



西南大學

贪心算法

张里博

lbzhang@swu.edu.cn



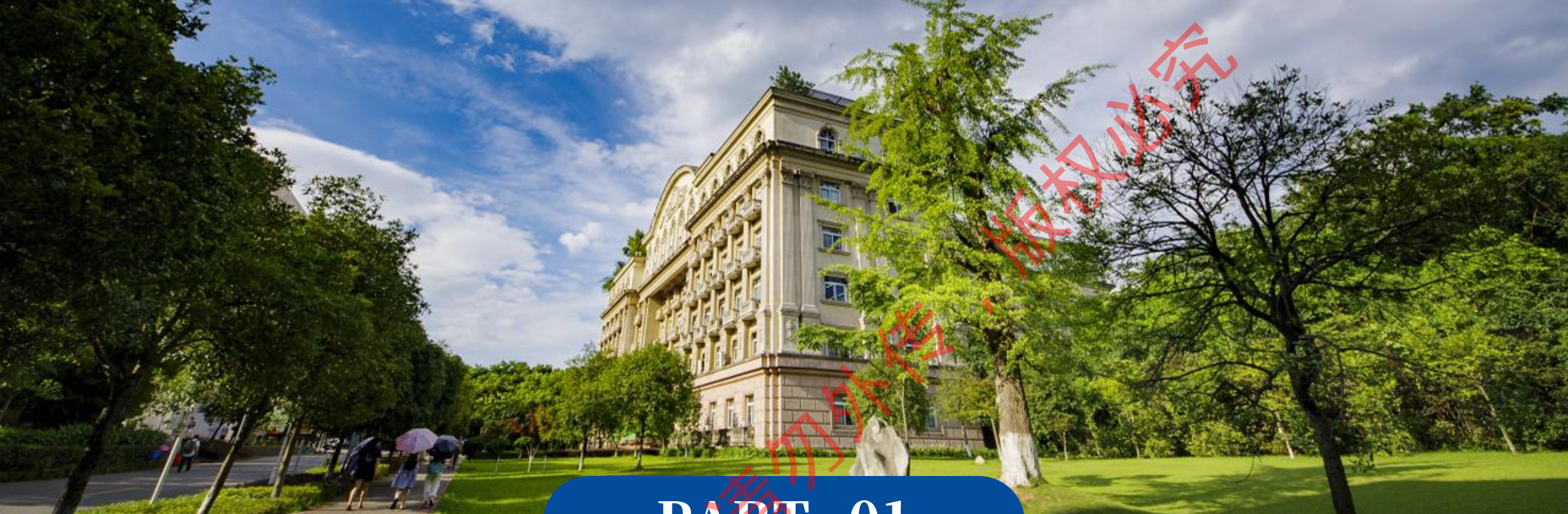
目录

1 贪心算法的设计思想

3 具体应用案例分析-Prim算法

4 正确性证明2：数学归纳法（规模）

5 正确性证明3：交换论证



PART 01

贪心算法的设计思想

Greedy Algorithm



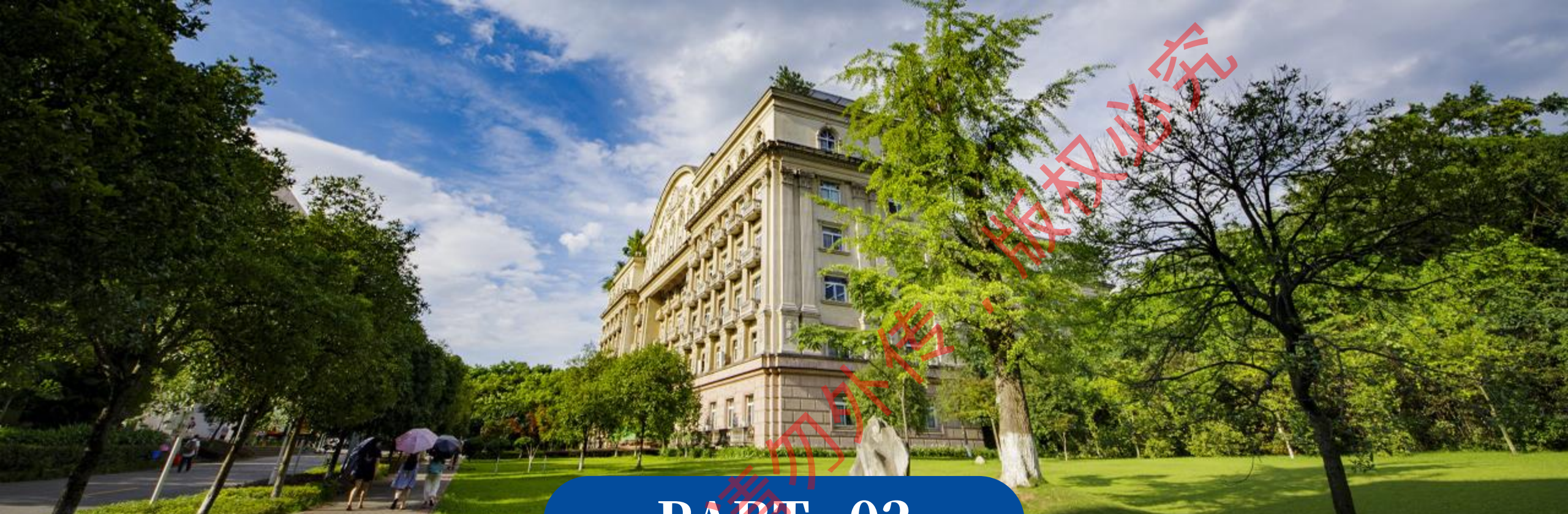


贪心算法

贪心算法（又称贪婪算法），是指在求解组合优化问题时，总是做出在当前看来是最好的选择。该算法不从整体最优上加以考虑，“只顾眼前”，某种意义上的局部最优解。

贪心算法不是对所有问题都能得到整体最优解。因此，贪心算法需要经过证明。

其基本步骤为：①把求解的问题分成若干个子问题；②对每一子问题求解，得到子问题的局部最优解；③把子问题的解局部最优解合成原来解问题的一个解。



PART 02

应用：最小生成树（PRIM）

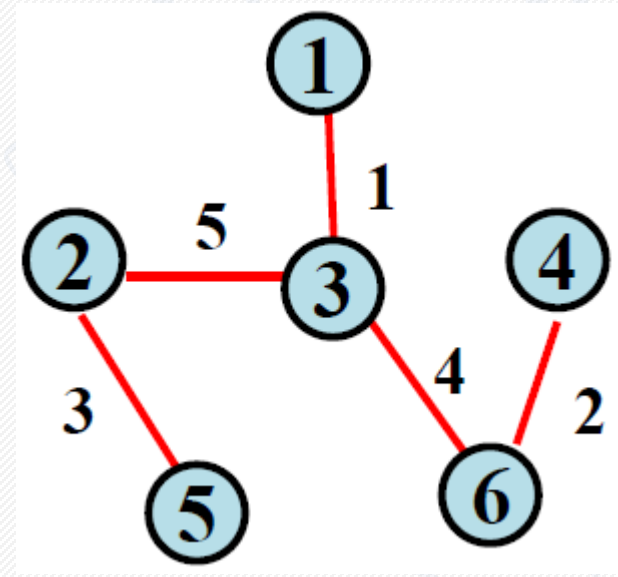
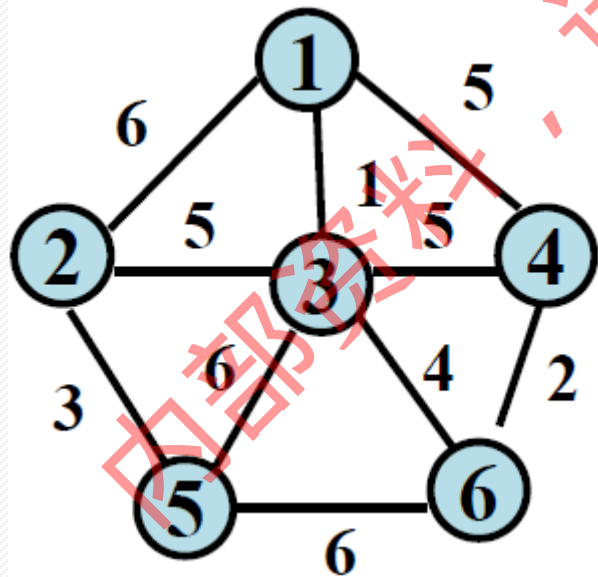
Greedy Algorithm





最小生成树 (PRIM)

- 无向连通带权图 $G=(V,E,W)$, $w(e) \in W$ 是边 e 的权.
