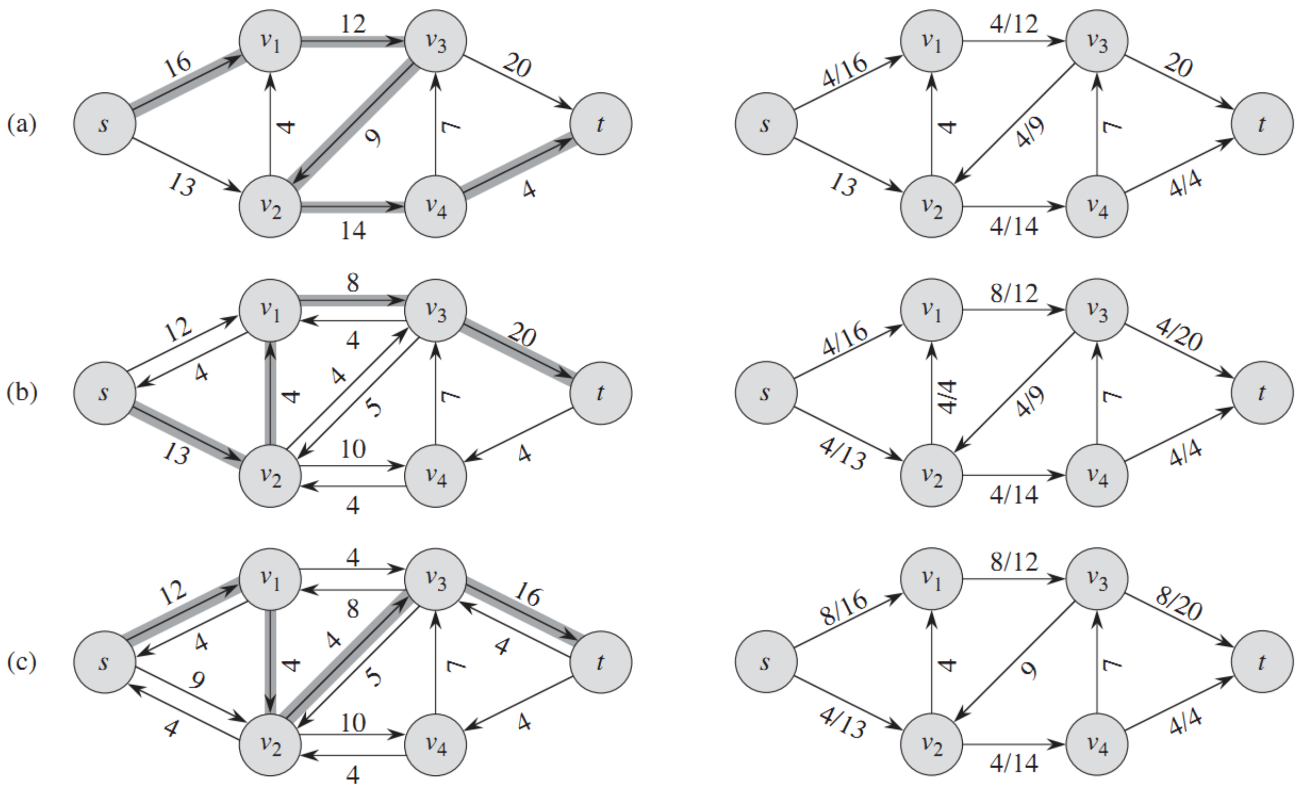
算法设计与分析

网络流



主讲： 朱东杰 博士、硕导
地点： M楼305
电话/微信： 18953856806
Email： zhudongjie@hit.edu.cn

网络流

- 学习要点
- 理解网络与网络流的基本概念
- 网络最大流问题
- 掌握网络最大流的增广路算法（Ford-Fulkerson）
- 网络最小费用流算法

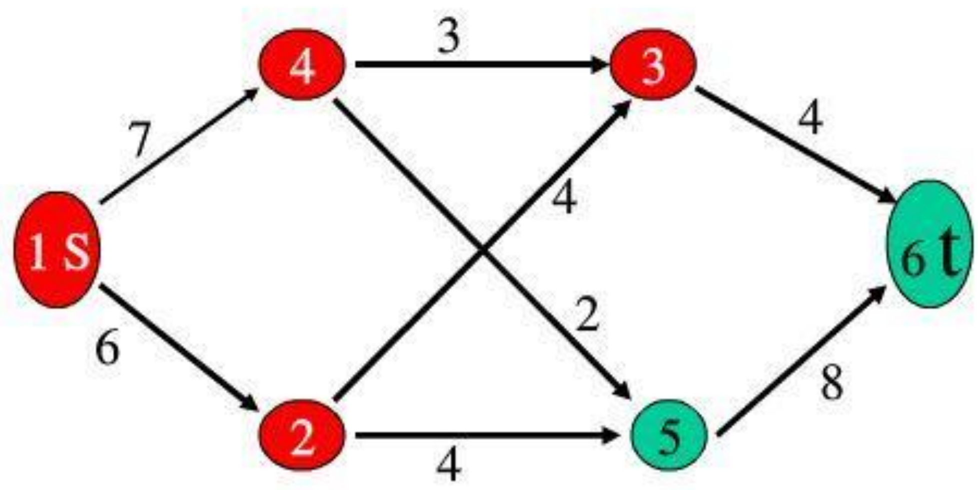
- 了解网络流其他算法
  - 了解网络最大流的其他算法
  - 了解网络最小费用流的其他算法

