

# EDA 软件设计 I

Lecture 16

# Bellman-Ford算法核心原理

什么样的机制使其可以处理负权边？

**Dijkstra = Greedy, Bellman-Ford  $\neq$  Greedy**

# Bel lman-Ford算法核心原理


 Idea behind:

 Same as Dijkstra: 从单个源点开始，计算到每个节点的距离

 距离最初是未知的，并假定为无限远

 Different from Dijkstra :


 核心：通过“**松弛**”**每条边**  $(u, v)$  来获取最短路径

 **松弛原理：每轮松弛后，最短路径的长度会逐步逼近最终值**——  
检查是否可以通过一个更短的路径到达节点  $v$ ，并在找到更短路径时更新  $d[v]$

# Bellman-Ford算法步骤

 **具体如何松弛**（假设图中有  $V$  个节点和  $E$  条边，源点为  $S$ ）：

- ① **初始化**：设置源点  $S$  的距离为 0，即  $d[S]=0$ ，其他所有节点的初始距离为无穷大 ( $\infty$ )，表示这些节点还没有被找到有效路径
- ② **松弛循环  $V-1$  次**：在每次循环中，对图中的每一条边  $(u, v)$ ，检查是否可以计算出更短的距离（松弛操作），如果计算出的距离更短，则更新距离： $\text{dist}[v] = \min(\text{dist}[v], \text{dist}[u] + \text{weight}(u, v))$ 。经过  $V-1$  次之后，所有节点的最短路径距离都将收敛为最终结果

 **如何检测负循环**：执行第  $V$  次松弛，如果第  $V$  次和第  $V-1$  次松弛后的结果（每个节点的最短路径）不一致，则负循环存在，否则不存在。

# 关键问题

- ❓为什么松弛边  $V-1$  次，为我们提供了单源最短路径？(为什么松弛  $V-1$  次就足够了)
- ❓有没有可能，在某一步松弛操作后，所有节点的最短路径数值已经不再发生变化？(松弛次数可能少于  $V-1$  次就结束吗？)
- ❓为什么第  $V$  次松弛后距离的减少表明存在负循环？
  - Hint: 图如何才能从非线性结构变成线性结构？图成为线性结构时对应  $V-1$  条边



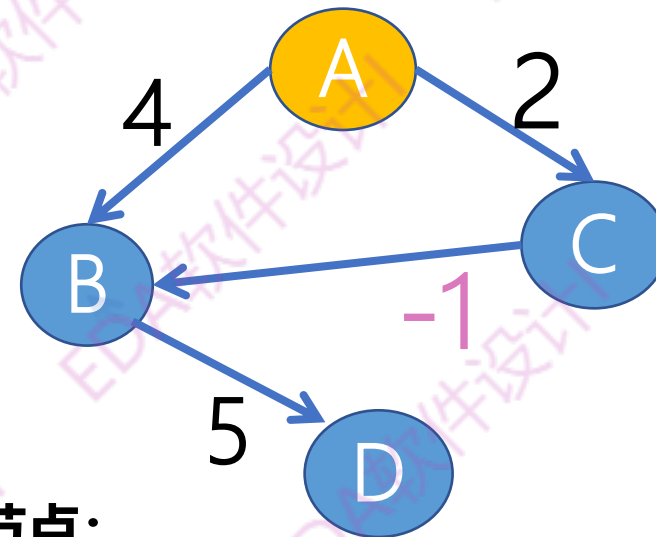
# Bellman-Ford具体步骤

松弛 (Relaxation)