- G 的一棵生成树是包含了 G 的所有顶点的树, 树中各边的权之和称为树的权
- (1) T 是 G 的生成树当且仅当 T 无圈且有 $n-1$ 条边.
- 目标: 寻找具有最小权的生成树称为 G 的最小生成树.





- 无向连通带权图 $G=(V,E,W)$, $w(e) \in W$ 是边 e 的权. 求 G 的最小生成树.
- 贪心算法: **Prim** 算法, **Kruskal** 算法
- 生成树在网络中有着重要应用:
 - 网络设计.
 - 电话, 电信, TV 电缆, 道路
 - 间接的应用.
 - 最大瓶颈路径
 - 网桥的自动配置协议构建 (避免环路)
 -

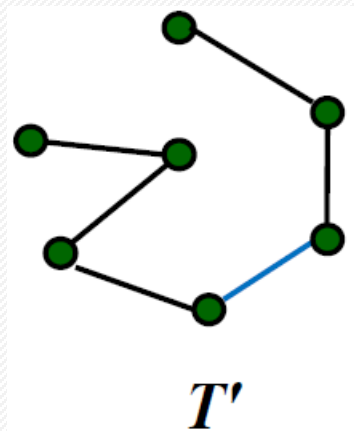
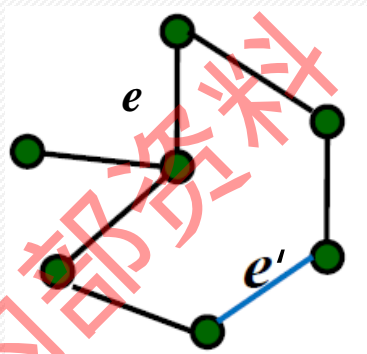
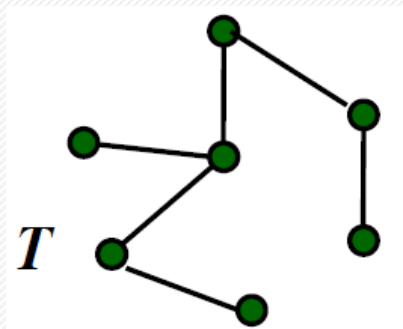


• **命题4.1** 设 G 是 n 阶连通图，那么：

(1) T 是 G 的生成树当且仅当 T 无圈且有 $n-1$ 条边.

(2) 如果 T 是 G 的生成树， $e' \notin T$ ，那么 $T \cup \{e'\}$ 含有一个圈 (回路).

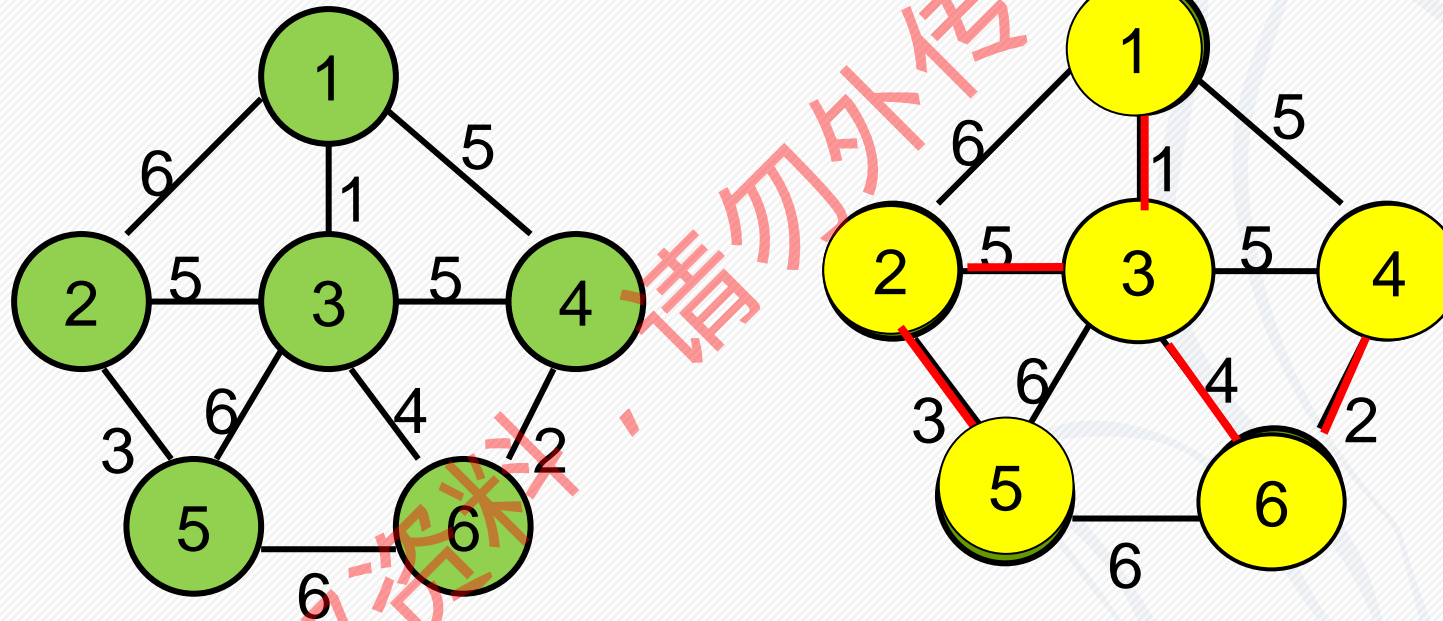
(3) 去掉 $T \cup \{e'\}$ 中圈的任意一条边 e ，能得到 G 的另一个生成树 T' .



