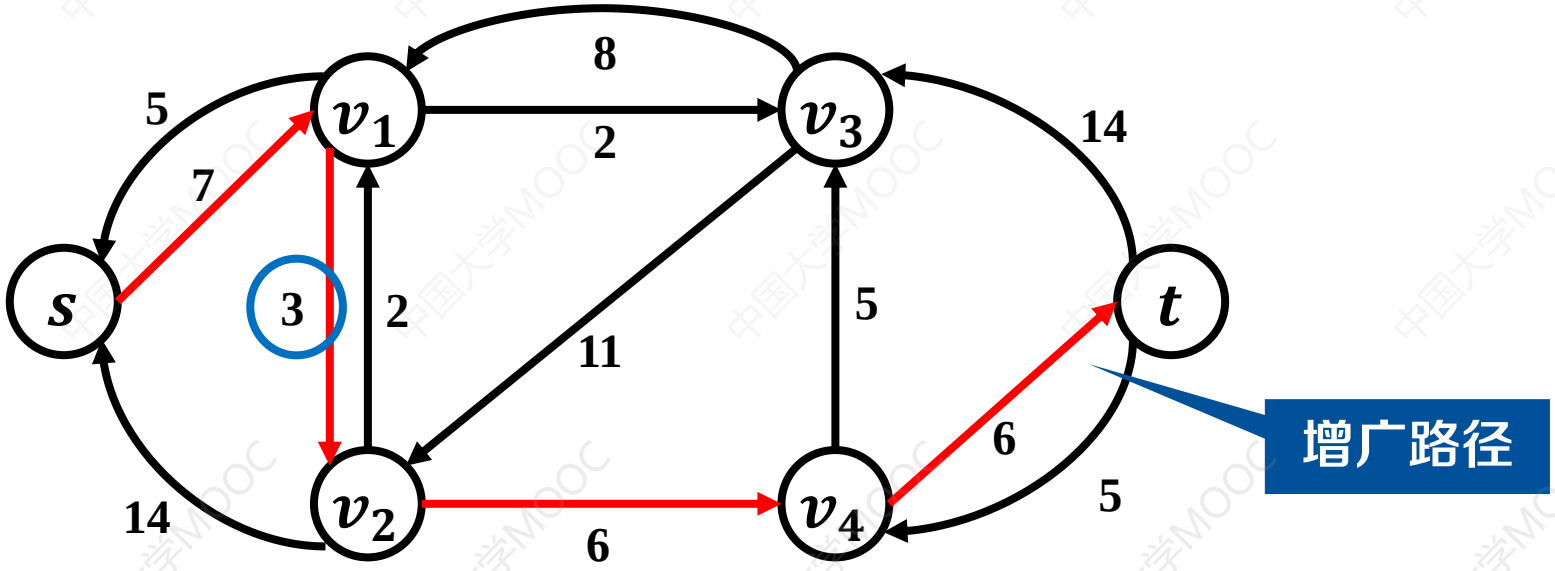
算法实例

流网络G



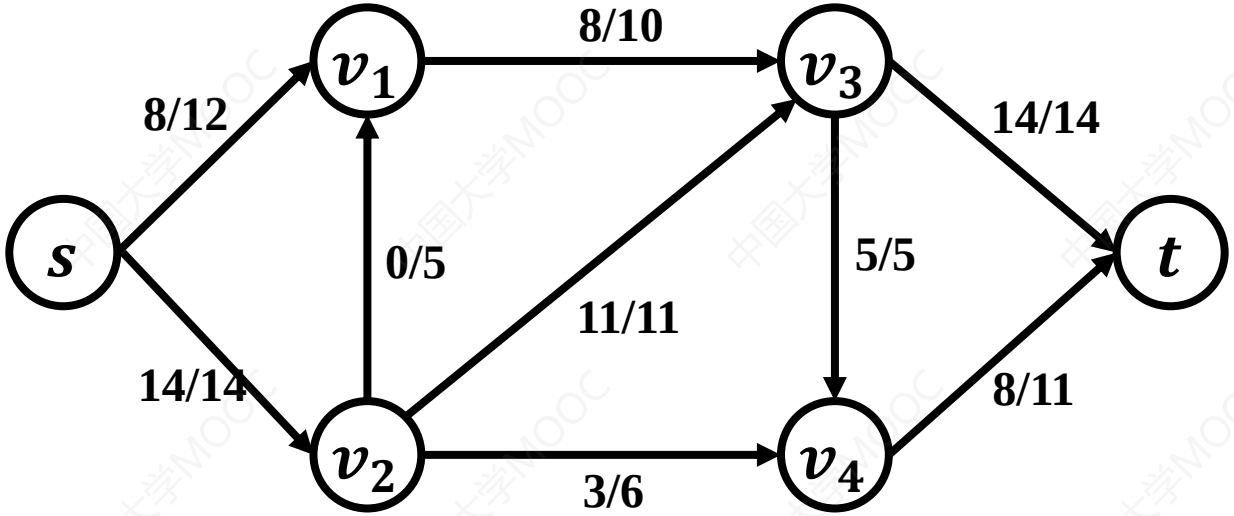
总流量：19+3=22

残存网络G_f



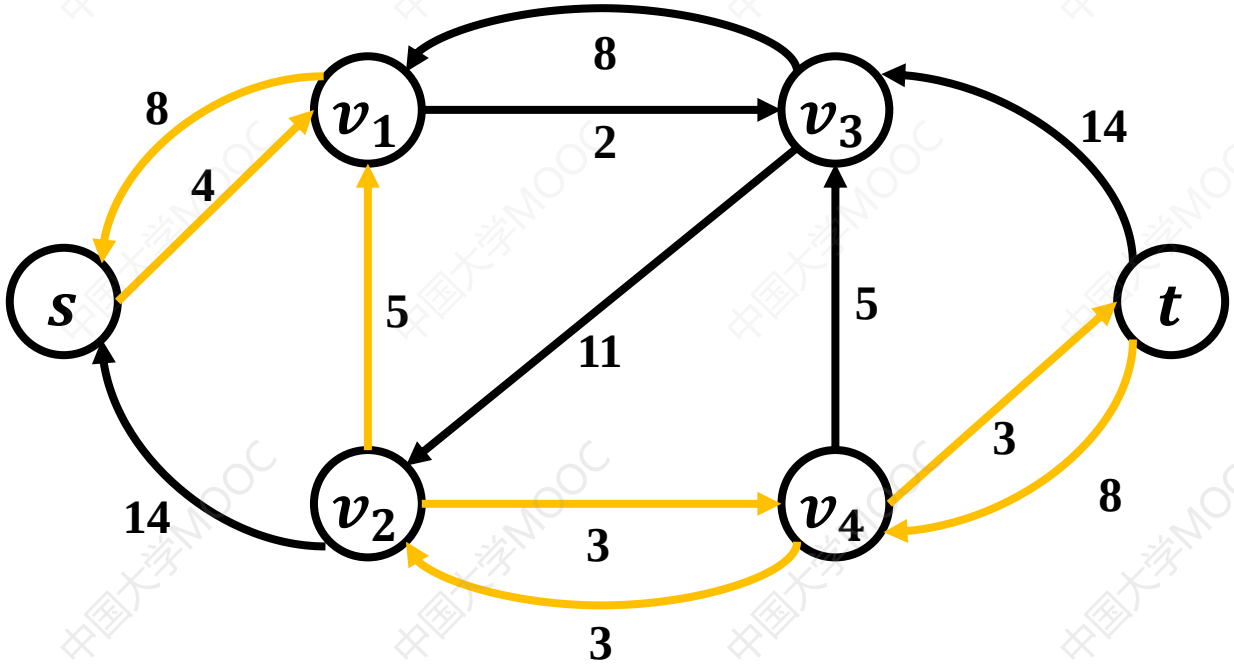
算法实例

流网络G



总流量：22

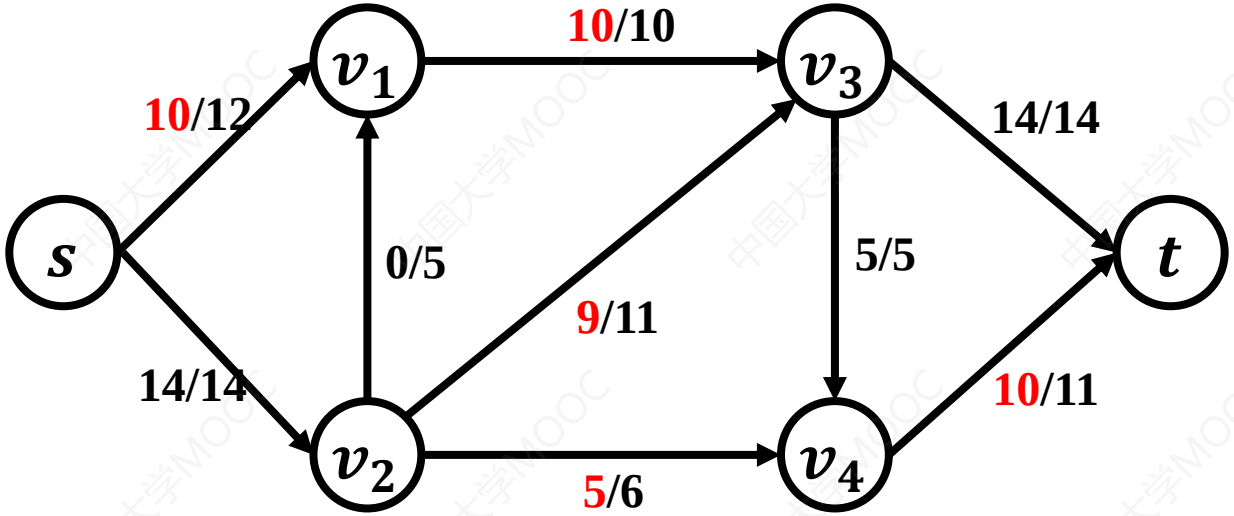
残存网络G_f



更新残存网络

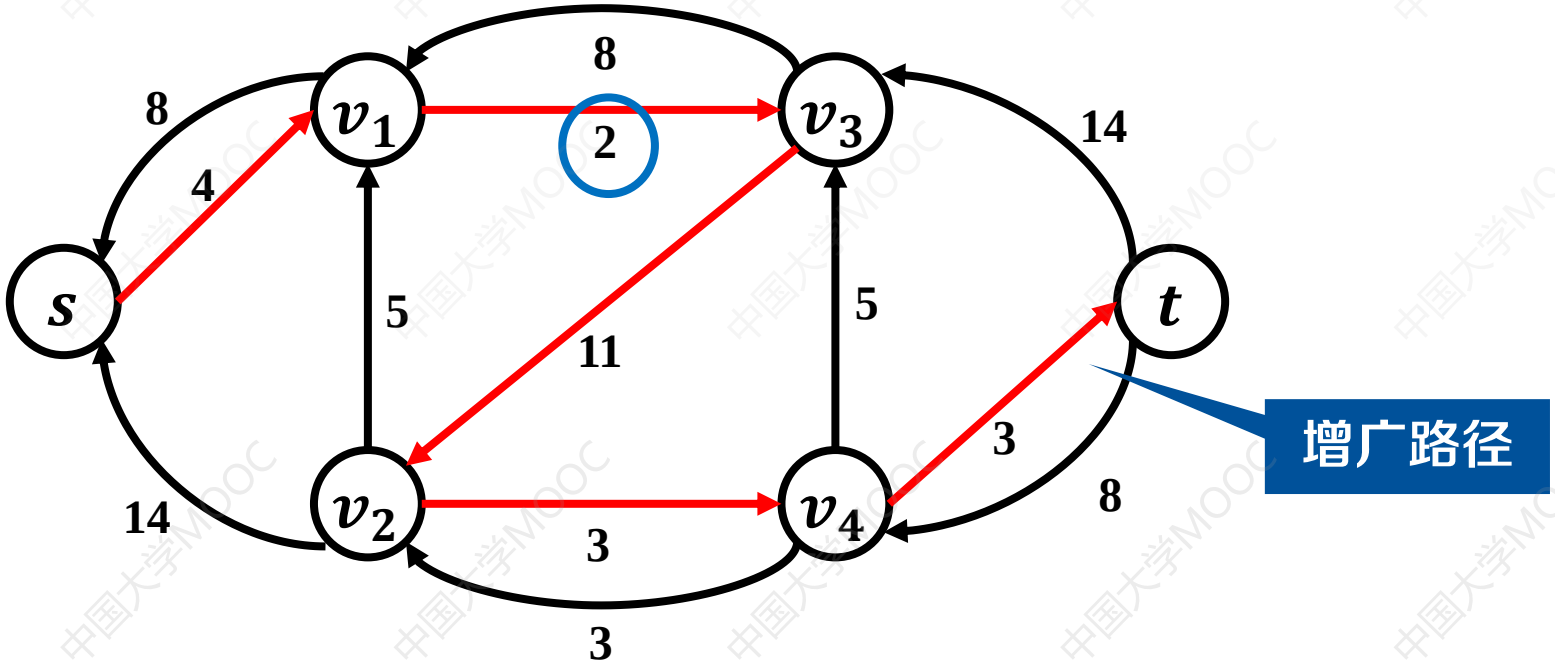
算法实例

流网络G



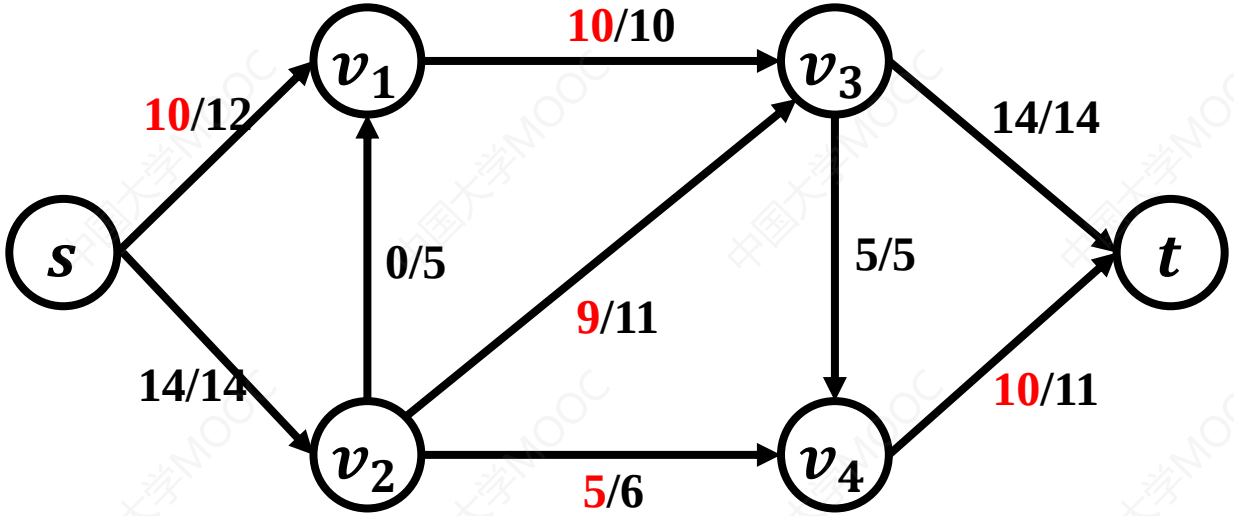
总流量：22+2=24

残存网络G_f



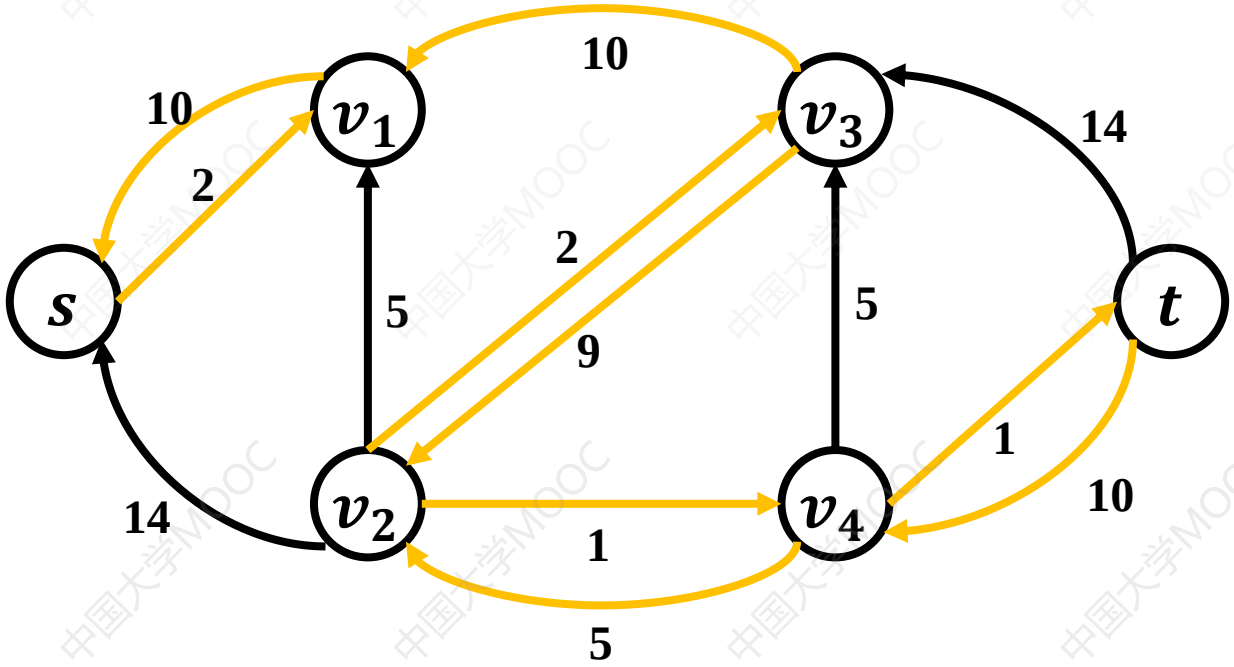
算法实例

流网络G



总流量：24

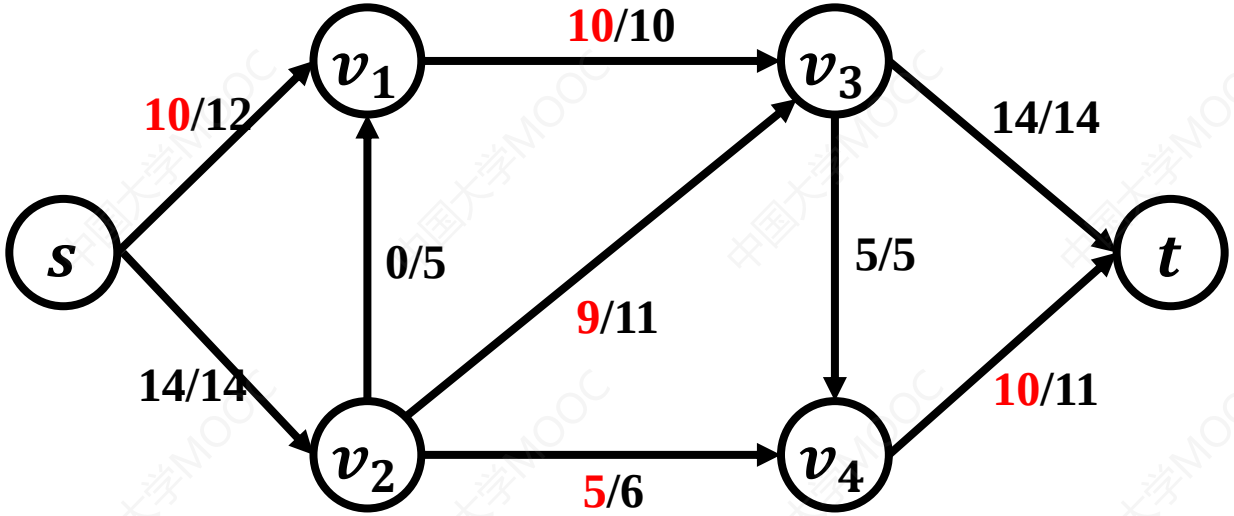
残存网络G_f



更新残存网络

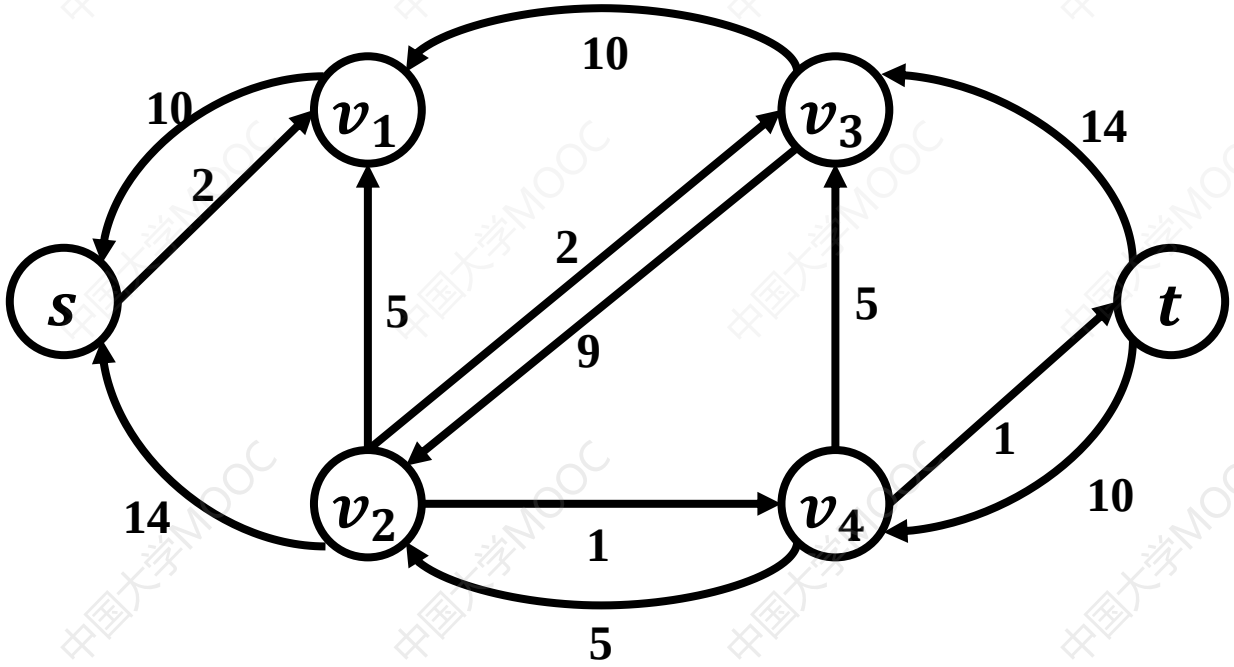
算法实例

流网络G



总流量：24

残存网络G_f



无增广路径
算法可终止

提纲

问题背景

算法思想

算法实例

算法分析

算法性质

伪代码

- **Ford-Fulkerson(G, s, t)**

```
输入: 图G =< V, E, C >，源点s，汇点t
输出: 最大流f*
//初始化边的流量
for each edge e∈G.E do
    e.f ← 0
end
G_f ← build residual network
//在G_f中寻找增广路径
while there exists a path p from s to t in the residual network G_f do
    c_f(p) ← min{c_f(e) : e is in p}
    //更新流f
    for each edge e in p do
        if e ∈ E then
            e.f ← e.f + c_f(p)
        end
        else
            e.f ← e.f − c_f(p)
        end
    end
    update G_f
end
f* ← f
return f*
```

初始化边的流量

伪代码

- **Ford-Fulkerson(G, s, t)**

输入: 图$G =< V, E, C >$，源点s，汇点t
输出: 最大流f^*
//初始化边的流量
for *each edge* $e \in G.E$ **do**
  $e.f \leftarrow 0$
end
$G_f \leftarrow$ *build residual network*
//在G_f中寻找增广路径
while *there exists a path* p *from* s *to* t *in the residual network* G_f **do**
  $c_f(p) \leftarrow min\{c_f(e) : e \text{ is in } p\}$
  //更新流f
  for *each edge* e *in* p **do**
    if $e \in E$ **then**
      $e.f \leftarrow e.f + c_f(p)$
    end
    else
      $e.f \leftarrow e.f - c_f(p)$
    end
  end
  update G_f
end
$f^* \leftarrow f$
return f^*

构造残存网络

伪代码

- **Ford-Fulkerson(G, s, t)**

```
输入: 图G =< V, E, C >，源点s，汇点t
输出: 最大流f*
//初始化边的流量
for each edge e∈G.E do
    e.f ← 0
end
G_f ← build residual network
//在G_f中寻找增广路径
while there exists a path p from s to t in the residual network G_f do
    c_f(p) ← min{c_f(e) : e is in p}
    //更新流f
    for each edge e in p do
        if e ∈ E then
            e.f ← e.f + c_f(p)
        end
        else
            e.f ← e.f − c_f(p)
        end
    end
    update G_f
end
f* ← f
return f*
```

寻找增广路径

伪代码

- **Ford-Fulkerson(G, s, t)**

输入: 图$G =< V, E, C >$，源点s，汇点t
输出: 最大流f^*
//初始化边的流量
for *each edge* $e \in G.E$ **do**
  $e.f \leftarrow 0$
end
$G_f \leftarrow$ *build residual network*
//在G_f中寻找增广路径
while *there exists a path* p *from* s *to* t *in the residual network* G_f **do**
