贪心法

算法设计与分析小班课

2022 年 4 月

注意: 以下内容主要是给课本作一些注解并补充习题, 可能不能准确地代表课程内容和考核的方向, 也不能替代阅读课本、听课、完成作业题、做往年题等活动; 其中的内容没有经过课程主管老师的审核, 也可能存在错误. 不过出现的所有错误都由作者本人负责.

1 证明方法

在贪心算法的设计中, 一个重要的元素是证明, 因为尽管我们所采用的设计策略比较自然、简单, 它们并不是明显地得到最优解, 而且最优解也可能不唯一. 课本上介绍了几种比较常见的正确性证明方法, 其中比较重要的是归纳法和交换论证法. 另外, 熟悉贪心算法的性质和结构, 会对之后学习近似算法的证明起到前置技能的作用 (简单的贪心策略有可能得到近似解).

若使用归纳法论证, 所提出的命题可以比较灵活, 但总是基于以下信念: "在任何一个阶段, 贪心法的表现都比任何算法好", 阶段的选择则需视问题决定, 通常是依据贪心法选择的 "最值" 来确定的. 如果对步数作归纳法, 一般的套路是证明: 假设有一个最优解, 算法在每部分的结果都不差于最优解的一部分, 或者存在一个最优解和算法的操作一样. 如果对问题规模作归纳法, 则需要问题及其子问题之间的结构关系比较清晰.

问题 1 给出两个字符串 S, s, 请在 $O(|S| + |s|)$ 时间内判断 s 是否为 S 的子序列 (子序列不一定连续).

解 算法非常自然和简单: 从 s 的第一个字符开始, 依次扫描 S 字符串, 并尝试匹配, 匹配后 s 的匹配指针右移; 若 s 全部匹配, 则报告 Yes, 否则报告 No. 因为指针从不左移, 所以时间 $O(|S| + |s|)$.

我们证明: 如果有子序列 $s_{k_1}, \ldots, s_{k_m}$, 那么贪心算法找到的子序列 $s_{\ell_1}, \ldots, s_{\ell_m}$ 一定满足 $\ell_i \le k_i$. 若这命题成立, 则算法报告找不到时, s 一定不是 S 的子序列. 事实上, 这用归纳法只需要两句话就可以证完了 (留给大家完成). □

问题 2 假设有一个序列 $w_1, \ldots, w_n$, 现在希望把它们划分为 k 个连续的序列, 使得每个序列的和不超过给定常数 W. 证明: 贪心算法——依次地检查 $w_1, w_2, \ldots$, 发现加上下一

个数之后和超过 W 就划分——可以使得 k 最小. 例如对于 1, 3, 2, 4, 5, 6 和 $W = 6$, 贪心算法划分为 $1+3+2, 4, 5, 6$, 但还有最优解 $1+3, 2+4, 5, 6$.

证明 这里按步数归纳可能不太直接, 我们转而证明: 对任意的 k, 假设某个算法将 $w_1, \dots, w_j$ 划分到前 k 组, 而贪心算法将 $w_1, \dots, w_i$ 划分到前 k 组, 那么 $i \geq j$.

用归纳法. 当 $k = 1$ 时, 算法总是装得尽可能多, 命题成立. 假设命题在 $k-1$ 时成立. 贪心算法将 $w_1, \dots, w_{i'}$ 放入前 $k-1$ 组, 另一个算法将 $w_1, \dots, w_{j'}$ 放入前 $k-1$ 组, 则 $j' \leq i'$. 于是贪心算法的第 k 组至少能划入 $w_{j'+1}, \dots, w_j$ 这些数, 所以 $j \geq i$, 命题成立.

根据上面命题, 贪心算法用 k 组装完时, 其他任何算法不一定装完, 所以贪心算法是最优解之一. □

另一种更加一般的方法是交换论证——实际上, 从交换论证中可以抽象出更广泛的模型, 详请参考最后的注解. 一个很有趣的关于操作系统的例子是高速缓存的最优替换策略, 其中涉及的证明同时具有归纳法和交换论证的思想.