若 $W(e') < W(e)$,
则 $W(T') < W(T)$.



- 输入：连通图 $G = \langle V, E, W \rangle$
- 输出： G 的最小生成树 T
- *Idea*
- 首先置 $S = \{1\}$ ，把顶点1放进 S 集合中；
- 然后，只要 S 是 V 的真子集，就做如下贪心选择：
 - 寻找连接 S 与 $V-S$ 的最短边（ $i \in S, j \in V-S$ ，且 $c[i][j]$ 是最小的边），将顶点 j 添加到 S 中，边添加到 T 中；
- 继续这个过程，直到 $S = V$ 时为止。





算法4.5 Prim

输入：连通图 $G = \langle V, E, W \rangle$

输出： G 的最小生成树 T

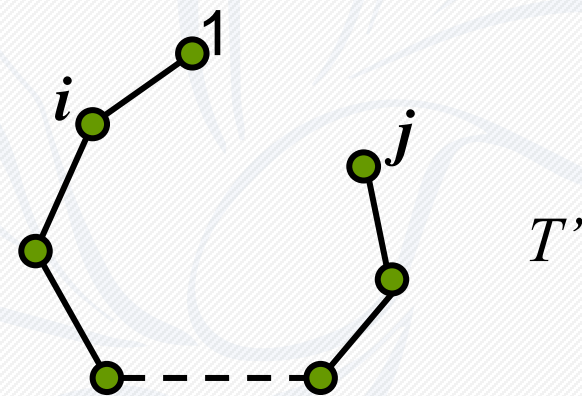
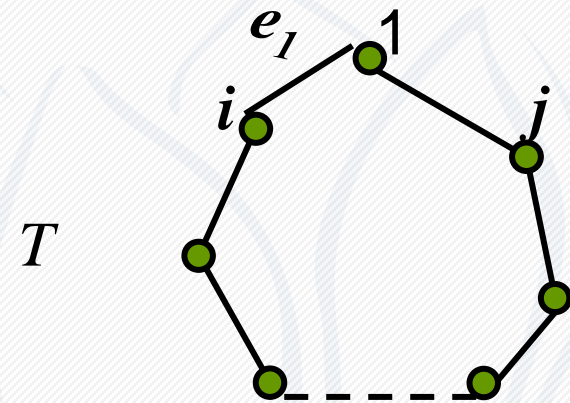
1. $S \leftarrow \{1\}; T \leftarrow \emptyset;$
2. while $V - S \neq \emptyset$ do
3. 从 $V - S$ 中选择 j 使得 j 到 S 中顶点的边 e 的权最小;
4. $S \leftarrow S \cup \{j\}; T \leftarrow T \cup \{e\};$
5. end



- **定理4.8** 对于任意 $k < n$, 存在一棵最小生成树包含Prim算法前 k 步选择的边。 (对步数归纳)
- **归纳基础:** $k=1$
- 存在一棵最小生成树 T 包含边 $e_1 = (1, i)$, e_1 是所有关联顶点 1 的边中权最小的。
- **归纳步骤:**
- 假设存在最小生成树包含Prim算法前 $k-1$ 步选择的边, 则一定存在最小生成树包含Prim算法前 k 步选择的边。

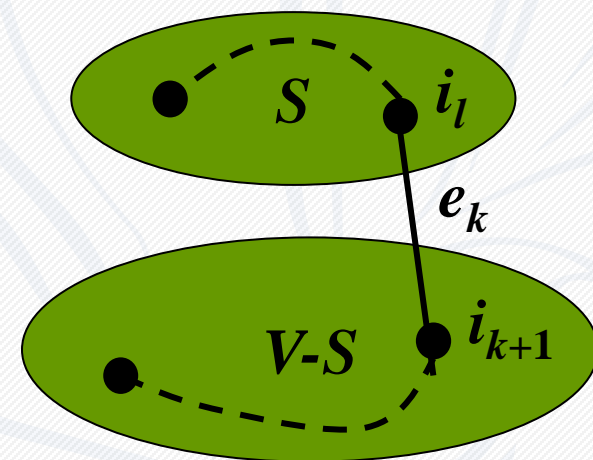


- **归纳基础:** 存在一棵最小生成树 T 含 $e_1 = (1, i)$ (含1的边中**权最小**).
- 证明: 设 T 为一棵**不包含 e_1 的最小生成树**
- 根据命题4.1(2), 图 $T \cup \{e_1\}$ 中一定有一条回路。
- 假设回路中**关联1的另一条边**为 $(1, j)$ (**$\geq e_1$ 的权**).
- 根据命题4.1(3), 令 $T' = (T - \{(1, j)\}) \cup \{e_1\}$, 则 T' 也是生成树, 且 **$W(T') \leq W(T)$** .
- 由于 T 是最小生成树 (**$W(T') \geq W(T)$**).
- 因此, **$W(T') = W(T)$** , T' 也是最小生成树。