# Bellman-Ford松弛操作的示意

假设有以下简单图，节点A为源点：

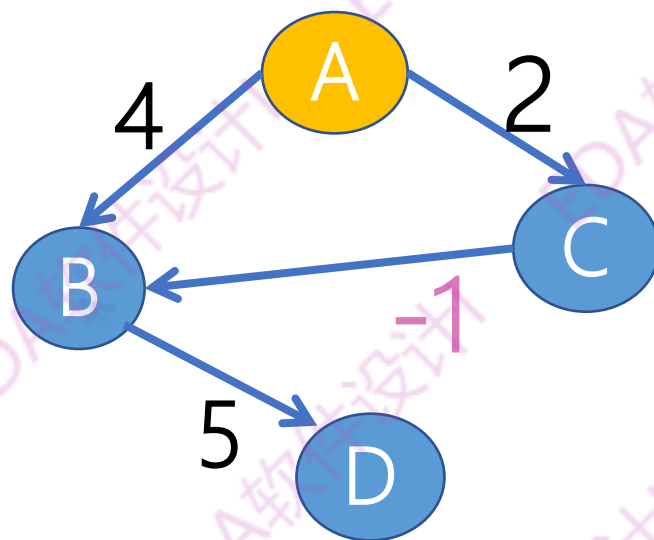
- 节点  $A \rightarrow B$ ，权重为 4
- 节点  $A \rightarrow C$ ，权重为 2
- 节点  $C \rightarrow B$ ，权重为 -1
- 节点  $B \rightarrow D$ ，权重为 5



**Theoretically, 通过每次松弛操作，最短路径逐步传播到所有节点：**

1. 第一次松弛：找到直接连接源点的节点的最短路径。
2. 第二次松弛：找到通过一个中间节点的更短路径。
3. 第三次松弛：确保所有节点的路径长度已经达到最短。

# Bellman-Ford 松弛操作的示意

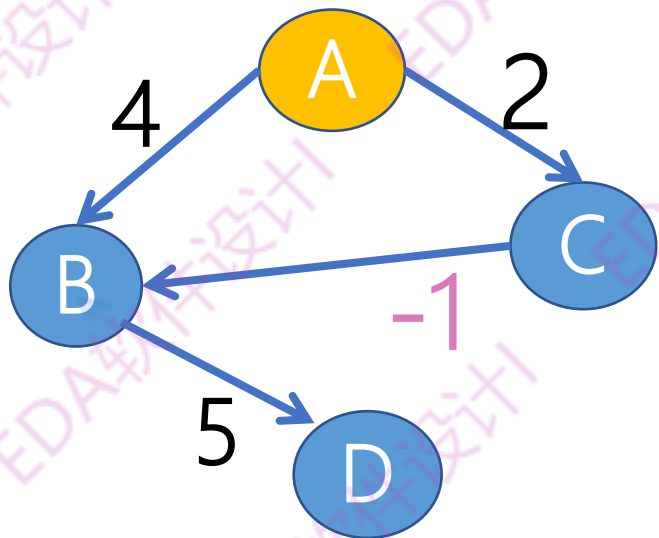


## 初始化

- 将源点  $A$  的距离初始化为 0, 即  $d[A] = 0$ 。
- 其他节点的初始距离设置为无穷大 ( $\infty$ ), 即  $d[B] = \infty$ ,  $d[C] = \infty$ ,  $d[D] = \infty$ 。