大家在 ICS 中都已经学到了高速缓存的概念. 那时介绍了各种替换方法, 比如 LRU 等等. 作为一个标杆, 如果我们已经如有先知地知道了未来的全部访存序列, 那么实际上最优替换方法是非常简单的 (操作系统课程会讲, 例如 OTEP [AA18, 第 244 页]).

问题 3 (Bélády, 1966) 在主存中有 n 字节的内容 U, 高速缓存的容量为 $k < n$, 初始时已满 (且其中的元素已经给定). 现在给出访存序列 $d_1, \dots, d_m \in U$, 假设不允许预取, 问如何设计高速缓存的替换策略, 使得缓存不命中次数最少?

解 方法为贪心法: 我们每次都替换在最远的将来所需要的那个字节的数据 (farthest in future, 注意之后不再需要的数据的请求时间为 ∞).

引理 4 *对于给定的 j, 假设存在一个解 $\mathcal{S}$ 和贪心算法的解 $\mathcal{S}_g$ 在前 j 步的操作都一样, 那么也存在一个解 $\mathcal{S}'$ 使得其前 $j+1$ 步的操作和贪心算法的解一样, 而且 $\mathcal{S}'$ 的不命中次数不超过 $\mathcal{S}$.*

证明 考虑 d_{j+1}, 如果它已经在内存中, 或者 $\mathcal{S}, \mathcal{S}_g$ 在此时的操作一样 (因为没有预取), 则只需取 $\mathcal{S}' = \mathcal{S}$.

否则, $\mathcal{S}_g$ 选择替换了 e, 而 $\mathcal{S}$ 选择替换了 $f \neq e$. 这时, 为了让 $\mathcal{S}_g, \mathcal{S}'$ 在 $j+1$ 步相同而且不命中次数不增加, 一个自然的想法是让 $\mathcal{S}'$ 在 $\mathcal{S}$ 的基础上改为替换 e, 并在后面作一些修改. 从 d_{j+2} 开始, $\mathcal{S}'$ 的行为可以和 $\mathcal{S}$ 一样, 直到下面三种情况之一发生. 注意, 在这之前, $\mathcal{S}, \mathcal{S}'$ 的缓存除了 e, f 之外都是一样的, $\mathcal{S}$ 中有 e, 而 $\mathcal{S}'$ 中有 f.

- ▷ **需要用 $g \neq e, f$ 替换某个缓存中元素, 但 $\mathcal{S}$ 选择替换了 e** 这时我们让 $\mathcal{S}'$ 选择替换 f, 二者的缓存此时就一模一样了, 接下来 $\mathcal{S}', \mathcal{S}$ 行为一样.
- ▷ **需要用 f 替换某个缓存中元素** 这时 $\mathcal{S}'$ 会发生命中, $\mathcal{S}$ 会替换掉某个 e'. 若 $e' = e$, 则此后二者的缓存又一样了. 如果 $e' \neq e$, 则在 $\mathcal{S}'$ 中用 e 替换 e', 之后二者行为一样. 不过如果未来还需要用到 e 的话, 这就发生了数据预取. 在这种情况下, 我们对 $\mathcal{S}'$ 做进一步的改造: 模拟已经替换的行为, 需要 e 时再做实际的替换. 这样得到的

$\mathcal{S}'$ 的不命中次数也不多于 $\mathcal{S}$ 的[1].

▷ **需要用 e 替换某个缓存中元素** 这是不可能的. 因为 $\mathcal{S}_{\mathrm{g}}$ 算法的设计策略是替换最远的将来所需要的那个字节的数据, 所以替换 e 说明在需要 f 之前不会需要 e. □

有了上面的引理之后, 我们便可提出下面的命题:

命题 5 *对于每个 j, 都存在一个最优解使得其前 j 步和贪心法选择前 j 步的一样.*

证明 对 j 归纳并利用上面的引理. □

特别地, 存在一个最优解使得它和贪心算法的解相同, 所以贪心法得到最优解. □

2 贪心法的补充例题

在以下的算法分析中, 我们都没有给出运行时间的分析, 其中有一些涉及排序、选择的步骤的具体时间消耗还和实现方法有关, 在应考时要注意优化.