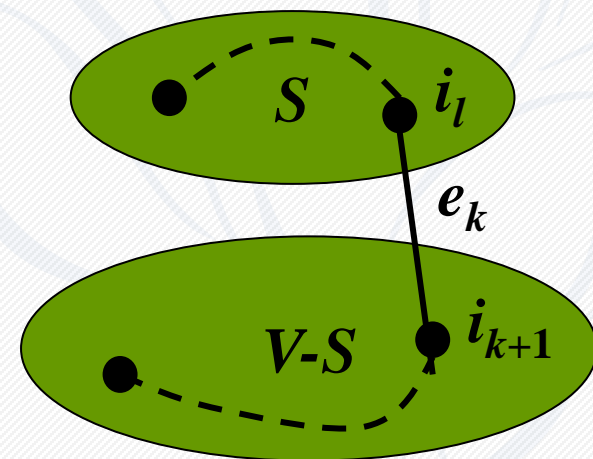
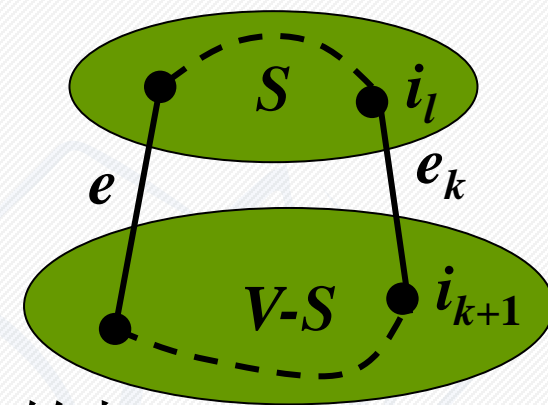


- **归纳假设**: 假设算法进行了 $k-1$ 步, 生成树的边为 e_1, e_2, \dots, e_{k-1} , 存在最小生成树 T 包含这些边.
- 需要证明的命题: 存在**最小生成树 T 包含算法前 k 步选择的边**.
- 假设Prim算法第 k 步**选择了顶点 i_{k+1}** , $e_k = (i_{k+1}, i_l)$, 则 e_k 是连接 S 和 $V-S$ 的边中权值最小的边
- **需要证明的命题:**
- 包含前 $k-1$ 步选择的**最小生成树包含 e_k**





- 证明：包含前 $k-1$ 步选择**最小生成树包含 e_k**
- 假设存在包含前 $k-1$ 步选择但**不含 e_k** 的最小生成树 T ,
- 该最小生成树连接 S 和 $V-S$ 的边存在且唯一, 设为 e ($\geq e_k$ 的权).
- 根据命题4.1 (2), 将 e_k 加到 T 中形成一个圈;
- 根据命题4.1 (3), 令 $T^*=(T-\{e\})\cup\{e_k\}$. T^* 也是 G 的一棵生成树, 其包含 $e_1, e_2, \dots, e_{k-1}, e_k$, 且 $W(T^*) \leq W(T)$.
- 由于 T 是最小生成树($W(T^*) \geq W(T)$).
- 因此, $W(T^*) = W(T)$, T^* 也是最小生成树。





最小生成树 (PRIM)

- Prim算法的设计

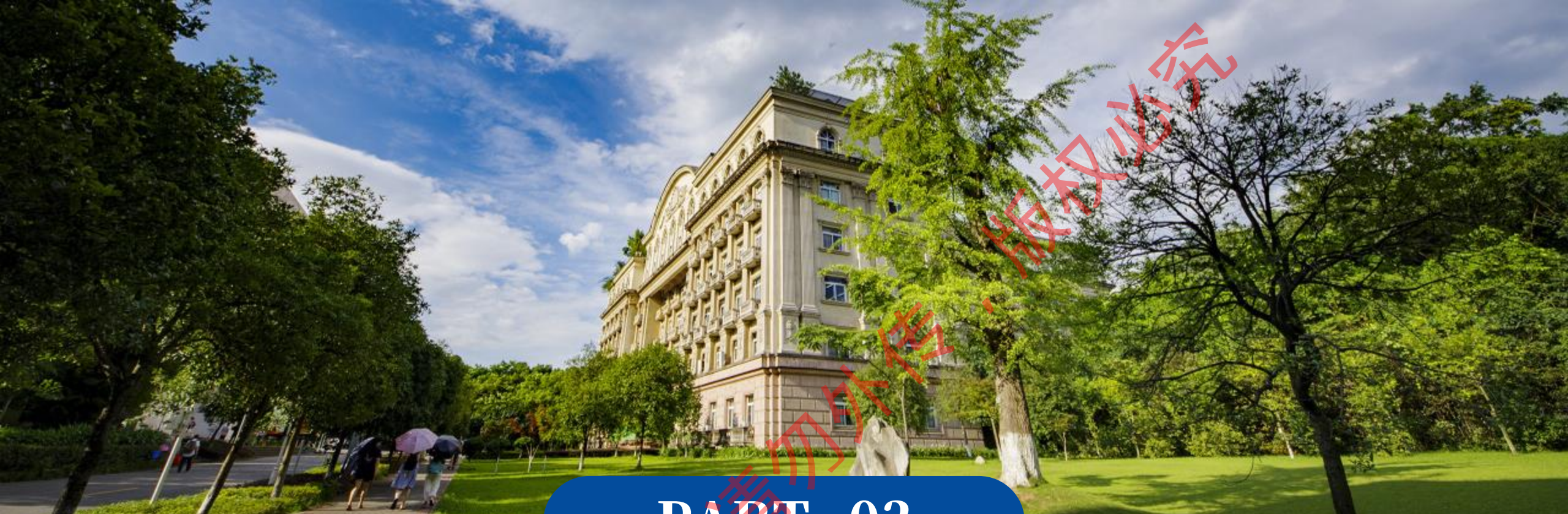
贪心策略：连接S与V-S的最短边

正确性证明：对步数归纳

- 时间复杂度： $O(n^2)$

算法步骤执行 $O(n)$ 次

每次执行 $O(n)$ 时间：~~V-S中最多有n-1个顶点~~



PART 03

正确性证明2：数学归纳法（规模）

Greedy Algorithm





例4.2 最优装载问题

- n 个集装箱 $1, 2, \dots, n$ ，集装箱 i 的重量 w_i ($w_i \leq c$)，轮船装载重量限制为 C ，**无体积限制**。问如何装使得上船的**集装箱最多**？

- 一个特殊的0-1背包问题

评价装载方案好坏的唯一标准：集装箱的数量

假设 $x_i=1$ 表示集装箱 i 被选中装上船，解是一个 n 维向量 $\langle x_1, x_2, \dots, x_n \rangle$ 。

$$\max \sum_{i=1}^n x_i$$

$$s.t.: \sum_{i=1}^n w_i x_i \leq C$$

$$x_i = 0, 1 \quad i = 1, 2, \dots, n$$



- n 个集装箱 $1, 2, \dots, n$ ，集装箱 i 的重量 w_i ，轮船装载重量限制为 C ，**无体积限制**。问如何装使得上船的**集装箱最多**？不妨设 $w_i \leq c$ 。