网络流

网络流图是一张只有一个源点和汇点的有向图，而最大流就是求源点到汇点间的最大水流量，下图的问题就是一个最基本，经典的最大流问题

实例：

有一自来水管道输送系统，起点是S，目标是T，途中经过的管道都有一个最大的容量。



- 问题：问从S到T的最大水流量是多少？

网络流

- **1 基本概念和术语**
- **（1） 网络**
- G是一个简单有向图，G=(V,E)，V={1，2，…，n}。
- 在V中指定一个顶点s，称为**源**和另一个顶点t，称为**汇**。
- 有向图G的每一条边(v,w)∈E，对应有一个值cap(v,w)≥0，称为边的**容量**。
- 这样的有向图G称作一个**网络**。
- **（2） 网络流**
- 网络上的**流**是定义在网络的边集合E上的一个非负函数flow={flow(v,w)}，并称flow(v,w)为边(v,w)上的**流量**。

网络流概念

- 容量：c(u,v)。表示边 <u,v>最大可以承载的流量。
- 流量：f(u,v)。表示边 <u,v>中已经有多少流量。
- 残量：r(u,v)。表示边 <u,v>还可以走多少的流量会达到饱和。$r = c - f$。
- 容量限制（Capacity Constraint）：对所有顶点对 $u, v \in V$，要求 $f(u, v) \le c(u, v)$。
- 反对称性（Skew Symmetry）：对所有顶点对 $u, v \in V$，要求 $f(u, v) = - f(v, u)$。
- 流守恒性（Flow Conservation）：对所有顶点对 $u \in V - \{s, t\}$，要求 $\Sigma v \in V f(u, v) = 0$。

网络流概念

- （1） 边流
- 对于网络G的一个给定的可行流flow，将网络中满足flow(v,w)=cap(v,w)的边称为**饱和边**；flow(v,w)<cap(v,w)的边称为**非饱和边**；flow(v,w)=0的边称为**零流边**；flow(v,w)>0的边称为**非零流边**。当边(v,w)既不是一条零流边也不是一条饱和边时，称为**弱流边**。
- （2） 最大流
- 最大流问题即求网络G的一个可行流flow，使其流量f达到最大。即flow满足：
- 0≤flow(v,w)≤cap(v,w)，(v,w)∈E；且

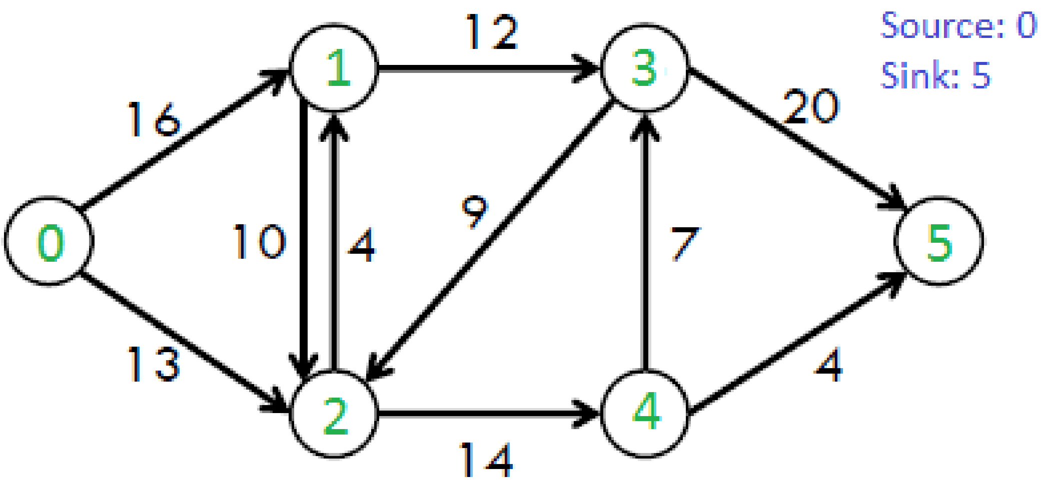
$$\sum flow(v,w) - \sum flow(w,v) = \begin{cases} f & v = s \\ 0 & v \neq s,t \\ -f & v = t \end{cases}$$

- （3） 流的费用
- 在实际应用中，与网络流有关的问题，不仅涉及流量，而且还有费用的因素。此时网络的每一条边(v,w)除了给定容量cap(v,w)外，还定义了一个单位流量费用cost(v,w)。对于网络中一个给定的流flow，其费用定义为：

$$\mathrm{cost}(flow) = \sum_{(v,w)\in E} \mathrm{cost}(v,w) \times flow(v,w)$$

最大流问题

- 最大流问题（Maximum-flow problem）中，给出源点 s 和汇点 t 的流网络 G，希望找出从 s 到 t 的最大值流。
- 满足流网络的性质的实际上定义了问题的限制：
  - 经过边的流不能超过边的容量；
  - 除了源点 s 和汇点 t，对于其它所有顶点，流入量与流出量要相等；