  $c_f(p) \leftarrow min\{c_f(e) : e \text{ is in } p\}$
  //更新流f
  for *each edge* e *in* p **do**
    if $e \in E$ **then**
      $e.f \leftarrow e.f + c_f(p)$
    end
    else
      $e.f \leftarrow e.f - c_f(p)$
    end
  end
  update G_f
end
$f^* \leftarrow f$
return f^*

确定增广路径残存容量

伪代码

- **Ford-Fulkerson(G, s, t)**

输入: 图$G =< V, E, C >$，源点s，汇点t
输出: 最大流f^*
//初始化边的流量
for *each edge* $e \in G.E$ **do**
  $e.f \leftarrow 0$
end
$G_f \leftarrow$ *build residual network*
//在G_f中寻找增广路径
while *there exists a path* p *from* s *to* t *in the residual network* G_f **do**
  $c_f(p) \leftarrow min\{c_f(e) : e \text{ is in } p\}$
  //更新流f
  for *each edge* e *in* p **do**
    if $e \in E$ **then**
      $e.f \leftarrow e.f + c_f(p)$
    end
    else
      $e.f \leftarrow e.f - c_f(p)$
    end
  end
  update G_f
end
$f^* \leftarrow f$
return f^*

更新流

伪代码

- **Ford-Fulkerson(G, s, t)**

```
输入: 图G =< V, E, C >，源点s，汇点t
输出: 最大流f*
//初始化边的流量
for each edge e∈G.E do
    e.f ← 0
end
G_f ← build residual network
//在G_f中寻找增广路径
while there exists a path p from s to t in the residual network G_f do
    c_f(p) ← min{c_f(e) : e is in p}
    //更新流f
    for each edge e in p do
        if e ∈ E then
            e.f ← e.f + c_f(p)
        end
        else
            e.f ← e.f − c_f(p)
        end
    end
    update G_f
end
f* ← f
return f*
```

更新残存网络

复杂度分析

- **Ford-Fulkerson(G, s, t)**

输入: 图$G =< V, E, C >$，源点s，汇点t
输出: 最大流f^*
//初始化边的流量
for *each edge* $e \in G.E$ **do**
  $e.f \leftarrow 0$
end
$G_f \leftarrow$ *build residual network*
//在G_f中寻找增广路径
while *there exists a path* p *from* s *to* t *in the residual network* G_f **do**
  $c_f(p) \leftarrow min\{c_f(e) : e \text{ is in } p\}$
  //更新流f
  for *each edge* e *in* p **do**
    if $e \in E$ **then**
      $e.f \leftarrow e.f + c_f(p)$
    end
    else
      $e.f \leftarrow e.f - c_f(p)$
    end
  end
  update G_f
end
$f^* \leftarrow f$
return f^*

$O(E)$

复杂度分析

- **Ford-Fulkerson(G, s, t)**

输入: 图$G =< V, E, C >$，源点s，汇点t
输出: 最大流f^*
//初始化边的流量
for *each edge* $e \in G.E$ **do**
  $e.f \leftarrow 0$
end
$G_f \leftarrow$ *build residual network*
//在G_f中寻找增广路径
while *there exists a path* p *from* s *to* t *in the residual network* G_f **do**
  $c_f(p) \leftarrow min\{c_f(e) : e \text{ is in } p\}$
  //更新流f
  for *each edge* e *in* p **do**
    if $e \in E$ **then**
      $e.f \leftarrow e.f + c_f(p)$
    end
    else
      $e.f \leftarrow e.f - c_f(p)$
    end
  end
  update G_f
end
$f^* \leftarrow f$
return f^*

