ALGORITHM NOTES

贪心算法
原理、证明与复杂度分析

从拟阵到交换论证，再到近似比

> 本教程不满足于「每步选当前最好」这一直觉描述。它把贪心算法还原为一个可判定的形式系统：先给出决策序列模型，再用拟阵刻画贪心最优的充要条件，用交换论证处理非拟阵结构，最后转向贪心失效时的近似保证与在线竞争比。每一处结论都附证明，每一处复杂度都给出推导。

版本 1.0

篇幅 8 章 · 附录 · 约 40 道习题

前置要求 算法分析、线性代数基础、概率论初步

日期 2026 年 10 月

写在前面

关于贪心算法的材料大多是同一个样子：先给直觉（每步选最好的），再列一堆经典例子，最后说「这个是正确的」。读者读完记住了案例，却没有获得判断新问题的能力——遇到一个没见过的贪心策略，仍然只能靠试。

这份教程换一条路。它把「贪心是否正确」当作一个可判定的命题来对待：第 1 章把贪心写成由评价函数、可行性判定、可行域构成的五元组，使「策略」成为可以讨论的对象；第 2 章给出唯一一个充要条件（拟阵）；第 3 章给出适用于其余情形的通用证明引擎（交换论证）；第 5、6 章处理贪心根本不可能正确时的补偿方案（近似比与竞争比）。

适合谁读

具备算法分析基础、熟悉大 O 记号与基本数据结构、愿意读证明的读者。线性代数与初等概率论在少数章节会用到（拟阵的线性表示、随机化的竞争比），不熟悉时可跳过相应小节而不影响主线。

三条阅读路线

- **想尽快用起来**：1 → 3 → 7 → 8。先建立判定框架，掌握交换论证，再处理实现陷阱与完整案例。
- **关心为什么对**：1 → 2 → 3 → 5。拟阵给出代数解释，交换论证覆盖非拟阵结构，近似比处理失效后的补偿。
- **做近似与在线**：5 → 6，对照阅读。两者形式相同（都是「与最优的比值」），但困难来源不同：一个来自计算复杂性，一个来自信息缺失。

约定

定理、引理、命题统一编号为「章.序号」，跨章引用写作「定理 3.4」。公式编号同理。复杂度记号采用 O、Ω、Θ 的标准含义；均摊与期望复杂度会明确标注。代码为 Python 伪码，优先可读性而非工程健壮性——第 7 章会专门讨论二者冲突时的取舍。

全书共 8 章、1 个附录，含 61 个编号公式、42 条定理与定义、30 处证明、5 幅图、8 张表与 35 道习题；部分习题的解答提示见 8.6 节。

目　录

01　贪心的形式化框架

1.1　一个问题，三种命运

1.2　决策序列模型

1.3　两个充分条件

1.4　三条证明路线

1.5　反例优先的设计心法

02　拟阵：贪心正确性的代数刻画

2.1　独立系统

2.2　拟阵的定义与交换公理

2.3　秩函数

2.4　拟阵上的贪心算法

2.5　推论与边界

2.6　拟阵交：贪心失效的代数解释

2.7　习题

03　交换论证：最通用的正确性引擎

3.1　论证的一般结构

3.2　区间调度：直接构造

3.3　最小化最大延迟：相邻交换

3.4　Huffman 编码：结构归纳

3.5　反例剖析：0-1 背包为何无法交换

04　图上的贪心：MST、最短路与并查集

4.1　割性质与环性质

4.2　Kruskal 与并查集的均摊分析

4.3　Prim 与优先队列的选择

4.4　Dijkstra：非负权的不变量证明

4.5　为什么最短路不能写成拟阵问题

4.6　习题

05　贪心的近似：当精确不可得

5.1　近似比的语言

5.2　集合覆盖：lnn+1

5.3 子模最大化：1−1/e
5.4 拟阵约束与更弱的保证
5.5 k-中心：2-近似及其紧性
5.6 习题

06 在线与随机：信息缺失下的贪心

6.1 竞争比
6.2 滑雪租赁
6.3 秘书问题：1/e的另一处现身
6.4 乘性权重与遗憾
6.5 在线集合覆盖
6.6 习题

07 复杂度工程与实现陷阱

7.1 同一策略，不同阶数
7.2 排序下界：为什么贪心常止步于O(nlogn)
7.3 比较器：正确性的隐形依赖
7.4 常见错误清单
7.5 用反例搜索代替直觉
7.6 习题

08 五个完整案例

8.1 案例一：缓存替换（Bélády 最优）
8.2 案例二：磁带存储（相邻交换的标准练习）
8.3 案例三：区间图着色（教室分配）
8.4 案例四：加权完成时间（Smith 法则）
8.5 案例五：带截止期的收益调度（拟阵落地）
8.6 部分习题提示

09 附录

A 符号表
B 定理索引
C 三个常数的来源
D 延伸阅读
E 排版说明

CHAPTER 01

贪心的形式化框架

> 大多数教材把贪心描述成一种“直觉”：每步选当前看起来最好的。但这个描述既不能判定一个贪心策略是否正确，也不能解释它错在哪里。本章要做的事，是把这种直觉换成一套可以书写、可以反驳、可以证明的形式语言。

1.1 一个问题，三种命运

考虑三个看起来同族的优化问题。给定n个物品，每件有重量w_i与价值v_i，容量为W。

- **分数背包**：物品可以任意切割。按v_i / w_i从大到小尽量装入。
- **0-1 背包**：物品要么整件装入，要么不装。同样按v_i / w_i贪心。
- **带截止期的任务调度**：每个任务有处理时间p_i与截止期d_i，最小化最大延迟。按d_i从小到大调度。

第一个贪心策略是**最优的**，第二个是**错的**，第三个又是**最优的**。三者的代码结构完全一致：排序、遍历、判定、接受。差别只藏在排序键与可行性判据里。

这说明「贪心算法」不是一个算法，而是一个**算法模板**。要谈它是否正确，必须先把这个模板的变量写清楚：候选集是什么、评价函数是什么、可行性怎么定义、决策顺序是否可回溯。这正是本章余下部分的任务。

1.2 决策序列模型

我们用一个五元组刻画贪心算法。这套记号在后续所有章节中通用。

定义 1.1（贪心算法的规范形式）

设E为有限候选集，$\mathcal{F} \subseteq 2^E$为可行解族。一个贪心算法由如下要素构成：

- 评价函数 $\sigma : E \to \mathbb{R}$，给出候选的优先级；
- 可行性判定 $\kappa : 2^E \to \{0, 1\}$，$\kappa(S) = 1$当且仅当$S \in \mathcal{F}$；
- 扩展规则：$S_{i+1} = S_i \cup \{e_{i+1}\}$，其中$e_{i+1}$是尚未考察的候选中$\sigma$值最大且满足$\kappa(S_i \cup \{e_{i+1}\}) = 1$者。

算法从$S_0 = \varnothing$出发，按σ降序扫描候选，得到序列$S_0 \subset S_1 \subset \cdots \subset S_k$，输出$S_k$。

这一模型包含三个隐含承诺，它们决定了贪心算法的能力边界。

(C1) **不可回退。** 一旦e被加入S，后续永不删除；一旦被跳过，永不再议。这使贪心成为**单遍**算法，也使它无法从早期错误中恢复。

(C2) **只看局部。** $\sigma(e)$只依赖e自身的属性，与当前已选集合S_i无关。若允许σ依赖S_i，算法就退化为一般的作用于集合函数的贪心，第 5 章会讨论这种推广。

(C3) **可行性单调。** 判定κ通常在S增长时越来越难满足。若κ不具备某种单调性，贪心甚至可能无法产生极大解。

> **一个必要的澄清**
>
> 条件 (C2) 并非所有被称为“贪心”的算法都满足。例如 Dijkstra 算法中，下一个确定标号的顶点取决于当前最短距离，是**动态**的优先级。严格说它属于“贪心 + 优先队列”的混合体。本书仍把它归入贪心，因为它的正确性论证（第 4 章）依赖的是同一套交换与归纳结构，而非全动态规划。

图 1.1 给出了这一模型的执行视图。注意那条被划掉的回边：它是贪心与回溯、局部搜索的分界线。

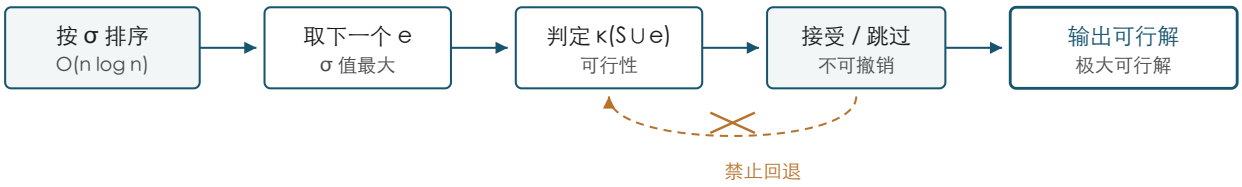


图 1.1　贪心算法的决策序列模型：排序、扫描、判定、输出，且不接受回退。

1.3 两个充分条件

形式化之后，可以写下正确性论证最常用的两个条件。

> **定义 1.2（贪心选择性质）**
>
> 称问题满足**贪心选择性质**，若存在某个最优解O^{*}，使得按σ排序后的第一个可行候选g_1满足$g_1 \in O^{*}$。

> **定义 1.3（最优子结构）**
>
> 称问题具有**最优子结构**，若对任意可行解S与任意最优解O^{*}，从O^{*}中删去已确定的决策后，剩余部分构成相应子问题的最优解。

命题 1.4

若问题同时满足贪心选择性质与最优子结构，则对每个实例，贪心算法输出的解都是最优解。

证明

对决策步数k作归纳。$k = 1$时，贪心选择性质直接给出$g_1 \in O^*$。

设贪心前k步的选择$g_1, \dots, g_k$均被某个最优解O^*包含。在剩余子问题中（候选集为E中尚未考察者，可行域为S_k的所有可行扩展），把$O^* \setminus S_k$视为该子问题的最优解。由最优子结构，$O^* \setminus S_k$是该子问题的最优解；对该子问题再次应用贪心选择性质，得到g_{k+1}属于某个最优扩展，从而存在最优解同时包含$g_1, \dots, g_{k+1}$。

归纳成立，故贪心输出S_k可扩展为最优解。又S_k本身是极大可行解，在目标函数单调（加入元素不会使目标变差）时S_k即最优。∎∎

这两个条件看似把问题解决了，实际上只是把困难后移：如何验证贪心选择性质？对具体问题逐个验证是可行的（正是第 3 章交换论证的做法），但每次都要重新动脑。第 2 章将说明，存在一整类结构——拟阵——在其中贪心选择性质**自动成立**，而且这个条件还是**必要**的。

1.4 三条证明路线

本书后续章节按三种技术组织，它们的适用范围与结论强度不同。

技术	适用结构	结论强度	主要代价
交换论证	几乎所有可证贪心	针对具体实例族，逐问题证明	需要构造交换，技巧性强
拟阵理论	独立系统上的加权选基	**充要条件**，一次刻画整类问题	只能覆盖拟阵结构
对偶与下界	近似算法、在线算法	给出比值保证与不可改进性	需构造对偶可行解或对抗实例

三者的关系可以这样理解：拟阵给出“为什么对”的代数解释；交换论证给出“对到什么程度”的归纳证明；对偶下界给出“能对到什么程度”的上限。一个完整的结论通常需要前两者之一加上第三者的紧性构造。

1.5 反例优先的设计心法

在设计阶段，最有价值的动作不是证明，而是**尽快杀死错误猜想**。给定一个候选贪心策略，按以下顺序检查：

1. **构造两元素反例。** 取$n = 2$，穷举所有参数组合，看贪心是否偏离最优。绝大多数错误策略在这一步就暴露。

2. **检查评价函数与目标函数是否同向**。若目标是最小化 $\max_j f_j$，而σ优化的是 $\sum_j f_j$，几乎必然出错（1.1 节的调度问题就是反例：按p_i排序会失败，按d_i排序才对）。
3. **检查不可回退是否致命**。若问题存在“早期选择严格排斥后期更优组合”的结构，贪心通常失效，应转向动态规划或近似算法。