# Bellman-Ford 松弛操作的示意



**第一次松弛**  $\text{dist}[v] = \min (\text{dist}[v], \text{dist}[u] + \text{weight}(u,v))$

遍历所有边，并更新每个节点的最短路径：

1. 边  $A \rightarrow B$ ，权重 4：

$$d[A] + 4 < d[B]$$

$0 + 4 < \infty$ ，所以更新  $d[B] = 4$ 。

2. 边  $A \rightarrow C$ ，权重 2：

$$d[A] + 2 < d[C]$$

$0 + 2 < \infty$ ，所以更新  $d[C] = 2$ 。

3. 边  $C \rightarrow B$ ，权重 -1：

$$d[C] + (-1) < d[B]$$

$2 - 1 < 4$ ，所以更新  $d[B] = 1$ 。

4. 边  $B \rightarrow D$ ，权重 5：

$$d[B] + 5 < d[D]$$

$1 + 5 < \infty$ ，所以更新  $d[D] = 6$ 。

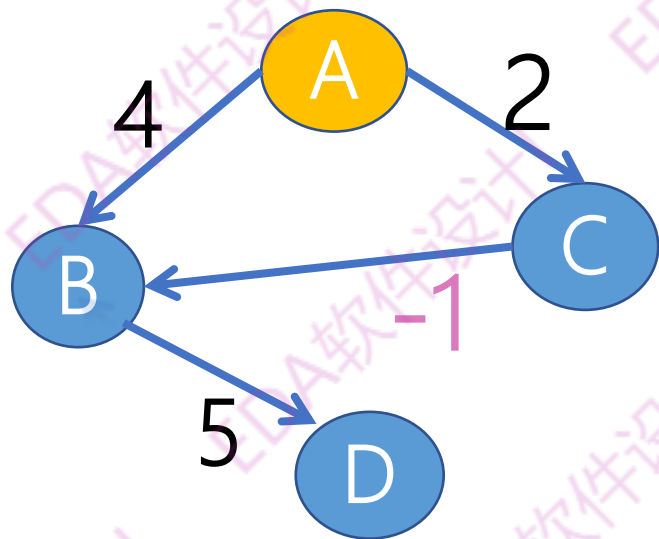
第一次松弛后，节点的最短路径距离如下：

- $d[A] = 0$
- $d[B] = 1$
- $d[C] = 2$
- $d[D] = 6$

# Bellman-Ford 松弛操作的示意

**第二次松弛**  $\text{dist}[v] = \min (\text{dist}[v], \text{distance}[u] + \text{weight}(u,v))$

再次遍历所有边，尝试更新最短路径：



1. 边  $A \rightarrow B$ , 权重 4:

$$d[A] + 4 < d[B]$$

$0 + 4 = 4$ , 但  $d[B]$  已是 1, 不需要更新。

2. 边  $A \rightarrow C$ , 权重 2:

$$d[A] + 2 < d[C]$$

$0 + 2 = 2$ , 但  $d[C]$  已是 2, 不需要更新。

3. 边  $C \rightarrow B$ , 权重 -1:

$$d[C] + (-1) < d[B]$$

$2 - 1 = 1$ , 但  $d[B]$  已是 1, 不需要更新。

4. 边  $B \rightarrow D$ , 权重 5:

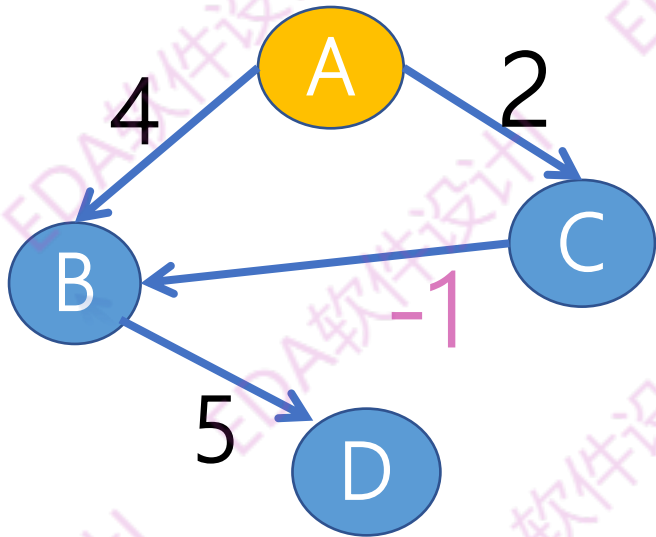
$$d[B] + 5 < d[D]$$

$1 + 5 = 6$ , 但  $d[D]$  已是 6, 不需要更新。

第二次松弛后，节点的最短路径距离没有发生变化：

- $d[A] = 0$
- $d[B] = 1$
- $d[C] = 2$
- $d[D] = 6$

# Bellman-Ford 松弛操作的示意



**第三次松弛**  $\text{dist}[v] = \min (\text{dist}[v], \text{dist}[u] + \text{weight}(u,v))$

进行第三次松弛，遍历所有边：

1. **边**  $A \rightarrow B$ 、**边**  $A \rightarrow C$ 、**边**  $C \rightarrow B$  和 **边**  $B \rightarrow D$  都不会再导致更新，因为最短路径距离已经稳定。

