浅谈一种基于比较的贪心思想在信息学竞赛中的应用

成都外国语学校 张隽恺

摘 要

本文介绍了一种基于比较的贪心思想在信息学竞赛中的应用。本文首先讲述了问题形式、该方法的过程以及使用该方法需要的一种条件，接着引入了一类树上形式与上一类问题类似的问题，并给出了这类问题基于上述贪心思想的算法。最后本文对信息学竞赛中的满足上述条件的常见模型和算法部分性质进行了分析。

1 引言

在信息学竞赛中，贪心思想的应用是极其重要的一个部分。在贪心思想中，存在一类通过分析在解中交换两个元素对答案的影响，从而得到最优解满足的性质的方法，这种方法通常被称为“Exchange Arguments”。这类方法在十年前就已经在信息学竞赛中出现，近年来也出现了一些该算法的拓展例子。但关于该算法的分析和研究仍然较少。本文将对这种算法进行较为详细的分析。

本文的第一部分通过一个例子引入了该方法，接着分析了该方法可以得到正确的最优解的一种充分条件。第二部分介绍了这种问题的一个经典拓展——一类在之前的问题上加入一种形如有根树的限制的问题，接着给出了一种与之前类似但更强的条件，以及问题满足该条件时问题的两种算法。两部分分别对几种常见的模型进行了讨论，并分析了该部分算法的部分性质及应用。

2 一类与排列相关的最优化问题及 Exchange Arguments 方法

2.1 问题引入

在本文的这一部分中，主要讨论如下形式的问题：

给定 n 个元素 $x_1, \cdots, x_n$，以及一个定义域为这些元素的序列，值域为有序集合的函数 F。求出对于所有的 n 阶排列 p，如下表达式的最小值：

$$F(\{x_{p_1}, x_{p_2}, \cdots, x_{p_n}\})$$

例如如下问题：

例题 1. [1]

给出 n 个二元正整数对 (a_i, b_i)，将它们按照任意顺序排成一列。定义排成一列的代价为每个二元组的 a 乘上序列中这个二元组之后的所有二元组的 b 之和的总和。求最小的代价。

$n \le 10^6, 1 \le a_i, b_i \le 10^6$

这可以看成上述问题的一种情况，其中 $F(\{(a_1, b_1), (a_2, b_2), \cdots, (a_k, b_k)\}) = \sum_{1 \le i < j \le k} a_i b_j$，显然实数中可以进行比较。

分析最优的排列满足的性质。考虑一个排列 $(p_1, \cdots, p_n)$ 以及在这个排列中交换两个相邻位置 $(i, i+1)$ 得到的排列，即 $(q_1, \cdots, q_n) = (p_1, \cdots, p_{i-1}, p_{i+1}, p_i, p_{i+2}, \cdots, p_n)$，则可以得到：

$$F(\{(a_{p_1}, b_{p_1}), \cdots, (a_{p_n}, b_{p_n})\}) - F(\{(a_{q_1}, b_{q_1}), \cdots, (a_{q_n}, b_{q_n})\}) = a_{p_i} b_{p_{i+1}} - a_{p_{i+1}} b_{p_i} \tag{1}$$

因而如果 $a_{p_i} b_{p_{i+1}} - a_{p_{i+1}} b_{p_i} > 0$，则 $F(\{(a_{q_1}, b_{q_1}), \cdots, (a_{q_n}, b_{q_n})\}) < F(\{(a_{p_1}, b_{p_1}), \cdots, (a_{p_n}, b_{p_n})\})$，则 $(p_1, \cdots, p_n)$ 不是最优解。因此只有满足 $\forall 1 \le i < n, a_{p_i} b_{p_{i+1}} - a_{p_{i+1}} b_{p_i} \le 0$，即 $\frac{a_{p_i}}{b_{p_i}} \le \frac{a_{p_{i+1}}}{b_{p_{i+1}}}$ 的排列可能是最优解。

如果一个排列 p 满足这样的限制，则 p 满足 $\forall 1 \le i < j \le n, \frac{a_{p_i}}{b_{p_i}} \le \frac{a_{q_i}}{b_{q_i}}$。因此对于任意数 v，所有满足 $\frac{a_i}{b_i} = v$ 的 i 在排列中一定构成连续的一段。不同的段在排列中按照 $\frac{a_i}{b_i}$ 从小到大排序。对于任意的 i, j，如果 $\frac{a_i}{b_i} = \frac{a_j}{b_j}$，则由 1 式，如果排列中 i, j 相邻，交换它们后得到的排列的 F 保持不变。因此可以得到，对于排列中满足内部所有 $\frac{a_i}{b_i}$ 相等的一段，这一段内任意交换不改变 $F(\{(a_{p_1}, b_{p_1}), \cdots, (a_{p_n}, b_{p_n})\})$。因此所有满足上述条件的排列 p 的 $F(\{(a_{p_1}, b_{p_1}), \cdots, (a_{p_n}, b_{p_n})\})$ 相同，从而它们都是最优解。

因此对于这个问题，只需要将所有的二元组按照 $\frac{a_{p_i}}{b_{p_i}}$ 排序，即可得到一种最优的排成一列的方式。时间复杂度为排序的时间复杂度 $O(n \log n)$。

2.2 问题分析与 Exchange Arguments 方法

考虑分析上一个问题中的做法。在这个问题中，使用的做法可以看作对于每两个元素 x_i, x_j 间建立了比较关系，比较关系可以看成 $F(\{x_i, x_j\}) \le F(\{x_j, x_i\})$。在这道题中，这样定义的比较关系以及原问题中满足一些特殊的性质，使得将所有元素按照比较关系排序可以得到一种最优解。但对于更一般的情况，上述做法显然不再成立。事实上对于一些 F，上述形式的问题目前只有指数级的算法。因此考虑分析上一个问题的性质，得到使得如下算法可以找到最优解的一种条件：

[1]题目来源：经典问题

设给出的所有元素构成的集合为 S，定义 S 上的一种二元比较关系 $\leq$，将所有元素按照比较关系排序。

在上一个问题中，不难发现存在如下性质：

性质 2.1. $\forall a, b \in S$，$a \leq b, b \leq a$ 至少有一者成立。即比较关系满足强完全性。

性质 2.2. $\forall a, b, c \in S$，如果 $a \leq b, b \leq c$，则 $a \leq c$。即比较关系满足传递性。

性质 2.3. $\forall a, b \in S$，如果 $a \leq b$，则对于任意一个包含 $\{a, b\}$ 作为连续子序列的元素序列 $\{s_1, \cdots, s_{k-1}, a, b, s_{k+2}, \cdots, s_l\}$，都有：

$$F(\{s_1, \cdots, s_{k-1}, a, b, s_{k+2}, \cdots, s_l\}) \leq F(\{s_1, \cdots, s_{k-1}, b, a, s_{k+2}, \cdots, s_l\})$$

存在如下结论：如果 F 满足存在一种 S 上的一种二元比较关系 $\leq$ 满足性质 2.1,2.2,2.3，则在上述算法中使用这种比较关系，上述算法可以得到问题的一个最优解。

证明. 由假设，则 $\leq$ 构成 S 上的一个弱序关系（Weak Ordering[1]，也被称为优先关系）。因此，设 $|S| = n$，存在一种 S 中元素排成序列 $\{x_1, \cdots, x_n\}$ 的方式，满足 $\forall 1 \leq i < n, x_i \leq x_{i+1}$。对于一个满足弱序关系的比较关系 $\leq$，使用基于比较的排序算法，即可在 $O(n \log n)$ 次比较中得到一个这样的序列。

此时有如下引理：

Lemma 2.1. 如果 S 中元素的一种排列 $\{x_1, \cdots, x_n\}$ 满足 $\forall 1 \leq i < n, x_i \leq x_{i+1}$，且 F 满足性质 *2.3*，则不存在一个 S 中元素的排列 $\{y_1, \cdots, y_n\}$，使得 $F(\{y_1, \cdots, y_n\}) < F(\{x_1, \cdots, x_n\})$。即 $F(\{x_1, \cdots, x_n\})$ 是所有可能的排列对应的函数值中最小的。

证明. 由 $\leq$ 的传递性，有 $\forall 1 \leq i < j \leq n, x_i \leq x_j$，设 $\{y_1, \cdots, y_n\} = \{x_{p_1}, \cdots, x_{p_n}\}$。
对 $\{x_{p_1}, \cdots, x_{p_n}\}$ 进行 n 轮交换，第 i 轮交换通过若干次交换 x_i 以及与其相邻的一个元素的顺序，将 x_i 交换至第 i 个位置，n 轮交换后即可得到 $\{x_1, \cdots, x_n\}$，可以得到第 i 轮交换中的每次交换一定是交换 x_i 以及 x_i 的前一个元素，且设前一个元素为 x_t，则一定有 $t > i$，从而 $x_i \leq x_t$。由性质 2.3，这样交换相邻两个元素后，F 不会增大。因此进行所有交换后 F 不会增大，从而有 $F(\{x_1, \cdots, x_n\}) \leq F(\{x_{p_1}, \cdots, x_{p_n}\})$。因此引理成立。 □