第 7 章会给出自动化的反例搜索脚本，把第 1 步变成可执行程序。

本章小结

- 贪心是一个模板，由$(\sigma, \kappa, \mathcal{F})$参数化；不谈这三者而谈“贪心对不对”没有意义。
- 不可回退、只看局部、可行性单调是它的三个结构性约束，也是它失效的根源。
- 贪心选择性质 + 最优子结构 $\Rightarrow$ 最优，但这个条件只提供验证框架，不提供发现方法。
- 证明路线有三条：交换论证（通用）、拟阵（充要）、对偶下界（近似与紧性）。

CHAPTER 02

拟阵：贪心正确性的代数刻画

> 第 1 章的贪心选择性质是一个需要逐问题验证的条件。本章给出它的代数版本：存在一类组合结构（拟阵），在其上贪心选择性质恒成立；更关键的是，这个条件不仅充分，而且必要。这就是 Rado – Edmonds 定理，贪心理论中唯一一个“充要”级别的结果。

2.1 独立系统

定义 2.1（独立系统）

设E为有限集，$\mathcal{I} \subseteq 2^E$为非空集族。称$(E, \mathcal{I})$为**独立系统**，若

- (I1) $\varnothing \in \mathcal{I}$；
- (I2) **遗传性**：若$I \in \mathcal{I}$且$J \subseteq I$，则$J \in \mathcal{I}$。

$\mathcal{I}$中的集合称为**独立集**。极大独立集称为**基**，基的大小记为**秩**$r(E)$。

遗传性是一个弱的、几乎无信息量的要求：去掉元素不会破坏可行性。它对应第 1 章的 (C3)，即可行性判定随集合增长而单调变难。

例子 2.1　设E为图的边集，$\mathcal{I}$为所有无圈子集（森林）。遗传性显然：森林的子集仍是森林。基是生成树。

例子 2.2　设E为物品集，给定背包容量W，$\mathcal{I} = \left\{ S : \sum_{e \in S} w_e \le W \right\}$。遗传性成立，但下面的交换性不成立。

例子 2.3　设$E = \{1, \dots, n\}$，$\mathcal{I} = \left\{ S : |S| \le k \right\}$。称为一致拟阵 $U_{k,n}$。

2.2 拟阵的定义与交换公理

定义 2.2（拟阵）

独立系统$(E, \mathcal{I})$称为**拟阵**，若它还满足

- (I3) **交换公理**：对任意$A, B \in \mathcal{I}$，若$|A| < |B|$，则存在$e \in B \setminus A$使$A \cup \{e\} \in \mathcal{I}$。

交换公理说的是：较小的独立集总能被较大的独立集“补”上一个元素。它的直接推论是所有基等大——这一点使“最大权基”与“最大权独立集”在拟阵上重合，是后续所有结论的技术基础。

引理 2.3

设 $(E, \mathcal{I})$ 满足 (I1) – (I3)，则任意两个基大小相同。

证明

设 A, B 为两个基且 $|A| < |B|$。由 (I3)，存在 $e \in B \setminus A$ 使 $A \cup \{e\} \in \mathcal{I}$，与 A 极大矛盾。故 $|A| = |B|$。∎∎

图拟阵是理解交换公理最好的载体。

定理 2.4（图拟阵）

设 $G = (V, E)$ 为无向图，$\mathcal{I}$ 为 E 中所有无圈子集构成的族。则 $(E, \mathcal{I})$ 是拟阵。

证明

(I1)(I2) 已在例子 2.1 中验证。对 (I3)，取森林 A, B 满足 $|A| < |B|$。令 V' 为 $A \cup B$ 中出现的顶点集，记 $c(F)$ 为森林 F 在 V' 上的连通分量数。对顶点集固定的森林有

$$c(F) = |V'| - |F| \tag{2.1}$$

由 $|A| < |B|$ 得 $c(A) > c(B)$。于是存在两个顶点 x, y，它们在 (V', B) 中连通，但在 (V', A) 中分属不同分量。取 B 中 x 到 y 路径上第一个跨越 A 的不同分量的边 e。这样的 e 存在，因为路径起点与终点分属 A 的不同分量。于是 $e \in B$；又 A 中每条边两端都在同一分量内，故 $e \notin A$，即 $e \in B \setminus A$。加入 e 后不产生圈，故 $A \cup \{e\} \in \mathcal{I}$。(I3) 成立。∎∎



图 2.1　交换公理在图拟阵中的含义：较大的森林 B 必然提供一条跨接 A 两个分量的边。

2.3 秩函数

拟阵可以用秩函数等价刻画，这种视角在算法设计中更直接。

定义 2.5（秩函数）

对独立系统$(E,\mathcal{I})$与子集$A\subseteq E$，定义$r(A)=\max\{|I|:I\subseteq A,\ I\in\mathcal{I}\}$。

定理 2.6（秩函数刻画）

函数$r:2^E\to\mathbb{Z}_{\geq 0}$是某个拟阵的秩函数，当且仅当它满足

- (R1) $r(\varnothing)=0$；
- (R2) $r(A)\leq r(A\cup\{e\})\leq r(A)+1$；
- (R3) 次模性：$r(A)+r(B)\geq r(A\cup B)+r(A\cap B)$；
- (R4) $r(A)\leq|A|$。

此时$\mathcal{I}=\{I:r(I)=|I|\}$。

(R3) 是整章最重要的不等式。它说明秩函数的边际增益随集合增大而递减——这正是第 5 章“子模函数”的雏形。事实上，拟阵的独立性系统与“整数、次模、单调、增一”的秩函数是同一对象的两副面孔，这一对应使得拟阵论与子模优化共享同一套工具。

2.4 拟阵上的贪心算法

给出了独立系统与秩函数，贪心算法可以写得极简。

```
def greedy_matroid(E, w, oracle):
    """E: 候选集; w: 权重; oracle(S, e) -> S ∪ {e} 是否独立"""
    S = []
    for e in sorted(E, key=lambda x: -w[x]):
        if oracle(S, e):
            S.append(e)
    return S
```

定理 2.7（Rado–Edmonds，1957）

设$(E,\mathcal{I})$为有限独立系统，$w:E\to\mathbb{R}_{\geq 0}$。则贪心算法对每一个权重函数$w$都返回最大权基，当且仅当$(E,\mathcal{I})$是拟阵。

这是全章的核心。它的两个方向含义截然不同：充分性说明拟阵是贪心的“安全区”；必要性说明这个安全区不能再扩大——一旦脱离拟阵，就必然存在某个权重函数使贪心失败。

证明（充分性）

设$(E,\mathcal{I})$是拟阵。令$G=\{g_1,\ \dots,g_k\}$为贪心输出，按选择顺序排列；令$O=\{o_1,\ \dots,o_k\}$为某个最优基，按权重降序排列。由引理 2.3，$|G|=|O|=k$。

断言：对每个$i\in\{1,\ \dots,k\}$有$w(g_i)\ge w(o_i)$。

固定i。令$A=\{g_1,\ \dots,g_{i-1}\}$，$B=\{o_1,\ \dots,o_i\}$。二者均为独立集，且$|A|=i-1<i=|B|$。由交换公理，存在$o_t\in B\setminus A$使$A\cup\{o_t\}$独立。

由于$o_t\in\{o_1,\ \dots,o_i\}$且$O$按权重降序，有$w(o_t)\ge w(o_i)$。贪心在第$i$步是从所有能扩展$A$的候选中选取权重最大者，而$A\cup\{o_t\}$可行，故

$$w(g_i)=\max\{w(e):A\cup\{e\}\in\mathcal{I}\}\ge w(o_t)\ge w(o_i) \tag{2.2}$$

断言成立。求和即得$w(G)=\sum_i w(g_i)\ge\sum_i w(o_i)=w(O)$，故贪心最优。■■

证明（必要性）

设$(E,\mathcal{I})$不是拟阵，即交换公理失效：存在$A,B\in\mathcal{I}$，$|A|<|B|$，且对任意$x\in B\setminus A$都有$A\cup\{x\}\notin\mathcal{I}$。

令$k=|A|$，$l=|B|$，故$l>k$。取ε满足$0<\varepsilon<l/k-1$，在$E'=A\cup B$上定义权重

$$w(e)=\begin{cases}1+\varepsilon, & e\in A\\ 1, & e\in B\setminus A\end{cases} \tag{2.3}$$

其余元素权重取0。

贪心按权重降序扫描，先处理A中全部k个元素。由于A独立，其子集均独立，贪心将它们全部接受，得到当前集A。随后处理$B\setminus A$中的元素，按假设它们都无法加入A，全部被跳过。剩余权重为0的元素至多贡献0的目标值。于是

$$w(G)=(1+\varepsilon)\,k \tag{2.4}$$

而B是可行解，其权重为$w(B)=(1+\varepsilon)\,|A\cap B|+|B\setminus A|\ge l$。于是最优值满足$w(O)\ge l$。由$\varepsilon<l/k-1$得$(1+\varepsilon)k<l$，故$w(G)<w(O)$，贪心失败。■■

两个方向的证明都很短，但它们解释了为什么拟阵理论值得学：它把“这个贪心对不对”从逐个实例的验证，变成了一个结构判定问题——检查交换公理。

2.5 推论与边界

推论 2.8（Kruskal 的正确性）

设G为连通无向图，令$w'=-w$。对图拟阵应用定理 2.7，贪心按w'降序（即按w升序）选边并保持无圈，输出最小生成树。

这个推论值得注意：MST 的正确性不需要任何割性质论证，它是拟阵定理的一个特例。第 4 章会给出割性质的独立证明，那套证明对理解算法的结构更有帮助，但逻辑上是冗余的。

> **推论 2.9（拟阵上的调度）**
>
> 设任务集带截止期，$\mathcal{I}$为“可全部在截止期前完成”的任务子集族（单机、单位处理时间）。则$(E,\mathcal{I})$是拟阵，按价值降序贪心得到最大价值可行调度。

边界一：Huffman 不在其中。 最优前缀码问题的可行解是二叉树结构，不是子集族，$\mathcal{I}$无从定义，拟阵理论完全失效。它的正确性必须靠交换论证（第 3 章）。

边界二：背包不在其中。 例子 2.2 的独立系统不满足交换公理：容量$W=10$，取$A=\{w=9\}$，$B=\{w=5,w=4\}$，则$|A|<|B|$但A无法再扩展。这与定理 2.7 的必要性方向一致——背包的比值贪心确实会失败。

边界三：最短路不在其中。 路径不能构成子集族意义上的独立系统：两条路径的并集通常不是路径。第 4.5 节会展开这一点。

2.6 拟阵交：贪心失效的代数解释

若可行解族是两个拟阵独立族的交集，贪心立刻失效。

> **定义 2.10（k-拟阵）**
>
> 设$(E,\mathcal{I}_1),\ldots,(E,\mathcal{I}_k)$为$k$个拟阵，称$\mathcal{I}=\bigcap_j \mathcal{I}_j$为$k$-拟阵的独立族。$k=1$时退化为拟阵。

例子 2.4（二分图匹配） 设$G=(L\cup R,E)$为二分图。令$\mathcal{I}_L$为“每个左顶点至多关联一条边”的边集族，$\mathcal{I}_R$同理。二者都是划分拟阵。$\mathcal{I}_L\cap\mathcal{I}_R$恰为匹配集族。

命题 2.11 二分图上的最大匹配不能用上述贪心求得。

> 证明
>
> 取P_4路径$a-b-c-d$，二分划分为$\{a,c\}$与$\{b,d\}$，边集$E=\{ab,bc,cd\}$。赋予权重$w(bc)=2$，$w(ab)=w(cd)=1$。
>
> 贪心先取bc（权重最大且可行），此后ab与cd都与bc共享端点而被拒绝，输出匹配大小1。而$\{ab,cd\}$是大小为2的匹配。贪心失败。∎