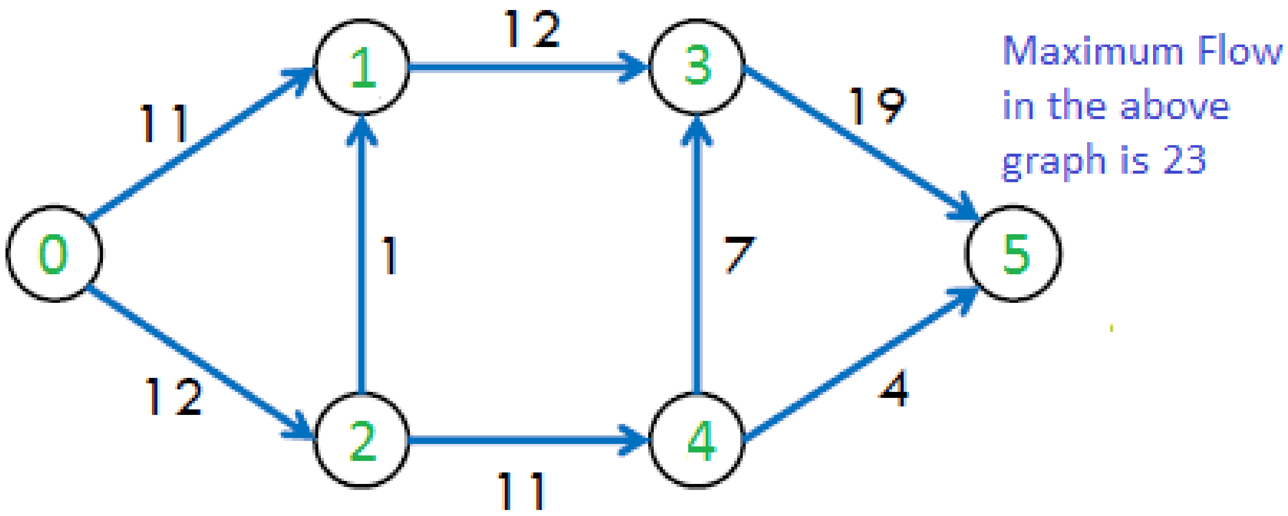


源点 s = 0，汇点 t = 5

为了理解最大流算法，可以形象的将网络中的各个边联想成水管。那么c和f表示水管内的最大水流量以及水流量。r表示这个水管还能够增加多少单位的水流量。而对于一条路径，其能够增加的最大水流量受限于该路径经过的所有水管中最小的r

最大流问题

源点 s = 0，汇点 t = 5



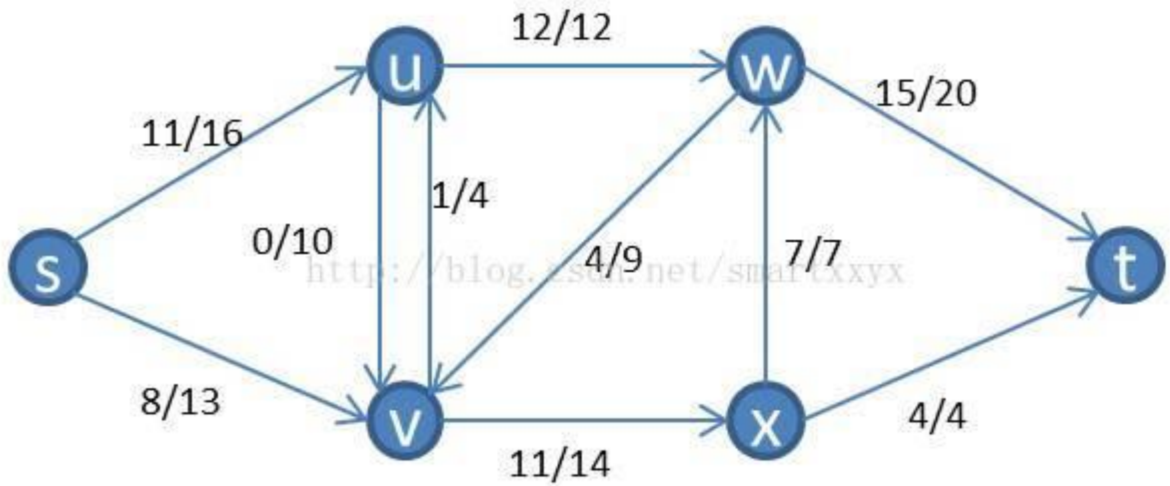
- Ford-Fulkerson方法的基本思想。首先需要了解的是Ford-Fulkerson是一种迭代的方法。开始时，对所有的u,v属于V，f(u,v)=0(这里f(u,v)代表u到v的边当前流量)，即初始状态时流的值为0。在每次迭代中，可以通过寻找一个“增广路径”来增加流值。增广路径可以看做是从源点s到汇点t之间的一条路径，沿该路径可以压入更多的流，从而增加流的值。反复进行这一过程，直到增广路径都被找出为止。
- **该方法依赖于三种重要思想：残留网络，增广路径和割**

最大流问题-残流网络

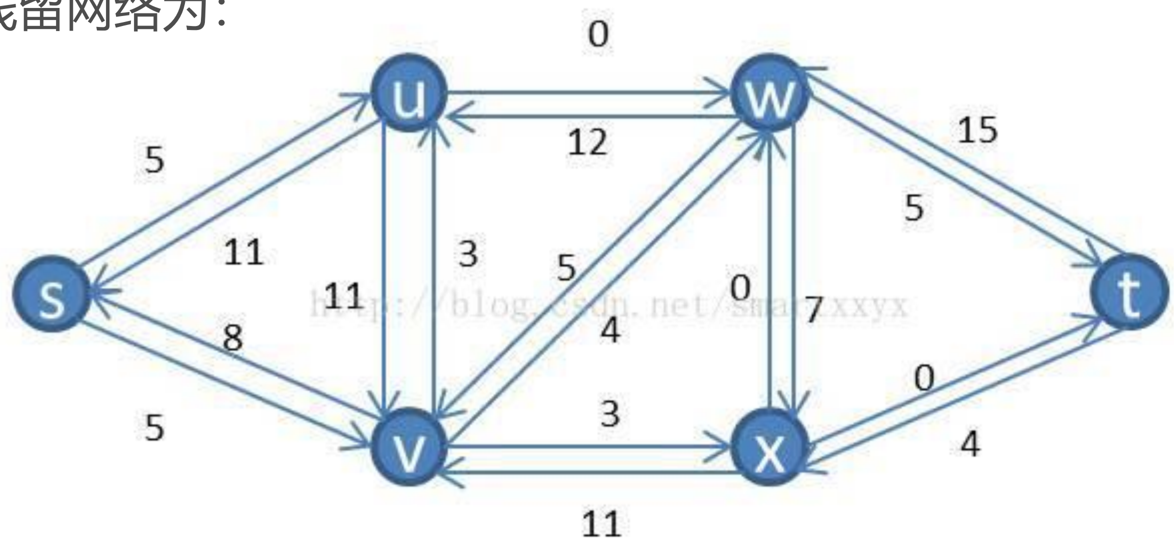
- 残留网络是指给定网络和一个流，其对应还可以容纳的流组成的网络。具体说来，就是假定一个网络G=（V，E），其源点s，汇点t。设f为G中的一个流，对应顶点u到顶点v的流。在不超过C（u，v）的条件下（C代表边容量），从u到v之间可以压入的额外网络流量，就是边（u，v）的残余容量（residual capacity）定义：**r（u，v）=c（u，v）-f（u，v）**
- 举个例子，假设（u，v）当前流量为3/4，那么就是说c（u，v）=4，f（u，v）=3，那么r（u，v）=1。我们知道，在网络流中还有这么一条规律。从u到v已经有了3个单位流量，那么从反方向上看，也就是从v到u就有了3个单位的残留网络，这时r（v，u）=3。可以这样理解，从u到v有3个单位流量，那么从v到u就有了将这3个单位流量的压回去的能力。