因此，在第三次松弛操作后，所有节点的最短路径依然保持不变：

- $d[A] = 0$
- $d[B] = 1$
- $d[C] = 2$
- $d[D] = 6$



# Bellman-Ford算法实现

1. 初始化 (same as Dijkstra)
2. 松弛循环
3. 检测负循环 (负权回路)

## Pseudocode

```
# Step 2: 松弛所有边  $V - 1$  次 (其中  $V$  为节点数)
for i from 1 to  $V - 1$ :
    for each edge  $(u, v)$  with weight  $w$  in graph:
        if  $\text{distance}[u] + w < \text{distance}[v]$ :
             $\text{distance}[v] = \text{distance}[u] + w$ 
```

# Progress So Far



**BFS**



**DFS**



**Topological Sort**



**MST**



**Shortest Path Problem: Dijkstra + Bellman Ford**



**Next?**



# 最大流最小割问题

max flow, min cut

- 历史背景（由来）
  - 什么是网络流
- 最大流问题 (what is a max flow)
- 最小割问题 (what is a min cut)

# 很多重大科技突破的原因：War

- 1.有史以来最大的科学项目之一是[曼哈顿计划](#)——它使核能和核武器变得实用
- 2.在破译德国信息的竞赛中，计算机变得很普遍。[恩尼格玛密码机](#)，在没有战争努力的情况下，它更像是一种数学奇观，其应用仅限于统计
- 3.[喷气式飞机](#)在战争期间变得实用，并给我们带来了航空革命
- 4.雷朋 (Rayban) 推出的偏光太阳镜最初是为了飞行员减轻恶心症状而推出的，现在普通民众也开始佩戴这种太阳镜
- 5.塑料：由于战争中主要金属变得稀缺，塑料开始被使用
- 6.M&M豆：被发明为小份巧克力，糖果涂层减少了战争前线巧克力的融化

# 最大流最小割历史背景

## Field of Birth: 军事通信网络

背景和需求：在20世纪50年代初期，随着冷战的加剧，美国空军和其他军事机构确实**投入大量资源研究如何在战时或攻击状态下保持通信网络的稳定性和高效性**。当时的重点之一是**确保关键基础设施（包括通信网络）在面对可能的攻击时不会完全瘫痪**。这些研究的确集中在提高网络的抗毁性和在受损情况下的传输能力。

## 流量和割的概念形成：

- 早期的军事需求推动了图论和网络流理论的发展，**具体来说，研究人员希望找到能够有效传输信息的最优路径，同时识别网络中哪些节点或连接是“瓶颈”。**
- 这些研究最终导致了**“最大流量-最小割定理”**的发展
- 该定理由**数学家L.R. Ford和D.R. Fulkerson**在1956年首次正式提出，他们的理论为分析和优化网络流量提供了关键工具——**最终引出形成了网络流这个研究领域**

# 网络流

## network flow

“网络流是研究如何在网络结构中优化资源分配与路径选择的数学理论”——  
运筹学角度

- 运筹学 (Operations Research) : 一门应用数学的学科, 专注于利用数学模型、统计分析和优化方法来辅助决策和解决复杂问题。

“网络流是研究在由节点和边构成的网络中, 如何高效传递和分配资源的数学理论”——图论角度

“网络流就是一种方法, 帮助我们找到从一个地方到另一个地方运送东西的最好路”——解释给7-year-old听的角度



# 网络流入门

## 网络流的基本概念

在一个网络流问题中，我们有以下要素：

- **源点 (Source)**：流的起点，通常记为  $S$
- **汇点 (Sink)**：流的终点，通常记为  $T$
- **容量 (Capacity)**：每条边都有一个容量限制，表示该边能够承载的最大流量
- **流量守恒原则**：除了源点和汇点外，网络中每个节点的进入流量之和等于离开流量之和
- **网络流问题**就是在满足这些容量和守恒条件的情况下，研究如何从源点向汇点传输流量



# 网络流入门

## 应用中的例子

- 假设有一个网络流模型，代表**城市的供水系统**：
- **最大流**表示在这个系统中，供水从**水源地（源点）**到达城市**各个需求区域（汇点）**的最大水量
- **最小割**表示**切断水源和城市需求区域的最低成本路径**，也就是找到整个系统中的最薄弱环节（比如一条容量较小的管道）