命题 2.11 的证明说明：两个拟阵的交不再是拟阵，交换公理在交集上失效。这不是算法设计的疏漏，而是结构层面的必然——由定理 2.7 的必要性方向，只要交集不是拟阵，就必然存在某个权重使贪心失败，P_4只是把这个存在性显式构造出来。

定理 2.12（Edmonds 拟阵交定理，简述） 两个拟阵的交上的最大基数独立集可在多项式时间内求得（$O\left(n^3\right)$量级的增广路算法）。因此匹配有多项式算法，但不是贪心算法。

2.7 习题

2.1 证明一致拟阵$U_{k,n}$满足交换公理。

2.2 设G为连通图，$r(A)$为边集A的秩（即A的森林部分的最大边数）。验证r满足 (R1) – (R4)，并说明 (R3) 中等号成立的条件。

2.3 举出一个不是拟阵的独立系统，使得贪心算法仍然对所有权重函数正确。这可能吗？给出证明或反例。

2.4 设$\mathcal{I}$为“顶点集V中不含三角形的子集”构成的族。判断它是否为拟阵，并说明理由。

2.5 证明：拟阵的基族满足“基交换公理”——对任意两个基B_1, B_2与任意$x \in B_1 \setminus B_2$，存在$y \in B_2 \setminus B_1$使$B_1 - x + y$仍是基。

2.6 利用定理 2.7 的必要性方向，构造一个权重使“按度数从小到大选边”的生成树贪心失败。

CHAPTER 03

交换论证：最通用的正确性引擎

> 拟阵只覆盖了贪心问题的一小片。绝大多数可证贪心既不满足交换公理，也无法写成加权选基。对它们，需要一个更朴素的工具：把贪心解与最优解逐项对齐，用一次“交换”消除差异。本章把这个动作标准化，并在四个经典问题上完整执行。

3.1 论证的一般结构

设贪心输出$G = \{g_1, \dots, g_k\}$（按选择顺序），最优解为O。交换论证分三步。

定理 3.1（交换引理模板）

设可行域向下封闭、目标函数为集合权重$w(S) = \sum_{e \in S} w(e)$且$w \geq 0$。若

1. **起步条件**：存在最优解包含g_1；
2. **归纳条件**：对任意i与任意包含$\{g_1, \dots, g_{i-1}\}$的最优解O，存在最优解O'包含$\{g_1, \dots, g_i\}$，

则贪心输出G是最优解。

证明

从某个最优解出发，对$i = 1, 2, \dots, k$反复应用归纳条件，得到包含$\{g_1, \dots, g_k\} = G$的最优解O^*。

另一方面，G是极大可行解：若存在$e \notin G$使$G \cup \{e\}$可行，设e在贪心第j步被考察并拒绝，则$S_{j-1} \cup \{e\}$不可行；由向下封闭性，$G \cup \{e\} \supseteq S_{j-1} \cup \{e\}$也不可行，矛盾。故$O^*$只能等于$G$。

于是$w(G) = w(O^*) = OPT$。■■

这个模板把证明负担集中在唯一一处：**归纳条件**。它问的是“给定一个已经包含前$i-1$步选择的最优解，能否改造它使其也包含第i步的选择，且不损失最优性”。以下四节给出四种不同的回答方式。

这个模板是骨架，每个问题的血肉在于**如何证明可交换性**。以下四节给出四种典型手段：直接构造、相邻交换、结构归纳、以及通过下界间接证明。

3.2 区间调度：直接构造

问题　给定n个区间$[s_i, f_i)$，选出最大数量的互不重叠区间。

贪心　按f_i升序排序，依次接受与已选区间不冲突者。

这个策略的直觉是“尽早结束，给后面留空间”。证明的关键正是把直觉写成不等式。

定理 3.2

按结束时间最早优先的贪心算法输出最大基数的不相交区间集。

证明

设贪心输出$g_1, \dots, g_k$，最优解按结束时间升序为$o_1, \dots, o_m$。

断言：对每个$i \le \min(k, m)$有$f(g_i) \le f(o_i)$。

$i = 1$：g_1是全体区间中结束最早者，故$f(g_1) \le f(o_1)$。

归纳步：设$f(g_{i-1}) \le f(o_{i-1})$。因$o_{i-1}$与$o_i$不相交，有$s(o_i) \ge f(o_{i-1}) \ge f(g_{i-1})$，即$o_i$与$g_1, \dots, g_{i-1}$均不相交，它在贪心第$i$步时仍是候选。贪心选取候选中的最小结束时间者，故$f(g_i) \le f(o_i)$。

收尾：若$k < m$，则$f(g_k) \le f(o_k) \le s(o_{k+1})$，说明$o_{k+1}$与全部$g_1, \dots, g_k$相容，贪心不会在第$k$步停止，矛盾。故$k \ge m$，贪心最优。■■

注意证明的结构：断言用归纳，收尾用反证。这是交换论证最常见的组合。

复杂度　排序$O(n \log n)$，扫描$O(n)$，总计$O(n \log n)$，空间$O(1)$（不计输入）。

失效边界　若给区间加权，目标改为最大化权重和，按结束时间贪心立刻失效。

例 3.3（加权区间调度的反例）

取$A = [0, 10)$，权重100；$B = [0, 1)$、$C = [1, 2)$、$D = [2, 3)$、$E = [3, 10)$，权重均为1。

按结束时间贪心依次接受B, C, D, E，总权重4。最优解为单取A，权重100。

这个反例的诊断意义在于：它违反了 1.5 节第 2 条——评价函数（结束时间）与目标函数（权重和）不同向。加权区间调度需要动态规划，复杂度$O(n \log n)$（按结束时间排序后二分查找前缀）。

3.3 最小化最大延迟：相邻交换

问题　n个任务，处理时间$p_j > 0$，截止期d_j。单机非抢占，从时刻0开始。完成时刻C_j，延迟$L_j = C_j - d_j$（可为负）。目标：最小化$L_{\max} = \max_j L_j$。

贪心（EDF）　按d_j升序调度。

定理 3.4（Jackson 法则）

EDF 调度最小化最大延迟。

证明

设S为某个最优调度，且在所有最优调度中与 EDF 的**前缀公共长度最长**。若S不是 EDF 序，则存在相邻两个任务i, j，i紧接在j之前，但$d_i > d_j$。

交换i, j的位置得到S'。设交换前i从t开始，则$C_i = t + p_i$，$C_j = t + p_i + p_j$。交换后j先在t开始，$C'_j = t + p_j$，$C'_i = t + p_i + p_j = C_j$。

逐个检查延迟：

- 任务j：$C'_j = t + p_j < C_j$，故$L'_j < L_j$，不会变差。
- 任务i：$C'_i = C_j$，故$L'_i = C_j - d_i$。由$d_i > d_j$得$C_j - d_i < C_j - d_j = L_j \le L_{\max}(S)$。
- 其余任务完成时刻不变。

因此$L_{\max}(S') \le L_{\max}(S)$，即$S'$也是最优调度。但$S'$与 EDF 的前缀公共长度更长（消除了一个逆序对），与S的选取矛盾。

故最优调度中不存在这样的逆序对，即 EDF 序本身最优。■■

这个证明的技巧值得单列：**不在贪心解与最优解之间交换，而在两个最优解之间交换**。它把“存在一个最优解与贪心一致”这一断言，转化为“任何最优解都可以被局部的相邻交换推向贪心序”这一更强、也更容易验证的命题。

复杂度　排序$O(n\log n)$，计算完成时刻$O(n)$。

> **一个常见误解**
>
> EDF 最小化**最大**延迟，但不最小化**总**延迟$\sum_j L_j$。后者是 NP-难的（$1\|\sum T_j$），EDF 在其上可以任意差。目标函数的范数选择决定了问题的可解性，这是贪心设计中最容易踩的陷阱。

3.4 Huffman 编码：结构归纳

前两节的解是子集或序列。Huffman 的解是**树**，可行解族无法写成$\mathcal{F} \subseteq 2^E$，拟阵与子集交换都不适用。它的证明依赖“存在一个最优解具有特定形态”这一结构引理。

问题　给定字符频率$f_1, \dots, f_n > 0$，构造前缀码使期望码长$L = \sum_i p_i \ell_i$最小，其中$p_i = f_i / \sum_j f_j$。

约束　前缀码的码长必须满足 Kraft 不等式。

定理 3.5（Kraft 不等式）

存在码长为 $\ell_1, \dots, \ell_n$ 的二元前缀码，当且仅当

$$\sum_{i=1}^{n} 2^{-\ell_i} \le 1 \tag{3.1}$$

证明

把前缀码对应到一棵满二叉树的叶子：深度为 ℓ_i 的叶子占据二叉树上比例为$2^{-\ell_i}$ 的“概率质量”。前缀性质保证任意两个叶子互不为祖先，故它们占据的质量区间互不相交。总质量不超过1，即得不等式。

充分性（构造）：将 ℓ_i 非降序排列，维护一个二进制计数器c（初值0）。处理第i个符号时，把c的二进制表示补零至长度 ℓ_i 作为其码字，然后令$c \leftarrow c+1$并在右侧补零至长度 ℓ_i 。Kraft 不等式保证此时$c < 2^{\ell_i}$，码字长度总够用；由于每次分配后计数器加一，已分配的码字既不为新码字的前缀，也不会成为其扩展。■■

引理 3.6（最深叶子兄弟引理）

设x, y是频率最小的两个符号。则存在一棵最优前缀码树，其中x, y是深度最大的叶子且互为兄弟。

证明

取任意最优树T。令a为深度最大的叶子，b为其兄弟。由于x频率最小，$f_x \le f_a$。

把x与a的位置互换，得到T'。码长变化为

$$\Delta L = \left(f_x - f_a\right)\left(\ell_a - \ell_x\right) \le 0 \tag{3.2}$$

因为$f_x - f_a \le 0$且 $\ell_a - \ell_x \ge 0$（a最深）。同理把y与b互换，成本不增。两步之后x, y成为最深层的兄弟叶子，且总成本不超过原最优值，故仍最优。■■

引理 3.7（收缩引理）

把x, y合并为新符号z，频率$f_z = f_x + f_y$。则原问题的最优树与收缩后问题的最优树一一对应，且

$$L\left(T\right) = L\left(T'\right) + f_x + f_y \tag{3.3}$$

证明

任取收缩问题的树T'，把叶子z展开为内部结点带两个子叶x, y。设z在T'中深度为 ℓ_z ，则x, y在T中深度为 $\ell_z + 1$。于是

$$L(T) = \sum_{i \neq x,y} f_i \ell_i + (f_x + f_y)(\ell_z + 1) = L(T') + f_x + f_y \tag{3.4}$$

常数项不影响最小化，故两问题的最优树对应。■■

定理 3.8（Huffman 最优性）

Huffman 算法输出最优前缀码。

证明

对符号数n归纳。$n = 2$时两符号码长均为1，唯一且最优。

设$n \geq 3$。由定理 3.6，存在最优树T^*使频率最小的x, y为最深兄弟。把x, y收缩得z，由定理 3.7，T^*对应收缩问题的一棵最优树。收缩问题有$n-1$个符号，由归纳假设 Huffman 在收缩问题上输出最优树T'。Huffman 在原问题上的输出恰为T'展开z所得，由定理 3.7 它满足$L = L(T') + f_x + f_y = L(T^*)$，故最优。■■