- 残流网络是设计与网络流有关算法的重要工具

最大流问题-残流网络

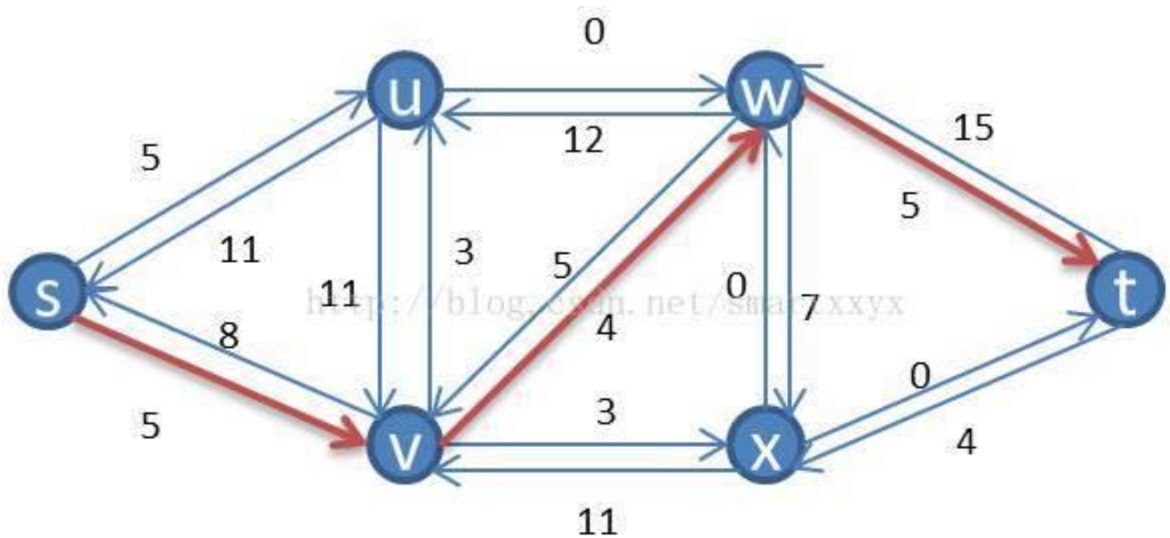


对应的残留网络为：



最大流问题-增广路径

- 已知一个流网络G和流f，增广路径p是其残留网络Gf中从s到t的一条简单路径。形象的理解为从s到t存在一条不违反边容量的路径，向这条路径压入流量，可以增加整个网络的流值。上面的残留网络中，存在这样一条增广路径：

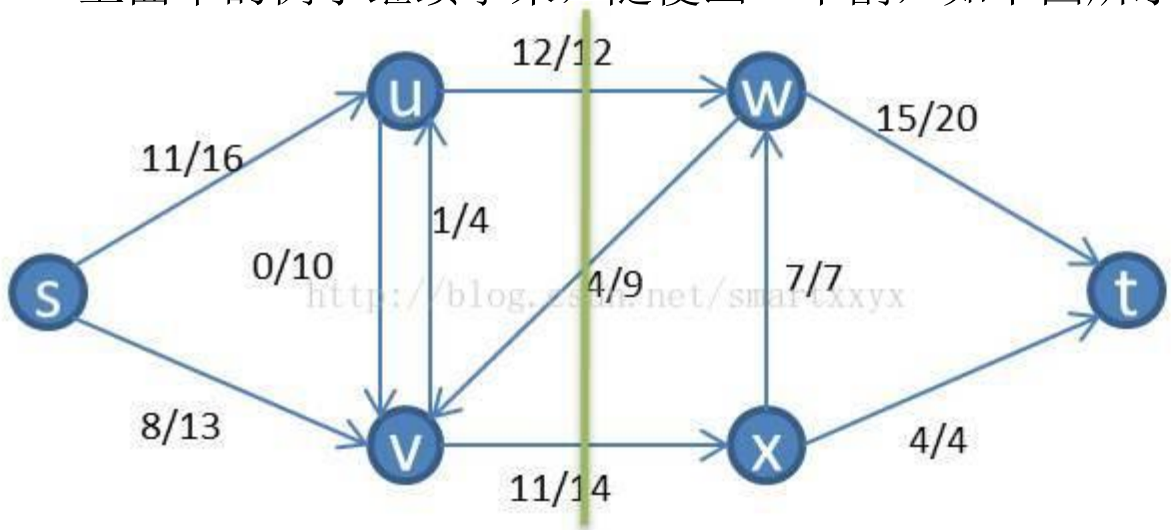


- 其可以压入4个单位的流量，压入后，我们得到一个新的流网络，其流量比原来的流网络要多4。这时我们继续在新的流网络上用同样的方法寻找增广路径，直到找不到为止。这时我们就得到了一个最大的网络流。

最大流问题-流网络的割

- 怎么证明当无法再寻找到增广路径时，就证明当前网络是最大流网络呢？这就需要用到最大流最小割定理

- 割的概念。流网络G（V，E）的割（S，T）将V划分为S和T=V-S两部分，使得s属于S，t属于T。割（S，T）的容量是指从集合S到集合T的所有边（有方向）的容量之和（不算反方向的，必须是S-àT）。如果f是一个流，则穿过割（S，T）的净流量被定义为f（S，T）（包括反向的，SàT的为正值，T—>S的负值）。将上面举的例子继续拿来，随便画一个割，如下图所示：