$O(E)$

残存网络最多有$2E$条边
且$O(V) \leq O(E)$

复杂度分析

- **Ford-Fulkerson(G, s, t)**

输入: 图$G =< V, E, C >$，源点s，汇点t
输出: 最大流f^*
//初始化边的流量
for *each edge* $e \in G.E$ **do**
  $e.f \leftarrow 0$
end
$G_f \leftarrow$ *build residual network*
//在G_f中寻找增广路径
while *there exists a path* p *from* s *to* t *in the residual network* G_f **do**
  $c_f(p) \leftarrow min\{c_f(e) : e \text{ is in } p\}$
  //更新流f
  for *each edge* e *in* p **do**
    if $e \in E$ **then**
      $e.f \leftarrow e.f + c_f(p)$
    end
    else
      $e.f \leftarrow e.f - c_f(p)$
    end
  end
  update G_f
end
$f^* \leftarrow f$
return f^*

$O(E)$

使用DFS，可在$O(V+2E) = O(E)$时间内找到一条增广路

复杂度分析

- **Ford-Fulkerson(G, s, t)**

输入: 图$G =< V, E, C >$，源点s，汇点t
输出: 最大流f^*
//初始化边的流量
for *each edge* $e \in G.E$ **do**
  $e.f \leftarrow 0$
end
$G_f \leftarrow$ *build residual network*
//在G_f中寻找增广路径
while *there exists a path* p *from* s *to* t *in the residual network* G_f **do**
  $c_f(p) \leftarrow min\{c_f(e) : e \text{ is in } p\}$
  //更新流f
  for *each edge* e *in* p **do**
    if $e \in E$ **then**
      $e.f \leftarrow e.f + c_f(p)$
    end
    else
      $e.f \leftarrow e.f - c_f(p)$
    end
  end
  update G_f
end
$f^* \leftarrow f$
return f^*

$O(E)$

$O(E)$

$O(E)$

$O(E)$

复杂度分析

- **Ford-Fulkerson(G, s, t)**

输入: 图$G =< V, E, C >$，源点s，汇点t
输出: 最大流f^*
//初始化边的流量
for *each edge* $e \in G.E$ **do**
  $e.f \leftarrow 0$
end
$G_f \leftarrow$ *build residual network*
//在G_f中寻找增广路径
while *there exists a path* p *from* s *to* t *in the residual network* G_f **do**
  $c_f(p) \leftarrow min\{c_f(e) : e \text{ is in } p\}$
  //更新流f
  for *each edge* e *in* p **do**
    if $e \in E$ **then**
      $e.f \leftarrow e.f + c_f(p)$
    end
    else
      $e.f \leftarrow e.f - c_f(p)$
    end
  end
  update G_f
end
$f^* \leftarrow f$
return f^*

每次循环中，
流的值至少提升一个单位

复杂度分析

- **Ford-Fulkerson(G, s, t)**

输入: 图$G =< V, E, C >$，源点s，汇点t
输出: 最大流f^*
//初始化边的流量
for *each edge* $e \in G.E$ **do**
  $e.f \leftarrow 0$
end
$G_f \leftarrow$ *build residual network*
//在G_f中寻找增广路径
while *there exists a path* p *from* s *to* t *in the residual network* G_f **do**
  $c_f(p) \leftarrow min\{c_f(e) : e \text{ is in } p\}$
  //更新流f
  for *each edge* e *in* p **do**
    if $e \in E$ **then**
      $e.f \leftarrow e.f + c_f(p)$
    end
    else
      $e.f \leftarrow e.f - c_f(p)$
    end
  end
  update G_f
end
$f^* \leftarrow f$
return f^*

每次循环后，
流的值至少增加1

最多循环$|f^*|$次

复杂度分析

- **Ford-Fulkerson(G, s, t)**

输入: 图$G =< V, E, C >$，源点s，汇点t
输出: 最大流f^*
//初始化边的流量
for *each edge* $e \in G.E$ **do**
  $e.f \leftarrow 0$
end
$G_f \leftarrow$ *build residual network*
//在G_f中寻找增广路径
while *there exists a path* p *from* s *to* t *in the residual network* G_f **do**
  $c_f(p) \leftarrow min\{c_f(e) : e \text{ is in } p\}$
  //更新流f
  for *each edge* e *in* p **do**
    if $e \in E$ **then**
      $e.f \leftarrow e.f + c_f(p)$
    end
    else
      $e.f \leftarrow e.f - c_f(p)$
    end
  end
  update G_f
end
$f^* \leftarrow f$
return f^*

$O(E)$

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

复杂度分析

- **Ford-Fulkerson(G, s, t)**

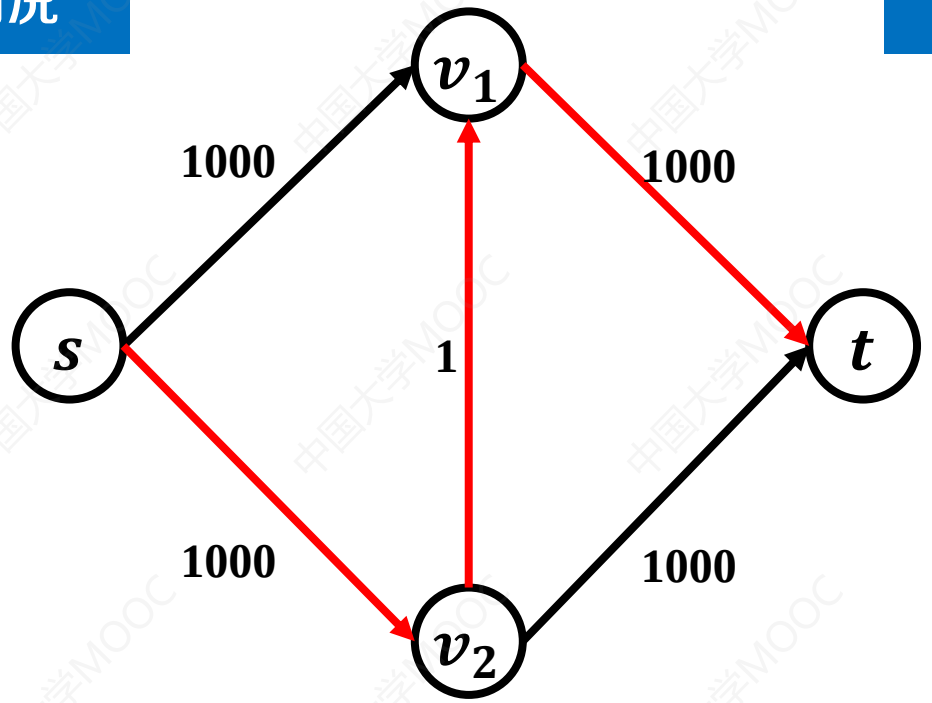
```
输入: 图G =< V, E, C >，源点s，汇点t
输出: 最大流f*
//初始化边的流量
for each edge e∈G.E do
    e.f ← 0
end
G_f ← build residual network
//在G_f中寻找增广路径
while there exists a path p from s to t in the residual network G_f do
    c_f(p) ← min{c_f(e) : e is in p}
    //更新流f
    for each edge e in p do
        if e ∈ E then
            e.f ← e.f + c_f(p)
        end
        else
            e.f ← e.f − c_f(p)
        end
    end
    update G_f
end
f* ← f
return f*
```