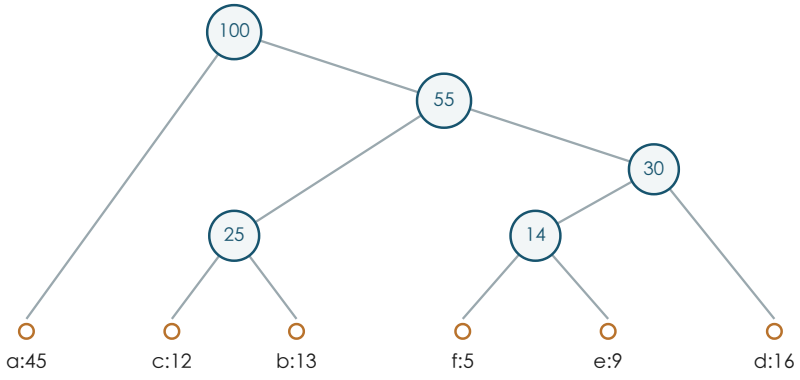


图 3.1　Huffman 树（频率 45, 12, 13, 5, 9, 16）：自底向上五步合并，内部结点标出合并后的频率。

复杂度　用二叉堆维护频率，每次合并两次弹出一次插入，共$n-1$次，每次$O(\log n)$，总计$O(n \log n)$。若频率是小整数，可用桶排序把排序降到$O(n)$（van Leeuwen 的两队列法可做到线性）。

3.4.1 熵下界与常数项的来源

Huffman 的最优性是对给定字母表而言的。它与信息熵的距离刻画了“前缀码的整数长度约束”到底代价多少。

定理 3.9（熵界）

设$H=-\sum_i p_i\log_2 p_i$，L为最优前缀码的期望码长，则

$$H\le L<H+1 \tag{3.5}$$

证明

下界：设最优码长为 ℓ_i，记$q_i=2^{-\ell_i}$，$K=\sum_i q_i\le 1$（Kraft）。令$q'_i=q_i/K$，则$\sum_i q'_i=1$，q'是一个概率分布。计算

$$H-L=-\sum_i p_i\log p_i+\sum_i p_i\log_2 q_i=-\sum_i p_i\log\frac{p_i}{q'_i}-\log_2 K \tag{3.6}$$

第一项是 KL 散度$D\left(p\,\middle\|\,q'\right)\ge 0$（由$\log x\le x-1$可得）；第二项因$K\le 1$而$-\log_2 K\ge 0$。故$H\le L$。

上界：取 $\ell_i^*=\left\lceil -\log_2 p_i\right\rceil$。则$2^{-\ell_i^*}\le p_i$，求和得$\sum_i 2^{-\ell_i^*}\le\sum_i p_i=1$，满足 Kraft，故存在这样的前缀码。其期望码长满足

$$L\le\sum_i p_i\,\ell_i^*<\sum_i p_i\left(-\log_2 p_i+1\right)=H+1 \tag{3.7}$$

最优码不劣于它，故$L<H+1$。■■

界$H\le L<H+1$是不可改进的：当所有$p_i=2^{-k}$时$L=H$；当分布极度偏斜时L可逼近$H+1$。这解释了为什么算术编码（可达到$H+O(1)$但常数更小）在压缩率上仅略优于 Huffman——差距被牢牢限制在1 bit 以内。

3.5 反例剖析：0-1 背包为何无法交换

第 3.1 节的模板要求“可交换性”。背包的失败可以从这个角度精确诊断。

贪心（比值序）　按v_i/w_i降序，能装则装。

例 3.10（比值贪心失败）

容量$W=10$。物品A：$w=6,v=6$；B：$w=5,v=4$；C：$w=5,v=4$。

比值：A为1.0，B,C为0.8。贪心先取A，剩余容量4装不下B或C，总价值6。最优取B,C，总价值8。

失败的机制在于：$g_1=A$之后，不存在包含A的最优解。断言“存在最优解包含g_1”直接破产。而这不是通过更好的排序键能修复的。

命题 3.11

对 0-1 背包，比值贪心满足$OPT < ALG + v_{\max}$，且该界紧。

证明

设贪心接受集合S后，物品j是第一个装不下的，记其比值为r_j。S中每个物品的比值不低于r_j，故

$$OPT \le v(S) + r_j\big(W - w(S)\big) < v(S) + r_j w_j = ALG + v_j \le ALG + v_{\max} \tag{3.8}$$

其中第一个不等式是分数背包给出上界，第二个用了$w(S) + w_j > W$。

紧性：容量$W = 2k$，一件物品$w = 2k$、$v = 2k$，另一件$w = k$、$v = k + 1$。贪心取后者，$ALG = k + 1$，而$OPT = 2k$，比值$(k + 1) / (2k) \to 1 / 2$。■■

推论是：比值贪心的近似比可以任意接近$1 / 2$，达不到常数级的好保证。真正有价值的方向是 FPTAS——Ibarra – Kim 算法用$O\big(n^3 / \varepsilon\big)$时间达到$1 - \varepsilon$近似，这是背包与集合覆盖的本质区别。

本章小结

- 交换论证的三段式：断言（用归纳）+ 交换（保持最优性）+ 收尾（用反证或计数）。
- 三种交换形态：直接构造（区间调度）、相邻交换（EDF）、结构归纳（Huffman）。
- 找到“存在最优解包含贪心首步”的**结构性理由**，是全部难点所在；找不到时，问题通常真的不可贪心。
- 熵界$H \le L < H + 1$说明 Huffman 的整数码长约束只值不到1 bit。

CHAPTER 04

图上的贪心：MST、最短路与并查集

> 图是贪心算法最密集的应用场。本章不满足于「Kruskal 是对的」这类结论，而是追问三个更细的问题：正确性的最小充分条件是什么（割性质）、同一条贪心在不同数据结构下的复杂度如何跨阶（并查集与堆）、以及为什么最短路明明也是图上贪心却不能纳入拟阵框架。

4.1 割性质与环性质

两个性质刻画了 MST 的局部最优结构，它们是所有 MST 算法的共同基础。

定义 4.1（割与跨边）

设$S \subset V$非空且非全集，称$(S, V \setminus S)$为一个割。两端分别在S与$V \setminus S$中的边称为跨边。若边集A中没有跨边，称割$(S, V \setminus S)$ 尊重 A。

定理 4.2（割性质）

设A是某棵最小生成树的子集，割$(S, V \setminus S)$尊重A，e是该割上权重最小的跨边。则$A \cup \{e\}$仍是某棵最小生成树的子集。

证明

设T为包含A的最小生成树。若$e \in T$，结论已成立。否则$T \cup \{e\}$含唯一环C。

由于e跨割，而C从S侧回到$V \setminus S$侧，C中必存在另一条跨边$e' \neq e$。由e的最小性，$w(e) \le w(e')$。

令$T' = T \cup \{e\} \setminus \{e'\}$。删除环上一条边后仍连通且边数为$n-1$，故$T'$是生成树，且

$$w(T') = w(T) + w(e) - w(e') \le w(T) \tag{4.1}$$

由T最小，$w(T') = w(T)$，T'也是最小生成树。又$e' \notin A$（否则A有跨边，与割尊重A矛盾），故$A \cup \{e\} \subseteq T'$。∎∎

定理 4.3（环性质）

设e不在最小生成树T中，则e在$T \cup \{e\}$所形成的环上是权重最大的边（之一）。

证明

若环上存在e'使$w(e') > w(e)$，则$T \cup \{e\} \setminus \{e'\}$是权重严格更小的生成树，与$T$最小矛盾。∎∎

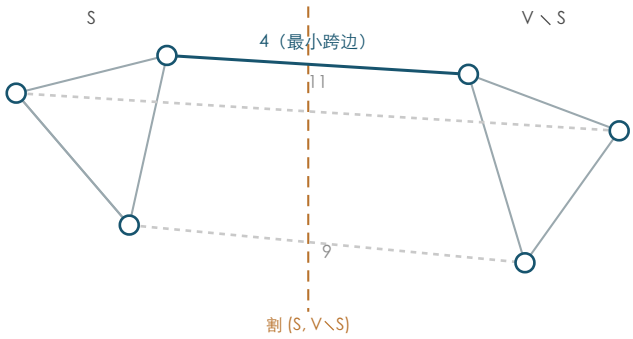


图 4.1　割性质：在尊重A的割上取最小跨边，加入A后仍可扩展为最小生成树。虚线为被割切断的边。

割性质的一句话版本是：**每条最小跨边都安全**。Kruskal 与 Prim 的差别只在于选择哪个割。

- **Kruskal**：固定按权重升序，割是隐式的——每次加入e时，取「e的两个端点各自的连通块」为割。
- **Prim**：固定割为「已选顶点集与其余顶点」，每次取该割上的最小边。

4.2 Kruskal 与并查集的均摊分析

Kruskal 的核心操作是「判断两端点是否已连通」与「合并两个连通块」，这正是并查集（disjoint set union）的职责。

```
class DSU:
    def __init__(self, n):
        self.p = list(range(n))
        self.r = [0] * n          # 秩 (rank)

    def find(self, x):
        while self.p[x] != x:
            self.p[x] = self.p[self.p[x]]   # 路径减半
            x = self.p[x]
        return x

    def union(self, a, b):
        a, b = self.find(a), self.find(b)
        if a == b:
            return False
        if self.r[a] < self.r[b]:
            a, b = b, a
        self.p[b] = a
        if self.r[a] == self.r[b]:
            self.r[a] += 1
        return True
```

定理 4.4（并查集的均摊界）

配合按秩合并与路径压缩，任意m次操作的均摊代价为$O(\alpha(n))$，其中α为反阿克曼函数。

反阿克曼函数的增长极慢：对$n < 2^{65536}$有$\alpha(n) \le 4$。因此在任何可想象的输入规模下，它都可视为常数。

证明要点

只有按秩合并：秩严格随合并递增，且秩为r的树至少含2^r个结点，故树高$O(\log n)$。

叠加路径压缩：Tarjan 的势能论证把结点分为「好」「坏」两类，对每个结点记录其秩的变化次数。关键观察是：一个结点每次被路径压缩“跳过”后，它所在子树的根秩至少翻倍，因此同一结点被压缩的次数不超过$\log n$次；把所有结点按其秩分级累计，总代价被$O\big((m+n)\log n\big)$收紧为$O(m\,\alpha(n))$。完整的双重归纳证明见 Tarjan (1975)。■■

复杂度　排序$O(m\log m)$，并查集$O(m\,\alpha(n))$，总计

$$T_{\text{Kruskal}} = O(m\log m) \tag{4.2}$$

对稀疏图$m = O(n)$时是$O(n\log n)$，已经达到比较模型下的排序下界（见 7.2 节），因此 Kruskal 在稀疏图上是最优阶的。

4.3 Prim 与优先队列的选择

Prim 维持一个顶点集U与当前最小跨边，用优先队列维护每个外部顶点到U的最小边权。

```
import heapq

def prim(adj, s):
    n = len(adj)
    done = [False] * n
    dist = [float('inf')] * n
    dist[s] = 0
    h = [(0, s)]
    total = 0
    while h:
        d, u = heapq.heappop(h)        # 惰性删除：跳过过期项
        if done[u]:
            continue
        done[u] = True
        total += d
        for v, w in adj[u]:
            if not done[v] and w < dist[v]:
                dist[v] = w
                heapq.heappush(h, (w, v))
    return total
```

复杂度 每条边至多触发一次入堆，堆操作$O(m)$次，每次$O(\log n)$，故$O(m\log n)$。若改用斐波那契堆，减键操作$O(1)$摊还，降到$O(m+n\log n)$——稠密图（$m=\Theta(n^2)$）下这是$O(n^2)$，与朴素的邻接矩阵实现同阶，但常数更小。

算法	朴素实现	二叉堆	斐波那契堆
Prim（MST）	$O(n^2)$	$O(m\log n)$	$O(m+n\log n)$
Dijkstra	$O(n^2)$	$O((m+n)\log n)$	$O(m+n\log n)$
Kruskal（MST）	$O(m\log m)$，排序主导	同左	同左

> **表格的读法**
>
> 不要把这看作“斐波那契堆总更好”。它的常数与实现复杂度都高，实践中几乎从不使用。表格真正的信息是：**同一贪心策略的复杂度阶由数据结构决定**，而渐近最优的数据结构往往因常数过大而不实用。第 7 章会把这一点展开为工程判据。