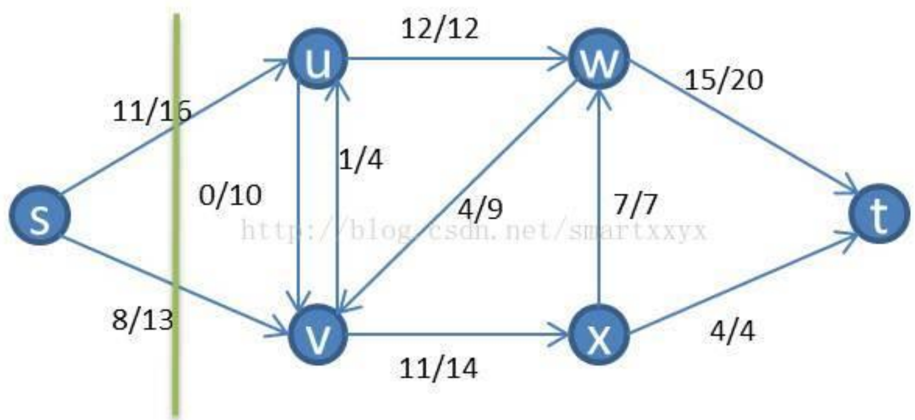


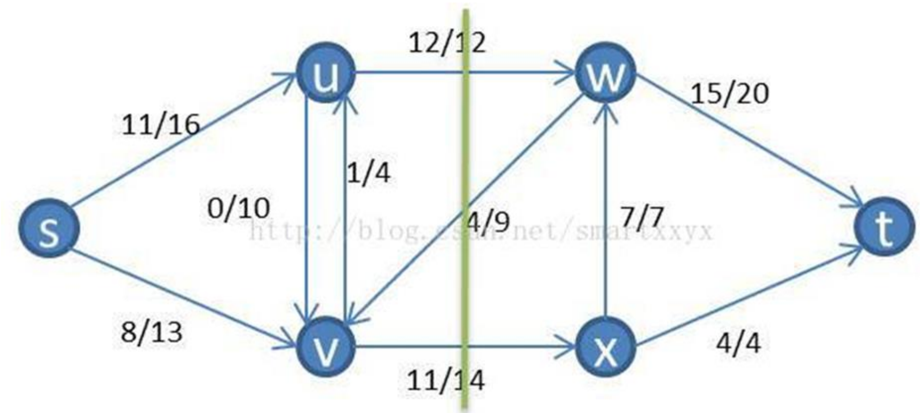
当前流网络的穿过割的净流量为f(u,w)+f(v,x)-f(w,v)=12+11-4=19

割的容量就是c(u,w)+c(v,x)=26

最大流问题-流网络的割

- **显然，我们有对任意一个割，穿过该割的净流量上界就是该割的容量，即不可能超过割的容量。所以网络的最大流必然无法超过网络的最小割。**
- 这跟残留网络上的增广路径有什么关系呢？

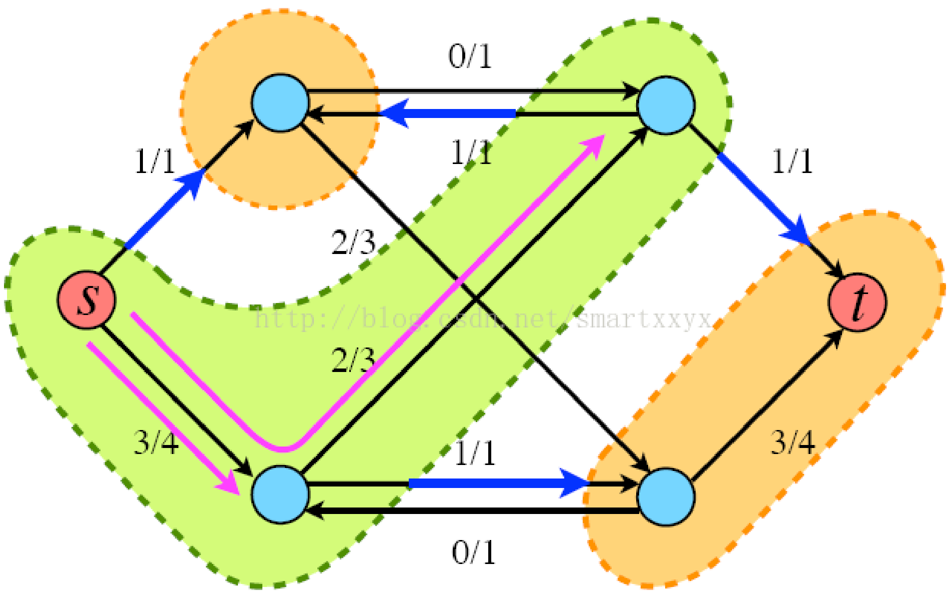




和上面的割相比，集合S中少了u和v，从源点s到集合T的净流量都流向了u和v，而在上一个割图中，集合S到集合T的流量是等于u和v到集合T的净流量的。其中w也有流流向了u和v，而这部分流无法流向源点s，因为没有路径，所以最后这部分流量加上s到u和v的流量，在u和v之间无论如何互相传递流，最终都要流向集合T，所以这个流量值是等于s流向u和v的值的。将s比喻成一个水龙头，u和v流向别处的水流，都是来自s的，其自身不可能创造水流。所以任意割的净流量都是相等的。

最大流问题-流网络的割

- **证明当残留网络Gf中不包含增广路径时，f是G的最大流**
- 假设Gf中不包含增广路径，即Gf不包含从s到v的路径，定义S={v:Gf中从s到v存在一条通路}，也就是Gf中s能够有通路到达的点的集合，显然这个集合不包括t，因为s到t没有通路。这时，我们令T=V-S。那么（S，T）就是一个割。如下图所示：