时间复杂度$O(E \cdot |f^*|)$

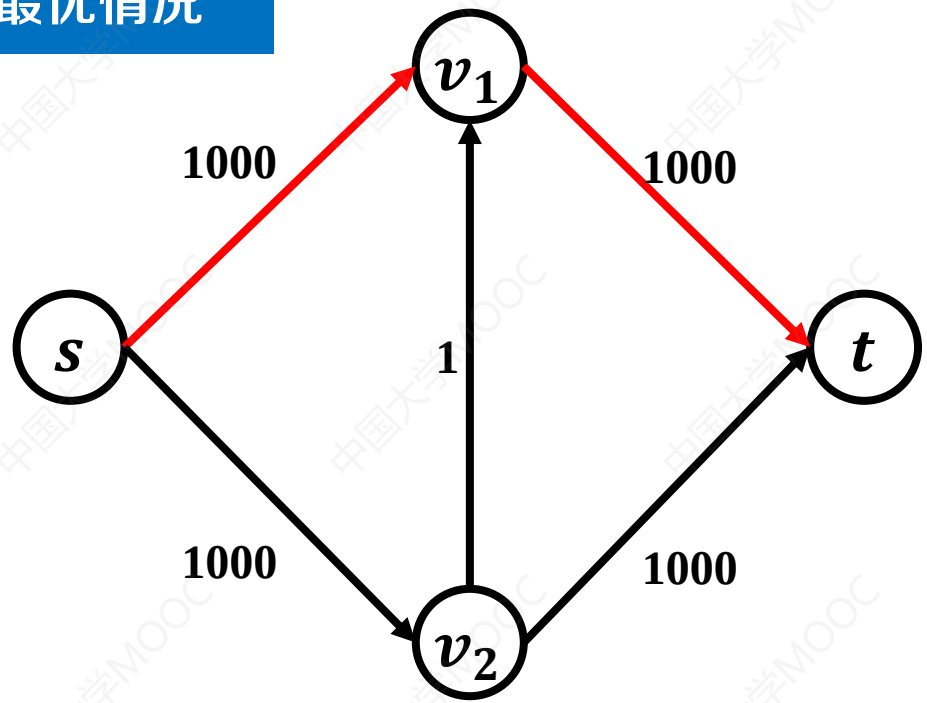
复杂度分析

- 运行时间依赖于增广过程

最坏情况



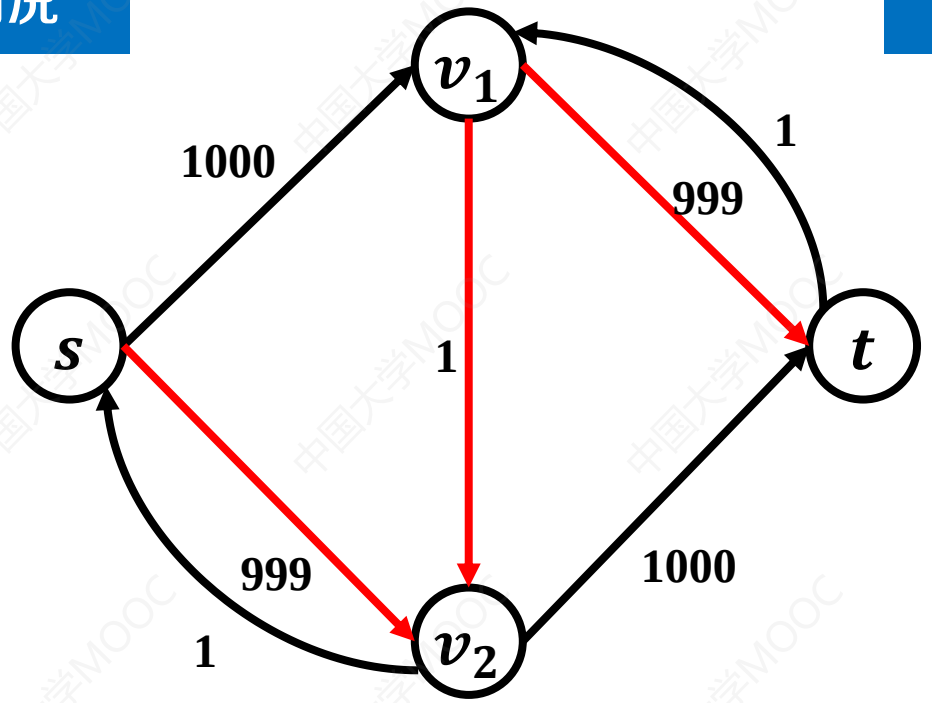
最优情况



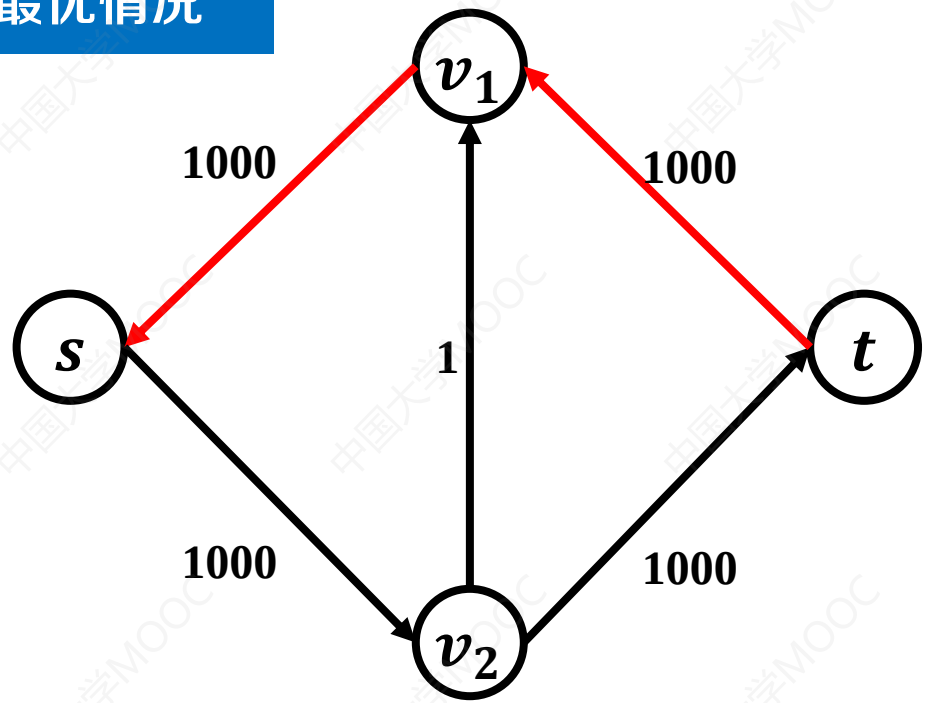
复杂度分析

- 运行时间依赖于增广过程

最坏情况



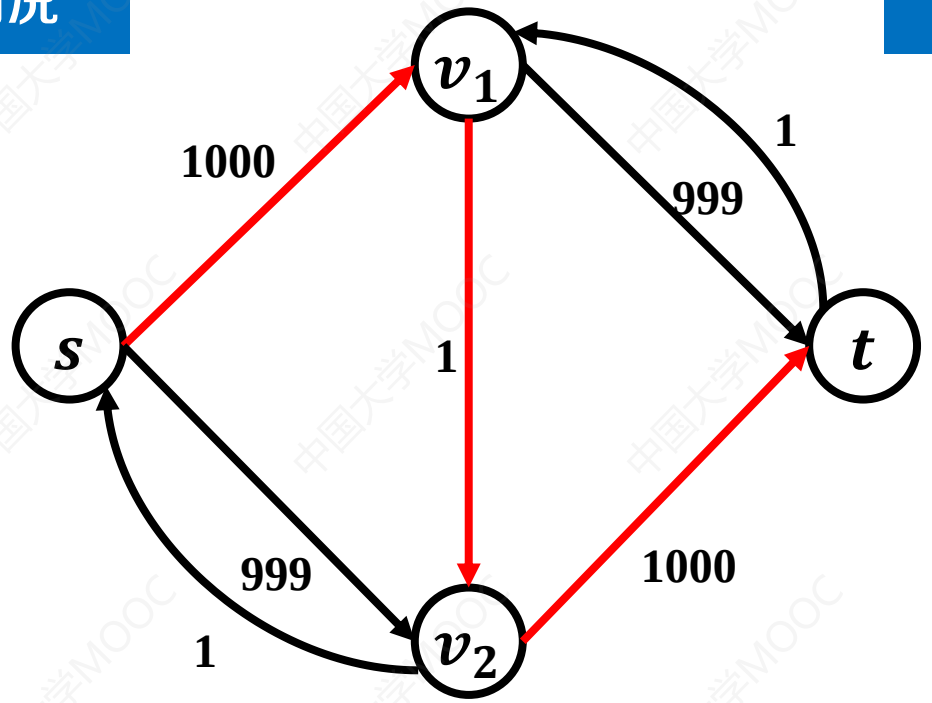
最优情况



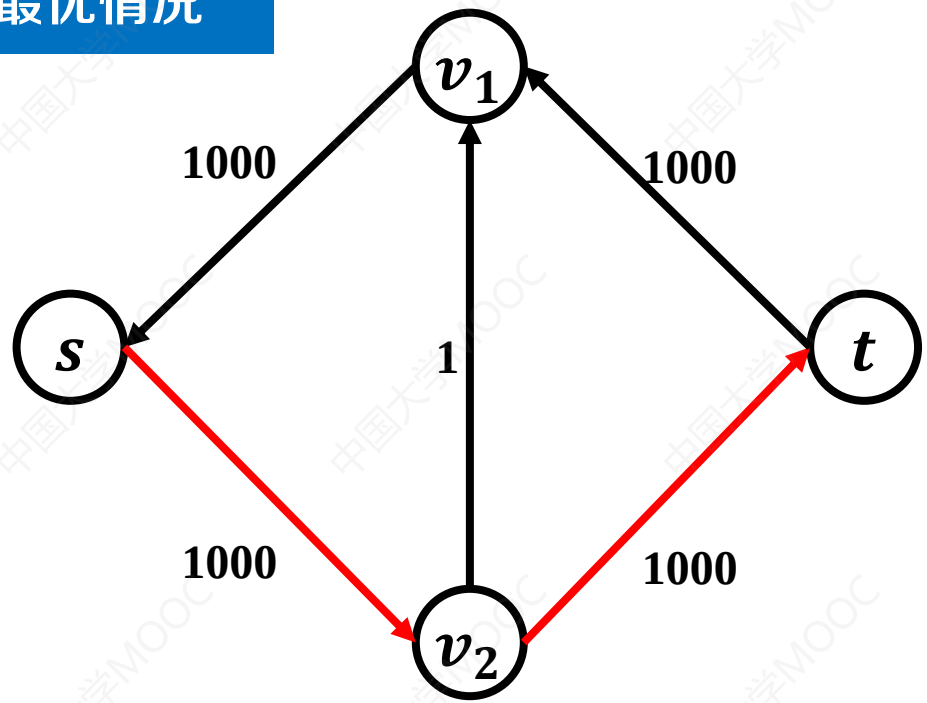
复杂度分析

- 运行时间依赖于增广过程

最坏情况



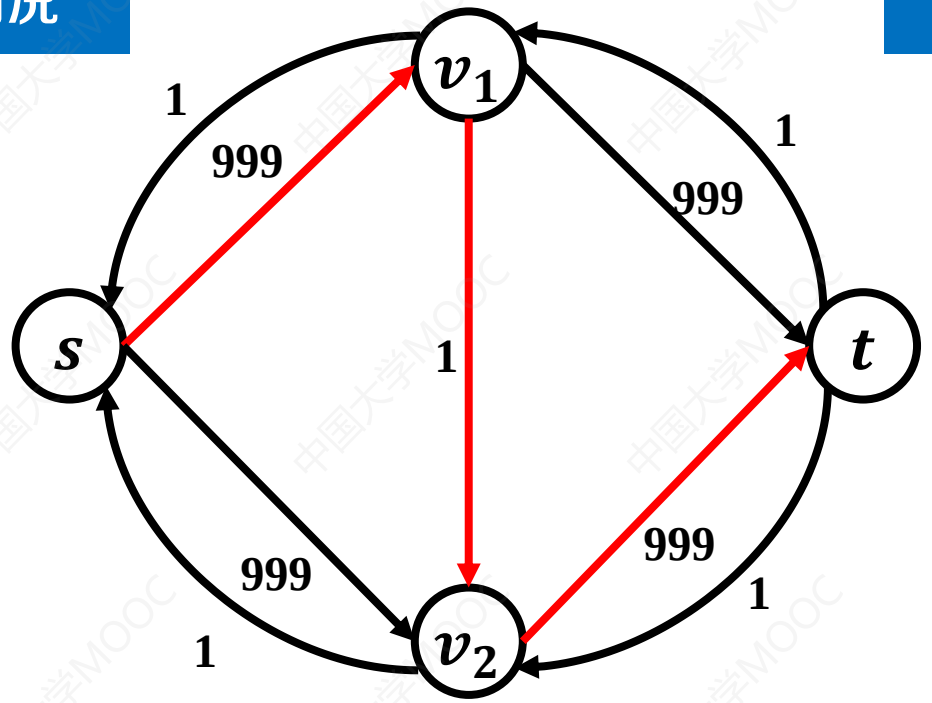
最优情况



复杂度分析

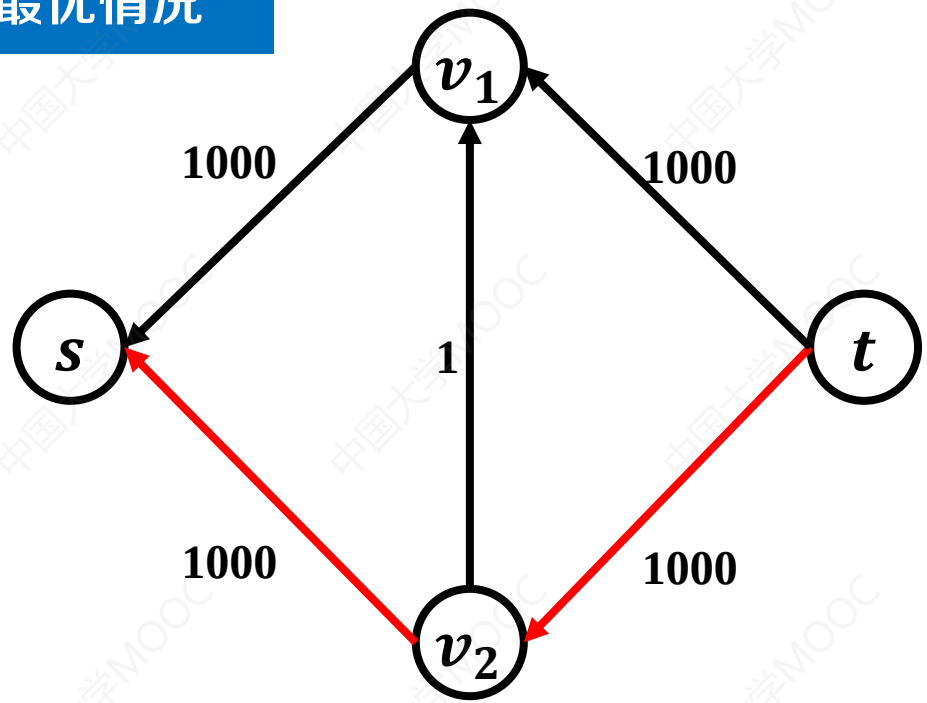
- 运行时间依赖于增广过程

最坏情况



每次增广总流量只增加1
需要执行2000次可完成

最优情况



每次增广总流量增加1000
仅需执行2次即可完成

提纲

问题背景

算法思想

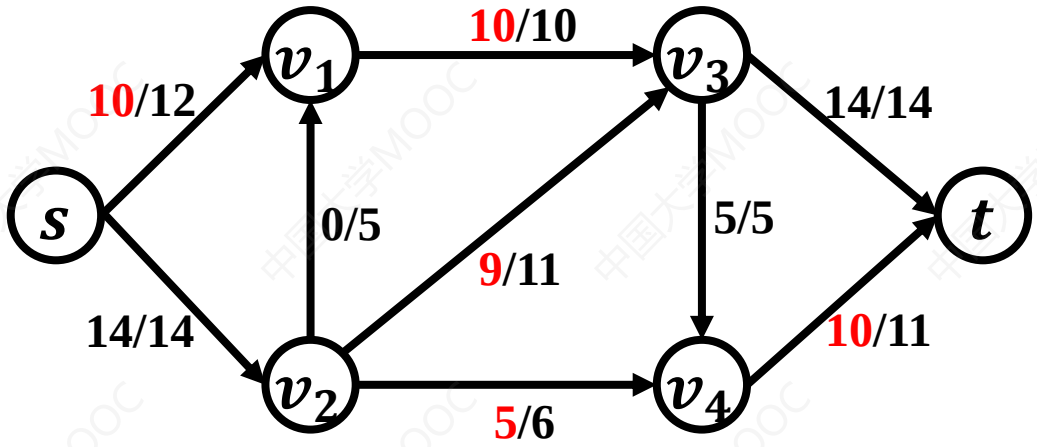
算法实例

算法分析

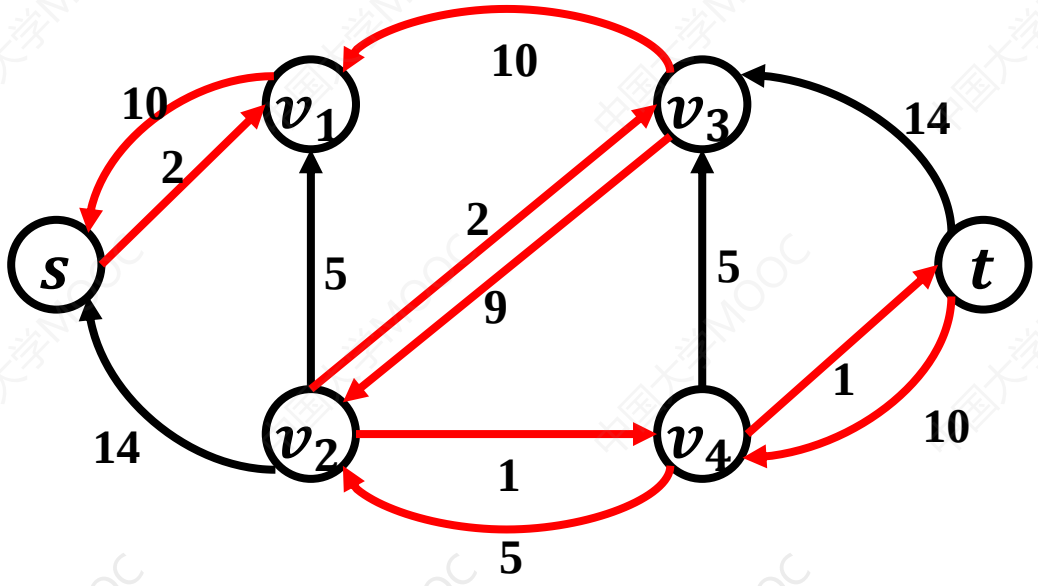
算法性质

正确性证明

流网络G



流网络G_f



问题：如何证明Ford-Fulkerson算法可获得最优解?

正确性证明

- 受启发于最大二分匹配证明
  - M是最大匹配⇔二分图G中无M的交替路径

正确性证明

- 受启发于最大二分匹配证明
  - M是最大匹配⇔二分图G中无M的交替路径

- 最大流证明思路
  - f是最大流⇔残存网络G_f中无增广路径
    - 充分性：f是最大流⇒残存网络G_f中无增广路径
    - 必要性：f是最大流⇐残存网络G_f中无增广路径

正确性证明

- f是最大流⇔残存网络G_f中无增广路径

  - 充分性：f是最大流⟹残存网络G_f中无增广路径

    - 反证：
    - 如果流f有增广路径p，则总流量$|f|$可被增加，f不是最大流，矛盾

正确性证明

- f是最大流$\Leftrightarrow$残存网络G_f中无增广路径
  - 必要性：f是最大流$\overset{?}{\Leftarrow}$残存网络G_f中无增广路径

直接证明困难

正确性证明

- f是最大流⇔残存网络G_f中无增广路径

  - **必要性：** f是最大流⇐?⇐?⇐残存网络G_f中无增广路径

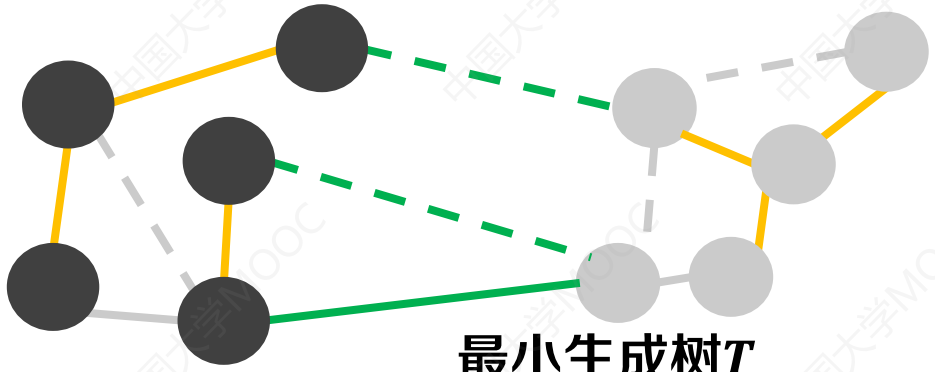
寻找最大流问题的上界

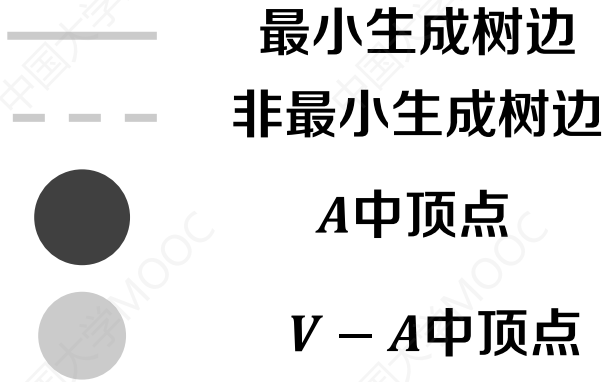
建立容量/流量和图结构的关系

引入割的概念，构造证明桥梁

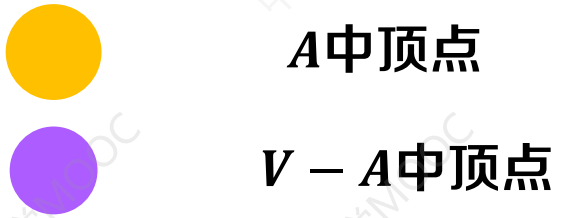
割(Cut)