由此，满足 $\forall 1 \leq i < n, x_i \leq x_{i+1}$ 的排列 $\{x_1, \cdots, x_n\}$ 是问题的一种最优解。 □

这也证明了上一个问题解法的正确性。

2.3 部分例子与对应模型

2.3.1 一些例子

还有其它一些问题满足类似的形式，下面列举了几个例子。

例题 2. [2]

给出 n 个包含小写字符的字符串 $s_1, \cdots, s_n$。找到一个 n 阶排列 p，将 $s_{p_1}, s_{p_2}, \cdots, s_{p_n}$ 顺序拼接得到 S，使 S 的字典序最小。

$\sum |s_i| \le 5 \times 10^5$

此时设 F 为将字符串顺序拼接的结果，字符串间的字典序比较显然合法。此时仍然令 $x \le y$ 当且仅当 $F(\{x,y\}) \le F(\{y,x\})$，则此时定义的比较关系为：对于字符串 s,t，$s \le t$ 当且仅当 $s+t$ 的字典序小于等于 $t+s$ 的字典序（这里 $s+t$ 表示字符串 s,t 按顺序拼接得到的结果）。显然 $\le$ 满足性质 2.1。字典序的比较关系满足在两个长度相同的字符串开头或结尾增加相同字符不影响字典序大小关系，因此 $\le$ 满足性质 2.3，只需证明比较关系满足传递性，即性质 2.2。此时有如下引理：

Lemma 2.2. 记 S^∞ 为将字符串 S 无限重复所得到的字符串，则 $S+T$ 的字典序小于等于 $T+S$ 的字典序当且仅当 S^∞ 的字典序小于等于 T^∞ 的字典序。

该引理的证明较为复杂，且与本文其余部分关系不大，故在此略去证明。

字典序的比较关系具有传递性，因此使用上述做法得到的 $\le$ 关系具有传递性，因此比较关系满足上述三条性质。使用上述做法，使用后缀数组或类似的算法支持预处理后 $O(1)$ 询问两个给出串的后缀的最长公共前缀，即可对于两个给出的字符串 S,T 在 $O(1)$ 的复杂度内比较 $S+T$ 与 $T+S$ 的字典序大小。后缀数组的实现时间复杂度为 $O((\sum |s_i|)\log(\sum |s_i|))$。

例题 3. [3]

有 n 块硬盘，需要对这些硬盘进行格式化。第 i 块硬盘在格式化前容量为 a_i，格式化后容量为 b_i。初始时每块硬盘中都装满了数据，但在格式化一块硬盘的过程中，需要将这块硬盘上的所有数据移动到其他硬盘或者存储空间中。数据可以进行任意转移，任意分割。求最少需要多少的额外存储空间，才能格式化所有的硬盘并保留数据。

$n \le 10^6, 0 \le a_i, b_i \le 10^9$

设第 i 次格式化的硬盘为第 p_i 块。则在格式化第 p_i 块硬盘前，需要的额外空间为 $\max(0, (\sum_{j=1}^{i-1} b_{p_j}) - (\sum_{j=1}^{i} a_{p_j}))$。因此问题相当于给定 n 对 (a_i, b_i)，找到它们的一个排列，最大化 $\min_{i=1}^{n}\{(\sum_{j=1}^{i-1} b_i) - (\sum_{j=1}^{i} a_j)\}$。记 $F(\{(a_1,b_1),\cdots,(a_n,b_n)\}) = -\min_{i=1}^{n}\{(\sum_{j=1}^{i-1} b_j) - (\sum_{j=1}^{i} a_j)\}$，即可看成最小化 F。

沿用之前的方法，定义 $x \le y$ 当且仅当 $F(\{x,y\}) \le F(\{y,x\})$，对于两个元素 $(a_1,b_1),(a_2,b_2)$，$(a_1,b_1) \le (a_2,b_2)$ 当且仅当 $\min\{-a_1, b_1 - a_1 - a_2\} \ge \min\{-a_2, b_2 - a_1 - a_2\}$。考虑这样定义的 $\le$ 关系是否满足上述性质。对于性质 2.3，设最后的序列为 $(a_1,b_1),\cdots,(a_n,b_n)$，交换的元素为

[2]题目来源：经典问题

[3]题目来源：2016 ACM-ICPC World Finals L Swap Space

$(a_t, b_t), (a_{t+1}, b_{t+1})$，且 $(a_t, b_t) \le (a_{t+1}, b_{t+1})$，则：

$$F(\{(a_1, b_1), \cdots, (a_n, b_n)\}) = -\min_{i=1}^{n}\{(\sum_{j=1}^{i-1} b_i) - (\sum_{j=1}^{i} a_j)\}$$

$$= -\min\{(\sum_{j=1}^{t-1} b_i) - (\sum_{j=1}^{t} a_j) + \min\{-a_t, b_t - a_t - a_{t+1}\}, \min_{1\le i\le n, i\ne t, t+1}\{(\sum_{j=1}^{i-1} b_i) - (\sum_{j=1}^{i} a_j)\}\}$$

$$F(\{(a_1, b_1), \cdots, (a_{t-1}, b_{t-1}), (a_{t+1}, b_{t+1}), (a_t, b_t), (a_{t+2}, b_{t+2}), \cdots, (a_n, b_n)\})$$

$$= -\min\{(\sum_{j=1}^{t-1} b_i) - (\sum_{j=1}^{t} a_j) + \min\{-a_{t+1}, b_{t+1} - a_{t+1} - a_t\}, \min_{1\le i\le n, i\ne t, t+1}\{(\sum_{j=1}^{i-1} b_i) - (\sum_{j=1}^{i} a_j)\}\}$$

因为 $\min\{-a_t, b_t - a_t - a_{t+1}\} \ge \min\{-a_{t+1}, b_{t+1} - a_t - a_{t+1}\}$，所以上式中前者小于等于后者，因此性质 2.3 成立。

分析此时大小关系的性质，即 $\min\{-a_1, b_1 - a_1 - a_2\} \ge \min\{-a_2, b_2 - a_1 - a_2\}$。如果这个不等式成立，则说明不等式左侧的两个元素都大于等于右侧的某一个元素，因此上述条件等价于满足以下两者之一：

1. 如果都大于等于第一个元素，则相当于 $a_1 \le a_2$ 且 $b_1 - a_1 \ge 0$。

2. 如果都大于等于第二个元素，则相当于 $b_1 \ge b_2$ 且 $b_2 - a_2 \le 0$。

由此可以考虑将所有元素按照 $b_i - a_i$ 的符号，即 $sgn(b_i - a_i)$[4]分成三类。此时有如下引理：

Lemma 2.3. *1.* 如果 $sgn(b_1 - a_1) > sgn(b_2 - a_2)$，则 $\min\{-a_1, b_1 - a_1 - a_2\} \ge \min\{-a_2, b_2 - a_1 - a_2\}$ 成立。

2. 如果 $sgn(b_1 - a_1) = sgn(b_2 - a_2) = 1$，则 $\min\{-a_1, b_1 - a_1 - a_2\} \ge \min\{-a_2, b_2 - a_1 - a_2\}$ 当且仅当 $a_1 \le a_2$。

3. 如果 $sgn(b_1 - a_1) = sgn(b_2 - a_2) = 0$，则 $\min\{-a_1, b_1 - a_1 - a_2\} \ge \min\{-a_2, b_2 - a_1 - a_2\}$ 成立。

4. 如果 $sgn(b_1 - a_1) = sgn(b_2 - a_2) = -1$，则 $\min\{-a_1, b_1 - a_1 - a_2\} \ge \min\{-a_2, b_2 - a_1 - a_2\}$ 当且仅当 $b_1 \ge b_2$。

[4] $sgn(x) = \begin{cases} 1 & x > 0 \\ 0 & x = 0 \\ -1 & x < 0 \end{cases}$

证明. 如果 $sgn(b_1-a_1)>sgn(b_2-a_2)$，则 $b_1-a_1\geq 0, b_2-a_2\leq 0$ 都成立。如果 $a_1\leq a_2, b_1\geq b_2$ 都不成立，则有 $b_1-a_1\leq b_2-a_2$，但这与 $sgn(b_1-a_1)>sgn(b_2-a_2)$ 矛盾，因此此时上述两种条件一定有一种得到满足。对于 sgn 相同的情况，由上述两种条件可以直接推出结论。 □

由引理 2.3 不难发现，如果按照 $sgn(b_i-a_i)$ 将所有元素分成三类，则在每一类内部，$\leq$ 都定义了合法的弱序关系。但此时在不同类之间，$\leq$ 定义的关系不一定满足传递性。例如，$(2,2)\leq(1,1)\leq(1,2)$，但 $(2,2)\not\leq(1,2)$。因此这样的比较关系不满足性质 2.2。

