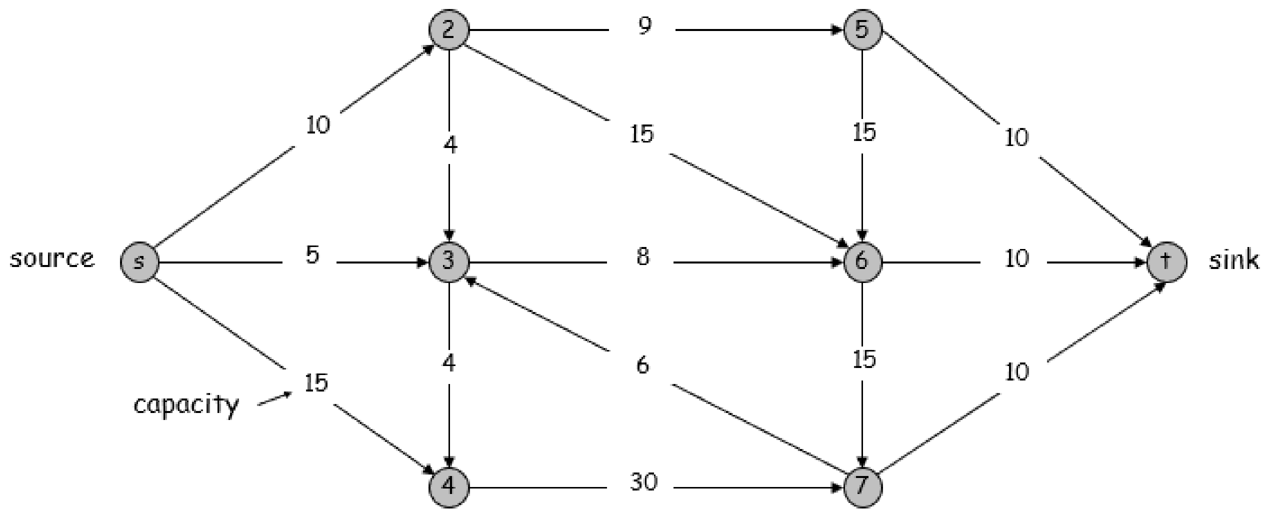
流网络

$G=(V,E)$是一个简单有向图，在V中指定顶点s和t，分别称为**源点**和**汇点**，有向图G中的每一条边$(u,v)\in E$，对应有一个值$cap(u,v)\geq 0$，称为边的容量，这样的有向图G称作一个**流网络**，下图是一个例子。



$f(v,u)$称作是从顶点u到顶点v的流，它满足以下性质：

- **容量限制**：对所有$u,v\in V$，要求$f(u,v)\leq c(u,v)$。
- **反对称性**：对所有$u,v\in V$，要求$f(u,v)=-f(v,u)$。