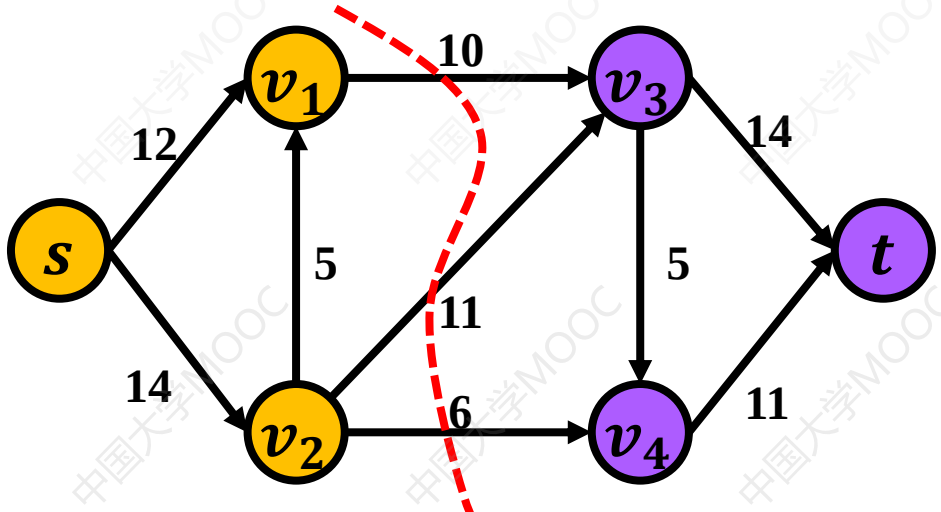
- 最小生成树的割
  - 图$G=<V,E>$是一个连通无向图，割$(A,V-A)$将图G的顶点集V划分为两部分





- 流网络的割
  - 图$G=<V,E,C>$是一个有向图，割$(A,V-A)$将图G的顶点集V划分为两部分
  - 其中源点$s \in A$，汇点$t \in V-A$