2.1 归纳论证

以下主要用 “stay-ahead” 型命题来证明.

问题 6 在一维公路的 $x_1 < \cdots < x_n$ 处各有一栋房子, 现在要在公路上设置基站来覆盖所有的房子. 若基站设置在 t 处, 则能覆盖 $[t-h, t+h]$, 其中 $h > 0$ 为给定常数. 设计算法给出一个设置方案, 使得需要的基站数目最小. (课本习题 4.3)

解 无妨设 $x_1 = 0$, 我们将第一个基站放在 h 处, 然后删去所有被这个基站覆盖的房子. 再令剩下房子中最左端的坐标为 0, 将下一个基站也放在 h 处, 依次类推. 我们可以发现, 贪心法总是尽可能 “晚” 地放置基站. 形式地说, 如果基站是 $s_1 < \cdots < s_m$ 的话, s_{i+1} 的位置有以下性质: s_{i+1} 是最大的满足 s_i, s_{i+1} 之间的房子被二者之一覆盖的位置. 这引导我们证明如下关键引理:

引理 7 *设 $s_1 < \cdots < s_m$ 是贪心法的解而 $t_1 < \cdots < t_k$ 是最优解, 那么对每个 $1 \le i \le \min\{m, k\}$ 都有 $s_i \ge t_i$.*

证明 用归纳法. $i = 1$ 时命题显然成立. 假设 $s_i \ge t_i$, 则 $s_1 < \cdots < s_i$ 覆盖了所有 $t_1 < \cdots < t_i$ 覆盖了的房子. 因此若将 t_{i+1} 加到贪心法的部分解中, s_i, t_{i+1} 之间的所有房子都会被这一新解覆盖, 又根据构造, s_{i+1} 是满足这一性质下最大的, 所以 $s_{i+1} \ge t_{i+1}$. □

如果 $k > m$, 那么根据我们的构造, 贪心法的部分解 $s_1 < \cdots < s_m$ 不能覆盖所有的房子, 结合上面引理导出的 $s_m \ge t_m$, 知 $t_1 < \cdots < t_m$ 也不能覆盖所有的房子, 这和它是最优解矛盾. 因此 $k \le m$, 所以贪心法得到最优解. □

问题 8 在某公司中, 每位员工都有一个工作时间的区间 $[s_i, f_i] (1 \le i \le n)$. 经理为了减轻自己的负担, 决定从这些员工中选出一个子集 $C \subseteq [n]$ 作为监督委员会, 要求所有

[1] 这实际上意味着在事先知道访存序列的情况下, 数据预取是对高速缓存的一个无效优化.

$[n] \setminus C$ 中的员工的工作时间至少和某个 C 中的委员工作时间有交集 (含端点). 设计算法求人数最小的监督委员会.

解 反复作以下操作: 从无人监督的员工, 找出结束时间最早的, 然后在所有和他工作时间相交的员工中找结束时间最晚的加入 C, 然后将被这个新委员覆盖的所有员工标记为“有人监督”.

设贪心法依次选择了 $i_1, \dots, i_k$, 任取其他解 $j_1, \dots, j_m$. 无妨设 $i_1, \dots, i_k; j_1, \dots, j_m$ 都已经按照起始时间从小到大排序, 因为监督委员会的两个区间不可能有包含关系, 所以这时它们的结束时间也已经从小到大排序.