如果有一组流满足以下条件，那么这组流就成为一个**可行流**：

- 源点s：流出量 = 整个网络的流量
- 汇点t：流入量 = 整个网络的流量
- 中间点：总流入量 = 总流出量

最大流即网络G所有的可行流中，流量最大的一个可行流。

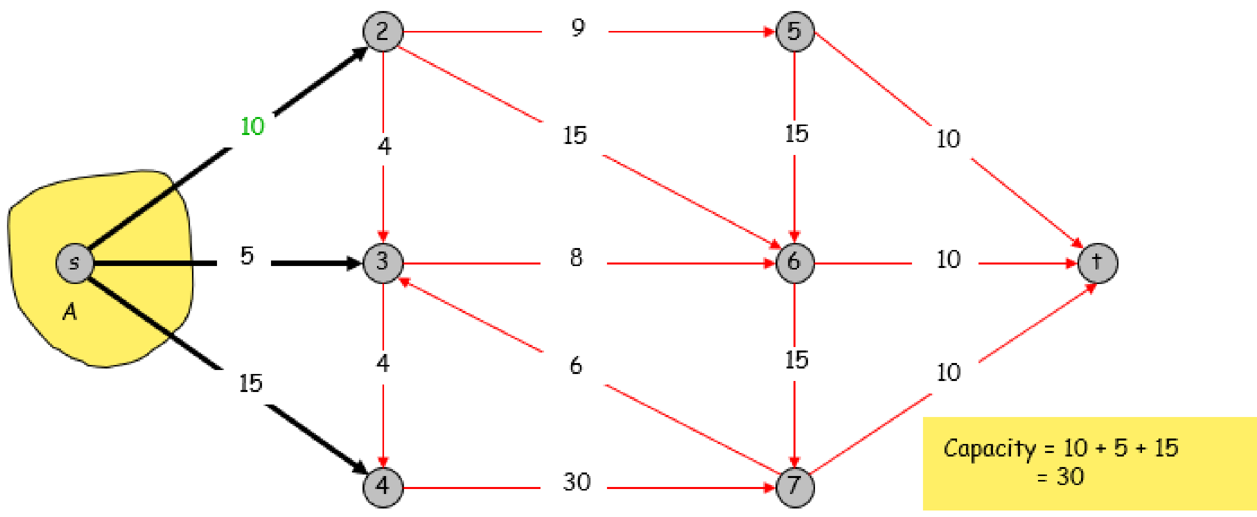
Ford-Fulkerson方法

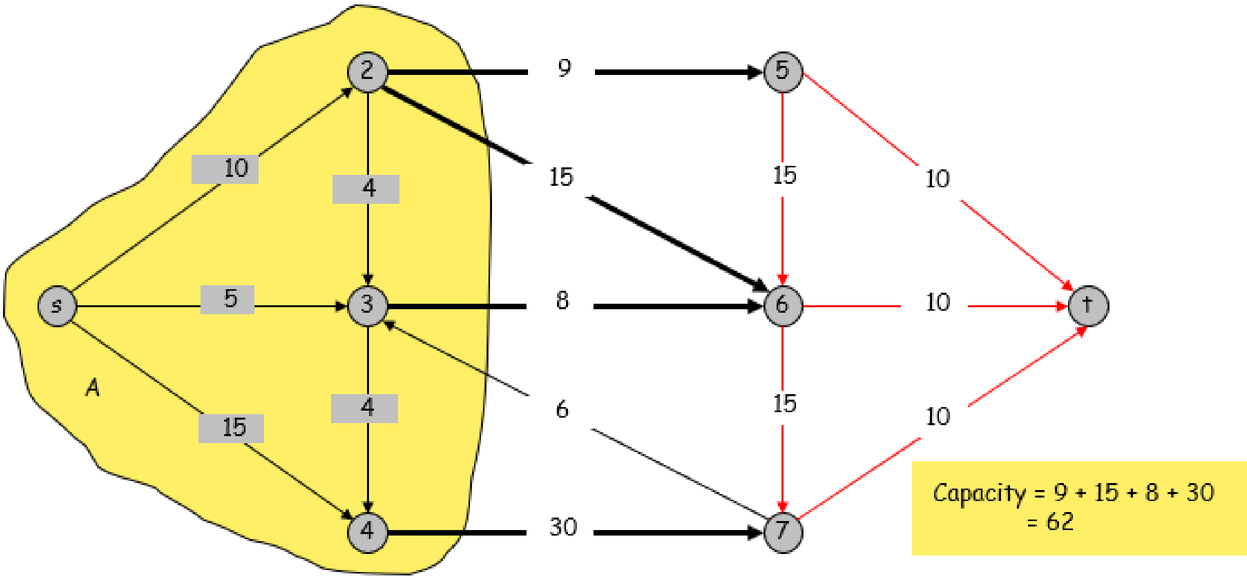
之所以称为Ford-Fulkerson方法而不是算法，是由于它包含具有不同运行时间的几种实现。Ford-Fulkerson方法依赖于三种重要思想：**残留网络**、**增广路径**、**割**。这三种思想是最大流最小割定理的精髓，该定理用流网络的割来描述最大流的值，我们将会在后面谈到。以下给出Ford-Fulkerson方法的伪代码：

```
Ford-Fulkerson-Method(G, s, t):
    initialize flow f to 0
    while there exists an augmenting path p:
        do augment flow f along p
    return f
```