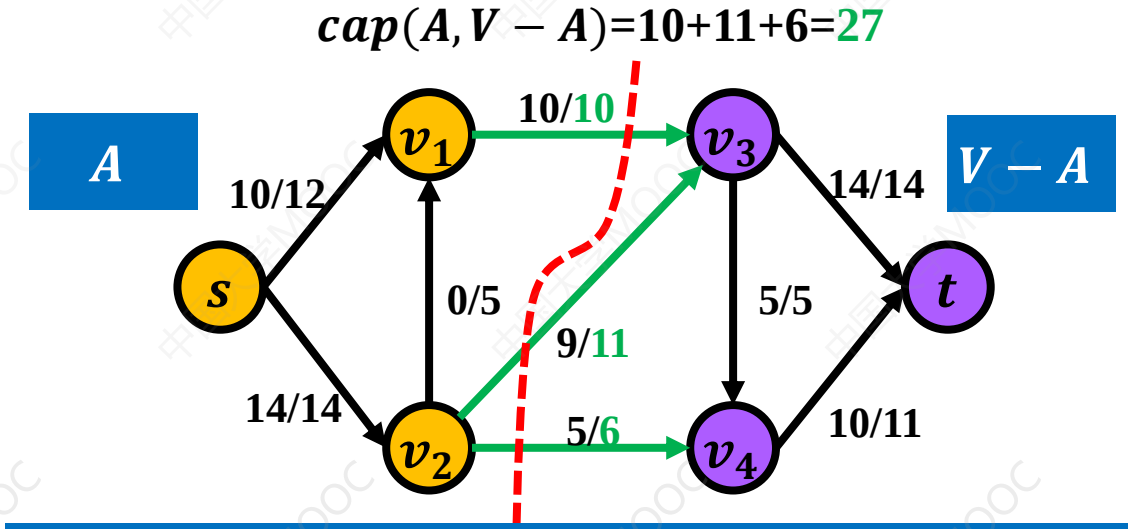


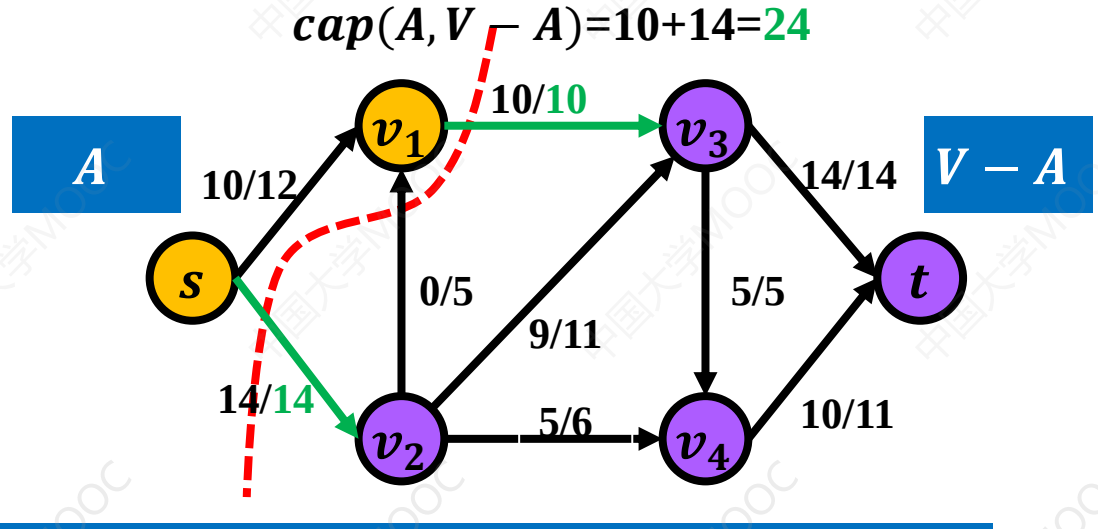
割的容量

- 横跨(Cross)
  - 给定割$(A, V-A)$和边(u, v)，$u \in A, v \in V-A$，称边(u, v)横跨割$(A, V-A)$

- 割的容量

$$cap(A, V-A) = \sum_{e\ out\ of\ A} c(e)$$



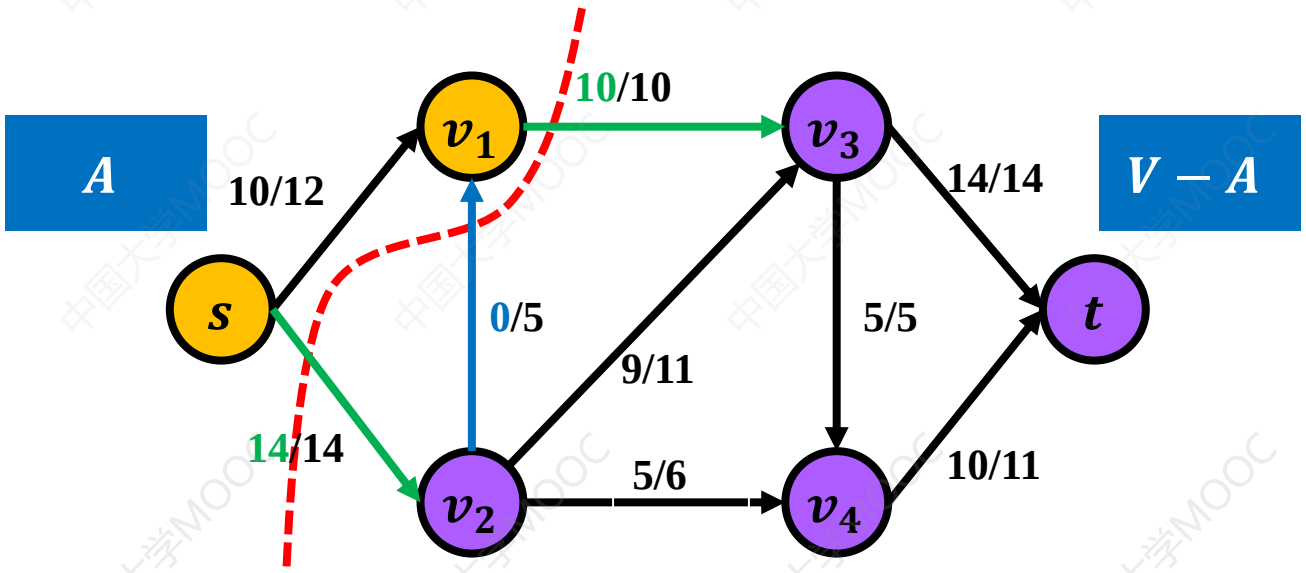


不同的割可能具有不同的容量

流值定理

- 给定割$(A, V-A)$和流f，那么

$$val(f) = \sum_{e\ out\ of\ A} f(e) - \sum_{e\ in\ to\ A} f(e)$$

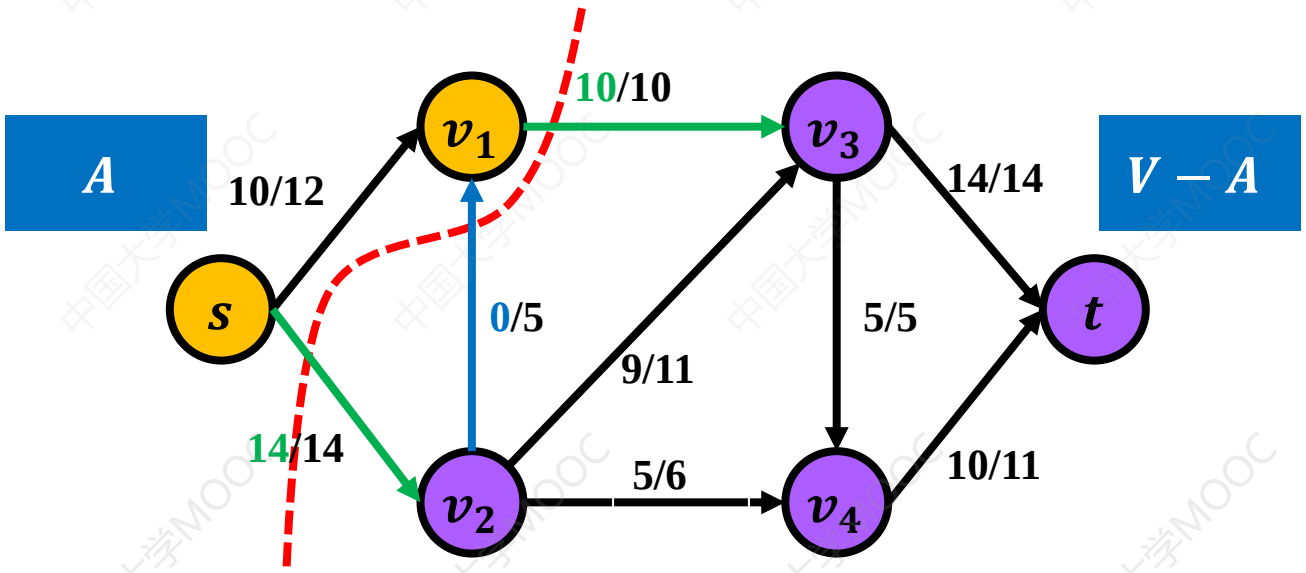


流值定理

- 给定割$(A, V-A)$和流f，那么

$$val(f) = \sum_{e\ out\ of\ A} f(e) - \sum_{e\ in\ to\ A} f(e)$$

流过割的流量=10+14-0



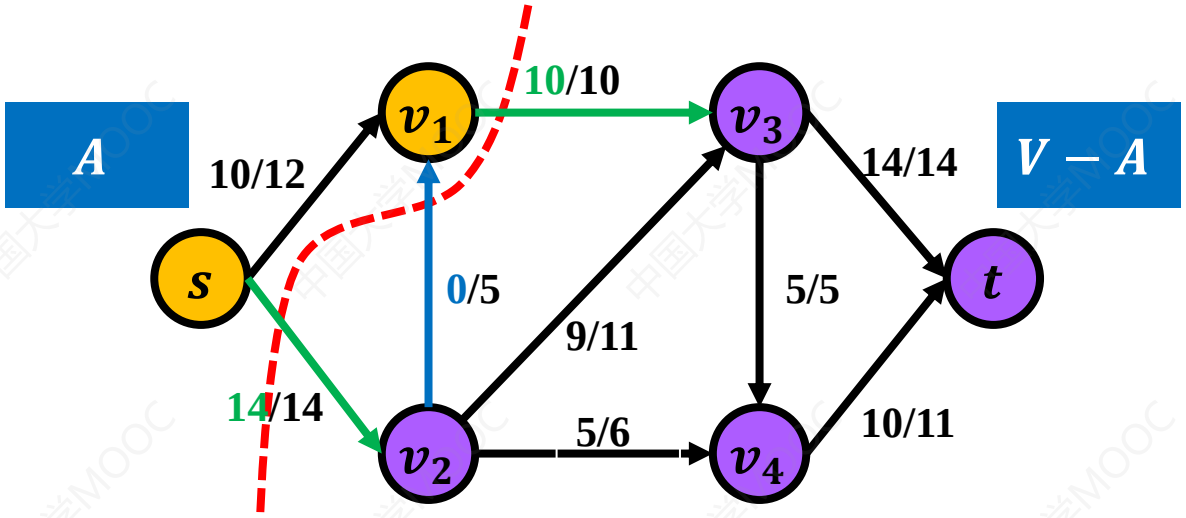
流值定理

- 给定割$(A, V-A)$和流f，那么

$$val(f) = \sum_{e\ out\ of\ A} f(e) - \sum_{e\ in\ to\ A} f(e)$$

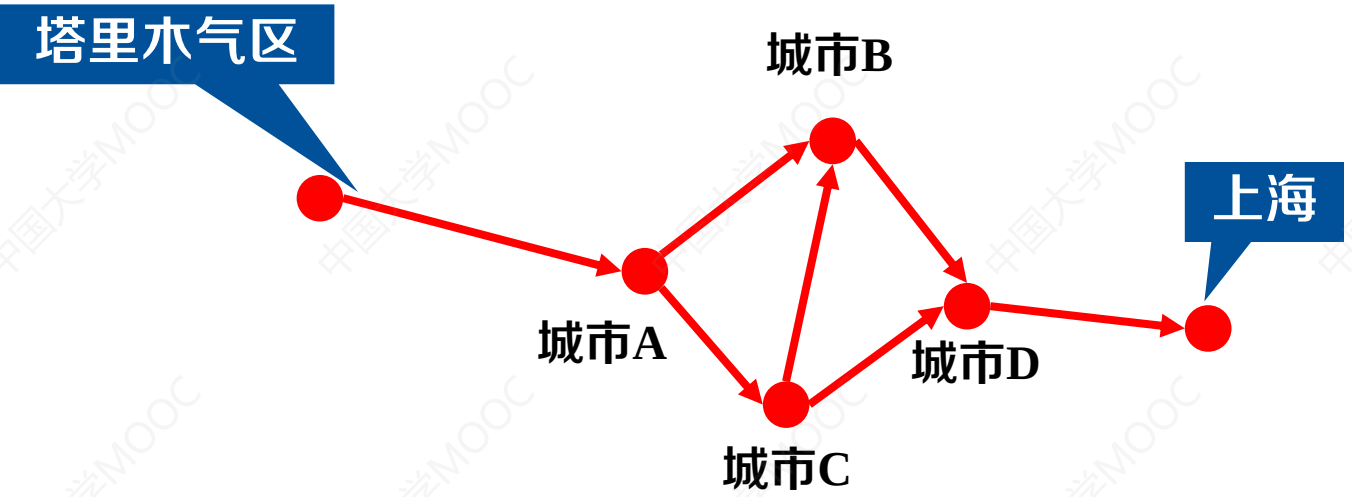
- 证明：

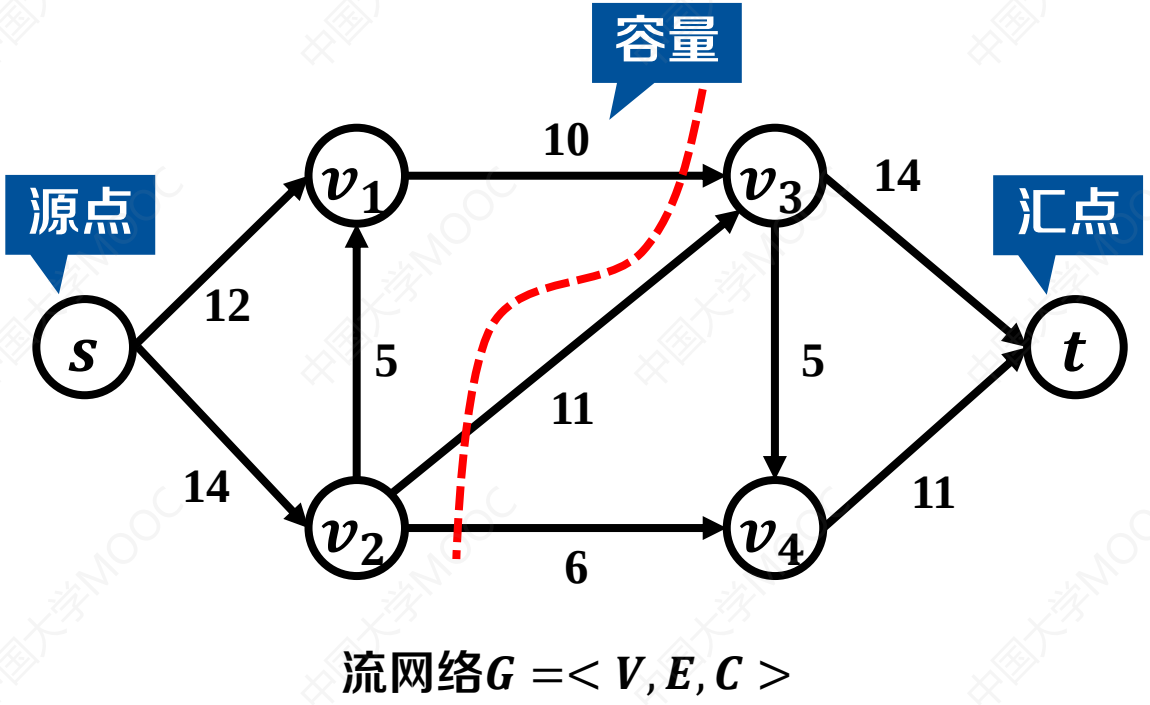
$$\begin{aligned} val(f) &= \sum_{e\ out\ of\ s} f(e) \\ &= \sum_{v \in A} \left(\sum_{e\ out\ of\ v} f(e) - \sum_{e\ in\ to\ v} f(e) \right) \\ &= \sum_{e\ out\ of\ A} f(e) - \sum_{e\ in\ to\ A} f(e) \end{aligned}$$



最小割

- 最大流：计算最大的天然气输送量
- 最小割：寻找天然气输送管道的瓶颈总容量





流网络$G =< V, E, C >$

最大流的流值 ≤ 最小割的容量（上界）

正确性证明

- f是最大流⇔残存网络G_f中无增广路径

  - **必要性：** f是最大流⇐？⇐？⇐残存网络G_f中无增广路径

引入**割**概念，构造证明桥梁

正确性证明

- f是最大流⇔残存网络G_f中无增广路径
  - 必要性：f是最大流$\overset{?}{\Leftarrow}$存在割是最大流的上界$\overset{?}{\Leftarrow}$残存网络G_f中无增广路径

正确性证明

- f是最大流⇔残存网络G_f中无增广路径
  - 必要性：f是最大流⇐存在割是最大流的上界⇐残存网络G_f中无增广路径

步骤1：存在割$(A, V-A)$使得$val(f) = cap(A, V-A)$ ⇐ 残存网络G_f中无增广路径

正确性证明

- f是最大流⇔残存网络G_f中无增广路径
  - 必要性：f是最大流⇐存在割是最大流的上界⇐残存网络G_f中无增广路径

    步骤1：存在割$(A, V-A)$使得$val(f) = cap(A, V-A)$ ⇐ 残存网络G_f中无增广路径

    步骤2：f是最大流⇐存在割$(A, V-A)$使得$val(f) = cap(A, V-A)$

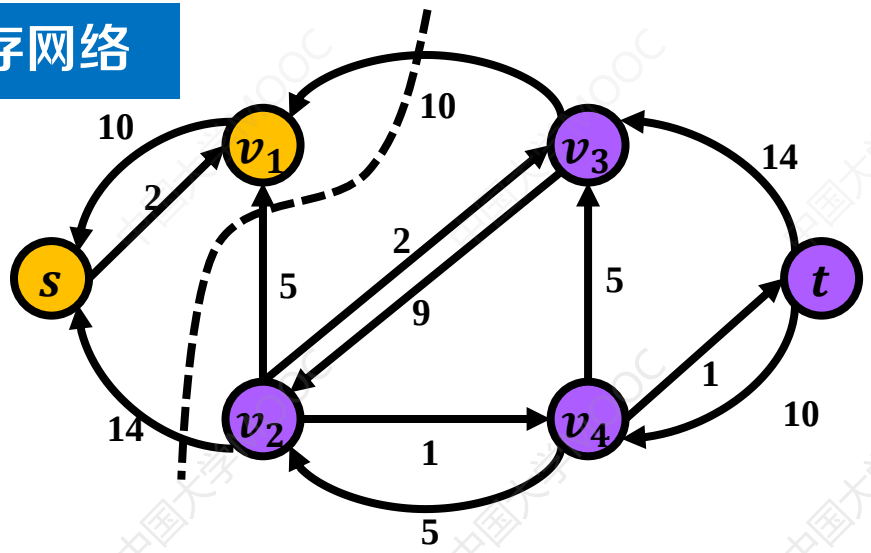
正确性证明

- f是最大流⇔残存网络G_f中无增广路径

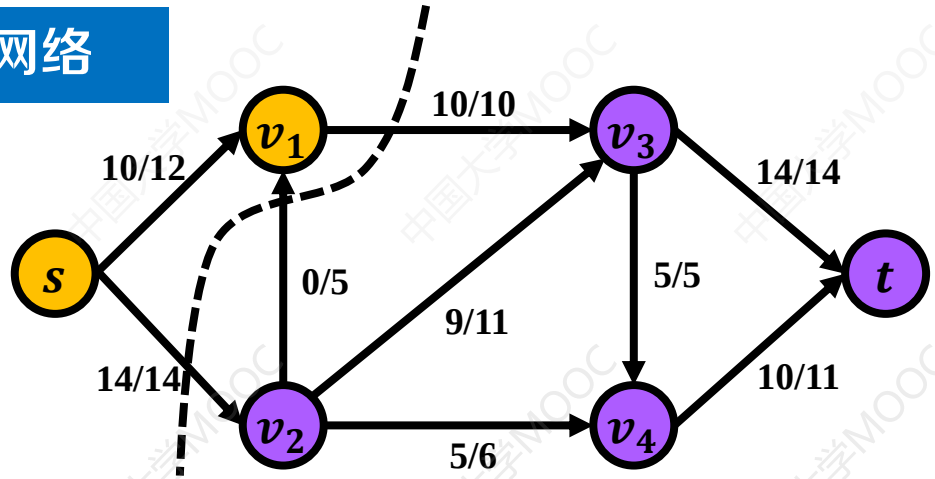
  步骤1：存在割$(A, V-A)$使得$val(f)=cap(A, V-A)$ ⇐ 残存网络G_f中无增广路径

  - 证明
    - 无增广路径，s只可达部分顶点，依据顶点可达性，划分形成割$(A, V-A)$

残存网络



流网络



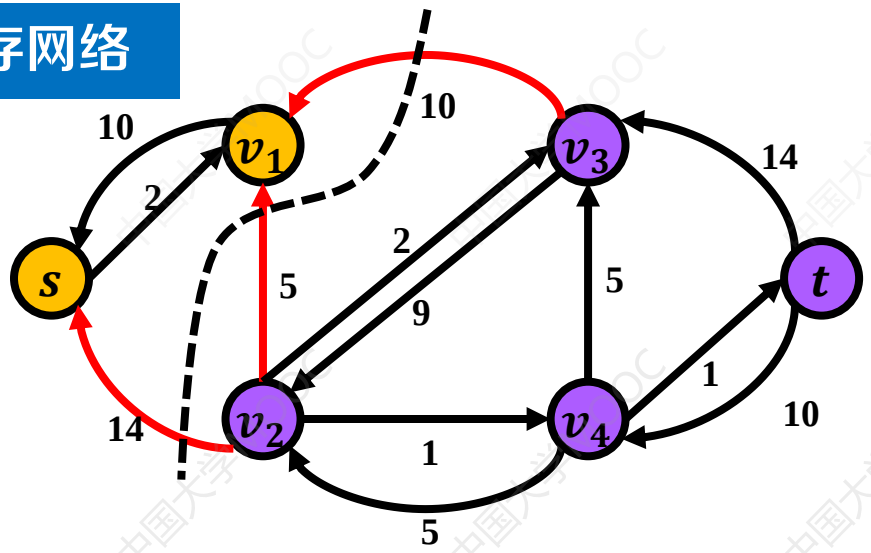
正确性证明

- f是最大流⇔残存网络G_f中无增广路径

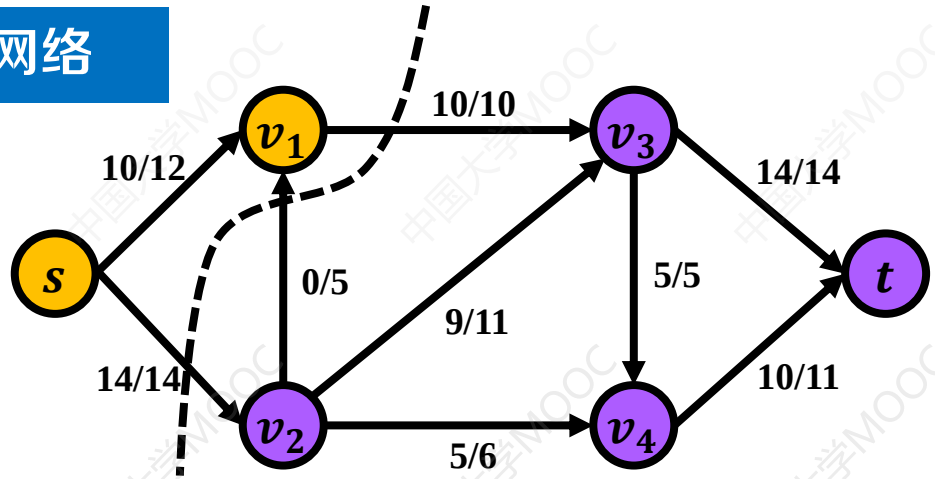
  步骤1：存在割$(A, V-A)$使得$val(f)=cap(A, V-A)$ ⇐ 残存网络G_f中无增广路径

  - 证明
    - 无增广路径，s只可达部分顶点，依据顶点可达性，划分形成割$(A, V-A)$
    - 残存网络中，因为s不可达$V-A$中的点，所以横跨割的残存边一定从$V-A$指向A