但对于不同类之间的情况，如果 $sgn(b_1-a_1)>sgn(b_2-a_2)$，则一定有 $(a_1,b_1)\leq(a_2,b_2)$。因此考虑使用如下方式定义 $\leq$：

$(a_1,b_1)\leq(a_2,b_2)$ 当且仅当满足如下两个条件之一：

1. $sgn(b_1-a_1)>sgn(b_2-a_2)$。
2. $sgn(b_1-a_1)=sgn(b_2-a_2)$ 且 $F(\{(a_1,b_1),(a_2,b_2)\})\leq F(\{(a_2,b_2),(a_1,b_1)\})$

容易验证使用这种方式定义的 $\leq$ 满足三条性质。因此使用这样定义的 $\leq$ 排序，即可使用上述算法求解。时间复杂度 $O(n\log n)$。

这个例子说明，由于性质 2.3 中给出的是蕴涵关系而不是双向关系，因而不一定可以直接使用 $F\{(x,y)\}\leq F\{(y,x)\}$ 定义比较关系。但由性质 2.3，如果 $x\leq y$ 则一定有 $F\{(x,y)\}\leq F\{(y,x)\}$，因此从这个比较关系出发仍然是一种好的思路。

在上一个例子中，不难发现如果没有 $a_i,b_i\geq 0$ 的条件，上述结论仍然成立。此时它可以表示下面这个例子：

例题 4. [5]

有 n 个箱子，第 i 个箱子有重量 w_i 和承载量 $v_i(w_i,v_i>0)$，将它们堆成一列使得每一个箱子上面的箱子总重量都不大于它的承载量。

$n\leq 10^6$

对于一个箱子的序列 $\{(w_1,v_1),\cdots,(w_n,v_n)\}$，它满足要求当且仅当 $\min_{i=1}^{n}\{v_i-\sum_{j=1}^{i-1}w_j\}\geq 0$。考虑最大化 $\min_{i=1}^{n}\{v_i-\sum_{j=1}^{i-1}w_j\}=\min_{i=1}^{n}\{\sum_{j=1}^{i-1}(-v_j-w_j)-\sum_{j=1}^{i}(-v_j)\}$。则问题可以看成上一个例题中的形式，其中 $a_i=-v_i, b_i=-v_i-w_i$，此时所有的 $b_i-a_i<0$。因此一种最优方案为将所有箱子按照 b_i 从大到小排序，即按照 v_i+w_i 从小到大排序。

类似地，对于例题 1 中的问题，如果将限制改为 $a,b\geq 0$ 或者 $a\geq 0$ 或者类似的条件，则对一些特殊元素进行处理后，可以通过类似于比较 a_ib_j, a_jb_i 的大小关系来比较 $(a_i,b_i),(a_j,b_j)$，使得比较关系满足性质 2.1,2.2,2.3。因此这种问题仍然可以使用上述算法求解。但如果没有限制，变为 a,b 为任意实数，则不难找到一组反例使得三条性质一定不能同时成立，因而此时的问题不再可以使用之前的算法。

[5]题目来源：改编自经典问题

2.3.2 部分模型

上文中给出的三个例子分别对应三种不同的元素、函数以及元素间的比较关系，这里将一组元素、函数以及比较关系称为模型，则上述例子对应了三种模型：

1. 元素形如 $(a,b)(a,b\in\mathbb{R},a\geq 0)$，$F(\{(a_1,b_1),(a_2,b_2),\cdots,(a_k,b_k)\})=\sum_{1\leq i<j\leq k}a_ib_j$，比较 F 的方式为实数比较。元素间具体的比较方式见上。

2. 元素形如 $(a,b)(a,b\in\mathbb{R})$，$F(\{(a_1,b_1),\cdots,(a_n,b_n)\})=-\min_{i=1}^{n}\{(\sum_{j=1}^{i-1}b_j)-(\sum_{j=1}^{i}a_j)\}$，比较 F 的方式为实数比较。元素间具体的比较方式见例题 3。

3. 元素形如字符串 s，$F(\{s_1,\cdots,s_n\}=s_1+\cdots+s_n$，比较 F 的方式为字典序比较。比较两个元素 s,t 的方式为比较 $s+t,t+s$ 的字典序。

上述几种模型（尤其是前两种）在信息学竞赛中有较为广泛的应用。还有很多问题可以转化为上面的几种模型或者其它满足上述性质的模型（例如，$F(\{(a_1,b_1),\cdots,(a_n,b_n)\}=\sum_{i=1}^{n}a_i\prod_{j=i+1}^{n}b_j$，其中 $a_i,b_i>0$），在此不再展开。

2.4 一些该方法的其它应用

使用上述算法中的性质和处理方式，可以解决更多的问题。例如如下一类问题：

给定 n 个元素 $x_1,\cdots,x_n$，以及一个定义域为这些元素的序列的函数 F。该函数满足上述三条性质。选出 k 个元素，将它们任意排列成 $(y_1,\cdots,y_k)$，最小化：

$$F(\{y_1,\cdots,y_k\})$$

按照上述算法，可以建立 x 间的弱序关系，不妨设 $x_1\leq x_2\leq\cdots\leq x_n$。设选择的元素为 $x_{s_1},\cdots,x_{s_k}(s_1<s_2<\cdots<s_k)$，则对于选择元素的集合为 $\{x_{s_1},\cdots,x_{s_k}\}$ 的情况，由上述算法，序列 $\{x_{s_1},\cdots,x_{s_k}\}$ 是这种情况中的一种最优方案。因此一定存在一种最优方案是 $\{x_1,\cdots,x_n\}$ 的子序列。根据这个性质，可以将按顺序选择若干个元素的问题变为选取序列的一个子序列的问题。这在一些情况下可以降低问题的难度，例如：

例题 5. [6]

有 n 个物品，第 i 个物品有非负费用 c_i 和价值 v_i，两个人进行如下博弈：

1. 第一个人选择一个物品，付出 c_i 的代价。第一个人也可以选择直接让博弈结束。

2. 第二个人可以选择删除这个物品，这会使得博弈回到第一步（但第一个人已经付出的代价不会改变），第二个人最多可以进行 k 次这种操作。第二个人也可以选择不操作，此时第一个人获得 v_i 的收益，博弈结束。

[6]题目来源：2014-2015 ACM-ICPC, Central Europe Regional Contest L The Imp

第一个人的总收益为收益减去付出的所有代价。第一个人希望最大化收益，第二个人希望最小化它。求双方最优操作下博弈的结果。

$n \le 1.5 \times 10^5, k \le 9$

如果第一个人选择不拿物品，则收益为 0。如果第一个人有更优的策略，则他一定会在博弈结束时得到一个物品的收益。此时第一个人的策略可以看成，选择一个由 $k+1$ 个物品组成的序列 $\{s_1, \cdots, s_{k+1}\}$，第 i 次操作时选择第 s_i 个物品。如果第一个人的策略固定，则容易得到第二个人任意操作时，第一个人的最小总收益为：

$$\min_{i=1}^{k+1}\{v_{s_i} - \sum_{j=1}^{i} c_{s_j}\}$$

考虑对于第一个人所有的策略，上式可能的最大值。此时，第一个人通过选择这种策略，可以使总收益不小于这个值。同时，不存在一种方案使得 min 更大，因此对于第一个人的一种策略，一定存在一个物品使得如果博弈在他选择这个物品时结束，则他的总收益小于等于这个值，因此第二个人一定可以使得总收益不超过这个值。从而游戏的结果即为这个值，问题变为最大化这个值。这与例题 4 中的情况类似，由例题 4 做法可以得到一种最优方案是将所有物品按照 v_{s_i} 从小到大排序。将所有物品按照价值从小到大排序，则第一个人的策略一定是选择排序后的一个子序列。设 $f_{i,j}$ 表示在排序后 $[i,n]$ 这段后缀中选择 j 个物品，上式的最大值，则容易得到 $f_{i,j} = \max\{f_{i+1,j}, \min\{v_i - c_i, f_{i+1,j-1} - c_i\}\}$。该算法时间复杂度为 $O(nk)$。

2.5 总结

该算法的出发点在于分析在一组方案中交换相邻两个元素后，当前方案权值的变化，由此得到最优的方案需要满足的性质。因此它被称为”Exchange Argument”。而该算法的实质在于找到一种比较方式构成序关系，使得只要交换一对相邻且满足序关系中后者小于等于前者的元素，则交换后的的方案的优秀度不会变差，因而可以推出满足序关系的序列一定是最优方案。直接使用交换后是否不会变差作为比较方式可能会出现问题，但可以发现从交换的比较关系出发是一种优秀的方案。上文中给出了几个可以使用这种算法的经典例子。这些例子之间看似没有关联，但它们都可以使用同一种方法求解，这说明这种算法存在广泛的应用。