- 算法设计：轻者优先

将集装箱排序，使得 $w_1 \leq w_2 \leq \dots \leq w_n$ ；

按照标号从小到大装箱，直到下一个箱子**不满足约束**（装入将使得集装箱总重超过轮船装载重量限制），则停止；

将解重新定义为装上轮船的集装箱标号的集合；

假设到第 m 个箱子停止了，满足的条件是什么？

$$\sum_{i=1}^m w_i \leq C, \sum_{i=1}^{m+1} w_i > C$$



定理4.2

- 命题：对装载问题任何规模为 n 的输入实例，算法得到最优解。
- 设集装箱从轻到重记为 $1, 2, \dots, n$.
- 归纳基础
- 对只含1个集装箱的输入实例，贪心法得到最优解。显然正确。
- 归纳步骤
- 假设对于有 k 个集装箱的问题，贪心法都能得到最优解；那么对于任何 $k+1$ 个集装箱的问题，贪心法也得到最优解。



正确性证明2

- $N = \{1, 2, \dots, k+1\}$, 其中 $w_1 \leq w_2 \leq \dots \leq w_{k+1}$, 重量限制为 C . 标号集合 I 为问题的解。

- **归纳假设:** 对于 $N' = \{2, 3, \dots, k+1\}$, $C' = C - w_1$, 贪心法得到的解 $I' = \{2, 3, \dots, m\}$ 是 $\langle N', C' \rangle$ 问题最优解。

$$\sum_{i=2}^m w_i \leq C - w_1, \sum_{i=2}^{m+1} w_i > C - w_1$$

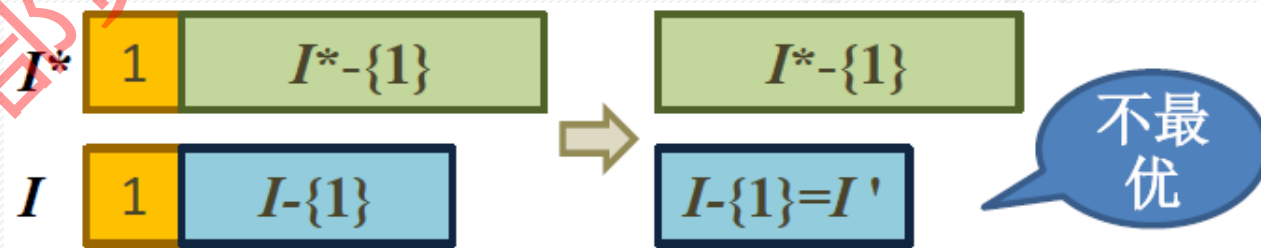
- 要证明的命题: 对于 $\langle N, C \rangle$ 问题, **贪心算法得到的解 I 是最优解**
- 对于 $\langle N, C \rangle$ 问题, **I 如何表示?**
- $I = \{1\} \cup I' = \{1, 2, \dots, m\}$

$$\sum_{i=1}^m w_i \leq C, \sum_{i=1}^{m+1} w_i > C$$

- 要证明的命题: 对于 $\langle N, C \rangle$ 问题, **I 是最优解。**



- 已知 I' 是 $\langle N', C' \rangle$ 问题最优解，求证 $I = \{1\} \cup I'$ 是 $\langle N, C \rangle$ 问题的最优解
- 证明：
- 假设存在 $\langle N, C \rangle$ 问题的包含 1 的最优解 I^* ，且 $\|I^*\|_0 > \|I\|_0$ ；
(如果 I^* 中没有 1，用 1 替换 I^* 中的首元素得到的解也是最优解)
- 那么 $I^* - \{1\}$ 是关于 N' 和 C' 的可行解（不超重） $\sum_{i=2}^k w_i \leq C - w_1$
- 因此， $\|I^* - \{1\}\|_0 > \|I - \{1\}\|_0 = \|I'\|_0$ ，这与已知矛盾。





归纳步骤证明思路

归纳假设 \Rightarrow

I 是贪心算法对于 $\langle N, C \rangle$ 问题的解

要证明的命题 \Rightarrow

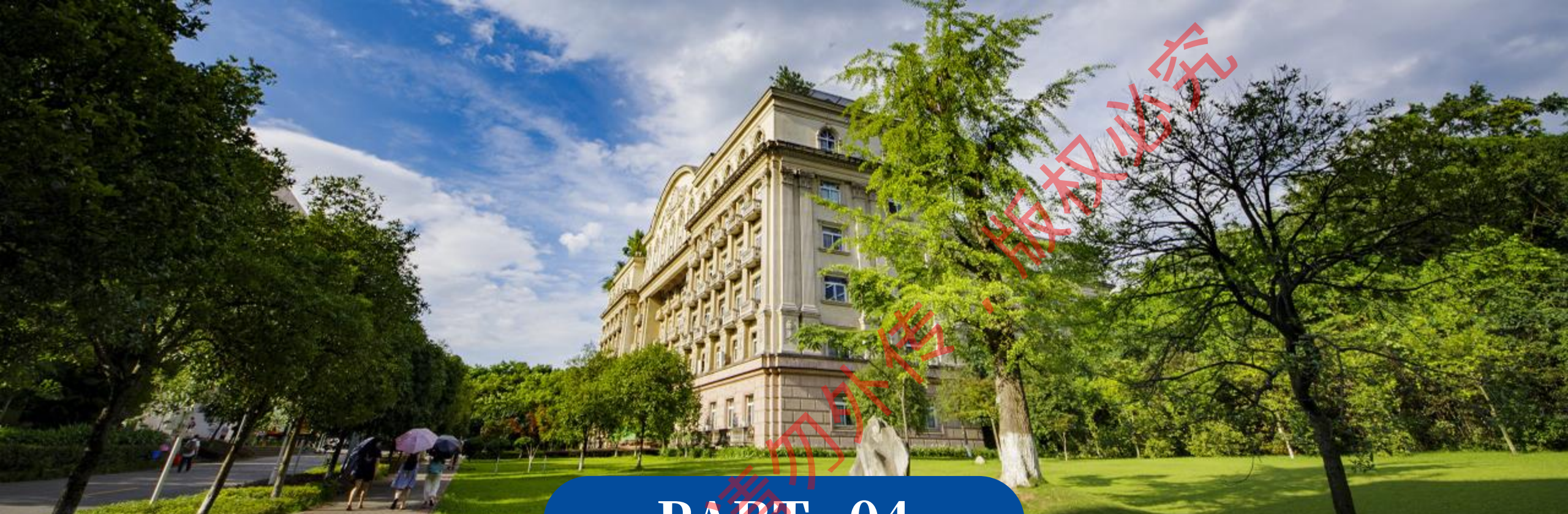
$N = \{1, 2, \dots, n+1\}, w_1 \leq w_2 \leq \dots \leq w_{n+1}$