残存网络



流网络



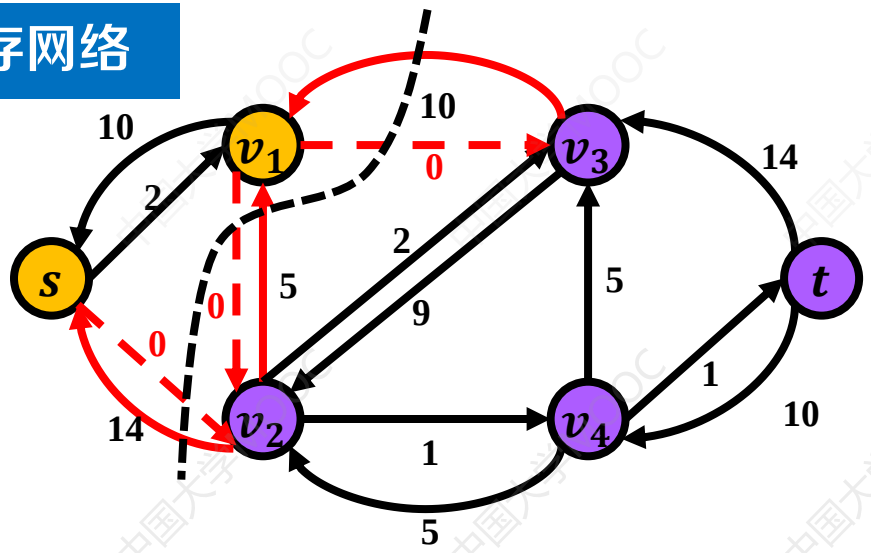
正确性证明

- f是最大流⇔残存网络G_f中无增广路径

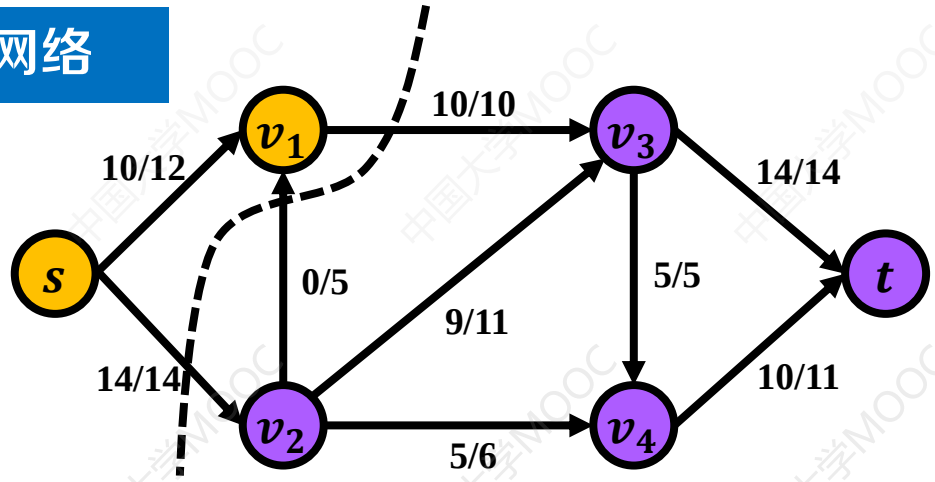
  步骤1：存在割$(A, V-A)$使得$val(f)=cap(A, V-A)$ ⇐ 残存网络G_f中无增广路径

  - 证明
    - 无增广路径，s只可达部分顶点，依据顶点可达性，划分形成割$(A, V-A)$
    - 残存网络中，因为s不可达$V-A$中的点，所以横跨割的残存边一定从$V-A$指向A
    - 因此，一组横跨隔的残存边只剩一条，说明$f(e)=0$或$f(e)=c(e)$

残存网络



流网络



正确性证明

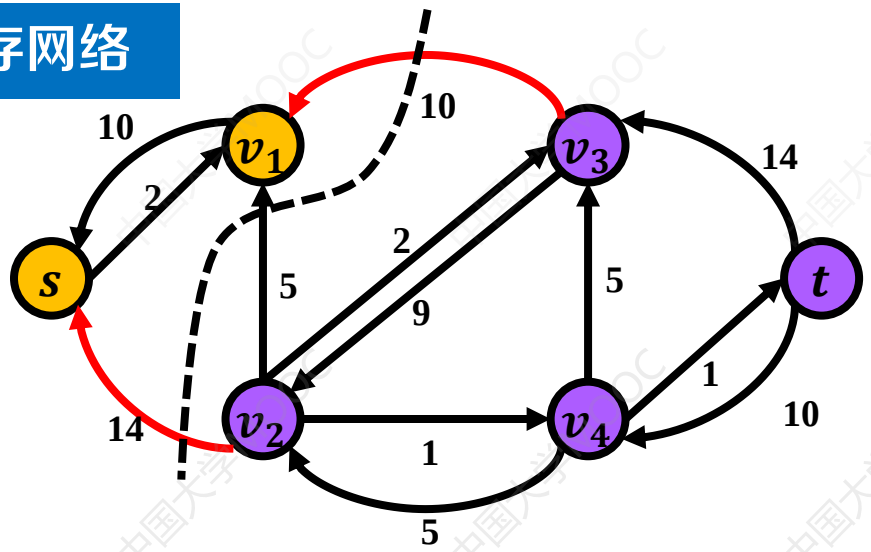
- f是最大流⇔残存网络G_f中无增广路径

  步骤1：存在割$(A, V-A)$使得$val(f)=cap(A, V-A)$ ⇐ 残存网络G_f中无增广路径

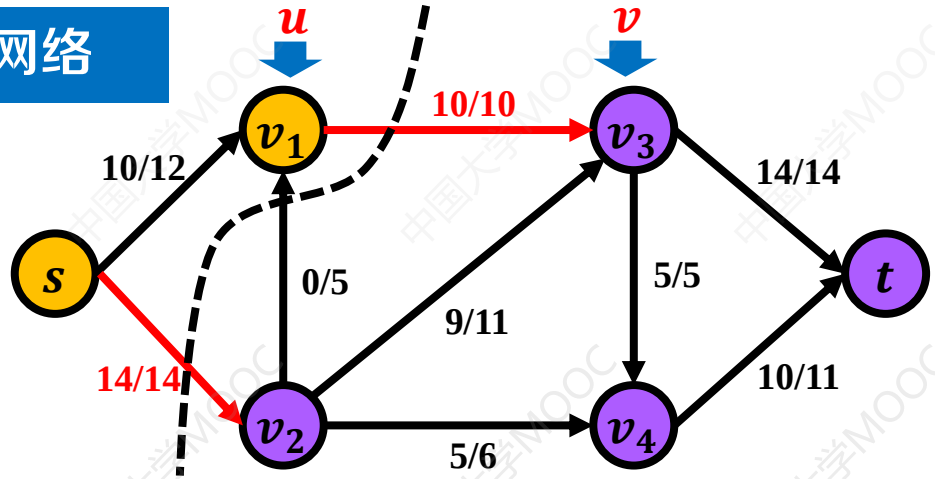
  - 证明
    - 无增广路径，s只可达部分顶点，依据顶点可达性，划分形成割$(A, V-A)$
    - 残存网络中，因为s不可达$V-A$中的点，所以横跨割的残存边一定从$V-A$指向A
    - 因此，一组横跨隔的残存边只剩一条，说明$f(e)=0$或$f(e)=c(e)$
    - 流网络中，若$(u,v)\in E$，则必有$f(u,v)=c(u,v)$，否则s可达顶点v，$v\in V-A$

$u\in A,$
$v\in V-A$

残存网络



流网络



正确性证明

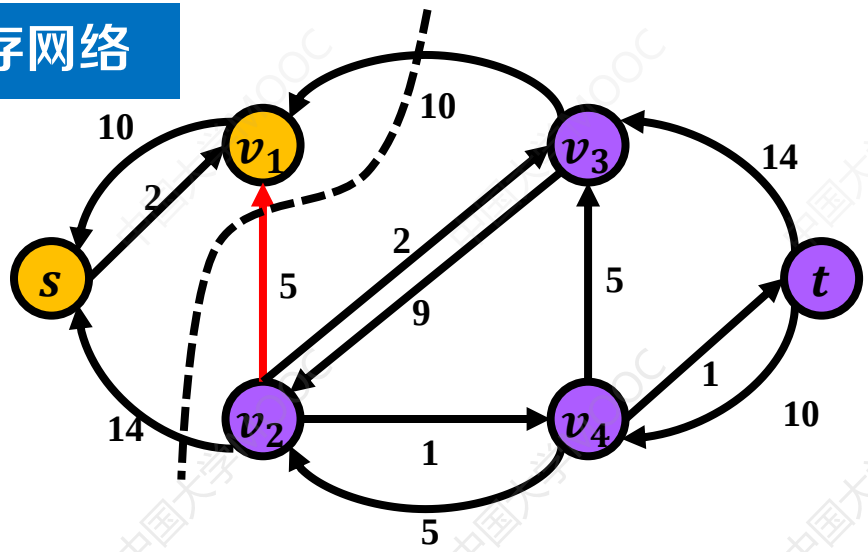
- f是最大流⇔残存网络G_f中无增广路径

  步骤1：存在割$(A, V-A)$使得$val(f)=cap(A, V-A)$ ⇐ 残存网络G_f中无增广路径

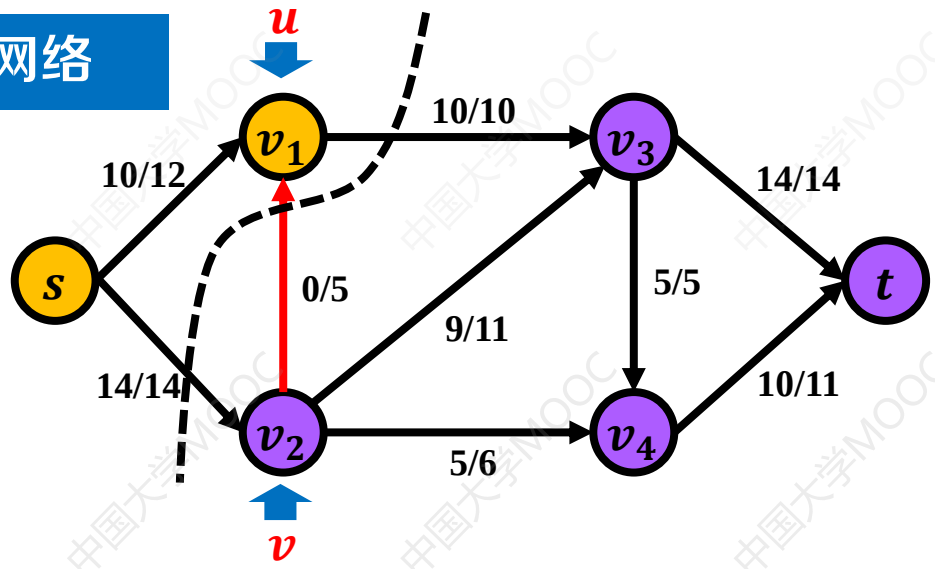
  - 证明
    - 无增广路径，s只可达部分顶点，依据顶点可达性，划分形成割$(A, V-A)$
    - 残存网络中，因为s不可达$V-A$中的点，所以横跨割的残存边一定从$V-A$指向A
    - 因此，一组横跨隔的残存边只剩一条，说明$f(e)=0$或$f(e)=c(e)$
    - 流网络中，若$(u,v)\in E$，则必有$f(u,v)=c(u,v)$，否则s可达顶点v，$v\in V-A$
    - 流网络中，若$(v,u)\in E$，则必有$f(v,u)=0$，否则$c_f(u,v)=f(v,u)>0$，顶点$v\in A$

$u\in A$,
$v\in V-A$

残存网络



流网络



正确性证明

- f是最大流⇔残存网络G_f中无增广路径

  步骤1：存在割$(A, V-A)$使得$val(f) = cap(A, V-A)$ ⇐ 残存网络G_f中无增广路径

  - 证明

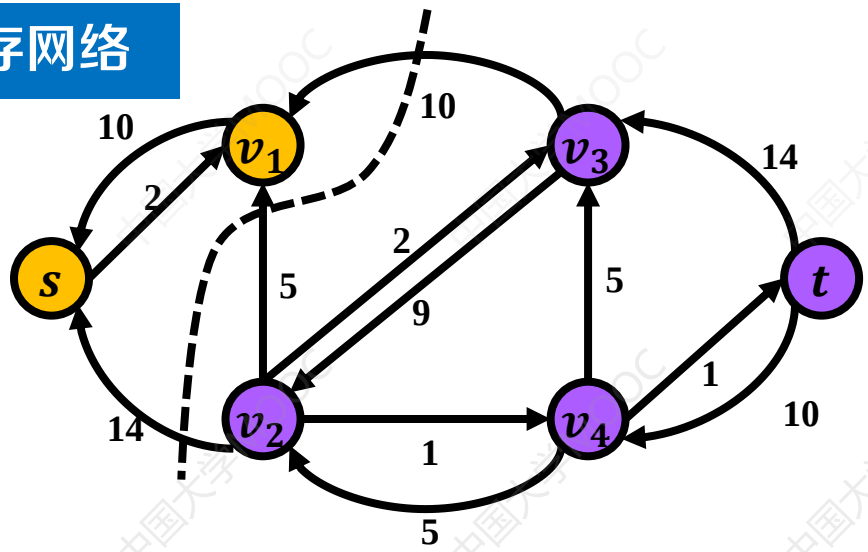
$u \in A,$
$v \in V - A$

  - 流网络中，若$(u,v) \in E$，则必有$f(u,v) = c(u,v)$
  - 流网络中，若$(v,u) \in E$，则必有$f(v,u) = 0$

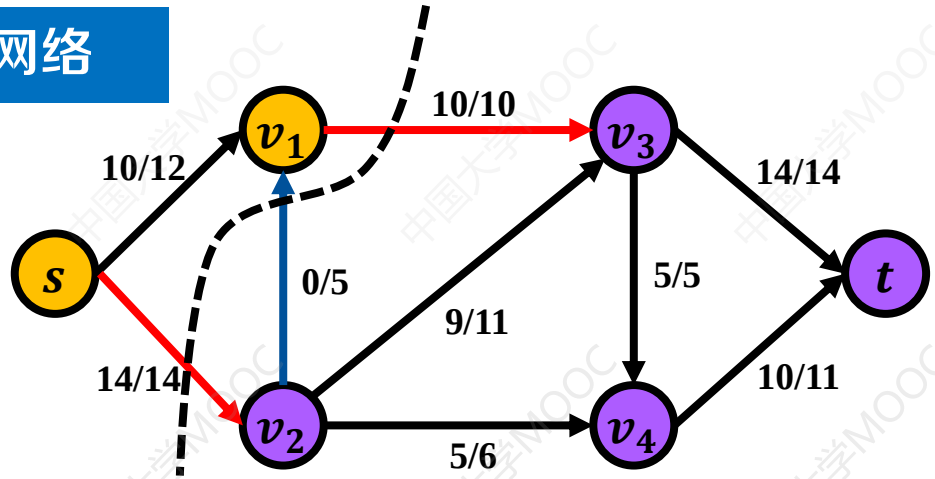
$$val(f) = \sum_{e\ out\ of\ A} f(e) - \sum_{e\ in\ to\ A} f(e)$$

流值定理

残存网络



流网络



正确性证明

- f是最大流⇔残存网络G_f中无增广路径

  步骤1：存在割$(A, V-A)$使得$val(f) = cap(A, V-A)$ ⇐ 残存网络G_f中无增广路径

  - 证明

    $u \in A,$ $v \in V-A$

    - 流网络中，若$(u,v) \in E$，则必有$f(u,v) = c(u,v)$
    - 流网络中，若$(v,u) \in E$，则必有$f(v,u) = 0$