引理 9 *对任意的 t, 设 x_t 为贪心法第 t 轮重复中找出的结束时间最早的无人监督员工, $j_1, \dots, j_{t-1}$ 必然和 x_t 不相交.*

证明 假设命题对所有 $\le t$ 的步数均成立, 考虑 $t+1$. 归纳假设指出 $j_1, \dots, j_{t-1}$ 和 x_t 都不交, 换言之和 x_t 相交的结束时间最早的区间 j_u 满足 $u \ge t$. 另外根据构造 $x_{t+1} \cap i_t = \varnothing$, 而且 i_t 是和 x_t 相交的区间中结束时间最迟的, 所以 $j_1, \dots, j_u$ 都和 x_{t+1} 不相交 (形象地说, 即 j_u 在 i_t “左边”, 而 x_{t+1} 还在 i_t 的 “右边”), 这就完成了归纳步. □

如果 $k > m$, 那么根据我们的构造, 贪心法的部分解 $i_1, \dots, i_m$ 不是一个正确的委员会, 结合上面引理导出的 $j_1, \dots, j_m$ 不能覆盖 x_{m+1}, 这和它是最优解矛盾. 因此 $k \le m$, 所以贪心法得到最优解. □

问题 10 在一段数据通路上有 n 个文件要被发送. 假设文件 i 的大小为 b_i 字节, 传输需要的时间是 t_i. 假设因为某些原因, 运营商要求在整个传输过程中, 对于任意的 t, 零时刻至 t 时刻所传输的数据的平均速率不能超过 r. 设计一个算法判定是否存在一个传输顺序满足运营商的要求.

解 记码率 $r_i = \frac{b_i}{t_i}$, 我们按照码率递增的顺序安排传输. 如果这种方案无法满足运营商的要求, 就回答 No, 否则回答 Yes. 因为整个过程中都要保持码率比较低, 所以可以想象这种安排已经是极限了, 这一性质就是: 任给时间 t, 贪心算法在 0~t 时间内的平均传输速率不超过任何一个其他传输方案 (易证). 由此可见算法的正确性. □

问题 11 某系统中一天的不同时段有一些比较关键的进程运行, 它们的开始和结束时间分别为 $s_i, f_i\ (1 \le i \le n)$. 现在有一个检查程序, 每运行它一次, 便可报告当前在运行的所有比较关键的进程的信息. 设计一个运行检查程序的方案, 使得在满足每个关键进程都被检查一次的情况下, 运行检查的次数最小. (课本习题 4.9)

解 关键进程 i 被检查的 ddl 在 f_i, 由此启发, 我们尝试将 f_i 递增排列, 然后每次在结束时间最早的进程结束前一刻运行一次检查程序, 并移去所有新增被检查的进程, 以此类推, 直到全部检查完毕. 根据构造我们知道算法是正确的: 最后每个关键进程都被检查一次. 最优性:

引理 12 对于任意的 k, 在贪心法刚好运行了前 k 次检查所处的时间段中, 任何其他得到正确解的算法都至少运行 k 次检查.

证明 当 $k = 1$ 时, 因为任何得到正确解的算法都必须在最早结束的进程运行时进行检查, 所以命题成立. 假设命题对 k 成立, 考虑 $k + 1$. 已经知道贪心法前 k 次检查能覆盖的进程不能覆盖第 $k + 1$ 次检查所处理的进程, 而贪心法总是在最晚的时刻运行新的检查, 所以任何正确的算法都必须在 $k, k + 1$ 之间运行一次检查, 这就完成了归纳步. □

根据以上引理知道贪心法得到最优解. □

2.2 改造最优解 (交换论证)

交换论证最基本情况聚焦于相邻逆序对的交换, 这种方法在运用的时候一般比较机械, 计算就可以了.

问题 13 某人组织了一次迷你铁人三项训练赛. 每人需要依次在泳池中游一段距离, 然后骑行一段距离, 最后跑一段距离. 假设由于场地问题, 泳池只能供一人使用, 而骑行和跑步的道路可以供任意多人同时使用. 已知 n 位同学分别在三个阶段所需要花费的时间 $\{(s_i, c_i, r_i) : 1 \le i \le n\}$, 现在希望适当地安排他们的出发顺序, 使得所有人全部完成训练的时间最短, 问应该如何设计算法? (课本习题 4.15)

解 我们按 $c_i + r_i$ 的递减顺序来安排, 正确性的证明使用交换论证.