同时该算法的思想，即分析最优方案中局部的性质从而推出最优方案中整体的性质，是一种分析问题性质的有力做法。关于该思想的更多分析在本文中不再展开。

3 问题在有根树上的拓展以及解法

3.1 问题引入

在这一部分中，主要讨论下面给出的一类问题，这类问题的形式与本文上一部分中给出的问题形式类似，但加入了一种与有根树相关的，类似偏序的限制。定义 y 在排列 p 中比 x 更先出现，当且仅当 $\exists 1 \le i < j \le n, p_i = y, p_j = x$。则这类问题可以表示为：

给定 n 个元素 $x_1, \cdots, x_n$，以及一个定义域为这些元素的序列，值域为有序集合的函数 F。再给出一棵 n 个点的有根树 T，其中 1 为根。记点 $i(2 \le i \le n)$ 的父亲为 fa_i。在接下来的部分中，为了使描述更简洁，本文使用“点 i 对应元素 p_i”这种方式来描述元素与树中点的关系。

称一个排列满足 T 的限制，当且仅当 $\forall 2 \le i \le n$，fa_i 在排列中比 i 更先出现。求出对于所有满足 T 的限制的 n 阶排列 p，如下表达式的最小值：

$$F(\{x_{p_1}, x_{p_2}, \cdots, x_{p_n}\})$$

例如下述问题：

例题 6. [7]

给出一棵 n 个点的有根树，其中 1 为根。点 i 上有数字 $v_i (v_i \in \{0, 1\})$。现在需要将这些点排成一列 $p_1, \cdots, p_n$，满足对于任意一个点 x，在排列中 x 除去自身外的所有祖先都比 x 更先出现。最小化序列 $\{v_{p_1}, v_{p_2}, \cdots, v_{p_n}\}$ 的逆序对数（满足 $1 \le i < j \le n$ 且 $v_{p_i} > v_{p_j}$ 的对数）。

$n \le 2 \times 10^5$

在本题中，$v_{p_i} > v_{p_j}$ 当且仅当 $v_{p_i} = 1, v_{p_j} = 0$，因此如果将 $v_i = 1$ 的元素看作上一部分中第一种模型中的 $(1, 0)$，将 $v_i = 0$ 的元素看作 $(0, 1)$，则该问题可以被表述为上述形式。

如果不存在 T 的限制，则由上一部分的结果，只需要问题满足存在一种元素间的比较关系满足上一部分给出的三条性质，就可以使用 Exchange Arguments 方法求解。但在加入树的限制后，即使 F 满足这一条件，也可能不存在时间复杂度较优的算法，例如如下模型：

给出 n 个数 $v_1, \cdots, v_n$，将它们排序，使得前 k 个数的和最大。

定义 $v_i \le v_j$ 即为在实数意义的进行比较。这显然满足性质 2.1,2.2，不难证明这样的比较关系也满足性质 2.3。因此 F 满足上文提出的限制。但考虑如下问题：

3.1.1 一种反例

给一棵 n 个点的有根树，点有点权。给定正整数 k，选择一个树上包含根的大小为 k 的连通块，使得连通块内点权和最大。

[7] 题目来源：AGC023F 01 on Tree

该问题等价于选择一个 n 个点的排列，满足 T 的限制，并最大化排列中前 k 个点的点权和。没有树上限制时该问题等价于上述模型。该问题可以使用动态规划的方式以 $O(n^2)$ 的时间复杂度解决，且如果树形如根节点有三个子树、每个子树都是一条链，则问题等价于给出三个任意序列 a,b,c，求出 $\max_{i_1+i_2+i_3=k} a_{i_1}+b_{i_2}+c_{i_3}$，因而这个问题难以在低于 $O(n^2)$ 的时间复杂度内解决。沿用上一部分中的思路考虑也难以解决问题。这说明如果希望使用上一部分中基于交换的贪心思路解决这类问题，则需要 F 满足更强的性质以及对问题进一步的分析。

3.2 对树上问题的分析

设问题中存在 $\leq$ 满足性质 2.1,2.2,2.3，则可以得到如下引理：

Lemma 3.1. 设元素为 $x_1,\cdots,x_n$，有根树的根为 1。设 v 满足 x_v 是 $x_2,\cdots,x_n$ 中的最小元素：$\forall i\in\{2,\cdots,n\}, x_t\leq x_i$，$u$ 是 v 在树上的父亲，则存在一种最优的排列，使得 u,v 在排列中相邻，即 $\exists i\in\{1,\cdots,n-1\}, p_i=u, p_{i+1}=v$ 或者 $p_{i+1}=u, p_i=v$。

证明. 对于任意一个满足 T 的限制的排列 p，设 a,b 满足 $p_a=u, p_b=v$，易得 $a<b$。因为 p_1 一定是 T 的根，因而 $\forall i\in\{a+1,\cdots,b-1\}, x_v\leq x_{p_i}$。由性质 2.3，有：

$$F(\{x_{p_1},\cdots,x_{p_{b-1}},x_{p_b},\cdots,x_{p_n}\})\geq F(\{x_{p_1},\cdots,x_{p_b},x_{p_{b-1}},\cdots,x_{p_n}\})\geq\cdots$$
$$\geq F(\{x_{p_1},\cdots,x_{p_a},x_{p_b},x_{p_{a+1}},\cdots,x_{p_{b-1}},x_{p_{b+1}},\cdots,x_{p_n}\})$$

如果 $\exists t\in\{a+1,\cdots,b-1\}$ 使得 x_{p_t} 是 x_v 的祖先，则因为 $p_t\neq v,u$，而 u 是 v 的父亲，因此 p_t 是 u 的祖先且 $p_t\neq u$，与 $p_1,\cdots,p_n$ 满足 T 的限制矛盾。因此不存在 $t\in\{a+1,\cdots,b-1\}$ 使得 p_t 是 v 的祖先，由于其它点对间的相对顺序不变，因而 $p_1,\cdots,p_a,p_b,p_{a+1},\cdots,p_{b-1},p_{b+1},\cdots,p_n$ 满足 T 的限制。因此对于任意一个满足 T 的限制的排列 p，都存在一个满足 T 的限制，且 x_u,x_v 在排列中相邻的排列 q，使得 $F(\{x_{q_1},\cdots,x_{q_n}\})\leq F(\{x_{p_1},\cdots,x_{p_n}\})$。因此存在一种最优的排列方式满足 u,v 在排列中相邻。 □

由引理 3.1，只考虑所有 u,v 相邻的满足 T 的限制的排列时得到的最优解是原问题的一个最优解。考虑在树上将 u,v 合并为一个点，这个点代表两个元素组成的序列 $\{x_u,x_v\}$。

此时问题相当于，给出一棵 $n-1$ 个点的有根树 T'，每个点对应一个元素组成的序列 s_i，选择一个满足 T 的限制的 $n-1$ 阶排列 p，将所有序列按照排列顺序拼接后得到序列 $S=s_{p_1}+s_{p_2}+\cdots+s_{p_n}$，最小化 $F(S)$。不难发现，在新的问题中，所有满足 T' 限制的排列拼接出的序列即为所有原问题中满足 T 的限制，且 x_u,x_v 在排列中相邻的排列所得到的序列。因此，新问题的最优解对应原问题的一个最优解。

新的问题与之前的问题形式相同，但每个点不再对应一个元素，而对应一个元素组成的序列。因而如果希望继续使用上述引理，则需要将元素间的比较关系拓展为元素构成的序列间的比较关系，即如下条件：

设所有给出的元素组成的集合为 S，所有由 S 中元素构成的非空序列组成的集合为 T。存在一种 T 中元素间的二元比较关系 $\leq$，满足如下三条性质：

性质 3.1. $\forall a, b \in T$，$a \leq b, b \leq a$ 中至少有一者成立。

性质 3.2. $\forall a, b, c \in T$，如果 $a \leq b, b \leq c$，则 $a \leq c$。

性质 3.3. $\forall a, b \in T$，如果 $a \leq b$，则对于任意由 S 中元素组成的序列 s_1, s_2，下列不等式都成立：

$$F(s_1 + a + b + s_2) \leq F(s_1 + b + a + s_2)$$

如果问题满足上述性质，则存在两种可以在 $O(n \log n)$ 次序列比较，$O(n)$ 次序列合并中得到问题的一个最优解的算法。下一部分将会给出对两种算法的分析。

3.3 树上问题的两种算法

接下来的算法都在问题满足上述性质的基础上考虑。此时序列满足和元素相同的比较性质，可以将问题中每个点对应一个元素 x_i 变为每个点对应一个序列 $s_i = \{x_i\}$，因此接下来只考虑序列间的比较和合并。