那么，对于顶点u属于S，v属于T，有f（u，v）=c（u，v）。否则（u，v）就存在残余流量，因而s到u加上u到v就构成了一条s到v的通路，所以v就必须属于S，矛盾。因此这时就表明当前流f是等于当前的割的容量的，因此f就是最大流。

最大流问题-Ford-Fulkerson

- **1 算法基本思想**
- 设P是网络G中联结源s和汇t的一条路。定义路的方向是从s到t。
- 将路P上的边分成2类：
- 一类边的方向与路的方向一致，称为**向前边**。向前边的全体记为P+。
- 另一类边的方向与路的方向相反，称为**向后边**。向后边的全体记为P-。
- 设flow是一个可行流，P是从s到t的一条路，若P满足下列条件：
- （1）在P的所有向前边(v,w)上，flow(v,w)<cap(v,w)，即P+中的每一条边都是非饱和边；
- （2）在P的所有向后边(v,w)上，flow(v,w)>0，即P-中的每一条边都是非零流边。
- 则称P为关于可行流flow的一条可增广路。
- 可增广路是残流网络中一条容量大于0的路。
- 将具有上述特征的路P称为可增广路是因为可以通过修正路P上所有边流量flow(v,w)将当前可行流改进成一个流值更大的可行流。

最大流问题-Ford-Fulkerson

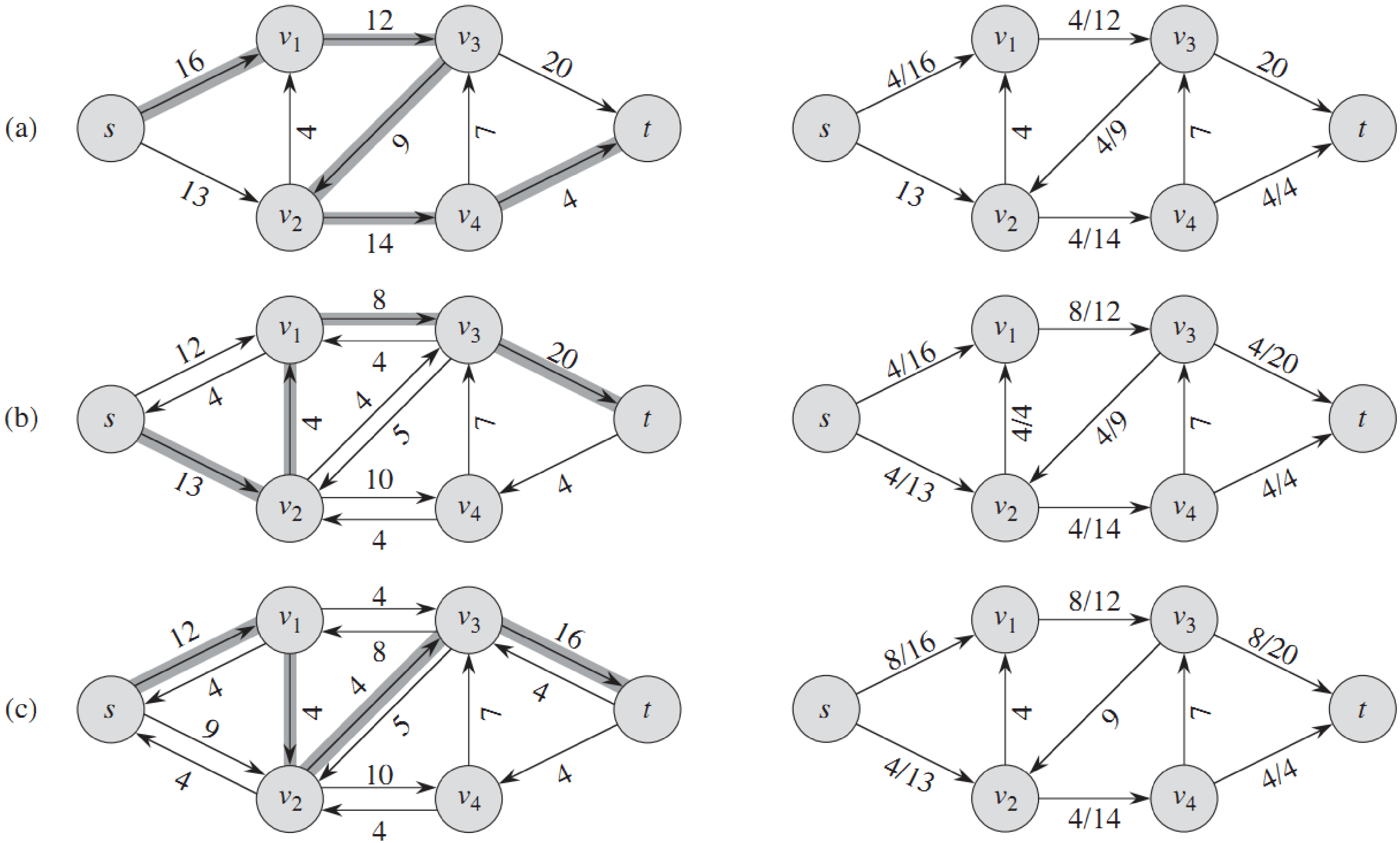
- 增流的具体做法是：
- （1）不属于可增广路P的边(v,w)上的流量保持不变；
- （2）可增广路P上的所有边(v,w)上的流量按下述规则变化：
- 在向前边(v,w)上，flow(v,w)+d；
- 在向后边(v,w)上，flow(v,w)-d。
- 按下面的公式修改当前的流。

$$flow(v,w)=\begin{cases} flow(v,w)+d & (v,w)\in P^{+} \\ flow(v,w)-d & (v,w)\in P^{-} \\ flow(v,w) & (v,w)\notin P \end{cases}$$