去掉箱子1, 令 $C' = C - \{w_1\}$,
得到规模 n 的输入 $N' = \{2, 3, \dots, n+1\}$

关于输入 N' 和 C' 的最优解 I'

在 I' 加入箱子1, 得到 I

证明 I 是关于输入 N 的最优解



PART 04

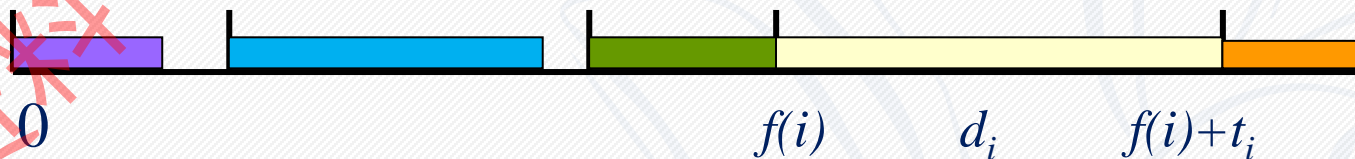
正确性证明3：交换论证

Greedy Algorithm





- **例4.3** 最小延迟调度
- 已知：客户集合 A ， $\forall i \in A$ ， t_i 为服务时间（**持续时间**）， d_i 为（**预期**）完成时间， t_i, d_i 为正整数。
- 一个**调度**是函数 $f: A \rightarrow \mathbb{N}$ ， $f(i)$ 为客户 i 的**实际开始时间**，满足**相容性**。
- 求**最大延迟**达到**最小**的调度



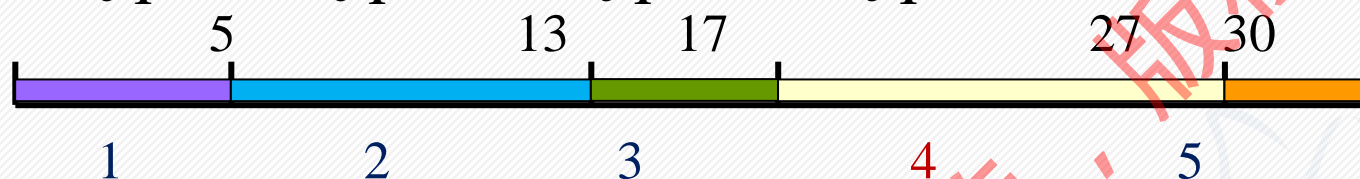
实际完成时间是？ 延迟？ $p_i(f, t, d) = \max\{0, f(i) + t_i - d_i\}$



实例

$A=\{1, 2, 3, 4, 5\}$, $T = \langle 5, 8, 4, 10, 3 \rangle$, $D=\langle 10, 12, 15, 11, 20 \rangle$

调度1: $f_1(1)=0, f_1(2)=5, f_1(3)=13, f_1(4)=17, f_1(5)=27$

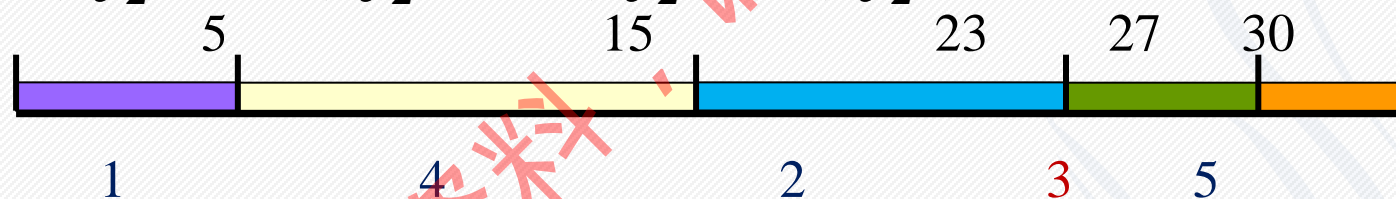


各任务延迟: $0=\max\{0, 5-10\}$, $1=\max\{0, 5+8-12\}$, $2=\max\{0, 13+4-15\}$,
 $16=\max\{0, 17+10-11\}$, $10=\max\{0, 27+3-20\}$;

最大延迟: 16

评价调度好坏的标准:
最大延迟

调度2: $f_2(1)=0, f_2(2)=15, f_2(3)=23, f_2(4)=5, f_2(5)=27$



各任务延迟: $0=\max\{0, 5-10\}$, $11=\max\{0, 15+8-12\}$, $12=\max\{0, 23+4-15\}$,
 $4=\max\{0, 5+10-11\}$, $10=\max\{0, 27+3-20\}$;

最大延迟: 12

同样的顺序, 没有空闲间隔比有间隔好



• 例4.3 最小延迟调度

- 已知：客户集合 A ， $\forall i \in A$ ， t_i 为服务时间（持续时间）， d_i 为（预期）完成时间， t_i, d_i 为正整数。
- 一个调度是函数 $f: A \rightarrow \mathbb{N}$ ， $f(i)$ 为客户 i 的实际开始时间，满足相容性。

- 求最大延迟达到最小的调度，即

$$\min_f \{ \max_{i \in A} \{ p_i(f, t, d) \} \} = \min_f \{ \max_{i \in A} \{ \max \{ 0, f(i) + t_i - d_i \} \} \}$$

$$s.t.: \forall i, j \in A, i \neq j, f(i) + t_i \leq f(j) \text{ or } f(j) + t_j \leq f(i)$$