    $f(e) = 0$

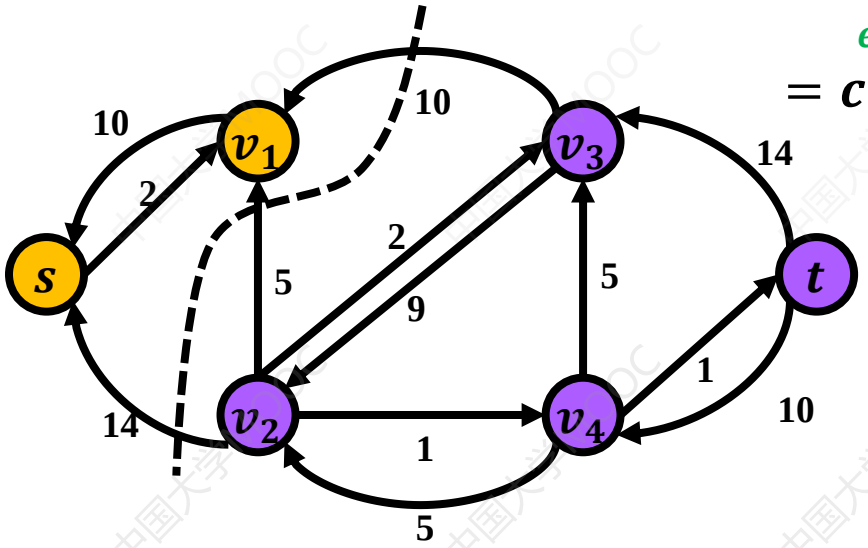
    $f(e) = c(e)$

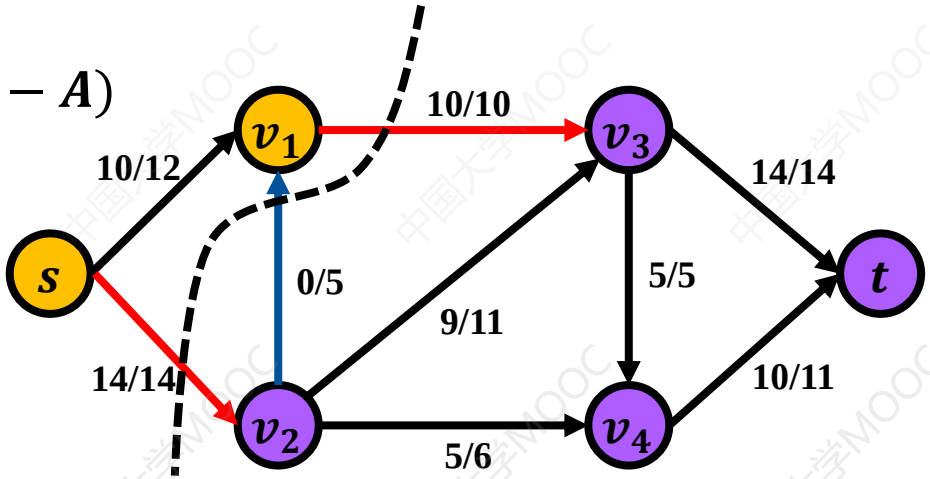
$$val(f) = \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e)$$

流值定理

$$= \sum_{e \text{ out of } A} c(e)$$

$$= cap(A, V-A)$$





正确性证明

- f是最大流⇔残存网络G_f中无增广路径
  - 步骤2：f是最大流⇐存在割$(A, V-A)$使得$val(f) = cap(A, V-A)$
  - 弱对偶性：对于任意流f与任意割$(A, V-A)$，$val(f) \leq cap(A, V-A)$
    - 证明：

$$
\begin{aligned}
val(f) &= \sum_{e\ out\ of\ A} f(e) - \sum_{e\ in\ to\ A} f(e) \\
&\leq \sum_{e\ out\ of\ A} c(e) \\
&= cap(A, V-A)
\end{aligned}
$$

流值定理

每条边流量必小于容量

弱对偶性(Weak Duality)：最大流的流值小于等于任意割容量

正确性证明

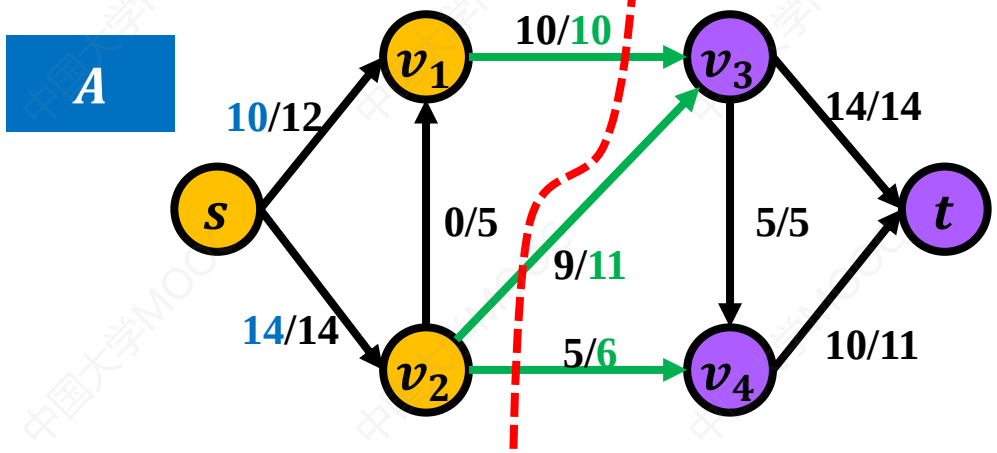
- f是最大流⇔残存网络G_f中无增广路径
  - 步骤2：f是最大流⇐存在割$(A, V-A)$使得$val(f) = cap(A, V-A)$
  - 强对偶性：最大流可取到其上界（最小割）
    - 证明：
      - 设割$(A, V-A)$使得$cap(A, V-A) = val(f)$
      - 根据弱对偶性，对于任意流f'，有$val(f) = cap(A, V-A) \geq val(f')$
      - 流f是最大流

弱对偶性

强对偶性(Strong Duality)：最大流的流值等于最小割的容量

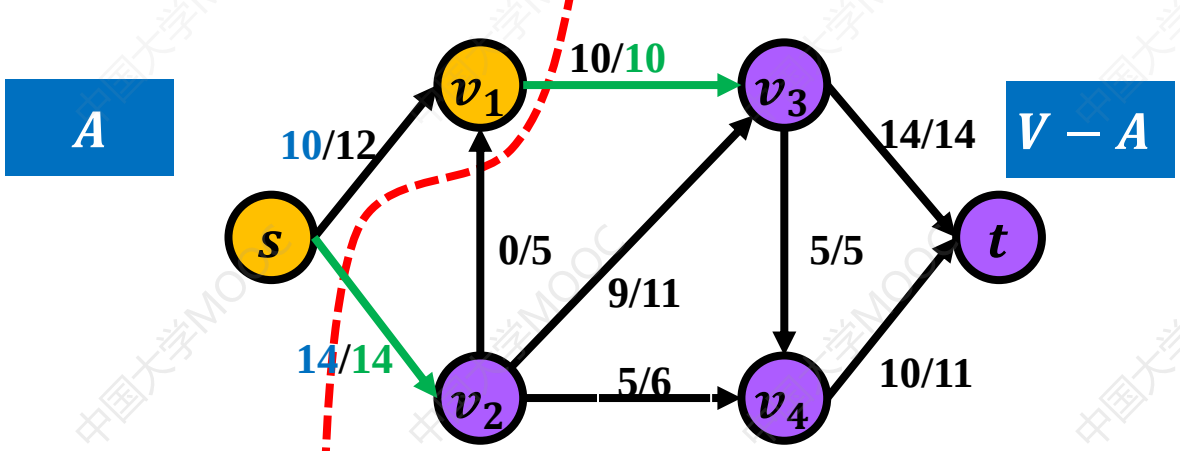
对偶性

- 弱对偶性(Weak Duality)
  - 对于任意流f与任意割$(A, V-A)$，$val(f) \leq cap(A, V-A)$



$cap(S, V-S)$=10+11+6=27
$val(f)$=10+14=24

- 强对偶性(Strong Duality)
  - f是最大流，存在割（最小割）$(A, V-A)$使得$val(f)$= $cap(A, V-A)$



$cap(S, V-S)$=10+14=24
$val(f)$=10+14=24

正确性证明

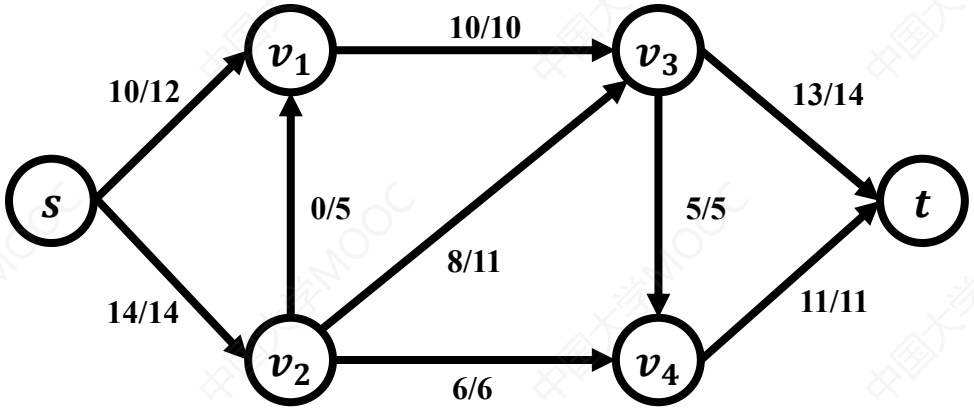
- 二分图
  - M是最大匹配⇔二分图G中无增广路径

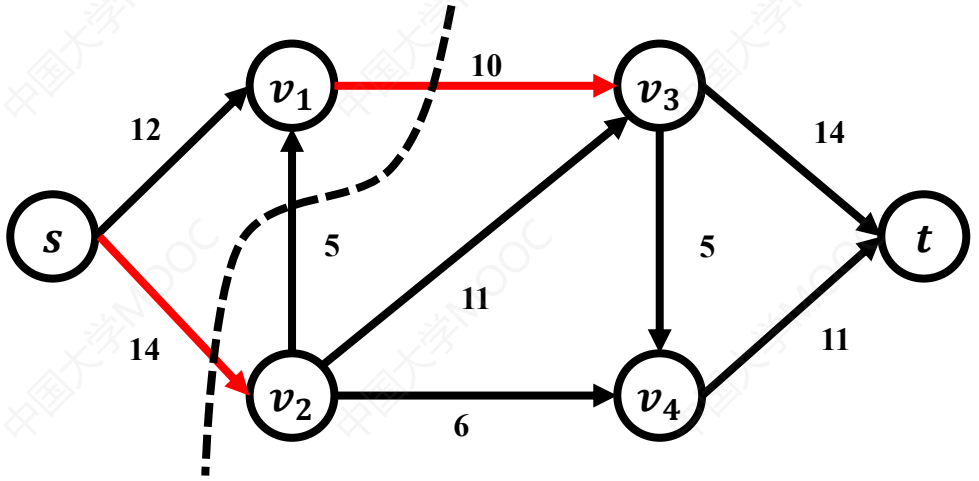
- 最大流
  - f是最大流⇔残存网络G_f中无增广路径
    - 充分性：f是最大流⇒残存网络G_f中无增广路径
    - 必要性：f是最大流⇐存在割是最大流的上界⇐残存网络G_f中无增广路径

回顾以上证明过程，得到最大流最小割定理

最大流最小割定理

- f为流网络$G =< V, E, C >$中一个流，该流网络源点为s，汇点为t，以下三条件相互等价
  - (i) 存在割$(A, V-A)$使得$val(f) = cap(A, V-A)$
  - (ii) 流f是最大流
  - (iii) 流f没有增广路径



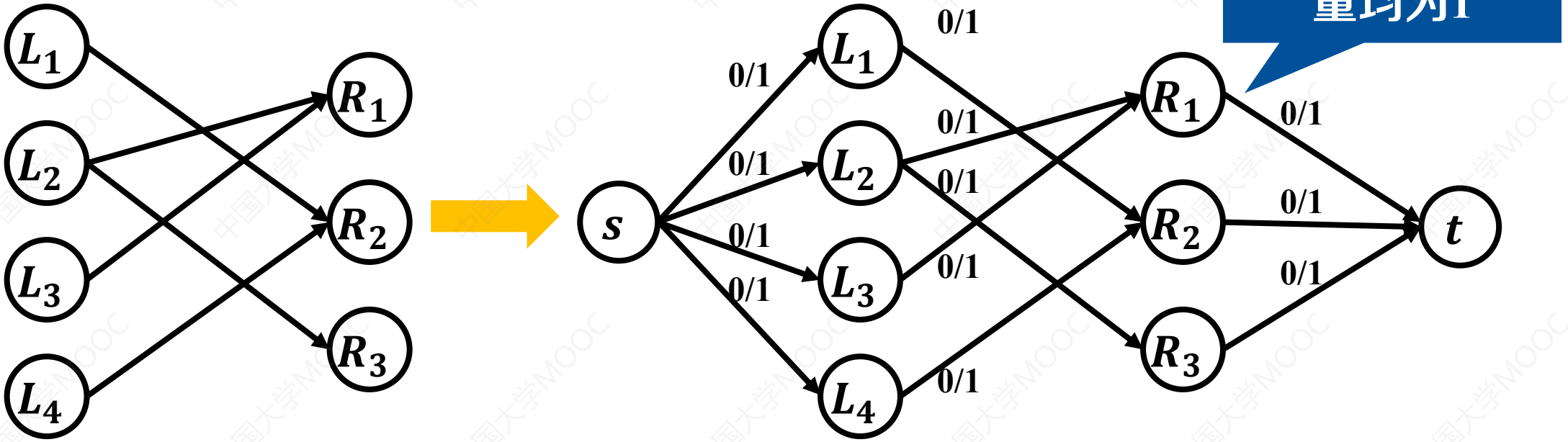


最大流
$|f^*| = 24$

最小割
$cap(A, V-A) = 24$

小结

- **Ford-Fulkerson算法可求解最大二分匹配问题**
  - **求解思想：将二分图转化为流网络**



小结

- 最大二分匹配问题和最大流问题比较

	最大二分匹配	最大流
核心思想	寻找交替路径，增加匹配数	寻找增广路径，扩充流量
图的结构	二分图	流网络
边的性质	匹配边、非匹配边	流量、容量、残存容量
增益过程	找到一条交替路径， 匹配数增加一	找到一条增广路径， 流量增加路径的最小剩余容量