最大流最小割定理

首先来介绍割的概念，一个割会把图G的顶点分成两个不相交的集合，其中s在一个集合中，t在另外一个集合中。割的容量就是**从A指向B**的所有边的容量和，最小割问题就是要找到割的容量最小的情况。下面给出两个例子，割的容量分别为30和62。

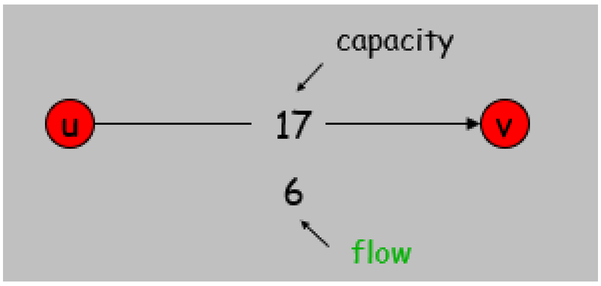


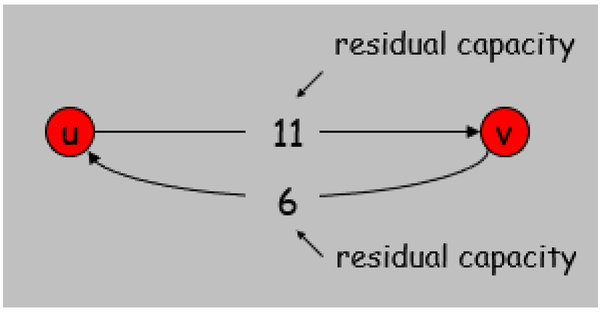


接着介绍**残留网络**和**增广路径**的概念，给定一个流网络G和一个可行流，流的**残留网络**G_f拥有与原网相同的顶点。流网络G中每条边将对应残留网中一条或者两条边，对于原流网络中的任意边(u, v)，流量为f(u, v)，容量为c(u, v)：

- 如果f(u, v) > 0，则在残留网中包含一条容量为f(u, v)的边(v, u);
- 如果f(u, v) < c(u, v)，则在残留网中包含一条容量为c(u, v) - f(u, v)的边(u, v)。