4.4 Dijkstra：非负权的不变量证明

Dijkstra 与 Prim 的代码几乎同构，差别只在“距离”的定义：Prim 中$d(v)$是v到**已选集**的最小边权，Dijkstra 中$d(v)$是v到**源点**的最短路径长度。

定理 4.5

设所有边权$w(e) \geq 0$，Dijkstra 算法正确计算单源最短路径。

证明

维护不变量：**每次从堆中取出并“确定”顶点u时，$d[u] = \delta(s,u)$**，其中δ为真实最短距离。

下界方向$d[u] \geq \delta(s,u)$恒成立，因为$d[u]$始终对应某条实际路径的长度。

上界方向用反证。设u是第一个被确定但$d[u] > \delta(s,u)$的顶点。取s到u的最短路P。因s已确定而u未确定，P上存在第一条离开已确定集S的边(x,y)，$x \in S$，$y \notin S$。

由于P是最短路且边权非负，P的前缀也是最短的，故

$$\delta(s,y) = \delta(s,x) + w(x,y) \tag{4.3}$$

由归纳$d[x] = \delta(s,x)$。x被确定时对所有出边做过松弛，故$d[y] \leq d[x] + w(x,y) = \delta(s,y)$。

现在比较y与u：u是当前未确定顶点中d值最小者。又$\delta(s,y) \leq \delta(s,u)$（$y$在$P$上，位于$u$之前），于是

$$d[u] \leq d[y] \leq \delta(s,y) \leq \delta(s,u) \tag{4.4}$$

与假设$d[u] > \delta(s,u)$矛盾。故不变量成立，算法正确。■■

证明中“边权非负”用在两处：P的前缀最短，以及$\delta(s,y) \leq \delta(s,u)$。去掉任何一处都会破坏结论。

例 4.6（负权反例）

三点s,a,b，边权$w(s,a) = 2$，$w(s,b) = 3$，$w(b,a) = -2$。

Dijkstra 先确定s（$d=0$），松弛得$d[a]=2$、$d[b]=3$；接着取出a（$d=2$）并确定，输出$\delta(s,a) = 2$。但真实最短路为$s \to b \to a$，长度$3-2=1$。错误来自“a的前缀最短”这一前提失效。

负权图需要 Bellman – Ford（$O(nm)$）或 Johnson 重标号。这不是贪心策略的改进问题，而是贪心所依赖的性质不存在。

4.5 为什么最短路不能写成拟阵问题

MST 与最短路在代码上如此相似，为什么一个能用拟阵解释、另一个不能？原因是两者的解空间结构根本不同。

维度	最小生成树	单源最短路
解的形态	边子集（生成树）	路径（顶点序列）
可行性封闭性	向下封闭（森林的子集仍是森林）	不封闭（路径的子集不是路径）
目标函数	边权之和（线性）	路径长度（路径上的和）
贪心合法性	拟阵交换公理	非负权三角不等式
失效条件	无	存在负权边

最短路解族不满足遗传性，甚至不能写成2^E的子集族——一条路径去掉中间某条边就不再连通。因此拟阵框架从第一步就无法启动。Dijkstra 的正确性完全依赖δ的**最优子结构**（最短路的子路最短），这恰好是贝尔曼方程的前提，也是动态规划的条件。

> **一个容易混淆的点**
>
> 「MST 可以用 Kruskal 贪心」与「MST 问题可以写成拟阵上的加权选基」是同一件事的两种说法；而「Dijkstra 是贪心」只是在描述**行为**，不是在描述**结构**。前者有代数刻画，后者没有。混淆这两者，会导致在设计新算法时错误地期待一个“贪心就够了”的结论。

4.6 习题

4.1 证明：若图中所有边权互不相同，则最小生成树唯一。

4.2 举出一个图，使得 Kruskal 与 Prim 产生边权相同但边集不同的两棵 MST。

4.3 设T为 MST。若把某条非树边e的权重减小，如何用$O(m)$时间更新 MST？给出算法并证明正确性。

4.4 证明：Dijkstra 在边权允许为0时仍正确。

4.5 分析用「d数组 + 线性扫描最小值」实现的 Dijkstra 在稠密图上的复杂度，并与二叉堆版本比较，给出切换阈值。

4.6 构造一个图，使得按「边权 / 两端剩余度」排序的生成树贪心失败。

CHAPTER 05

贪心的近似：当精确不可得

> 前面的章节都在回答「贪心何时给出最优解」。但大量问题是 NP-难的，最优解不可能在多项式时间内得到。此时贪心的角色发生转变：它不再是求解器，而是一个**比值保证的构造器**。本章的核心量是近似比，核心方法是把贪心的每一步失败量化为一个可求和的上界。

5.1 近似比的语言

定义 5.1（近似比）

设Π为最小化问题，$ALG(I)$与$OPT(I)$分别为算法与最优解在实例I上的目标值。若对一切实例I有

$$ALG(I) \le \rho \cdot OPT(I) \tag{5.1}$$

则称算法为ρ-近似算法，$\rho \ge 1$为近似比。最大化问题取反号，$\rho \le 1$。

近似比是最坏情况保证，因此它的证明结构天然是对抗式的：必须对所有实例同时成立。常用的两种技术是**势函数法**（找出一个单调下降的上界量）与**收费论证**（把算法支付的成本分摊到最优解的元件上）。

与近似比相关的还有两个更强的概念：PTAS（对任意$\varepsilon > 0$，存在$(1+\varepsilon)$-近似算法，复杂度可依赖ε）与 FPTAS（要求复杂度对$1/\varepsilon$也是多项式）。背包有 FPTAS，集合覆盖没有——这个差别来自问题结构的根本不同：集合覆盖的困难是组合爆炸式的，而背包的困难只在数值精度。

5.2 集合覆盖：$\ln n + 1$

问题　全集U，$|U| = n$，集合族$\mathcal{S} = \{S_1, \dots, S_m\}$，$\bigcup_j S_j = U$。求最少的集合覆盖$U$。

贪心　每次选取能覆盖最多**未覆盖**元素的集合。

定理 5.2

集合覆盖贪心是$(\ln n + 1)$-近似的。

证明

设$OPT = t^*$。令n_i为第i步开始前未被覆盖的元素数，$n_1 = n$。

关键不等式：$n_{i+1} \le n_i\left(1 - \frac{1}{t^*}\right)$。

理由：最优解中的t^*个集合覆盖全部n_i个未覆盖元素，故其中至少一个覆盖不少于n_i / t^*个。贪心选的是覆盖最多者，故本步至少新覆盖n_i / t^*，剩余不超过$n_i\left(1 - 1/t^*\right)$。

递推得

$$n_i \le n\left(1 - \frac{1}{t^*}\right)^{i-1} \le n\,e^{-(i-1)/t^*} \tag{5.2}$$

取$i = t^*\ln n + 1$，得$n_i \le n e^{-\ln n} = 1$。由于n_i是整数，$n_i \le 1$且继续下去必然归零，故贪心至多用$t^*\ln n + 1$个集合，即$ALG \le OPT \cdot \ln n + 1 \le (\ln n + 1)\,OPT$。■■

紧性　该比值不可改进到$o(\ln n)$：Feige (1998) 证明，除非 NP 有准多项式算法，任何多项式时间算法都无法达到$(1-\varepsilon)\ln n$的近似比。集合覆盖因此是「贪心即最优阶」的典型——不是因为贪心多聪明，而是因为问题本身就这么难。

加权版本　用收费论证可把结论推广到加权集合覆盖：设贪心第i步选中集合S，覆盖c_i个新元素，代价$w(S)$。给每个新覆盖元素记价$w(S)/c_i$。可以证明$w(S)/c_i \le OPT/n_i$（否则最优解在剩余元素上就会超支），于是

$$\sum_{i} w(S_i) \le OPT \sum_{i} \frac{c_i}{n_i} \le OPT \cdot H_n \le OPT\left(\ln n + 1\right) \tag{5.3}$$

其中$\sum_i c_i / n_i$恰是调和级数的前若干项，因为每步n_i递减而c_i恰好是它的减少量。

5.3 子模最大化：$1 - 1/e$

集合覆盖的贪心有一个隐含结构：目标函数「覆盖元素数」是**子模**的。把这一点抽象出来，就得到一类统一的近似结论。

定义 5.3（子模函数）

集合函数$f: 2^E \to \mathbb{R}_{\ge 0}$称为**子模**的，若对任意$A \subseteq B \subseteq E$与$e \notin B$，有

$$f(A \cup \{e\}) - f(A) \ge f(B \cup \{e\}) - f(B) \tag{5.4}$$

即**边际收益递减**。若还有$A \subseteq B \Rightarrow f(A) \le f(B)$，称为**单调**子模。

记$\Delta(e \mid S) = f(S \cup \{e\}) - f(S)$。子模性就是说$\Delta(e \mid \cdot)$随集合增大而单调不增。拟阵的秩函数（定理 2.6 的 (R3)）正是子模函数的一个特例——这不是巧合，而是拟阵论与子模优化共享同一套代数结构的表现。

问题　单调子模f，基数约束$|S| \le k$，求 $\max f(S)$。

贪心　每次选边际收益最大者。

> **定理 5.4（Nemhauser–Wolsey–Fisher，1978）**
> 对单调子模最大化问题，贪心算法满足$f\left(S_k\right) \ge \left(1-\frac{1}{e}\right) f\left(S^*\right)$。

证明

设S_i为贪心的前i步输出，S^*为最优解，$|S^*| = k$，并设$f(\varnothing) = 0$。记$S^* = \{e_1, \dots, e_k\}$。

第一步（单调性）：$f\left(S^*\right) \le f\left(S_i \cup S^*\right)$。

第二步（子模性）：把S^*逐个加入S_i，每步的边际收益不超过单独加入时的边际收益，

$$f\left(S_i \cup S^*\right) = f\left(S_i\right) + \sum_{j=1}^{k} \Delta\left(e_j \mid S_i \cup \left\{e_1, \dots, e_{j-1}\right\}\right) \le f\left(S_i\right) + \sum_{j=1}^{k} \Delta\left(e_j \mid S_i\right) \tag{5.5}$$

第三步（贪心选择）：每项都不超过贪心本步的最优边际收益，

$$\sum_{j=1}^{k} \Delta\left(e_j \mid S_i\right) \le k \cdot \max_{e} \Delta\left(e \mid S_i\right) = k\left(f\left(S_{i+1}\right) - f\left(S_i\right)\right) \tag{5.6}$$

合并三步得$f\left(S^*\right) \le f\left(S_i\right) + k\left(f\left(S_{i+1}\right) - f\left(S_i\right)\right)$，整理为

$$f\left(S^*\right) - f\left(S_{i+1}\right) \le \left(1-\frac{1}{k}\right)\left(f\left(S^*\right) - f\left(S_i\right)\right) \tag{5.7}$$

对$i = 0, \dots, k-1$迭代（初值$f\left(S_0\right) = 0$）：

$$f\left(S^*\right) - f\left(S_k\right) \le \left(1-\frac{1}{k}\right)^{k} f\left(S^*\right) \le e^{-1} f\left(S^*\right) \tag{5.8}$$

最后一步用了$\left(1-1/k\right)^{k} \le e^{-1}$。移项即得$f\left(S_k\right) \ge \left(1-1/e\right) f\left(S^*\right)$。∎∎

这个证明的结构值得记住：**三次不等式，每次只用一个性质**（单调、子模、贪心最优性）。它之所以能给出与k无关的常数$1-1/e$，关键在于第三步把k项的求和统一放大为同一个最大值，再由$\left(1-1/k\right)^{k}$的极限回收k。