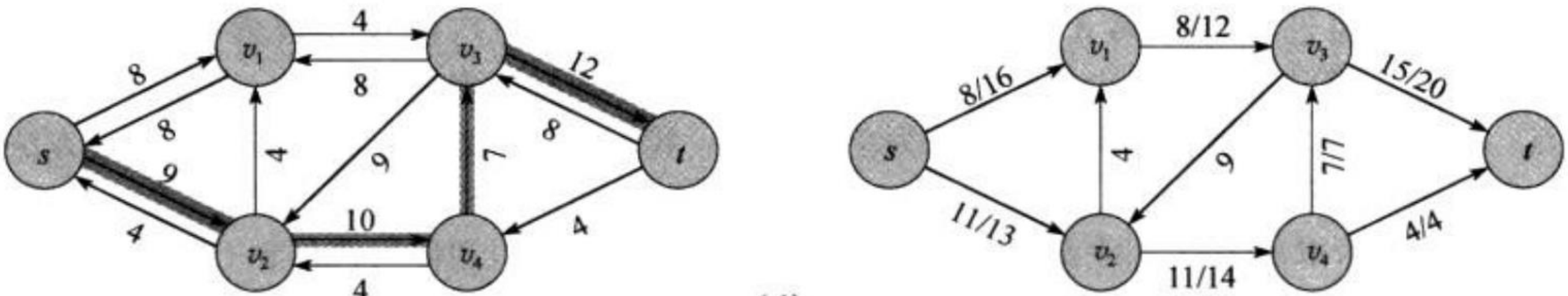
- 其中d称为可增广量，可按下述原则确定：d取得尽量大，又要使变化后的流仍为可行流。
- 按照这个原则，d既不能超过每条向前边(v,w)的cap(v,w)-flow(v,w)，也不能超过每条向后边(v,w)的flow(v,w)。
- 因此d应该等于向前边上的cap(v,w)-flow(v,w)与向后边上的flow(v,w)的最小值。也就是残流网络中P的最大容量。
- **增广路定理：**设flow是网络G的一个可行流，如果不存在从s到t关于flow的可增广路P，则flow是G的一个最大流。

最大流问题-Ford-Fulkerson

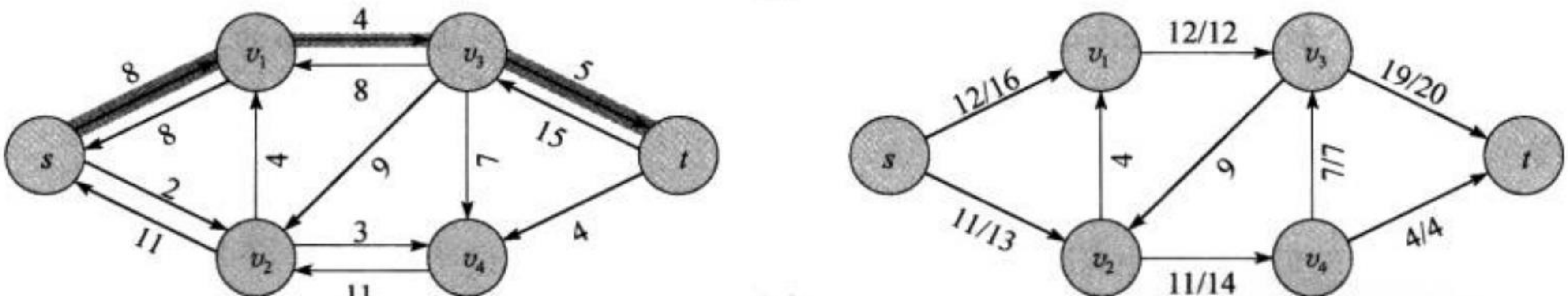
找出一条增广路径 ——2.修改其上点的值（残流网络）——3.继续重复1，直至找不出增广路。则此时源点的汇出量即为所求的最大流。



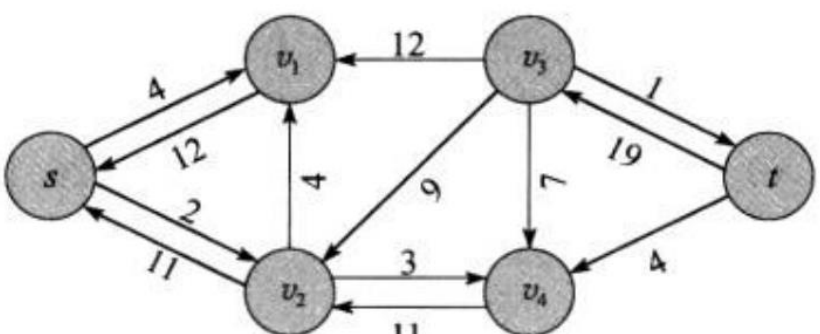
最大流问题-Ford-Fulkerson



(d)



(e)



(f)

最大流问题-Ford-Fulkerson

```
template <class Graph, class Edge> class MAXFLOW
{   const Graph &G;
    int s, t,maxf;
    vector<int> wt;
    vector<Edge *> st;
    int ST(int v) const { return st[v]->other(v); }
    void augment(int s, int t)
    {   int d = st[t]->capRto(t);
        for (int v = ST(t); v != s; v = ST(v))
          if (st[v]->capRto(v) < d)  d = st[v]->capRto(v);
        st[t]->addflowRto(t, d);
        maxf+=d;
        for ( v = ST(t); v != s; v = ST(v))  st[v]->addflowRto(v, d);
    }
    bool pfs();
public:
      MAXFLOW(const Graph &G, int s, int t,int &maxflow) :
         G(G), s(s), t(t), st(G.V()), wt(G.V()),maxf(0)
      {   while (pfs()) augment(s, t); maxflow+=maxf;}
};
```