下图为一个例子：





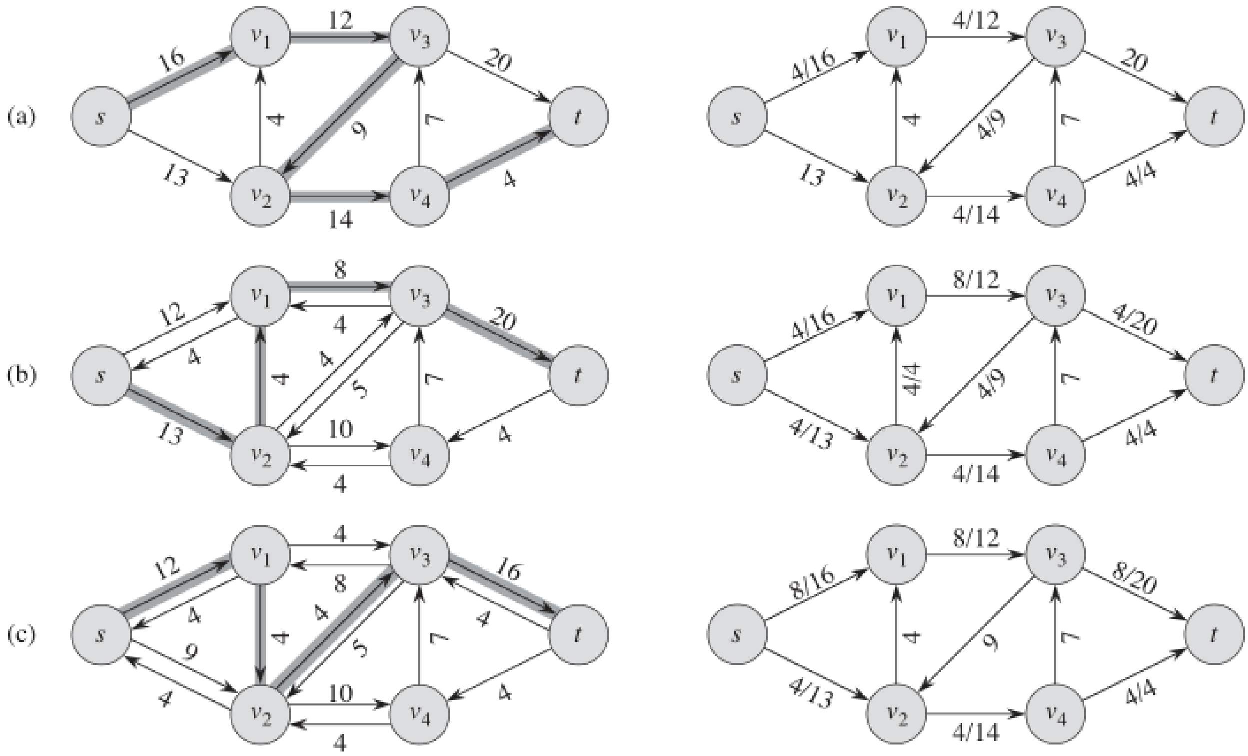
对于一个已知的流网络$G=(V,E)$和流f，**增广路径**p为残留网络G_f中从s到t的一条简单路径。