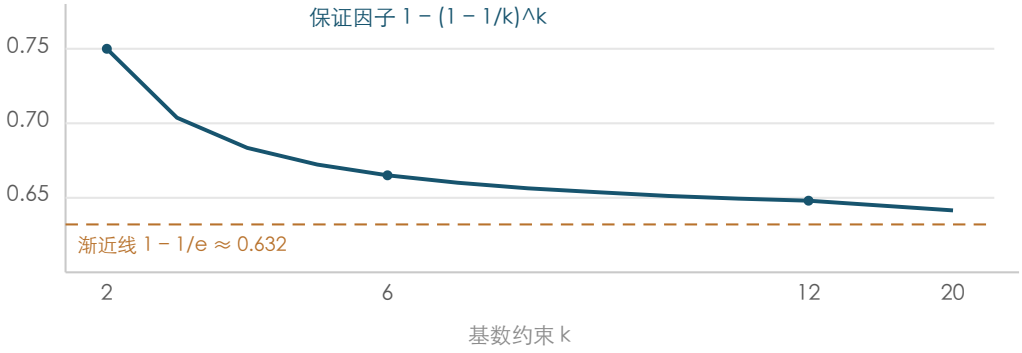


图 5.1　贪心保证因子$1-(1-1/k)^k$随基数约束k的收敛：从$k=2$的0.75单调下降到$1-1/e$，与渐近线的差距以$O(1/k)$的速度消失。

紧性　$1-1/e$不能再改进。对最大覆盖问题（f为覆盖元素数，是单调子模的），Feige (1998) 证明：除非 P = NP，任何多项式算法都达不到$(1-1/e+\varepsilon)$。因此定理 5.4 不只是贪心的下界，它同时是这一类问题多项式可解性的天花板。

5.4 拟阵约束与更弱的保证

把基数约束换成拟阵约束，$1-1/e$会退化。

问题　单调子模f，拟阵$\mathcal{M}=(E,\mathcal{I})$，求 $\max\{f(S):S\in\mathcal{I}\}$。

贪心（按f的边际收益排序，保持独立性）此时只能保证$1/2$。原因是定理 5.4 第二步的「逐个加入」论证不再成立：S^*的元素不一定能逐个加入S_i而保持独立，交换公理只保证能加入**某个**元素，不能保证加入**指定的**元素。这一处损失直接削掉了$1-1/e-1/2$的差距。

更好的算法（连续贪心 + 舍入，Calinescu 等 2011）也能达到$1-1/e$，但需要解一个连续松弛，不再是纯组合贪心。这个例子说明：**约束越强，纯贪心的保证越弱；恢复保证通常要付出算法复杂度的代价。**

5.5 k-中心：2-近似及其紧性

问题　度量空间中的点集P，$|P|=n$，选k个中心使最大服务半径最小。

贪心（Gonzalez 算法）　任选一个中心；此后每次选距离当前中心集**最远**的点作为新中心，直到选满k个。

> **定理 5.5**
>
> Gonzalez 算法是2-近似的。

证明

设$c_1, \dots, c_{k+1}$为算法依次选出的前$k+1$个中心，记$r_j = \max_{x \in P} d\left(x, \{c_1, \dots, c_j\}\right)$。由「选最远点」的规则，$c_{j+1}$满足$d\left(c_{j+1}, \{c_1, \dots, c_j\}\right) = r_j$，且$r_1 \geq r_2 \geq \cdots$。

于是对任意$i < j \leq k+1$，有$d\left(c_i, c_j\right) \geq r_{j-1} \geq r_k$。即前$k+1$个中心两两距离不小于$r_k$。

算法用k个中心时的半径恰为$ALG = r_k$。另一方面，最优解的k个中心要把这$k+1$个两两距离$\geq r_k$的点全部覆盖，由抽屉原理，必有两个点由同一个中心服务，该中心到其中至少一个点的距离$\geq r_k / 2$。故$OPT \geq r_k / 2$。

合并得$ALG = r_k \leq 2\,OPT$。■■

k-中心是 NP-难的，且2-近似是紧的：除非 P = NP，不存在$(2-\varepsilon)$-近似算法。这与集合覆盖的$\ln n$一样，说明某些问题的近似比上限被计算复杂性锁死，贪心恰好触到了那条线。

本章小结

- 近似比的证明有两种主流结构：势函数（集合覆盖）与三次不等式链（子模最大化）。
- 集合覆盖$\ln n + 1$，子模最大化$1 - 1/e$，k-中心2——三个常数分别对应三种不同的结构性障碍。
- 三个界都是紧的（在 P ≠ NP 假设下），说明贪心在这些问题上的表现已经到了理论极限。
- 拟阵约束会削弱保证（$1 - 1/e \to 1/2$），因为交换公理只保证「存在」可加入元素，不保证「指定」元素可加入。

5.6 习题

5.1 证明：集合覆盖的目标函数是单调子模的。

5.2 在定理 5.2 的证明中，若把t^*换成OPT的实际值，界是否还能改进？说明为什么不能。

5.3 构造一个k-中心实例，使 Gonzalez 算法的近似比恰为2。

5.4 设f为单调子模，g为单调超模（边际收益递增）。判断 $\max f$与 $\min g$哪个能被贪心有效近似，给出理由。

5.5 证明$1 - 1/e$是$1 - (1 - 1/k)^k$当$k \to \infty$的极限，并给出误差项的量级。

5.6 加权集合覆盖的收费论证中，为什么$w(S) / c_i \leq OPT / n_i$？给出完整论证。

CHAPTER 06

在线与随机：信息缺失下的贪心

> 前几章都假设能看到全部输入。在线场景下，输入逐日到达，算法必须在看到未来之前做决定，且决定不可撤销——这正是第 1 章 (C1) 的极端情形。此时“最优”不再可比，替代品是**竞争比**：与事后最优（全知者）的比值。

6.1 竞争比

定义 6.1（竞争比）

设算法A对在线实例序列σ的输出为$A(\sigma)$，事后最优为$OPT(\sigma)$。若对一切σ有$A(\sigma) \le \rho\, OPT(\sigma)$，称$A$为$\rho$-竞争的。确定性算法取$\rho$的下确界为竞争比；随机化算法对期望值定义。

竞争比与近似比的形式完全一致，但内涵不同：近似比的困难来自计算复杂性（算不完），竞争比的困难来自**信息缺失**（看不到）。这个区别决定了证明工具：近似比用不等式链，竞争比用**对抗构造**（给每个确定性策略配一个最坏输入）。

6.2 滑雪租赁

问题　滑雪场每日租金1，购买雪板一次B。你不知道会滑多少天，每天开始时决定租还是买（买了之后不再付费）。目标：最小化总花费。

若知道总天数d，最优花费为 $\min(d, B)$。在线情形下：

定理 6.2

确定性竞争比恰为$2 - \frac{1}{B}$，由策略「先租$B-1$天，第B天购买」达到。

证明（上界）

按策略，第B天购买。

- 若$d < B$：全程租赁，花费$ALG = d$，而$OPT = \min(d, B) = d$，比值为1。
- 若$d \geq B$：租赁$B-1$天并购买，$ALG = (B-1) + B = 2B-1$，而$OPT = B$，比值为

$$\frac{2B-1}{B} = 2 - \frac{1}{B} \tag{6.1}$$

故竞争比不超过$2 - 1/B$。■■

证明（下界）

设某确定性策略首次购买发生在第t天。分两种情形构造对抗输入：

- 若$t \leq B-1$：令$d = t$。策略花费$ALG = (t-1) + B$，而$OPT = t$，比值

$$\frac{t-1+B}{t} = 1 + \frac{B-1}{t} \geq 1 + \frac{B-1}{B-1} = 2 > 2 - \frac{1}{B} \tag{6.2}$$

- 若$t \geq B$：令$d \geq t$充分大。策略花费$ALG = (t-1) + B \geq 2B$，而$OPT = B$，比值≥ 2。

两种情形下比值都不低于$2 - 1/B$，故任何确定性策略都无法突破此界。■■

随机化　若把购买日随机化，可把竞争比降到

$$\frac{e}{e-1} \approx 1.582 \tag{6.3}$$

构造思路：令购买日服从某个在$\{1, \dots, B\}$上的分布p，使任意截止日d下的期望花费与$OPT(d)$的比值一致地被$e/(e-1)$控制；对p解一个线性规划即得最优分布。下界方向由 Yao 原理给出：构造一个输入分布，使任何确定性策略在该分布上的期望花费至少是$e/(e-1)$倍的事后期望最优。

从2到1.58，随机化把确定性下界削掉了约21%。这是在线算法中反复出现的模式：随机化不能突破“信息缺失”的定性障碍，但能把常数压到接近最优。

6.3 秘书问题：$1/e$的另一处现身

问题　n个候选人随机顺序逐个到达，面试后必须立即决定录用或拒绝，不可回退。目标：以最大概率选中绝对最优者。

策略　先面试并全部拒绝前$k = \lfloor n/e \rfloor$人（只观察），此后录用第一个优于前k人者。

定理 6.3

该策略的成功概率趋于$1/e$。

证明要点

最优者出现在位置i的概率为$1/n$。给定其在位置$i>k$，策略成功当且仅当前$i-1$人中的最优秀者落在前k个位置（否则策略会在i之前录用某人）。该条件概率为$k/(i-1)$。故

$$P\left(\text{成功}\right)=\sum_{i=k+1}^{n}\frac{1}{n}\cdot\frac{k}{i-1}=\frac{k}{n}\sum_{j=k}^{n-1}\frac{1}{j}\approx\frac{k}{n}\ln\frac{n}{k} \tag{6.4}$$

取$k=n/e$得$P\approx\frac{1}{e}\ln e=\frac{1}{e}$。对$k$求导可验证这是最优的观察长度。■■

$1/e$在本书出现了两次：子模最大化的保证因子是$1-1/e$，秘书问题的成功率是$1/e$。二者都源于$\left(1-1/n\right)^{n}\to e^{-1}$这一极限，但机理不同——前者是“每步损失$1/k$的累积”，后者是“最优者恰好落在观察窗之后的概率”。

6.4 乘性权重与遗憾

在线学习的框架更弱：不仅看不到未来，连“最优固定解”也是事后的。此时用**遗憾**（regret）度量：

$$R_T=\sum_{t=1}^{T}\ell_t\left(a_t\right)-\min_{a\in\mathcal{A}}\sum_{t=1}^{T}\ell_t\left(a\right) \tag{6.5}$$

Hedge（乘性权重更新）算法维护每个动作的权重w_a，损失后按$w_a\leftarrow w_a\left(1-\eta\,\ell_t\left(a\right)\right)$衰减。取$\eta=\sqrt{8\ln|\mathcal{A}|/T}$可得

$$R_T\le\sqrt{\frac{T\ln|\mathcal{A}|}{2}} \tag{6.6}$$

即平均遗憾$O\left(\sqrt{\ln|\mathcal{A}|/T}\right)\to 0$。这是“贪心”在信息最匮乏的设定下能达到的最好结果：它不再保证与最优解只差常数倍，而是保证**随时间趋于不差**。

6.5 在线集合覆盖

在线集合覆盖中，元素逐个到达，算法必须立即选一个包含它的集合。Alon 等（2003）证明贪心策略的竞争比为$O\left(\log n\right)$，且这个阶是最优的——与离线情形的$\ln n+1$同阶。

这个结论的意义在于：**信息缺失不一定改变复杂度阶**。滑雪租赁中信息缺失把比值从1拉到2（常数倍），而在线集合覆盖只把常数从$\ln n$拉到$c\ln n$。判断一个在线问题“难到什么程度”，要看最优解对未来的依赖是常数级的还是结构级的。

本章小结

- 竞争比衡量信息缺失的代价，用对抗构造证明下界，用策略分析证明上界。
- 滑雪租赁：确定性$2-1/B$，随机化$e/(e-1)$。
- 秘书问题：观察n/e后录用，成功率$1/e$。
- 乘性权重把“与最优的比值”换成“随时间消失的遗憾”，是在线学习的基本范式。