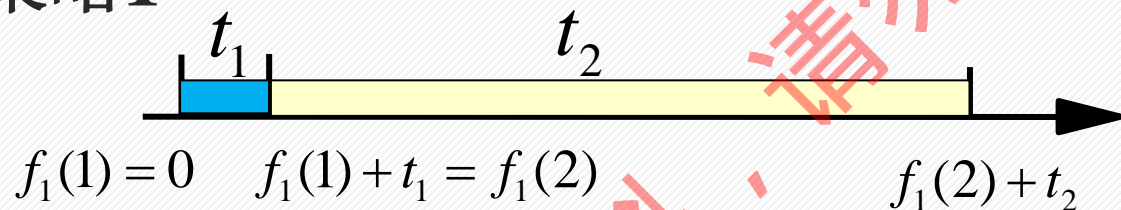
贪心策略1: 按照 t_i 从小到大安排任务

贪心策略2: 按照 $d_i - t_i$ 从小到大安排任务

贪心策略3: 按照 d_i 从小到大安排任务

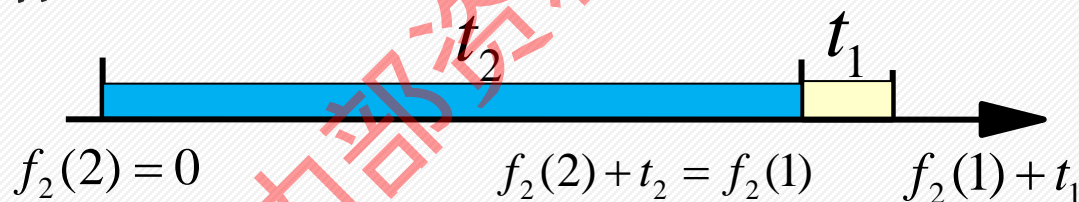
例: $t_1=1, d_1=100, t_2=10, d_2=10$

策略1



任务延迟分别为0, 1
最大延迟为1

策略2、3



任务延迟分别为0, 0
最大延迟为0



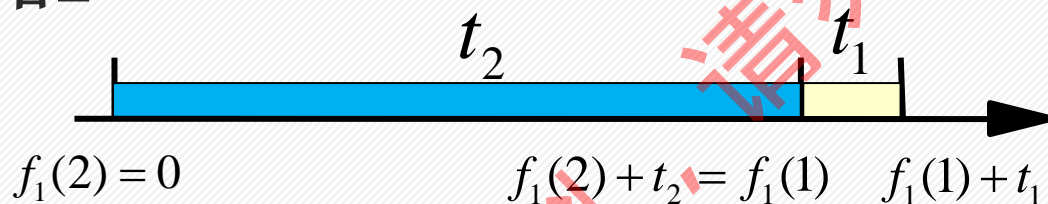
贪心策略1: 按照 t_i 从小到大安排任务

贪心策略2: 按照 $d_i - t_i$ 从小到大安排任务

贪心策略3: 按照 d_i 从小到大安排任务

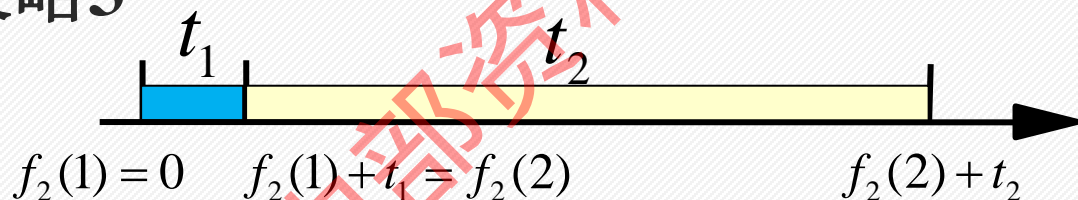
例: $t_1=1, d_1=2, t_2=10, d_2=10$

策略2



任务延迟分别为: 9, 0
最大延迟: 9

策略3



任务延迟分别为: 0, 1
最大延迟: 1



算法4.3 Schedule

输入: A, T, D

输出: f

1. 按照 d_i 从小到大对 A, T, D 重新排序;
2. $f(1) \leftarrow 0$;
3. $i \leftarrow 2$;
3. while $i \leq n$ do
4. $f(i) \leftarrow f(i-1) + t_{i-1}$; // 没有空闲
5. $i \leftarrow i+1$;
6. end

设计思想: 按预期完成时间从早到晚安排任务, 没有空闲间隔



思路:

1. 分析算法解的结构特征, 即一般最优解与算法解的区别 (成分、顺序)
2. 设计一种转换操作(替换成分、交换次序等), 在有限步将任意一个普通最优解逐步转换成算法的解。
3. 证明: 每步变换都保持解的最优性不降低。即从一个最优解转换成另一个最优解。

贪心算法解的特点: 没有空闲间隔时间, 没有逆序 ($f(i) < f(j)$ 且 $d_i > d_j$)

最优解没有空闲间隔时间

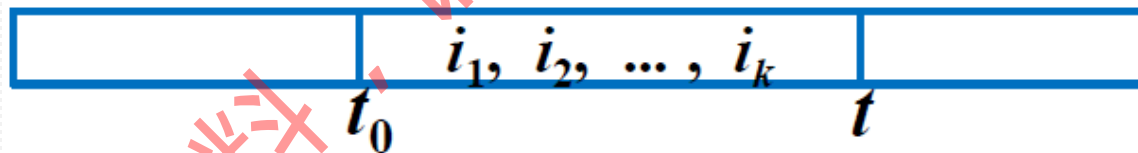
但是, 可能有: 1.存在逆序; 2.具有相同预期结束时间的任务顺序不同。



引理4.1 所有没有逆序、没有空闲时间的调度具有相同的最大延迟。

证明：由于调度 f 中 没有逆序，因此，在 f 中具有相同预期完成时间 d 的客户 i_1, i_2, \dots, i_k 必被连续安排。

假设这 k 个客户的开始服务时刻为 t_0 , 完成时刻为 t 。



- 在这 k 个客户中，最大延迟发生在最后一个客户，被延迟的时间是 $\max\{0, t-d\}$ ，与 i_1, i_2, \dots, i_k 的排列次序无关。
- 因此，所有没有逆序、没有空闲时间的调度具有相同的最大延迟



- 从一个没有空闲时间（但是有逆序）的最优解出发，不改变最优性的条件（最大延迟不变）下，转变成没有逆序的解

思考：

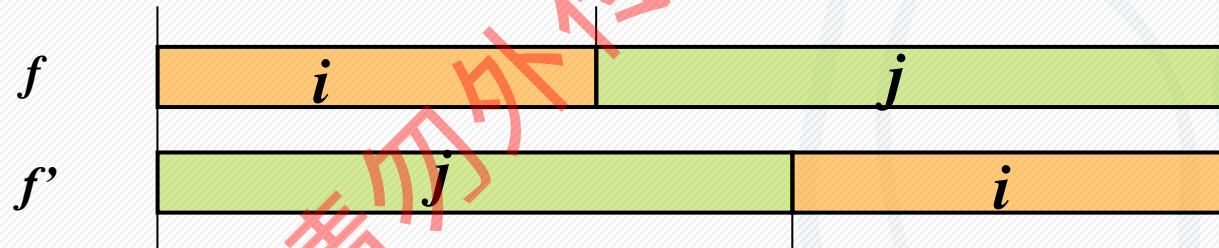
- (1) 相邻逆序： (i, j) 构成一个相邻逆序： $f(i) < f(j)$ 且 $d_i > d_j$.
- (2) 所有的逆序都可以通过相邻逆序的有限次（最多 $n(n-1)/2$ ）交换得到。