# 3 Facts about Max Flow & Min Cut

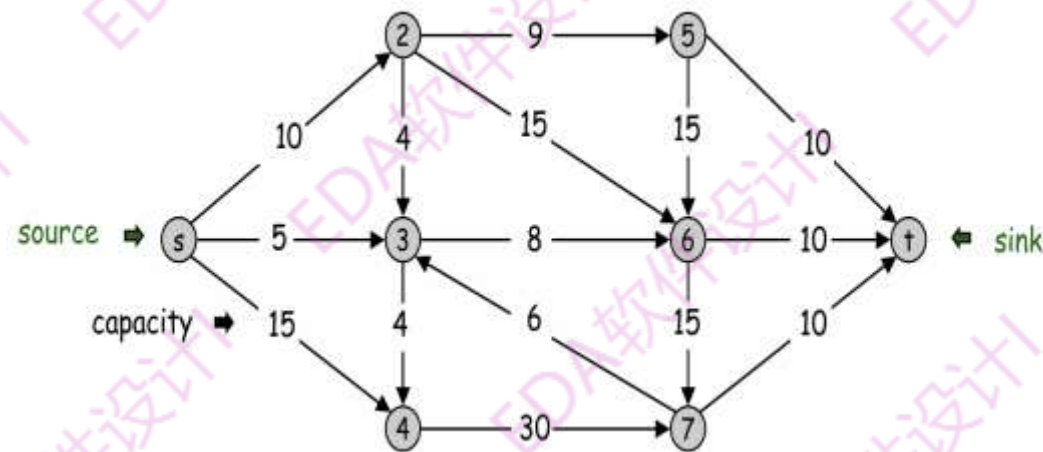
1. 两个非常重要的算法问题
2. 组合优化中的基石问题
  - 组合优化：数学优化的一个分支，专注于在**离散的或有限的解空间**中找到最优解。它涉及在有限或可数的解集中，通过组合和排列不同的元素，来满足某些约束并优化目标函数。典型问题：
    1. TSP（旅行商问题）
    2. MST + Shortest Path
    3. 背包问题（Knapsack Problem）
3. 优美的数学对偶性
  - 一个问题可以通过另一个相关联的“对偶问题”来解决或描述