6.6 习题

6.1 证明滑雪租赁的随机化策略不存在优于$e/(e-1)$的竞争比（用 Yao 原理构造输入分布）。

6.2 若租金为r、购买价为B，重做定理 6.2，给出竞争比关于B/r的表达式。

6.3 秘书问题中若允许录用后被更优者替换（但替换有代价c），问题结构如何变化？

6.4 证明 Hedge 算法的遗憾界，并说明为什么η的取值是$\Theta\left(1/\sqrt{T}\right)$而不是$\Theta(1/T)$。

6.5 构造一个在线实例，使“每隔固定天数购买”的策略竞争比趋于无穷。

CHAPTER 07

复杂度工程与实现陷阱

> 一个贪心算法在纸上是$O(m\log m)$，在代码里可能是$O(mn)$；一个在实数上正确的比较器，在浮点上可能违反传递律并导致未定义行为。本章处理从证明到实现的落差，重点在三件事：数据结构的阶数影响、比较模型的下界、以及比较器与浮点的具体陷阱。

7.1 同一策略，不同阶数

贪心的骨架几乎不变：排序、扫描、判定。真正决定复杂度的是判定用的数据结构。

问题	判定操作	朴素结构	进阶结构	复杂度变化
区间调度	与前一个结束时间比较	变量	变量	$O(n\log n)$（排序主导）
MST	连通性查询	DFS 每次$O(n+m)$	并查集	$O(mn)\to O(m\log m)$
MST	取最小跨边	线性扫描	二叉堆	$O(n^2)\to O(m\log n)$
单源最短路	取最小d	线性扫描	二叉堆 / 斐波那契堆	$O(n^2)\to O(m\log n)\to O(m+n\log n)$
Huffman	取两个最小频率	排序一次后双指针	二叉堆	$O(n\log n)$，双指针可到$O(n)$
集合覆盖	统计新覆盖数	每轮重扫$O(mn)$	惰性堆（lazy greedy）	$O(mn)\to O(mn\log)$均摊更优

最后一行值得单独说。集合覆盖的朴素实现每轮重算所有集合的边际收益，复杂度$O(mnk)$。惰性贪心利用子模性带来的一个性质：边际收益只会下降，因此可以用最大堆缓存上界，弹出后重算，若仍不小于堆顶则确定选中。实测中它把常数降低一到两个数量级，而渐近界不变。

阶数与常数的分工

渐近阶只在n足够大时才有指导意义。对$n\le 1000$的输入，$O(n^2)$的数组实现常常快过$O(n\log n)$的堆实现，因为堆的每次操作有 cache miss 与常数开销。工程判据不是“选阶数更低的”，而是“先测，再按输入规模分段选择”。

7.2 排序下界：为什么贪心常止步于$O(n\log n)$

大量贪心算法的复杂度被排序卡住。这不是实现问题，而是**模型下界**。

> **定理 7.1（比较排序下界）**
>
> 任何基于两两比较的排序算法在最坏情况下至少需要$\Omega(n\log n)$次比较。

证明

n个元素有$n!$种排列，算法必须区分全部$n!$种输出。把算法的执行过程看成二叉决策树：每个内部结点是一次比较，叶子是一个输出排列。高度为h的二叉树至多有2^h个叶子，故

$$2^h \geq n! \Longrightarrow h \geq \log_2(n!) \tag{7.1}$$

用 Stirling 估计或简单放缩：$\log_2(n!) = \sum_{k=1}^{n} \log_2 k \geq \sum_{k=\lceil n/2 \rceil}^{n} \log_2 k \geq \frac{n}{2}\log_2 \frac{n}{2} = \Omega(n\log n)$。∎∎

这条下界解释了为什么 Kruskal 在稀疏图上的$O(n\log n)$是最优的：任何 MST 算法都必须“知道”边的相对顺序，而在比较模型下这需要$\Omega(m\log m)$时间。要突破它，只能放弃比较模型——例如整数权重的基数排序，可把排序降到$O(m)$，此时 MST 的瓶颈转移到并查集，整体$O(m\,\alpha(n))$。

7.3 比较器：正确性的隐形依赖

所有基于排序的贪心都隐含假设：排序键定义了一个**严格弱序**。

> **定义 7.2（严格弱序）**
>
> 二元关系 $\prec$ 称为严格弱序，若
>
> - 非自反：$\neg(a \prec a)$；
> - 非对称：$a \prec b \Longrightarrow \neg(b \prec a)$；
> - 传递：$a \prec b \wedge b \prec c \Longrightarrow a \prec c$；
> - **不可比传递**：若a与b不可比、b与c不可比，则a与c不可比。

第四条最容易被忽略。浮点比较是重灾区。

```
# 错误：用 epsilon 比较浮点数会破坏传递性
def cmp_bad(a, b):
    if abs(a.key - b.key) < 1e-9:
        return 0
    return -1 if a.key < b.key else 1

# 正确：用确定的全序，把 epsilon 只用于结果判断
def cmp_ok(a, b):
    if a.key != b.key:
        return -1 if a.key < b.key else 1
    return (a.id > b.id) - (a.id < b.id)   # 显式 tie-break
```

`cmp_bad` 的问题是：a与b相差10^{-9}视为相等，b与c也相等，但a与c相差2×10^{-9}却判定不等。此时排序算法（如 Timsort 的归并）可能产生不一致的结果，甚至越界——这是 C++ `std::sort` 中真实存在的未定义行为来源。

> **三条硬规则**
>
> 1. **比较器必须是纯函数**：不能依赖外部可变状态（如“当前已选集合”），否则同一元素在不同时刻的比较结果不同。
> 2. **必须给出全序**：键相等时用唯一标识符打破平局，避免结果依赖输入顺序。
> 3. **不要在比较器里做浮点容差**：容差属于“是否接受”的判断，不属于“谁在前”的判断。

平局处理会影响正确性吗？ 对 Kruskal：不影响总值，只影响选出哪棵 MST（边权全异时唯一）。对区间调度：按结束时间排序后，结束时间相同的区间顺序不影响最优值。对 Dijkstra：平局顺序不影响距离结果，但影响生成的路径。对 Huffman：平局顺序会影响树形，但不影响期望码长——因为定理 3.6 的证明对任意满足“最小频率”的选择都成立。

7.4 常见错误清单

下面每一条都对应本书前面某个定理的反面。

错误	症状	对应分析
评价函数与目标不同向	加权问题用了计数用的键	例 3.3
可行性判据过早收紧	提前拒绝本可扩展的候选	1.2 节 (C3)
假设贪心无需回退	早期选择排斥最优组合	2.6 节
混淆“最大权基”与“最大权独立集”	权重为负时结果错	引理 2.3
在负权图上用 Dijkstra	距离偏大	例 4.6
用零容差比较浮点键	排序结果不稳定	7.3 节
忽略平局导致的最坏情形	对抗输入下退化	6.2 节

7.5 用反例搜索代替直觉

设计新贪心时，最有效的验证手段是**小规模穷举对拍**。下面的脚本枚举所有小实例，找出第一个反例。

```
from itertools import combinations
import random

def greedy_by_start(intervals):
    """错误策略：按开始时间最早贪心"""
    cur, cnt = float('-inf'), 0
    for s, f in sorted(intervals):
        if s >= cur:
            cnt, cur = cnt + 1, f
    return cnt

def brute(intervals):
    """枚举所有子集，求最大相容区间数"""
    best = 0
    for r in range(len(intervals), 0, -1):
        for S in combinations(intervals, r):
            if all(a[1] <= b[0] or b[1] <= a[0]
                   for a, b in combinations(sorted(S), 2)):
                return r
    return best

random.seed(0)
for _ in range(200000):
    n = random.randint(2, 6)
    iv = [(s := random.randint(0, 9), s + random.randint(1, 5)) for _ in range(n)]
    if greedy_by_start(iv) != brute(iv):
        print("反例:", iv, greedy_by_start(iv), brute(iv))
        break
```

运行后很快会输出类似 `反例: [(0, 5), (1, 2), (2, 3)] 1 2` 的结果：`(0,5)` 起点最早，但选它会挡住后两个。这个反例比任何文字解释都直接。

> **本章小结**
>
> - 复杂度由判定操作的数据结构决定，同一贪心可从$O(mn)$优化到$O(m\,\alpha(n))$。
> - 基于排序的贪心受$\Omega(n\log n)$下界约束，突破它必须放弃比较模型。
> - 比较器必须构成严格弱序；epsilon 容差会破坏传递性并引发未定义行为。
> - 反例搜索脚本是设计阶段性价比最高的工具。

7.6 习题

7.1　把 Kruskal 的并查集实现换成朴素 DFS 判连通，给出复杂度并分析在什么输入规模下反而更快。

7.2　证明惰性贪心（lazy greedy）不改变集合覆盖的近似比。

7.3　构造一个浮点比较器违反不可比传递律的具体数值例子。

7.4　设边权为$[0, 2^{32})$内的整数，用基数排序实现 Kruskal，给出总复杂度。

7.5　改写 7.5 节的脚本，验证“按处理时间升序”能否最小化单机总完成时间。

CHAPTER 08

五个完整案例

> 本章把前面的工具串起来。每个案例都从问题陈述走到可运行算法，中间不跳过证明，重点展示“如何判断这个贪心该不该被相信”。这些案例覆盖了交换论证的四种形态：相邻交换、结构归纳、割结构与拟阵。

8.1 案例一：缓存替换（Bélády 最优）

问题　容量为k的缓存，请求序列$\sigma = r_1, \dots, r_T$已知。缓存满时需淘汰一页，代价为缺页次数。求最优淘汰策略。

这个问题的特殊之处是：**已知未来**。这正是贪心最难奏效的场景（全局信息通常意味着局部规则不够），但 Bélády 的策略却精确最优。

策略　淘汰“下次使用时刻最远”的页；若某页不再使用，优先淘汰它。

> **定理 8.1（Bélády）**
>
> 最远下次使用优先策略最小化缺页次数。

证明

设A为该策略，O为缺页次数最少的算法，且在**最长的前缀**上与A行为一致。

设两者首次分歧发生在时刻t：同一请求r_t导致缺页，缓存内容相同（记为C），A淘汰$x \in C$，O淘汰$y \in C$，$x \neq y$。由策略定义，x的下次使用不早于y的下次使用，记作$n(x) \geq n(y)$（不再使用视为$+\infty$）。

构造O'：在时刻t淘汰x而非y，其余决策与O相同，并做如下修补——当O首次访问y时，O'的缓存中y仍在（因为O'从未淘汰y），此时O'与O的缓存内容一致，此后完全同步。

关键比较：在$[t, n(y))$区间内，O'的缓存包含y而不含x，O相反。由于x在$n(x) \geq n(y)$之前不被访问，这段区间内O'不会因缺少x而缺页；而O可能因缺少y而缺页。故O'的缺页次数不超过O。

于是O'也是最优算法，且与A的一致前缀严格更长（t处也一致了），与O的选取矛盾。■■

实现的复杂度　需预知每次请求的下次出现位置：从后向前扫描一遍，用哈希表记录，$O(T)$空间与时间；每次淘汰用最大堆取最远者，$O(\log k)$。总$O(T \log k)$。

为什么实际系统不用它

Bélády 需要预知未来，只能作离线基准。它的价值在于给出下界：任何在线策略的缺页数都不少于 Bélády，因此可用它评估 LRU 等在线策略的差距。这个案例也说明“贪心最优”与“贪心可用”是两件事。

8.2 案例二：磁带存储（相邻交换的标准练习）

问题 n个文件长度$l_1, \dots, l_n$，顺序存放于磁带。读取文件i需从头扫描，代价为完成时刻 $C_i = \sum_{j \le i} l_j$。最小化总代价 $\sum_i C_i$。

定理 8.2

按l_i升序排列最优。

证明

设某个排列中相邻两项为i后接j，且$l_i > l_j$。设i之前的长度和为t。

交换前这两项的贡献：$C_i = t + l_i$，$C_j = t + l_i + l_j$，合计$2t + 2l_i + l_j$。交换后：$C_j = t + l_j$，$C_i = t + l_i + l_j$，合计$2t + l_i + 2l_j$。

差值

$$\left(2t + 2l_i + l_j\right) - \left(2t + l_i + 2l_j\right) = l_i - l_j > 0 \tag{8.1}$$