假设最优解中存在紧邻逆序对 $i < j$ 使得 $c_i + r_i < c_j + r_j$, 现在交换二者的安排顺序. 原来 i 结束消耗的时间是 $s_i + c_i + r_i + C$, j 结束消耗的时间 $s_i + s_j + c_j + r_j + C$, 二者结束的总消耗为 $s_i + s_j + c_j + r_j + C$ (C 为某个常数); 交换后则分别变为 $s_i + s_j + c_i + r_i + C$ 和 $s_j + c_j + r_j + C$, 消耗总时间为 $\max\{s_i + s_j + c_i + r_i + C, s_j + c_j + r_j + C\}$. 而

$$(s_i + s_j + c_j + r_j + C) - \max\{s_j + c_i + r_i + C, c_j + r_j + C\} = \min\{c_j + r_j - c_i - r_i, s_i + s_j\} \ge 0,$$

因此交换之后总时间不上升, 故通过一系列的交换便可得到贪心法的解, 即后者是最优的. □

问题 14 图书馆的有一台打印机, 某时刻有一系列打印任务, 它们所需要的时间为 $t_1, \dots, t_n$, 重要性分别为 $w_1, \dots, w_n$. 请设计一个算法给出一个打印队列 $\sigma : [n] \longrightarrow [n]$, 使得加权等待时间

$$\sum_{i=1}^{n} w_i \left(t_i + \sum_{j:\sigma(j)<\sigma(i)} t_j \right)$$

最小. (课本习题 4.19 的一般情况)

解 我们按 $\frac{w_i}{t_i}$ 的递减顺序来安排, 正确性的证明使用交换论证.

假设最优解中存在紧邻逆序对 $i < j$ 使得 $w_i/t_i < w_j/t_j$, 交换之前二者的加权等待时间为 $w_i(C + t_i) + w_j(C + t_i + t_j)$, 而交换之后为 $w_j(C + t_j) + w_i(C + t_i + t_j)$, 前后的差为

$$w_i(C + t_i) + w_j(C + t_i + t_j) - w_j(C + t_j) - w_i(C + t_i + t_j) = w_j t_i - w_i t_j \ge 0,$$

所以交换导致加权等待时间不增, 故通过一系列的交换便可得到贪心法的解, 即后者是最优的. □

问题 15 某银行进行反洗钱调查. 现在已经知道了 n 笔可疑转账, 它们发生的时间只能近似地知道, 分别为 $t_i \pm e_i (1 \le i \le n)$. 另外近期某账户记录了准确的转账时间 $x_1 \le \cdots \le x_n$. 设计算法确定该账号是否可能对应这 n 笔转账. 换言之, 是否存在置换 $\sigma\colon [n] \longrightarrow [n]$ 使得 $x_{\sigma(i)} \in [t_i - e_i, t_i + e_i]$?

解 算法如下: 依次考虑每个 x_i, 如果有某个还未被分配转账的区间包含 x_i, 我们就将 x_i 分配给 $t_i + e_i$ 最小的那个区间; 如果每一步都顺利进行, 则汇报 Yes, 如果有一步卡住了, 就汇报 No.

我们下面只需要证明, 如果算法回答 No, 那么一定想要的 σ 一定不存在. 如果不然, 假设有一个置换 σ' 满足要求, 令 i 是最大的满足 "贪心算法对前 i 个转账的分配和 σ' 一样" 的整数. 假设 $\sigma'(i+1) = \ell$ 而贪心算法满足 $\sigma(i+1) = j \ne \ell$ (注意贪心算法在这一步还是能执行下去的, 因为之前的状态相同). 再设 $\sigma'(k) = j$, 由 i 的选择知 $k \ge i + 1$. 根据算法的构造, $t_j + e_j \le t_\ell + e_\ell$, 于是

$$t_\ell + e_\ell \le x_{i+1} \le x_k \le t_j + e_j \le t_\ell + e_\ell.$$

所以其实我们可以把 σ' 修改一下, 令 $\sigma'(k) = \ell, \sigma'(i+1) = j$, 这就和 i 的选择矛盾. 因此算法一定输出正确的答案. □

问题 16 田忌赛马的故事出自《史记·孙子吴起列传》, 讲的是孙膑的谋略智慧. 原文为: "忌数与齐诸公子驰逐重射. 孙子见其马足不甚相远, 马有上、中、下辈. 于是孙膑谓田忌曰: '君弟重射, 臣能令君胜.' 田忌信然之, 与王及诸公子逐射千金. 及临质, 孙膑曰: '**今以君之下驷与彼上驷, 取君上驷与彼中驷, 取君中驷与彼下驷.**' 既驰三辈毕, 而田忌一不胜而再, 卒得王千金."