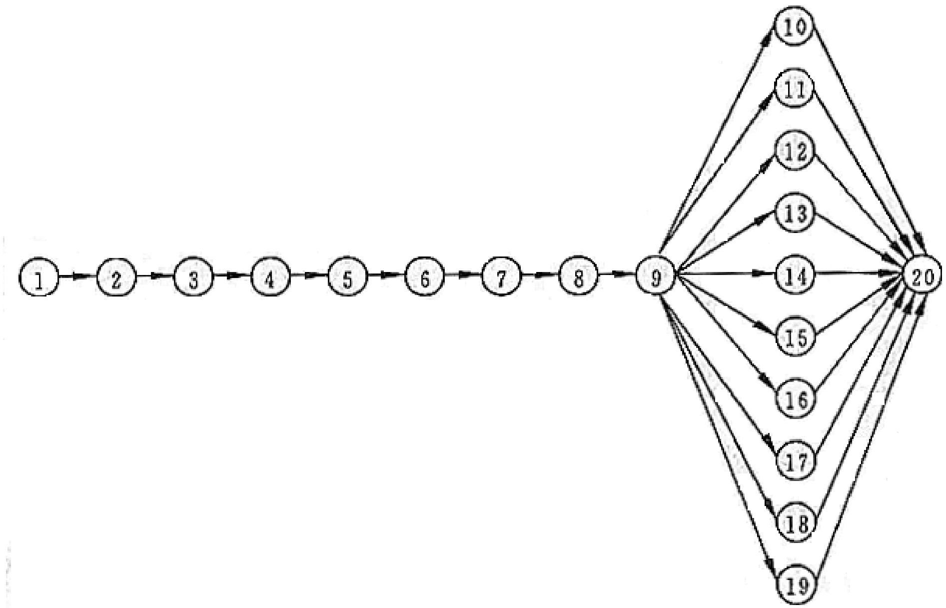
最大流问题-Ford-Fulkerson

- **3 算法的计算复杂性**
- 增广路算法的效率由下面2个因素所确定。
- （1）整个算法找增广路的次数；
- （2）每次找增广路所需的时间。
- 给定的网络中有n个顶点和m条边，且每条边的容量不超过M。
- 可以证明，在一般情况下，增广路算法中找增广路的次数不超过nM次。
- 最短增广路算法在最坏情况下找增广路的次数不超过nm/2次。
- 找1次增广路最多需要$O(\mathrm{m})$计算时间。
- 因此，在最坏情况下最短增广路算法所需的计算时间为$O(\mathrm{nm}^2)$ 。
- 当给定的网络是稀疏网络，即$\mathrm{m}=O(\mathrm{n})$时，最短增广路算法所需的计算时间为$O(\mathrm{n}^3)$。
- 最大容量增广路算法在最坏情况下找增广路的次数不超过2mlogM次。
- 由于使用堆来存储优先队列，找1次增广路最多需要O(nlogn)计算时间。
- 因此，在最坏情况下最大容量增广路算法所需的计算时间为 $O(mn\log n\log M)$
- 当给定的网络是稀疏网络时，最大容量增广路算法所需的计算时间为 $O(n^2\log n\log M)$

最大流其他算法

预流推进算法

- **1 算法基本思想**
- 增广路算法的特点是找到增广路后，立即沿增广路对网络流进行增广。
- 每一次增广可能需要对最多n-1条边进行操作。
- 最坏情况下，每一次增广需要O(n)计算时间。
- 有些情况下，这个代价是很高的。下面是一个极端的例子。



最小费用流问题

- **1 网络流的费用**
- 在实际应用中，与网络流有关的问题，不仅涉及流量，而且还有费用的因素。
- 网络的每一条边(v,w)除了给定容量cap(v,w)外，还定义了一个单位流量费用cost(v,w)。对于网络中一个给定的流flow，其费用定义为：

$$\mathrm{cost}(flow) = \sum_{(v,w)\in E} \mathrm{cost}(v,w) \times flow(v,w)$$

- **2 最小费用流问题**
- 给定网络G，要求G的一个最大用流flow，使流的总费用最小。
- **3 最小费用可行流问题**
- 给定多源多汇网络G，要求G的一个可行流flow，使可行流的总费用最小。
- 可行流问题等价于最大流问题。最小费用可行流问题也等价于最小费用流问题。

最小费用流算法

- **1 算法基本思想**
- 利用求最大流的增广路算法的思想，不断在残流网络中寻找从源s到汇t的最小费用路，然后沿最小费用路增流，直至找到最小费用流。
- 残流网络中从源s到汇t的最小费用路是残流网络中从s到t的以费用为权的最短路。
- 残流网络中边的费用定义为：

$$wt(v,w)=\begin{cases}\cos t(v,w) & (v,w)\in P^{+}\\ -\cos t(w,v) & (v,w)\in P^{-}\end{cases}$$

- 当残流网络中边(v,w)是向前边时，其费用为cost(v,w)；
- 当(v,w)是向后边时，其费用为-cost(w,v)。

最小费用流的最小费用路算法

最小费用流的最小费用路算法

步骤0：初始可行0流。

步骤1：如果不存在最小费用路，则计算结束，已经找到最小费用流；

否则用最短路算法（bellman-ford）在残流网络中找从s到t的最小费用可增广路，

转步骤2。

步骤2：沿找到的最小费用可增广路增流，并转步骤1。

最小费用流的最小费用路算法

- **3 算法的计算复杂性**
- 算法的主要计算量在于连续寻找最小费用路并增流。
- 给定网络中有n个顶点和m条边，且每条边的容量不超过M，每条边的费用不超过C。
- 每次增流至少使得流值增加1个单位，因此最多执行M次找最小费用路算法。
- 如果找1次最小费用路需要s(m,n,C)计算时间，则求最小费用流的最小费用路算法需要O(Ms(m,n,C))计算时间。

网络流总结

- 学习要点
- 理解网络与网络流的基本概念
- 网络最大流问题
- 掌握网络最大流的增广路算法（Ford-Fulkerson）
- 网络最小费用流算法

- 了解网络流其他算法
  - 了解网络最大流的其他算法
  - 了解网络最小费用流的其他算法