# 流 (Flow)

**Flow (流)**：从源节点 (source) 流向汇节点 (sink) 的某种数量 (如物质、信息或资源)，并且沿着图中的边逐步传递，满足容量限制和流量守恒条件：

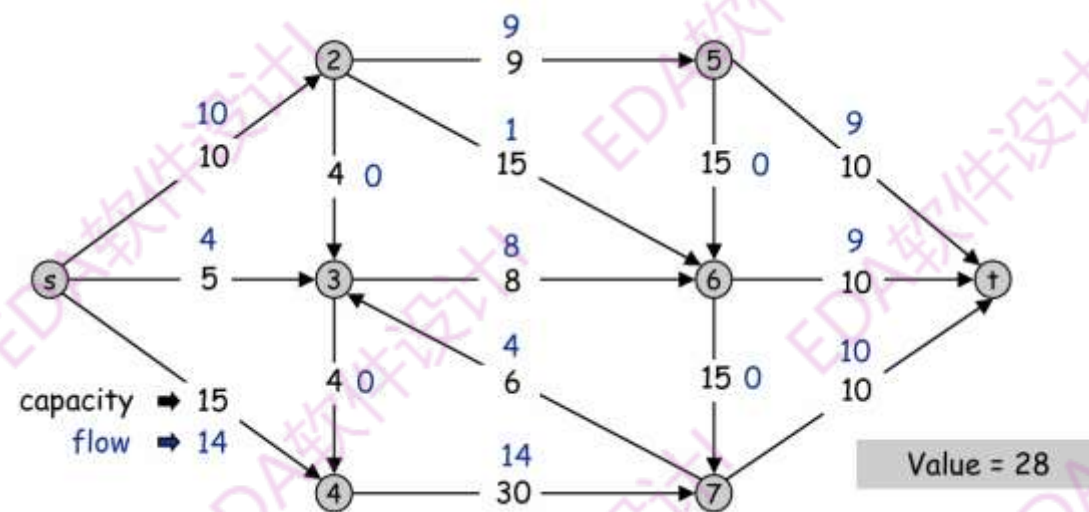
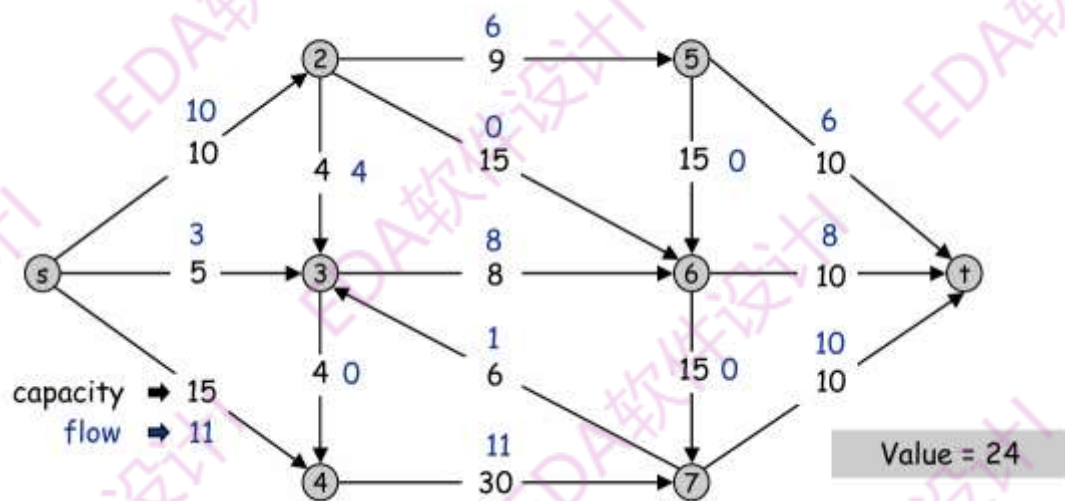
1. 容量限制：对于网络中的每条边  $e$ ，流量  $f(e)$  必须在 0 到该边的容量  $u(e)$  之间，即  $0 \leq f(e) \leq u(e)$
2. 流量守恒：对于源节点和汇节点之外的每个节点，流入流和流出流必须相等，即

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



# Max Flow Problem

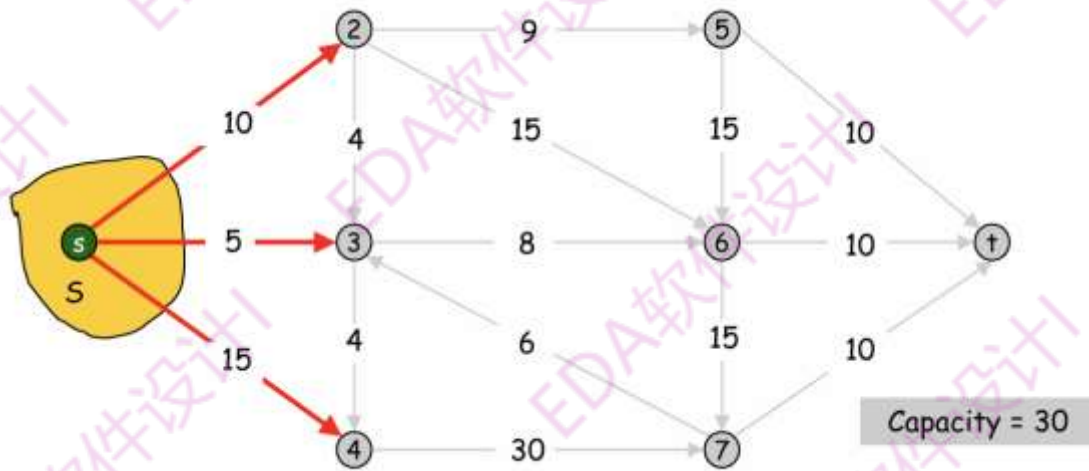
- 最大流问题：为边分配流量（flow），使得：
  - ① 在每个中间节点上流入（inflow）和流出（outflow）相等（约束条件：流量守恒）
  - ② 最大化从  $s$  到  $t$  的流量（优化目标）