现在假设齐王和田忌各有 n 匹马, 这些马都有战力之分, 用一个数值表示; 当两匹马比赛时, 战力高的胜, 为方便起见, 假设不可能出现平局 (此扩展情形由同学们自己补全). 接下来, 齐王和田忌要比赛 n 轮, 假设赢一局田忌可获得 1 两黄金, 否则输掉 1 两黄金. 齐王的马的出战顺序是按战力从高到低, 设计一个算法使得田忌能赢得最多的黄金.

解 根据三匹马的情形的启发, 我们可以得出以下策略:

(i) 如果田忌最慢的马比齐王最慢的马快, 则让二者比赛;

(ii) 否则, 让田忌最慢的马去和齐王最快的马比赛.

剩下的马递归运行即可.

我们先证明, 一定存在一个最优解是满足 (i) 或 (ii) 的. 若 (i) 的条件成立, 设齐王和田忌最慢的马分别是 n, n', 而在最优解中, $n, i'; i, n'$ 分别比赛. 今交换使得 $n, n'; i, i'$ 分别比赛. 因为 n 是所有马中最慢的, 所以 n, i' 和 n, n' 的对局田忌都必胜. 而 n' 的战力弱于 i', 所以对田忌来说 i, i' 的结果不差于 i, n', 所以改造后田忌的收益不会减少. 若 (ii) 的条

件成立, 设齐王最快的马是 1, 而在最优解中, $1, i'; i, n'$ 分别比赛. 同理交换使得 $1, n'; i, i'$ 分别比赛. 因为齐王最慢的马比 n' 快, 所以 i, n' 和 $1, n'$ 的对局田忌必输. 而 1 的战力强于 i, 所以对田忌来说 i, i' 的结果不差于 $1, i'$. 最后, 我们对问题规模 n 作归纳法并利用以上交换论证即完成证明. □

问题 17 (Deferred Merging Embedding, [Cha+92]) 在大多数数字集成电路系统中, 各种信号都是根据系统时间脉冲信号的时钟频率进行同步的, 这样这些信号就能在相同的步调上工作. 假设有一棵有 $n = 2^k$ 个顶点的完全二叉树, 根结点处有一个时钟, 时钟信号在一条边上传播的时间由边权给定 (例如从根到叶子的延迟就是路径上的权的和). 我们要求时钟信号传播到每个叶结点的时间都一样, 这样才能达成**同步**的效果. 为此我们想要适当地拉长某些边 (人为增加一些边上的延迟), 同时希望时钟信号传播的延迟时间尽可能低. 设计一个算法完成这件事.

解 因为 n 已经给定, 故无妨考虑时钟信号在所有 2^k 个路径上传播的总延迟时间.

利用二叉树递归的结构. 记 v 为根结点, v', v'' 为其两个孩子. 再设 d', d'' 分别为 v, v'' 到相应子树的叶子结点的最大延迟. 然后我们在延迟比较小的那边的 v-v'/v'' 边上人为增加 $|d' - d''|$ 的延迟, 接着在子树上递归运行这个算法.

为了证明算法的最优性, 我们需要用到下面两个有关改造的性质:

引理 18 (i) 任何最优解都不可能同时在某个内部结点向下的两条边上同时增加延迟;
(ii) 任何最优解都不可能在某个非根的内部结点到其所在子树的全部叶子上的路径上都增加延迟.

证明 (i) 几乎是显然的, 因为如果同时增加了 $\delta', \delta'' > 0$, 那么可以将这两条边同时缩短 $|\delta' - \delta''|$, 和最优解矛盾.