3.3.1 算法一

在 T 满足上述性质的基础上继续考虑上述做法，此时在引理 3.1 中将元素变为序列，引理仍然成立。因此在一个点对应一个元素组成的序列时，也可以进行上文中合并两个点的操作。上述每次操作都会在树上将两个点以及它们对应的序列合并，合并 $n-1$ 次后，有根树变为一个点，由引理这个点对应的序列即为一种最优解。通过上述步骤，可以得到如下算法：

进行 $n-1$ 次操作，每次找到除去根节点对应的序列外，其余点对应序列中最小的序列。设这个点为 x，将 x 对应的序列拼接到 x 父亲的序列的末尾，并在树上将 x 与 x 的父亲合并为一个点。进行 $n-1$ 次操作后，根节点对应的序列即为一种最优解。

使用引理 3.1 容易证明，每一次合并点后，新得到的问题的最优解对应上一个问题的一个最优解，因此上述算法的正确性得以保证。

在实现时，如果使用树上并查集维护缩点，将并查集中一个连通块的信息记录在连通块的根处，使用任意一种支持插入删除元素并查询最小值的数据结构（例如平衡树）维护除去根节点外所有点对应的序列，则上述算法需要进行 $O(n \log n)$ 次序列比较，$O(n)$ 次序列合并，其余部分时间复杂度为 $O(n \log n)$。

3.3.2 算法二

通过从不同的角度分析问题，可以得到另外一种解决这类问题的算法。这个算法基于如下两个引理：

Lemma 3.2. 称一个排列 p 满足序列 $\{v_1,\cdots,v_k\}$ 的顺序关系，当且仅当在排列中 v_1 比 v_2 更先出现，v_2 比 v_3 更先出现，……，v_{k-1} 比 v_k 更先出现。

设有根树 T 中节点 u 有若干个儿子节点 $v_1,\cdots,v_k$，且这些儿子节点都是叶子节点。则如果满足 $s_{v_1}\le s_{v_2}\le\cdots\le s_{v_k}$，则存在一种满足 T 的限制的最优的元素排列，其满足序列 $\{v_1,\cdots,v_k\}$ 的顺序关系。

为了证明这个引理，这里给出另一个引理：

Lemma 3.3. 设有根树中 u 有两个儿子节点 v_1,v_2，如果 $s_{v_1}\le s_{v_2}$，对于任意一种满足 T 要求的排列 p，设 $p_i=v_1,p_j=v_2$，则如果 $j\le i$，则可以通过只改变排列中 $p_j,p_{j+1},...,p_i$ 的值，得到另外一个排列 q，满足 $F(s_{q_1}+\cdots+s_{q_n})\le F(s_{p_1}+\cdots+s_{p_n})$ 且在排列 q 中 v_1 比 v_2 先出现。

证明. 因为 $s_{v_1}\le s_{v_2}$，由性质3.1,3.2，可以得到不等式 $s_{p_{j+1}}+\cdots+s_{p_{i-1}}\le s_{v_2},s_{v_1}\le s_{p_{j+1}},\cdots,s_{p_{i-1}}$ 中至少有一个成立。

如果前者成立，由性质 3.3，可以得到：

$$\begin{aligned}&F(s_{p_1}+\cdots+s_{p_{j-1}}+s_{v_2}+s_{p_{j+1}}+\cdots+s_{p_{i-1}}+s_{v_1}+s_{p_{i+1}}+\cdots+s_{p_n})\\\ge&F(s_{p_1}+\cdots+s_{p_{j-1}}+s_{p_{j+1}}+\cdots+s_{p_{i-1}}+s_{v_2}+s_{v_1}+s_{p_{i+1}}+\cdots+s_{p_n})\\\ge&F(s_{p_1}+\cdots+s_{p_{j-1}}+s_{p_{j+1}}+\cdots+s_{p_{i-1}}+s_{v_1}+s_{v_2}+s_{p_{i+1}}+\cdots+s_{p_n})\end{aligned}$$

因为 u,v 为叶子节点且父亲相同，不难得到 $p_1,\cdots,p_{j-1},p_{j+1},\cdots,p_{i-1},v_1,v_2,p_{j+1},\cdots,p_n$ 满足 T 的限制，不难发现这个排列满足引理的要求。后者成立的情况同理，因此引理成立。 □

由这个引理，可以给出前一个引理的证明：

证明. 对于任意一个由 $\{v_1,\cdots,v_k\}$ 中元素组成，且元素两两不同的序列 $\{v_{i_1},\cdots,v_{i_l}\}$，定义 $g(\{v_{i_1},\cdots,v_{i_l}\})$ 为所有满足 T 的限制，且满足序列 $\{v_{i_1},\cdots,v_{i_k}\}$ 的顺序关系的排列 p 中，$F(s_{p_1}+\cdots+s_{p_n})$ 的最小值。由于 $v_1,\cdots,v_k$ 为父亲相同的叶子节点，因此不难发现对于任意满足条件的序列，都存在至少一种合法的排列，因此 g 是良定义的。

考虑两个满足 $s_{v_x}\le s_{v_y}$ 的元素 x,y。对于任意一个满足条件的序列 $\{v_{i_1},\cdots,v_{i_l}\}$，如果 $\exists t\in\{1,\cdots,l-1\}$ 使得 $i_t=y,i_{t+1}=x$，令 i' 为在序列 $\{i_1,\cdots,i_l\}$ 中交换 i_t,i_{t+1} 的结果，考虑 $g(\{v_{i_1},\cdots,v_{i_l}\})$ 与 $g(\{v_{i'_1},\cdots,v_{i'_l}\})$ 间的大小关系。

对于任意一组在计算 $g(\{v_{i_1},\cdots,v_{i_l}\})$ 时满足条件的排列 p，不难发现在 p 中，v_y 比 v_x 更先出现，且设 $p_j=v_y,p_i=v_x$，则 $v_{i_1},...,v_{i_l}$ 没有在 $p_{j+1},...,p_{i-1}$ 中出现。由引理 3.3，存在一个在 p 上只改变 $p_j,\cdots,p_i$ 的值，且使得 v_x 比 v_y 更先出现的排列 q，使得 q 满足 T 的限制

且 $F(s_{q_1}+\cdots+s_{q_n}) \le F(s_{p_1}+\cdots+s_{p_n})$。此时 q 满足序列 $\{v_{i_1},\cdots,v_{i_{l-1}},v_x,v_y,v_{i_{l+1}},\cdots,v_{i_l}\}$ 的顺序关系。因此 q 是在计算 $g(\{v_{i_1},\cdots,v_{i_{l-1}},v_x,v_y,v_{i_{l+1}},\cdots,v_{i_l}\})$ 中一种满足要求的排列。从而可以得到 $g(\{v_{i_1},\cdots,v_{i_{l-1}},v_x,v_y,v_{i_{l+1}},\cdots,v_{i_l}\}) \le g(\{v_{i_1},\cdots,v_{i_l}\})$。

此时如果将 g 看作上一部分中算法需要最小化的函数，则当前的 $\le$ 关系满足性质 2.3，因为 $\le$ 满足性质 3.1,3.2，所以 $\le$ 一定满足性质 2.1,2.2。因此由上一部分中的 Exchange Arguments 方法，可以得到如果 $s_{v_1} \le \cdots \le s_{v_k}$，则 $g(\{v_1,\cdots,v_k\})$ 是所有 k 阶排列 p 中 $g(\{v_{p_1},\cdots,v_{p_n}\})$ 的最小值。因此存在一种最优的排列 p 满足序列 $\{v_1,\cdots,v_k\}$ 的顺序关系。因此引理成立。 □

另一个引理为：

Lemma 3.4. 设在有根树 T 中，点 u 的所有儿子节点为 $v_1,...,v_k$，且 u 的所有儿子节点都是叶子节点。假设 $s_{v_1} \le s_{v_2} \le \cdots \le s_{v_k}$。如果 $s_{v_1} \le s_u$，则对于任意一个满足 T 的限制，且满足序列 $\{v_1,\cdots,v_k\}$ 的顺序关系的排列 p，都存在一个排列 q 满足 T 的限制，且满足序列 $\{v_1,\cdots,v_k\}$ 的顺序关系，并且 $F(s_{q_1}+\cdots+s_{q_n}) \le F(s_{p_1}+\cdots+s_{p_n})$。

证明. 设在排列 p 中 $p_j = u, p_i = v_1$，则 $j \le i$。如果 $s_{p_i} \le s_{p_j}$，则不等式 $s_{p_{j+1}}+\cdots+s_{p_{i-1}} \le s_{v_2}, s_{v_1} \le s_{p_{j+1}},\cdots,s_{p_{i-1}}$ 中至少有一者成立。