最大流最小割定理：**网络的最大流等于某一最小割的容量**，并且下列条件是等价的：

- f是G的一个最大流。
- 残留网络G_f不包含增广路径。
- 对G的某个割(S,T)，有$|f|=c(S,T)$。

基本的Ford-Fulkerson算法

根据，我们可以求给定有向图的最大流。下面给出《算法导论》中的一个实例：

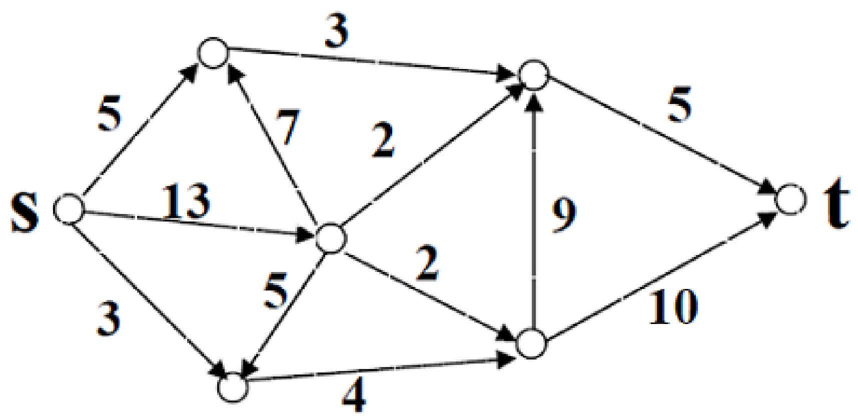


上图中的左边表示开始时的残留网络，右边表示将增广路径加入残留网络后得到的新的可行流，通过三次迭代即可得到最大流，根据最大流最小割定理，我们同样可以得到最小割。

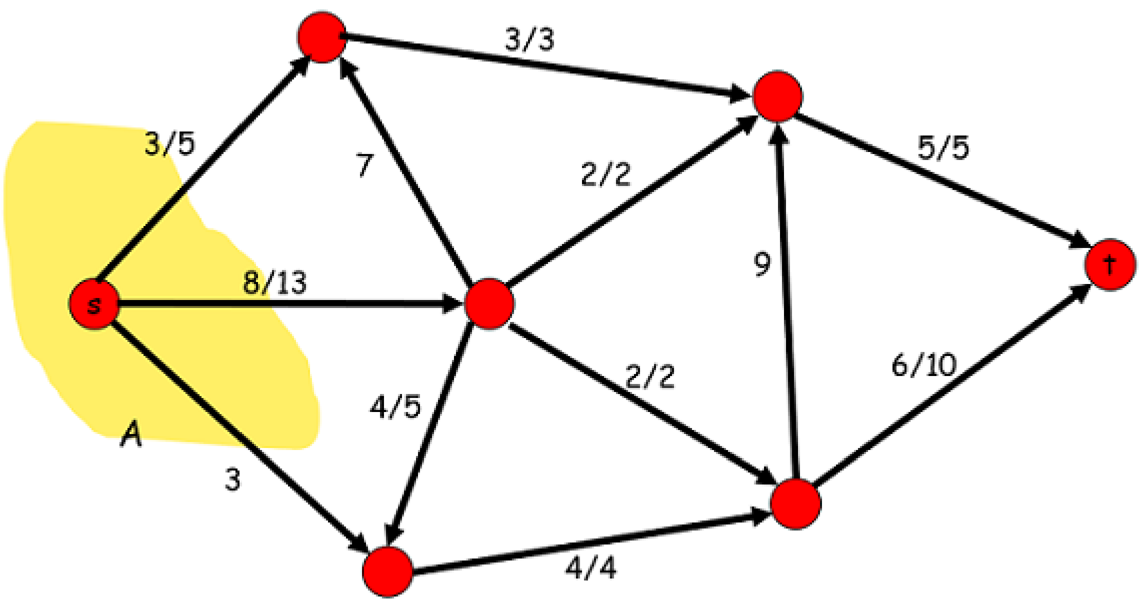
再通过本课程课件上的一个例题进行练习。

- 给定有向图G如下所示，起点为s，终点为t。

- 1. 计算(s,t)间的最大流，最小割
- 2. 将满足(s,t)间最大流的路径以及路径上流的权重一一列出
- 3. 写出s，t之间的一个最小割



同样通过基本的Ford-Fulkerson算法，可得到答案如下。



Edmonds-Karp算法

Edmonds和Karp曾经证明了如果每步的增广路径都是最短，那么整个算法会执行$O(mn)$步，Edmonds-Karp算法是用广度优先搜索来实现对增广路径p的计算的，实现的伪代码如下图所示。

ShortestAugmentingPath(V, E, s, t)

```
FOREACH e ∈ E
    f(e) ← 0
Gf ← residual graph

WHILE (there exists augmenting path)
    find such a path P by BFS
    f ← augment(f, P)
    update Gf
RETURN f
```

由于在广度优先搜索时最坏情况下需要$O(m)$次操作，所以此算法的复杂度为$O(m^2n)$。之后，Dinitz改进了Edmonds-Karp算法，得到一个时间复杂度为$O(mn^2)$的算法，下面给出一张关于最短增广路径算法研究历史的表格，这里就不再展开了。

Year	Discoverer	Method	Big-Oh
1951	Dantzig	Simplex	mn^2U
1955	Ford, Fulkerson	Augmenting path	mnU
1970	Edmonds-Karp	Shortest path	m^2n
1970	Dinitz	Shortest path	mn^2
1972	Edmonds-Karp, Dinitz	Capacity scaling	$m^2 \log U$
1973	Dinitz-Gabow	Capacity scaling	$mn \log U$
1974	Karzanov	Preflow-push	n^3
1983	Sleator-Tarjan	Dynamic trees	$mn \log n$
1986	Goldberg-Tarjan	FIFO preflow-push	$mn \log (n^2 / m)$
. . .	. . .	. . .	. . .
1997	Goldberg-Rao	Length function	$m^{3/2} \log (n^2 / m) \log U$ $mn^{2/3} \log (n^2 / m) \log U$