# Cut (割)

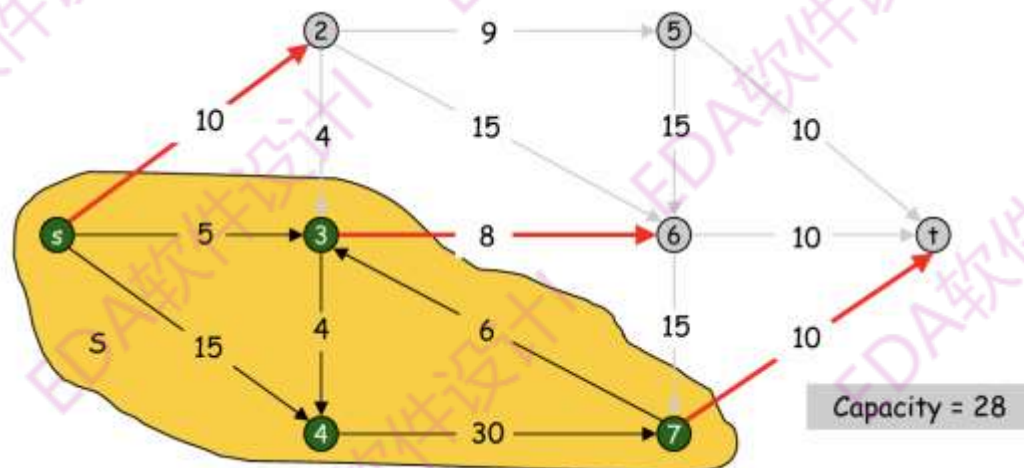
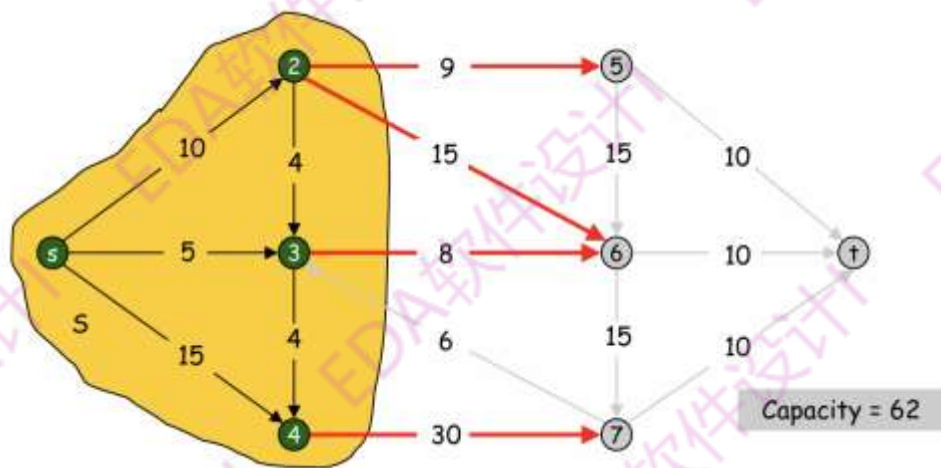
- 割是一个节点的划分  $(S, T)$  , 其中源节点  $s$  属于  $S$  , 汇节点  $t$  属于  $T$
- 割的容量  $Capacity(S, T)$  等于从  $S$  离开的边的权重之和





# Min Cut Problem

- 最小割问题：删除一组**“最佳”**边以使节点  $t$  与节点  $s$  断开连接
- **“最佳”**：找一个从源节点  $s$  到汇节点  $t$  的**最小容量的割**



# 鼓励页：Algorithm after BFS、DFS

- 不像DFS、BFS那么直观
- Dijkstra底层思想是Greedy
- Bellman ford的底层思想是Dynamic Programming
- 最大流最小割是“组合优化”理论里的明珠
  - 无处不在的应用
  - 走向计算理论中深邃奥秘的“第一步”