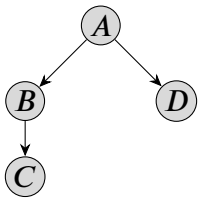
(ii) 是一个简单的推广, 我们沿用类似的改造方法. 假设不然, 设这个非根的内部结点是 w, 而其所在子树的全部叶结点为 $x_1, \ldots, x_k$. 令 e_i 为满足如下条件的边: 在 w-x_i 路径上, e 是第一个被增加延迟的边. 这些 e_i 的集合记为 F, 那么显然 $|F| \geq 2$. 令 δ 为 F 中边所被增加的延迟中最小的, 现在从这些边增加的延迟中都减去 δ, 然后在 w 到其父亲结点的边上增加 δ 的延迟. 此时时钟仍然同步, 而总延迟时间的改变为 $+\delta - |F|\delta < 2\delta - |F|\delta \leq 0$, 这和最优解矛盾. □

利用以上引理, 我们来证明上述贪心算法的最优性. 假设某个最优解在结点 v 以下的两条边所增加的延迟和贪心算法不一样, 以下沿用上面的 $v', v''; \delta', \delta''; d', d''$ 记号, 无妨设 $d' \geq d''$.

- 如果 $\delta'' - \delta' = d' - d''$, 因为该最优解的行为不一样, 所以 $\delta', \delta'' > 0$, 这和引理的 (i) 矛盾.
- 如果 $\delta'' - \delta' > d' - d''$, 那么算法肯定还要将 v' 到其对应子树上所有的叶子的路径上的延迟都增加以同步时钟, 这和引理的 (ii) 矛盾.
- 如果 $\delta'' - \delta' < d' - d''$, 那么算法肯定还要将 v'' 到其对应子树上所有的叶子的路径上的延迟都增加以同步时钟, 这仍和引理的 (ii) 矛盾.

由此可见, 贪心算法给出的解不仅是最优的, 还是唯一的. □

注 19 类似问题: 某公司有 n 位员工, 存在树形的上下级关系, 根结点为总经理. 假设现在总经理有一个消息需要传达. 在单位时间内, 上级可以向紧邻的下级 (父结点和子结点) 发邮件. 这个过程可以统筹进行, 例如下面的图中, A 先发给 B 则消息在 2 时刻末就广播完毕, 而如果先发给 D 就需要 3 时刻末才能结束广播.



设计一个算法确定消息传递的最短时间. (先用动态规划策略, 但在建立递推式时要用贪心算法.) ♤

参考文献

粗略地说, 贪心法 (以及典型的交换论证) 可以由一种叫作拟阵 (matroid) 的组合结构来刻画, 这一结构上的典型优化问题能概括最小生成树之类的问题, 直接搜索维基百科或者阅读 [Oxl11] 可获得更多信息.

贪心法在图论中的两个重要应用是 Kruskal 最小生成树算法和 Dijkstra 算法, 关于这些算法和连带的图论性质, 我们已经在数据结构课程中学过, 可以参考 [KT06, 第 4 章及其习题 1、2, 8~11, 18~23, 25~28, 30~33] 来复习. 另外, 同学们上学期学过的 Havel–Hakimi 算法 (判定度数序列是否可简单图化) 其实也是贪心算法 (为什么?).

[AA18] R. H. Arpaci-Dusseau and A. C. Arpaci-Dusseau. *Operating systems: Three easy pieces*. Arpaci-Dusseau Books LLC, 2018. ISBN: 978-1-98-508659-3.

[Cha+92] T.-H. Chao et al. “Zero skew clock routing with minimum wirelength”. 刊于: *IEEE Trans. on Circuits and Systems* 39.11 (1992), pp. 799–814.

[KT06] J. Kleinberg and É. Tardos. *Algorithm Design*. Pearson Education, 2006. ISBN: 978-0-32-129535-4.

[Oxl11] J. G. Oxley. *Matroid Theory*. 2nd ed. Oxford University Press, 2011. ISBN: 978-0-19-960339-8.

编写: WC
E-mail: wchang@pku.edu.cn