如果前者成立，则由性质 3.3 可以得到：

$$\begin{aligned} &F(s_{p_1}+\cdots+s_{p_{j-1}}+s_{p_j}+s_{p_{j+1}}+\cdots+s_{p_{i-1}}+s_{p_i}+s_{p_{i+1}}+\cdots+s_{p_n}) \\ &\ge F(s_{p_1}+\cdots+s_{p_{j-1}}+s_{p_{j+1}}+\cdots+s_{p_{i-1}}+s_{p_j}+s_{p_i}+s_{p_{i+1}}+\cdots+s_{p_n}\}) \end{aligned}$$

因为 p 满足序列 $\{v_1,\cdots,v_k\}$ 的顺序关系，因而 $v_2,...,v_k$ 在排列中都在 v_1 之后出现。不难得到 $p_1,\cdots,p_{j-1},p_{j+1},\cdots,p_{i-1},u,v_1,p_{j+1},\cdots,p_n$ 满足 T 的限制，且满足序列 $\{v_1,\cdots,v_k\}$ 的顺序关系。后者成立的情况同理。因此引理成立。 □

由上述引理，可以推出如下引理：

Lemma 3.5. 对于 T 中的任意点 u，设 u 子树中所有点构成集合 $subtree_u$，将 $subtree_u$ 中元素分为若干个序列 $t_1,...,t_k$（每个 $subtree_u$ 中元素在所有序列中正好出现一次），设 $t_i = \{t_{i,1},...,t_{i,l_i}\}$，令 $c_i = s_{t_{i,1}}+\cdots+s_{t_{i,l_i}}$，则存在一种方式满足如下条件：

1. $c_1 \le c_2 \le \cdots \le c_k$
2. 如果在原问题中删去 u 的子树，并在树中向 u 的父亲加入 k 个儿子节点 $v_1,\cdots,v_k$，它们依次对应序列 $c_1,\cdots,c_k$，则新问题中所有满足序列 $\{v_1,\cdots,v_k\}$ 的顺序关系的排列中的最优解对应原问题的一组最优解。

证明. 在树上进行归纳。对于叶子节点 u，令 $k = 1, d_1 = \{u\}$，此时显然成立。

对于非叶子节点 u，设它有 m 个儿子节点为 $v_1,...,v_m$。由归纳假设，它的每个儿子都满足引理。因而对于每个儿子 v_i，都存在满足引理条件的点的序列 $t_{i,1},\cdots,t_{i,k_i}$ 以及对应元素

的序列 $c_{i,1},\cdots,c_{i,k_i}$。对每个子树使用第二条条件后，书中 u 的子树变为 u 有 $\sum_{j=1}^{m}k_i$ 个儿子节点 $v'_{1,1},\cdots,v'_{1,k_1},\cdots,v'_{m,1},\cdots,v'_{m,k_m}$，这些节点都是叶子节点，且它们对应上文中的那些序列。因为 $c_{i,1}\le\cdots\le c_{i,k_i}$，因而使用归并的方式可以将所有儿子节点排成一列，使得在这个排列中，对于任意一个儿子节点 v_i，排列都满足 $\{v'_{i,1},\cdots,v'_{i,k_i}\}$ 的顺序限制，且排列中所有点对应的序列在比较关系下单调不降：对于排列中两个相邻的元素 $v'_{i,a},v'_{j,b}$，满足 $c_{i,a}\le c_{j,b}$。记 $l=\sum_{j=1}^{m}k_i$，当前点的排列为 $p_1,\cdots,p_l$，点对应的序列为 $s_{p_1},\cdots,s_{p_l}$。由引理 3.2，存在当前问题的一组最优解，使得解的排列满足 $\{p_1,\cdots,p_l\}$ 的顺序限制。又因为 $\{p_1,\cdots,p_l\}$ 满足所有 $\{v'_{i,1},\cdots,v'_{i,k_i}\}$ 的顺序限制，从而对每个子树考虑第二条性质，可以得到满足 $\{p_1,\cdots,p_l\}$ 的顺序限制的最优解对应原问题的一组最优解。

考虑这样一组最优解的性质，如果 $s_{p_1}\le s_u$，则由引理 3.4，存在一种排列满足 $\{p_1,\cdots,p_l\}$ 的顺序限制，满足 u,p_1 在排列中相邻，且这组解的 F 不大于最优解的 F，因此这也是一组最优解。考虑在树中将节点 p_1 与 u 合并，将对应的序列进行拼接（拼接后的 $s'_u=s_u+s_{p_1}$）。则不难得到新问题中满足 $\{p_2,\cdots,p_l\}$ 的顺序限制的解对应原问题中一组满足 $\{p_1,\cdots,p_l\}$ 的顺序限制的解，且原问题中任意一组满足 u,p_1 在排列中相邻的解都对应一组新问题的解。因此由引理 3.2，新问题中存在一组满足 $\{p_2,\cdots,p_l\}$ 的顺序限制的最优解，从而这组解对应之前问题中所有满足 $\{p_1,\cdots,p_l\}$ 的顺序限制的排列的最优解，从而对应原问题的一组最优解。

不断进行上面的操作，直到当前排列中的第一个点满足 $s_{p_i}\not\le s_u$ 或者所有叶子节点都被合并。对于第二种情况，取 $k=1,d_1,c_1$ 为之前合并的结果即可满足引理条件。对于第一种情况，设合并了 $p_1,\cdots,p_{i-1}$，则此时满足 $s_u\le s_{p_i}\le\cdots\le s_{p_k}$。考虑令 t_1 为当前 u 对应的点序列，$t_{2,\dots}$ 为 $p_i,\cdots,p_k$ 对应的点序列，则这满足第一条要求。同时因为 $p_i,\cdots,p_k$ 都是叶子节点且父亲为 u，因此在第二条要求中的新问题中，所有满足序列 $\{u,p_i,\cdots,p_k\}$ 的顺序关系的排列都可以对应此时一种满足序列 $\{p_i,\cdots,p_k\}$ 的顺序关系的最优解，由上述分析可得这个最优解对应一组原问题的最优解，因此新问题满足这个条件的最优解对应一组原问题的最优解，引理成立。 □

使用上述引理证明中的构造方法，即可得到第二种算法：

考虑对于每个点 u 求出引理 3.5 中该点的所有 c_i 按顺序排列构成的序列 S_u。对于一个 u，进行如下过程：

首先对于 u 的每个儿子 v_i 求出 S_{v_i}，接着将所有的 S_{v_i} 按照序列的比较关系归并得到 S_u。

令 $C=\{x_u\}$，重复进行下一步：

找到 S_u 中的第一个元素，设为 c。如果 S_u 为空或者 $c\notin C$ 则结束过程。否则，从 S_u 中删去 c，并将序列 c 拼接在 C 后得到新的 C。

在重复的过程结束后，将 C 加入 S_u 的开头，结束当前点的构造过程。

可以在树中进行深度优先搜索，自下向上求出所有的 S_u，求出根节点的 S 后，由引理 3.5 可知，将 S 中所有序列拼接即可得到问题的一组最优解。该算法需要 $O(n)$ 次对两个 S

进行归并，$O(n)$ 次询问/删除 S 的第一个序列以及 $O(n)$ 次合并序列。如果使用平衡树或类似的数据结构维护 S，则该算法需要 $O(n\log n)$ 次比较序列，与上一个算法的复杂度相同。

3.4 对具体模型的分析

对于满足存在一种比较关系满足性质 3.1,3.2,3.3 的问题，上述两种算法都可以使用 $O(n\log n)$ 次序列比较以及 $O(n)$ 次序列合并中解决问题。接下来考虑哪些模型满足这一条件。这里对上一部分中提到的三种模型进行分析：

3.4.1 第一种模型

在第一种模型中，元素形如 $(a,b)(a,b\in\mathbb{R},a\geq 0)$，$F(\{(a_1,b_1),\cdots,(a_n,b_n)\})=\sum_{1\leq i<j\leq n}a_ib_j$。考虑对于两个序列 $c_1=\{(a_1,b_1),\cdots,(a_n,b_n)\},c_2=\{(a_1^{'},b_1^{'}),\cdots,(a_m^{'},b_m^{'})\}$，$F(c_1+c_2)$ 与 $F(c_2+c_1)$ 的关系（+ 表示序列拼接）。则不难得到如下结果：

$$F(c_1+c_2)-F(c_2+c_1)=(\sum_{i=1}^{n}a_i)*(\sum_{i=1}^{m}b_i')-(\sum_{i=1}^{m}a_i')*(\sum_{i=1}^{n}b_i)$$

因此可以将序列看成 $(\sum a_i,\sum b_i)$，这样得到的结果仍然是一个元素。因此考虑使用上一部分中这种模型的比较方式对两个序列的 $(\sum a_i,\sum b_i)$ 进行比较，比较的结果即为序列比较的结果。不难发现这样比较满足性质 3.1,3.2,3.3。

3.4.2 第二种模型

在第二种模型中元素形如 $(a_i,b_i)(a,b\in\mathbb{R})$，$F(\{(a_1,b_1),\cdots,(a_n,b_n)\})=-\min_{i=1}^{n}\{(\sum_{j=1}^{i-1}b_i)-(\sum_{j=1}^{i}a_j)\}$。此时从 F 进行分析可以得到如下结论：