故交换严格降低总代价。反复消除逆序对即得升序排列，且它在所有排列中最小。■■

复杂度$O\left(n\log n\right)$，由排序主导。这个证明是相邻交换的模板：**只比较局部两项的贡献差，其余项不受影响**。凡是目标函数可以写成 $\sum_i f_i\left(C_i\right)$且$f_i$单调递增的问题，都能用这个模板（例如 8.4 节的加权版本）。

8.3 案例三：区间图着色（教室分配）

问题 n个课程各占一个时间区间$\left[s_i, f_i\right)$，分配教室使同一教室内不冲突，最小化教室数。

贪心 按s_i升序处理，为每个课程分配任意一个空闲教室；无空闲则新开。

定理 8.3

贪心所用教室数等于最大重叠深度ω，因而是最优的。

证明

设贪心在处理课程I时新开了第k间教室。这说明在I开始时，前$k-1$间教室全部被占用，即存在$k-1$个课程$J_1, \dots, J_{k-1}$满足$s_J \le s_I < f_J$。它们与I在时刻s_I同时活跃，构成k个重叠区间的集合。

故$\omega \ge k$，即任何方案至少需要k间教室。贪心用k间，达到下界，最优。■■

这个证明只用了一次观察：**新开教室的时刻必然存在一个宽度为k的重叠**。它属于“下界匹配”型论证——不构造交换，而是证明算法输出恰等于某个显然的下界。

复杂度$O(n\log n)$（排序）+ $O(n\log n)$（用最小堆维护空闲教室）。

8.4 案例四：加权完成时间（Smith 法则）

问题　n个任务，处理时间p_j，权重w_j，单机从时刻0开始。最小化$\sum_j w_j C_j$。

定理 8.4（Smith 法则）

按p_j / w_j升序调度最优。

证明

设相邻两项为i后接j，i之前的总时长为t。

交换前贡献：$w_i(t+p_i) + w_j(t+p_i+p_j)$。交换后贡献：$w_j(t+p_j) + w_i(t+p_j+p_i)$。

差值为$w_j p_i - w_i p_j$。故交换前更优当且仅当

$$w_j p_i < w_i p_j \iff \frac{p_i}{w_i} < \frac{p_j}{w_j} \tag{8.2}$$

于是任何逆序对（$p_i/w_i > p_j/w_j$且i在j前）都可被交换改善，最优序必须按比值升序。■■

注意与 3.3 节的区别：EDF 最小化 $\max_j L_j$（无穷范数），Smith 法则最小化 $\sum_j w_j C_j$（加权一范数）。两个目标用两个不同的排序键，都正确。这再次印证 1.5 节的诊断原则：**目标函数的范数决定排序键**。

8.5 案例五：带截止期的收益调度（拟阵落地）

问题　n个任务，每个耗时1单位，截止期d_j，收益p_j。同一时刻只能做一个任务。求最大收益的可行任务集。

结构　令$\mathcal{I} = \{S : S\text{中任务可全部在各自截止期前完成}\}$。可以验证$(E, \mathcal{I})$是拟阵（交换公理成立）。因此按收益降序贪心，用独立性判定即可。

独立性的高效判定不需要真排序：对任务集S，把截止期升序排列得$d_{(1)} \le \cdots \le d_{(|S|)}$，则$S$可行当且仅当对所有$k$有$d_{(k)} \ge k$。

```
def schedule(jobs):
    """jobs: [(profit, deadline)]；按收益降序，用并查集找最近空槽"""
    n = len(jobs)
    parent = list(range(n + 2))          # 槽位 0..n+1

    def find(x):
        while parent[x] != x:
            parent[x] = parent[parent[x]]
            x = parent[x]
        return x

    total = 0
    for profit, d in sorted(jobs, reverse=True):   # 收益降序
        slot = find(min(d, n))                     # 最晚可用槽位
        if slot > 0:
            parent[slot] = find(slot - 1)          # 占用该槽
            total += profit
    return total
```

复杂度：排序$O(n\log n)$，并查集$O(n\,\alpha(n))$，总$O(n\log n)$。

例 8.5（贪心的最优性验证）

四个任务（收益，截止期）：$(100,1)$、$(19,2)$、$(27,2)$、$(25,1)$。

按收益降序：100 → 槽 1；27 → 截止2，占槽 2；25 → 槽 1、2 已占，拒绝；19 → 无槽，拒绝。总收益127。

若改为按截止期升序、同截止期按给定顺序贪心，则依次取$(100,1)$、$(19,2)$，得119，低于最优。差别只来自**排序键**，代码结构完全一致。

本章小结

- 四个案例给出了交换论证的四种形态：相邻交换（8.2、8.4）、结构归纳（8.1）、下界匹配（8.3）、拟阵（8.5）。
- 排序键的选择由目标函数的范数与结构决定： max 用截止期，$\sum C_j$用长度，$\sum w_jC_j$用p_j / w_j，带收益用收益。
- 相同的问题在不同目标下需要不同的贪心，不存在“万能排序键”。

8.6 部分习题提示

1.4 提示：把“贪心选择性质”写成「存在最优解O^*与贪心首步g_1满足$g_1 \in O^*$」，注意量词顺序：O^*可以依赖实例，但不能依赖贪心的后续步骤。

2.3 提示：由定理 2.7 的必要性方向，答案是不可能。若贪心对所有w都正确，则交换公理必须成立。

2.4 提示：考虑K_4的三角形。取A为一条边（不含三角形），B为另两条边构成的路径（仍不含三角形），$|A| < |B|$，但加入A的任一端点边都可能成三角。构造具体图即可反驳。

3.11 提示：紧性构造取$W = 2k$，一件$w = 2k, v = 2k$，另一件$w = k, v = k + 1$。

5.3 提示：在直线上取$2k$个相距略大于2的点对，使贪心从一端开始每次只覆盖一对，而最优解取每对的左端点。

6.1 提示：Yao 原理：对任意输入分布，确定性算法在该分布上的期望竞争比不超过随机化算法的竞争比下界。构造d的分布使期望比值最小化后仍为$e/(e-1)$。

7.2 提示：惰性贪心只在“堆顶的确等于当前最大边际收益”时才确定选择，其输出与朴素贪心逐项相同，故近似比不变。

APPENDIX

附录

> 符号约定、定理索引与延伸阅读。

A　符号表

符号	含义	首次出现
E	有限候选集（元素集）	1.2
$\mathcal{F}$	可行解族，$\mathcal{F} \subseteq 2^E$	1.2
$\sigma(e)$	评价函数（排序键）	1.2
$\kappa(S)$	可行性判定，取值$\{0,1\}$	1.2
$\mathcal{I}$	独立集族	2.1
$r(A)$	秩函数	2.3
$w(e)$	元素权重	2.4
$\Delta(e \mid S)$	边际收益$f(S\cup\{e\})-f(S)$	5.3
OPT	最优解目标值	3.1
ALG	算法输出目标值	5.1
ρ	近似比 / 竞争比	5.1
$\alpha(n)$	反阿克曼函数	4.2
$H(X)$	信息熵$-\sum_i p_i \log_2 p_i$	3.4
$\delta(s,v)$	源点到v的最短距离	4.4
t^*	集合覆盖的最优集合数	5.2

B 定理索引

编号	名称	章节
命题 1.4	两充分条件蕴含最优	1.3
引理 2.3	拟阵的基等大	2.2
定理 2.4	图拟阵	2.2
定理 2.6	秩函数刻画	2.3
定理 2.7	Rado – Edmonds 刻画	2.4
推论 2.8	Kruskal 正确性	2.5
定理 2.12	Edmonds 拟阵交	2.6
定理 3.1	交换引理模板	3.1
定理 3.2	区间调度最优性	3.2
定理 3.4	Jackson 法则（EDF）	3.3
定理 3.5	Kraft 不等式	3.4
定理 3.8	Huffman 最优性	3.4
定理 3.9	熵界$H \le L < H+1$	3.4
命题 3.11	背包比值贪心界	3.5
定理 4.2	割性质	4.1
定理 4.3	环性质	4.1
定理 4.4	并查集$O(m\alpha(n))$	4.2
定理 4.5	Dijkstra 正确性	4.4
定理 5.2	集合覆盖$\ln n+1$	5.2
定理 5.4	子模贪心$1-1/e$	5.3
定理 5.5	k-中心2-近似	5.5
定理 6.2	滑雪租赁$2-1/B$	6.2
定理 6.3	秘书问题$1/e$	6.3
定理 7.1	比较排序下界	7.2
定理 8.1	Bélády 最优性	8.1
定理 8.4	Smith 法则	8.4

C　三个常数的来源

书中反复出现的三个常数有统一来源，值得单独记一笔。

常数	近似值	出现位置	来源
$1-1/e$	0.632	子模最大化	$\lim_{k\to\infty}\left(1-1/k\right)^k = e^{-1}$
$1/e$	0.368	秘书问题	最优观察长度n/e
$e/(e-1)$	1.582	滑雪租赁（随机化）	最优分布的对偶解

三者都源于同一个极限$\left(1+1/n\right)^n \to e$，但分别对应“保证因子”“成功率”“竞争比”三种不同的度量，不可混用。

D　延伸阅读

教材与系统论述

1. Cormen, Leiserson, Rivest, Stein. *Introduction to Algorithms*, 4th ed. MIT Press, 2022.（第 15、16 章：贪心与拟阵）
2. Kleinberg, Tardos. *Algorithm Design*. Pearson, 2005.（第 4 章交换论证、第 11 章近似）
3. Oxley, J. *Matroid Theory*, 2nd ed. Oxford, 2011.
4. Borodin, El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge, 1998.

关键原始文献

5. Rado, R. “Note on independence functions.” *Proc. London Math. Soc.*, 1957.
6. Edmonds, J. “Matroids and the greedy algorithm.” *Mathematical Programming* 1:127 – 136, 1971.
7. Nemhauser, Wolsey, Fisher. “An analysis of approximations for maximizing submodular set functions I.” *Mathematical Programming* 14:265 – 294, 1978.
8. Feige, U. “A threshold of ln n for approximating set cover.” *JACM* 45(4):634 – 652, 1998.
9. Tarjan, R. “Efficiency of a good but not linear set union algorithm.” *JACM* 22(2):215 – 225, 1975.
10. Gonzalez, T. “Clustering to minimize the maximum intercluster distance.” *Theoretical Computer Science* 38:293 – 306, 1985.
11. Alon, Awerbuch, Azar, Buchbinder, Naor. “The online set cover problem.” *SIAM J. Comput.* 39(2):361 – 370, 2009.
12. Calinescu, Chekuri, Pál, Vondrák. “Maximizing a monotone submodular function subject to a matroid constraint.” *SIAM J. Comput.* 40(6):1740 – 1766, 2011.

13. Belady, L. “A study of replacement algorithms for a virtual-storage computer.” *IBM Systems Journal* 5(2):78 – 101, 1966.

14. Smith, W. “Various optimizers for single-stage production.” *Naval Research Logistics Quarterly* 3:59 – 66, 1956.

进阶方向

15. Schrijver, A. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer, 2003.（拟阵与次模的现代处理）

16. Krause, Golovin. “Submodular function maximization.” In *Tractability*, Cambridge, 2014.（近似算法的工程视角）

E　排版说明

本 PDF 的产生流程值得一提，因为它本身也是「用合适的工具做合适的事」的一个例子。内容以 Markdown 撰写；公式由自建的 LaTeX 子集转换器编译为 MathML（覆盖分式、根式、求和与积分、矩阵环境、定界符自动伸缩等）；最终由 Chromium 排版引擎按 A4 页面输出，页边距、页眉页脚与分页规则由 CSS Paged Media 控制。中文正文用宋体，西文与数学符号用 STIX Two，标题用黑体，行内公式与汉字的间距做了紧缩处理。

之所以不用现成的排版工具链，是因为需要在离线环境中同时满足三个要求：数学公式的排版质量接近 LaTeX、中文字体与标点的正确处理、以及可脚本化的批量修订。三者叠加后，这条路反而最短。