因此，只需要证明命题：

交换相邻的逆序，调度仍旧最优（最大延迟不会改变）。



- 假设 f 是一个**最优解**（**最大延迟记为 r** ），其中存在**相邻逆序**(i, j)，即 $d_j < d_i$ 。若交换 i 和 j 的顺序，得到解 f' 。
- 要证明的命题： f' 的最大延迟不超过 f 的最大延迟 **r** 。



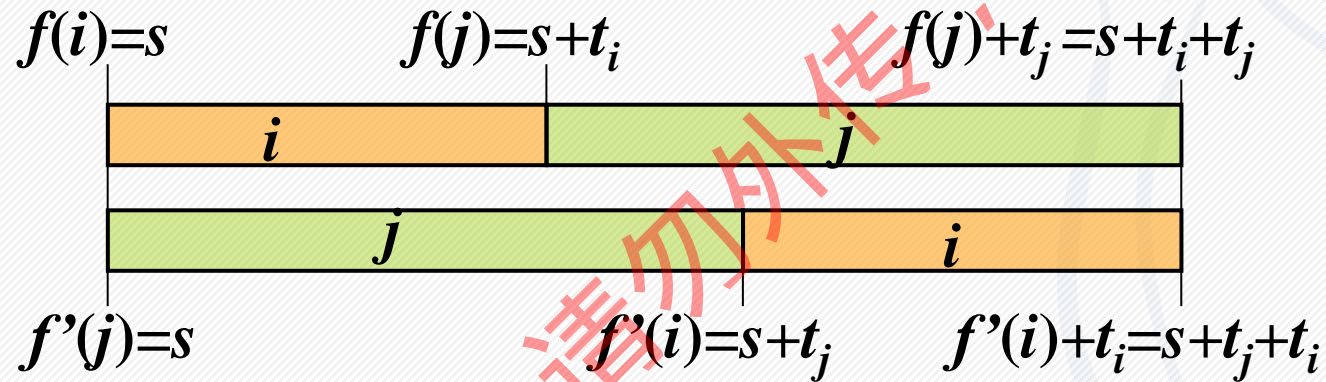
分析：

- (1) f 序列中任意一个客户的延迟表示为 **$\text{delay}(f, k) \leq r$** ;
- (2) 交换 i, j 对其他客户的延迟没影响，即 **f' 中其他客户的延迟 $\leq r$** ;
- (3) 交换 i, j 后不增加 j 的延迟，即 **$\text{delay}(f', j) \leq \text{delay}(f, j) \leq r$** ;
- (4) 交换 i, j 后可能增加 i 的延迟;
- (5) 若能证明 **$\text{delay}(f', i) \leq r$** ，则 **$f'$ 中所有客户的延迟都不超过 r** ;



定理4.3

若 i 在 f 的延迟小于 j 在 f 的延迟 $\text{delay}(f', i) \leq \text{delay}(f, j)$, 则 $\text{delay}(f', i) \leq r$



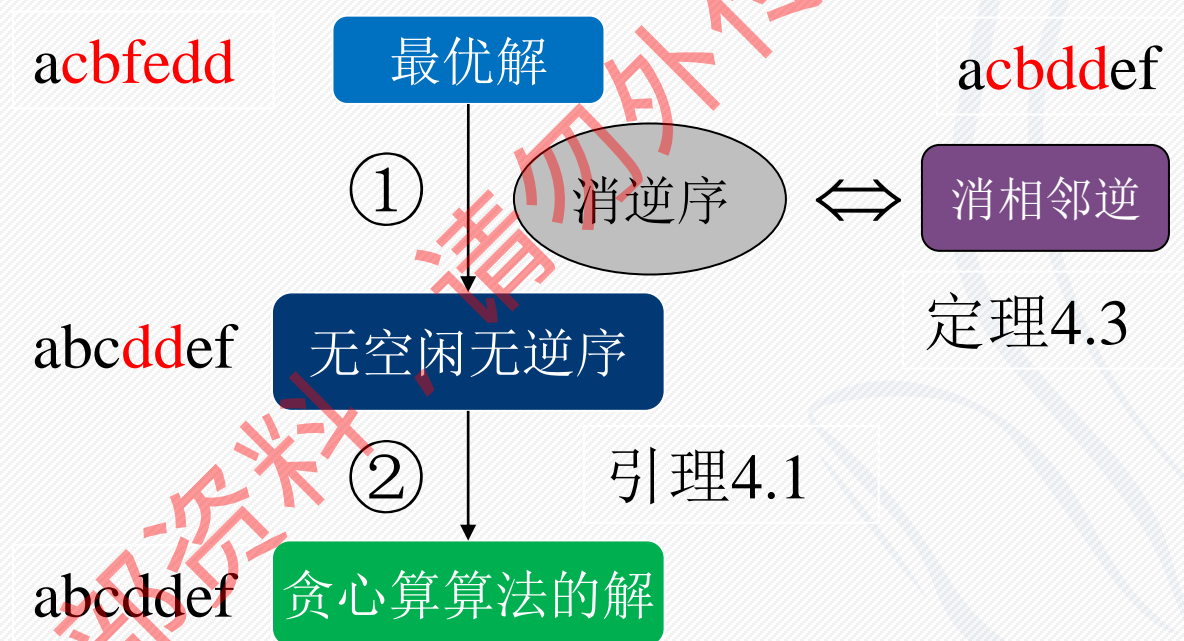
$$\text{delay}(f', i) = \max\{0, s + t_j + t_i - d_i\}$$

$$\text{delay}(f, j) = \max\{0, s + t_j + t_i - d_j\}$$

$$d_j < d_i \Rightarrow \text{delay}(f', i) \leq \text{delay}(f, j) \leq r$$



- 相对于贪心算法的解，最优解可能：
- 1.存在逆序； 2.具有相同预期结束时间的任务顺序不同。





数学归纳法

1. 一个描述算法正确性的命题 $P(n)$, n 为**算法步数**或者**问题规模**
2. 归纳基础: $P(1)$ 或 $P(n_0)$ 为真, n_0 为某个自然数
3. 归纳步骤: $P(k) \Rightarrow P(k+1)$

交换论证

1. 分析算法解与一般最优解的区别, 找到把一般解改造成算法解的一系列操作 (替换成份、交换次序);
2. 证明操作步数有限;
3. 证明每步操作后的得到解仍旧保持最优。