对于序列中两个位置连续的元素 $(a_1,b_1),(a_2,b_2)$，在序列中将它们合并为一个元素 $(-\min(-a_1,b_1-a_1-a_2),b_1+b_2-a_1-a_2-\min(-a_1,b_1-a_1-a_2))$ 不会改变 F 的值。

归纳可得，在计算 F 时任意一个元素组成的序列都可以被合并为一个等价的元素。因此在性质 3.3 中可以在不等式两侧分别将序列 s,t 合并为与之等价的元素，从而可以使用等价元素的比较结果来比较两个序列。由于元素间存在一种比较关系满足性质 2.1,2.2,2.3，因此使用这种比较关系得到的序列间比较关系满足性质 3.1,3.2,3.3。

3.4.3 第三种模型

对于第三种模型，序列中若干个字符串进行拼接后仍然是字符串，因此上述性质显然成立，但对字符串进行拼接及比较的问题则较为复杂。这部分的具体细节在本文中不再展开。

但对于这部分开头给出的反例，不存在一种 $\leq$ 关系满足上述条件。例如，令 $k=2$，则 $F(\{3,3\})>F(\{4,1\})$，但 $F(\{4,3,2\})<F(\{4,4,1\})$，由性质 3.1，$\{3,3\},\{4,1\}$ 中必定有一者小于等于另外一者，但两种情况都与性质 3.3 矛盾。因此上述条件一定不能被满足，该问题不能使用上述算法解决。

3.5 算法性质及应用

对于上述前两种模型，它们有较好的性质：在合并两个序列 c_1,c_2 时，只需要知道关于两个序列的 $O(1)$ 级别的信息，即可求出合并后序列的信息以及 $F(c_1+c_2)$。例如，对于第一种模型，只需要知道序列 c 的 $\sum a_i,\sum b_i,F(c)$，即可 $O(1)$ 求出合并后的这些信息，而在第二种模型中只需要知道等价元素 (a,b)。因为这些原因，对于这两种模型对应的问题，上述两种算法可以在 $O(n\log n)$ 的时间复杂度内得到答案。其中第一种模型对应上文给出的例题 6，而下面这个题目对应第二种模型：

例题 7. [8]

有一棵 n 个点的树，树上除去节点 1 外的每个节点有一个权值 v_i。有一个人初始在点 1，这个人有一个值 hp，初始为 0。这个人可以沿着树边移动，他第一次到达一个点 i 时，hp 会变为 $hp+v_i$，如果 $hp<0$ 则他会失败。求他能否在不失败的情况下到达一个给定的点 t。

$n\leq 10^5,|v_i|\leq 10^9$

考虑加入一个叶子节点连向 t，这个节点的权值为 $+\infty$，则如果能到达 t，则从 t 走到这个叶子节点后，一定可以经过其它所有节点。因此问题变为能否在不失败的情况下经过所有节点。如果维护一个序列，在第一次经过某个节点时将节点加入这个序列，则得到的序列是一个满足给出的树 T 的限制的排列，且经过所有节点时 v 的变化过程即为按照排列顺序经过每个点时，v 的变化过程。

此时问题变为找到一个满足 T 的限制的排列，使得按照排列顺序经过每个点，任意时刻 $v\geq 0$。考虑最大化 v 在所有时刻的最小值，则问题变为第二种模型的形式，其中一个大于等于 0 的元素 x 对应 $(0,x)$，小于 0 的元素 x 对应 $(-x,0)$。使用上述算法求解，时间复杂度为 $O(n\log n)$。

3.5.1 求一组最优解

上一部分提到，在前两种模型中，只需要维护序列的部分信息就可以比较序列并维护答案，考虑在这种情况下求出方案。上述两种算法中，都进行了 n 次合并序列操作，每次操作形如将序列 s_i 拼接到序列 s_j 末尾。在算法的过程中记录这些操作，执行完整个算法后，通过这些合并操作的信息，不难在 $O(n)$ 的时间复杂度内还原出算法求出的方案。

[8]题目来源：2013-2014 ACM-ICPC, Central Europe Regional Contest E Escape

3.5.2 求“每个点子树问题的答案”

考虑下面这类形式的问题：

给出一个上述形式的问题，称一个以 u 为根的子树的子问题为：只保留树中 u 子树内的点作为新的树 T'，以及这些点对应的元素，以 u 为 T' 的根，其余问题形式与之前的问题相同。

对于树中每个点 u，求出以 u 为根的子树的子问题的答案。

下面是一个这种问题的例子：

例题 8. [9]

给定一棵 n 个点的有根树，1 为根，点有点权 v_i。你有若干枚棋子，可以进行如下操作：

1. 选择一个上面有棋子的点，将这个点上的所有棋子拿回。
2. 选择一个上面没有棋子，且它的儿子上都有棋子的点 i，向这个点上放 v_i 个棋子。

对于每个点，求出如果需要在这个点上放上棋子，则初始时最小需要多少枚棋子。

$n \le 2 \times 10^5, 1 \le v_i \le 10^9$

分析题目不难得到如下性质：

1. 如果需要在 i 上放上棋子，则过程中只会向 i 的子树中放棋子，且正好会向 i 子树中的每个点放一次棋子。
2. 存在一种最优的操作方式，使得对于任意点 x，在向 x 上放上棋子后，下一步操作为拿回 x 儿子上的棋子。

这里略去证明。通过上述性质，设 $p_1, \cdots, p_k$ 为操作中所有向点上放棋子的操作的点按顺序构成的序列，则对于向 i 上放棋子的问题，存在一种最优解使得 p 是 i 子树中所有点的一个排列。该排列需要满足，对于树上任意一个点 u，设 u 的父亲为 v，如果 u, v 都在排列中出现，则在排列中 u 一定在 v 之前出现。同时，设 s_i 为点 i 所有儿子的 v 之和，则由第二条性质，在点 p_i 放下棋子后，此时总共放下的棋子数量为 $(\sum_{j=1}^{i} v_{p_j}) - (\sum_{j=1}^{i-1} s_{p_j})$，因此相当于最小化 $\max_i((\sum_{j=1}^{i} v_{p_j}) - (\sum_{j=1}^{i-1} s_{p_j}))$。这与上述第二种模型类似，唯一的区别是树上一条边限制的方向与之前相反。

注意到 $(\sum_{j=1}^{i} v_{p_j}) - (\sum_{j=1}^{i-1} s_{p_j}) = (\sum_{i=1}^{k} v_{p_i} - s_{p_i}) + (\sum_{i=1}^{k-i+1} s_{p_{k-i}}) - (\sum_{i=1}^{k-i} v_{p_{k-i}})$，因此将排列翻转后的 F 仍然与之前形式相同，而翻转后树的限制变为 u 的父亲 v 在排列中一定在 u 之前出现，这与上文中满足 T 的限制等价。因此通过上述方式，问题被转化为对于树中每个点 u，求出以 u 为根的子树的子问题的答案。

[9] 题目来源：UOJ418【集训队作业 2018】三角形

使用算法二中的思路容易解决这种问题。在算法二中，由引理 3.5 不难得到如下性质：

记 S_u 为引理 3.5 中 u 构造的所有序列构成的序列，将 S_u 内元素按顺序拼接，得到的序列即为一种以 u 为根的子树的子问题的最优解。

因此，如果在算法二中使用平衡树或者类似的数据结构维护 S_u，即可同时在平衡树中维护以 u 为根的子树的子问题的最优解。这样的过程复杂度与算法二相同。

使用算法一的思路也可以解决这个问题，但需要进行更多的分析，可以得到如下引理：

Lemma 3.6. 对于有根树 T 中的任意点 u，设 u 的儿子节点构成集合 son_u，则 $\forall S \in son_u$，令 $U(u,S)$ 为 u 以及 S 中所有点在树中的子树内的所有节点构成的集合，则：

对于任意一种在 T 中使用算法一求出的最优排列 p，存在一种在只保留树中 $U(u,S)$ 的导出子图，且以 u 为根的树上使用算法一求出的排列 q，使得 p 满足 $\{q_1,\cdots\}$ 的顺序关系。

证明. 对 T 的节点数进行归纳，只有一个节点时显然成立。

对于其余情况，考虑在 T 中进行算法一过程中的第一次合并操作。如果这次合并操作没有涉及到属于 $U(u,S)$ 的点，则操作后 $U(u,S)$ 不变，这部分中使用算法一求出的最优排列不变，由归纳假设可知此时结论成立。如果这次合并只影响到了 u（例如 u 与父亲节点合并或者某个不在 $U(u,S)$ 内的儿子节点与 u 合并），则合并后 $U(u,S)$ 在树中仍然有引理中要求的结构，且在合并后 $U(u,S)$ 的问题中，树结构不变，但根节点对应的序列会改变。但可以发现算法一中根节点对应的序列不影响算法的过程，因此此时结论仍然成立。

否则，合并操作一定合并了两个属于 $U(u,S)$ 的节点。设合并点为 u 与 u 的父亲节点 fa_u，则 s_u 为 T 中除去根节点外所有节点对应序列的一个最小值，因此它也是 $U(u,S)$ 除去根节点 u 外所有节点对应序列的一个最小值。因此存在在保留 $U(u,S)$ 的导出子图中以 u 为根得到的有根树中进行算法一的一种方式，使得第一步合并 u 与 fa_u。由归纳假设可知此时结论成立。因此引理成立。 □

由引理 3.6，对于任意一种算法一求出的最优排列 p，对于任意树中节点 x，在 p 中只保留属于 x 子树内的节点得到的排列即为对 x 子树内的问题使用算法一得到的一种最优解。因此通过这个最优排列，可以知道在所有子树内的问题中元素一定满足一种顺序。此时可以使用支持合并的数据结构，在树中进行深度优先搜索，自下向上维护每个节点的一种最优解。一种实现方式为使用动态开点线段树进行维护，合并时使用线段树合并，该算法的复杂度也与算法一相同。

由引理 3.5 不难发现，算法二求出的最优排列 p 也满足这一性质：对于任意树中节点 x，在 p 中只保留属于 x 子树内的节点得到的排列即为对 x 子树内的问题的一种最优解。但如果最优排列 p 不是由上述算法求出，而是任意一个最优的排列，则容易构造出上述性质的反例。因此上述结论只对本文中给出的两种算法求出的最优解有保证。

3.5.3 部分其它应用

最后，这里再给出一个与算法二的做法有关的题目：

例题 9. [10]

有 n 堆石子排成一列，第 i 堆石子有 p_i 个。对于每个 i，求出如下问题的答案：

你需要将所有石子合并成一堆，每次你可以选择相邻的两堆石子，将它们合并成一堆，代价为两堆石子数量的和。有一个额外限制：每次选择的两堆石子中，必须包含原来的第 i 堆石子。求出最小总代价。

$n \le 2 \times 10^5, 1 \le a_i \le 10^6$

考虑从位置 x 开始的问题，可以看成每次将一堆石子并入第 x 堆，设第 $i-1$ 次加入了第 p_i 堆石子（$p_1 = x$），则 p 构成一个排列，序列的限制相当于 $\forall i \in \{1, 2, \cdots, k-1\}$，$i+1$ 在排列中比 i 先出现，且 $\forall i \in \{k+1, \cdots, n\}$，$i-1$ 在排列中比 i 先出现。因此考虑一条 $1-2-\cdots-n$ 的链，设这条链以 k 为根得到的有根树为 T_k，则从位置 x 开始的问题中，合并顺序的限制等价于 p 满足 T_k 的限制。

容易发现总代价只和 p 有关，且代价为 $\sum_{i=1}^{n} a_{p_i} * (n-i+1)$。如果减去一个定值 $\sum_{i=1}^{n} a_i$，则上述代价相当于 $\sum_{1 \le i < j \le n} a_{p_i}$。因此如果令点 i 对应元素 $(a_i, 1)$，上述代价即为第一种模型中的形式。从而可以使用上述算法，在 $O(n \log n)$ 的时间复杂度内求出单个问题的答案。

如果使用算法二，则该算法求出以 x 为根的答案的步骤为：

求出在引理 3.5 中，$1-2-\cdots-(x-1)$ 的链，以 $x-1$ 为根时根节点得到的所有序列按顺序组成的序列 L_{x-1}，以及 $(x+1)-\cdots-n$ 的链，以 $x+1$ 为根时根节点得到的序列 R_{x+1}。由引理 3.2，3.4 可以得到，设将 L_{x-1}，R_{x+1} 按照比较关系归并后得到的序列为 S'_x，则将 S'_x 中所有序列按顺序拼接，再在得到的序列开头加入 x 后，得到的方案对应原问题的一组解，且是由引理 3.5 转换后问题的一组最优解，因此这组解是原问题的一组最优解。

考虑 L_x 与 L_{x+1} 间的差别，根据上述算法，求 L_{x+1} 时，$x+1$ 作为根，只有一个儿子 x，且对 x 的子树进行算法二可以得到 L_x。因此算法二中会进行如下操作：令 $C = \{(v_x, 1)\}, L_{x+1} = L_x$，取出 L_{x+1} 中第一个元素 c，如果 $c \le C$ 则将 c 从 L_{x+1} 中删去，并将 c 拼接到 C 末尾。重复执行上一步直到不满足条件，最后将 C 加入 L_{x+1} 开头。

在 x 从 1 依次增加到 n 时，对 L 进行的操作为向开头插入或删除序列并查询开头的序列。显然插入及删除的次数为 $O(n)$ 次。对 R 也只会进行 $O(n)$ 次向开头插入或删除的操作。只需要在每次插入或删除后维护当前 L, R 按比较关系归并的一种结果，就可以维护答案。考虑使用平衡树维护一种归并的结果，删除开头序列时可以直接在平衡树中删除。在开头插入序列时，设操作为插入到 L 开头，插入序列为 s，则由构造方式，插入的序列一定小于等于当前 L 中的所有序列。从而如果将这个序列插入到平衡树中最后一个满足 $t \le s, s \not\le t$ 的元素 t 后，则这种方式既保证了有序性，也保证了 L 中所有元素在当前序列中按照顺序排列，因此插入后仍然是一种归并的结果。在平衡树中维护答案，因为序列比较，拼接，合并答案可以 $O(1)$ 维护，因而时间复杂度为 $O(n \log n)$。

[10]题目来源：Yuhao Du Contest 7 H Heavy Stones

本题中只需要求出答案，而如果两个序列 a,b 满足 $a \le b, b \le a$，则在方案中交换相邻的这两个序列一定不改变 F，因此在插入时，对于满足 $a \le b, b \le a$ 的序列 a,b 按照任意顺序排列都不会改变答案。但如果按照任意顺序插入，虽然答案一定与最优解答案相同，但当前维护的方案不一定满足树的限制。

3.6 总结

这类问题有着与上一部分中问题类似的结构，但在上一种问题上额外增加了对排列的限制，对排列的限制可以看作一个有根树的形式。从需要满足的条件以及满足条件下的解法上看，两种问题都有很强的关联。但由于加入了限制，通过一些例子可以发现只满足之前的要求并不能解决问题。本文对上一部分中的要求加强，将比较元素变为了比较序列。在此基础上，本文给出了两种从不同角度出发的算法。第一种算法从局部单个点的情况入手，通过将比较关系下最小的序列所在的点与它的父亲合并得到规模更小的问题。第二种算法则是先解决每个儿子子树内的问题，通过若干性质，将儿子子树内问题的解合并得到自身子树内问题的解。两种算法可以以相同的时间复杂度解决问题。从两个算法的解题过程出发，可以得到更多的应用，本文对一些应用进行了分析，但因为笔者水平有限，本文只举出了笔者所知的部分应用。

4 后记及展望

在信息学竞赛中，贪心思想有着广泛的应用，有很多与贪心思想有关的值得深入研究的问题。本文所述的相关算法在信息学竞赛中已经有了多次出现，但这类问题的详细分析并不简单。由于笔者水平有限，还有有许多与这类算法相关的问题笔者没有得到很好的答案，例如下列问题：

1. 本文中两部分分别对比较关系提出了三条性质的限制，是否存在比这三条性质更弱/更简单，但也能使得本文所述算法得到最优解的性质？

2. 除了上文提到的三种模型及其变形，是否存在更多满足上述性质的模型？模型是否存在更加本质的描述？

希望本文能够起到抛砖引玉的作用，希望感兴趣的读者能够进行进一步研究，从而得到这类问题更多更有趣的做法与应用。笔者也希望看到上述问题更好的结果。

感谢

感谢中国计算机学会提供学习和交流的平台。

感谢国家集训队教练杨耀良的指导。
感谢父母对我的培养和教育。
感谢学校对我的栽培。
感谢教导我的多位教练对我各方面的指导。
感谢戴江齐同学与我交流讨论，给我启发。
感谢刘一平、金天、宋佳兴等多位同学为本文审稿。
感谢温铠瑞学长在我学习信息学竞赛的过程中给我的鼓励和帮助。

参考文献

[1] Wikipedia,"Weak Orderings",https://en.wikipedia.org/wiki/Weak_ordering

[2] Errichto,"Lecture #3 —Exchange arguments (sorting with dp)",https://codeforces.com/blog/entry/63533

[3] AtCoder Grand Contest 023 Editorial,https://atcoder.jp/contests/agc023/editorial

[4] CERC 2013: Presentation of solutions,https://cerc.tcs.uj.edu.pl/2013/data/cerc2013solutions.pdf