期中复习提纲

岳镝

这份提纲的目的是从考试角度整理助教认为重要的知识点，并解答小班课上没来得及讲的例题。并不提供关于期中考试的任何官方指向。

1 数学基础、函数的阶、递推方程

1.1 知识点

数学基础 读写伪代码，求和

函数的阶 记号 $O(\cdot), \Omega(\cdot), o(\cdot), \omega(\cdot)$ 定义。两个函数阶的比较（证明？）。常用结论，例如 $\log(n!) = O(n \log n)$, $a^{\log_b n} = n^{\log_b a}$。函数阶的排序。

递推方程 迭代法、递归树、尝试法、主定理
主定理：$T(n) = aT(n/b) + f(n)$ 的求解

- 直觉：$T_1(n) = aT_1(n/b)$, $T_2(n) = f(n)$，比较谁在多项式意义下 dominate
- 主定理失效：*sub-polynomial* 意义下 dominate。例子：$T(n) = 2T(n/2) + n \log n$
- 主定理的情形 (3)：$af(n/b) \le cf(n)$ 是比 $f(n) = \Omega(n^{\log_b a + \varepsilon})$ 更强的条件，因此必须验证

1.2 例题

例 1.1. 设 $k \in \mathbb{Z}$，函数 $f(n) = \Theta(n^{\log_b a} \log^k n)$，求解递推方程 $T(n) = aT(n/b) + f(n)$。

解. 该递推方程不符合主定理的任何一种情况，故不能使用主定理。不失一般性，设 $n = b^\ell$。用递归树方法求和：

$$T(n) = \sum_{i=0}^{\ell} a^i f(n/b^i) = \sum_{i=0}^{\ell} a^i f(b^{\ell-i}).$$

注意到

$$f(b^{\ell-i}) = \Theta((b^{\ell-i})^{\log_b a} \log^k (b^{\ell-i})) = \Theta(a^{\ell-i} \cdot (\ell - i)^k).$$

故

$$T(n)=a^{\ell}\sum_{i=0}^{\ell}\Theta((\ell-i)^k)=a^{\ell}\sum_{i=1}^{\ell}\Theta(i^k)=a^{\log_b n}\sum_{i=1}^{\ell}\Theta(i^k)=n^{\log_b a}\sum_{i=1}^{\ell}\Theta(i^k)$$

当 $k>-1$ 时，$\sum_{i=1}^{\ell}i^k=\Theta(\ell^{k+1})=\Theta((\log_b n)^{k+1})$。

当 $k=-1$ 时，$\sum_{i=1}^{\ell}i^k=\Theta(\log\ell)=\Theta(\log\log n)$。

当 $k<-1$ 时，$\sum_{i=1}^{\ell}i^k=\Theta(1)$。

综上，

$$T(n)=\begin{cases}\Theta(n^{\log_b a}\cdot\log^{k+1}n), & k>-1;\\ \Theta(n^{\log_b a}\cdot\log\log n), & k=-1;\\ \Theta(n^{\log_b a}), & k<-1.\end{cases}$$

□

注. 除非明确说明，否则只需要求出 $T(n)$ 的阶，不需要写出准确的表达式。

2 分治

2.1 知识点

- 重要技术
  - 芯片测试：两两分组，每轮淘汰至少一半
  - 多项式在单位根上求值：按奇偶次项划分（这在数学上对应着快速傅里叶变换 (FFT) 的深刻背景，和考试可能关系不大）
- 重要结果
  - 选第 k 小：$O(n)$
  - 方幂 a^n：$O(\log n)$
  - 最近点对距离：$O(n\log n)$

2.2 例题

例 2.1. 设 A 是包含 n 个元素的数组，设常数 $k\in\mathbb{Z}_+$。设计算法判断 A 中是否存在出现次数大于 n/k 的元素。

解. 维护 $k-1$ 个“计数器” $c_1,c_2,\ldots,c_{k-1}$；及对应的 $k-1$ 个临时元素 $m_1,m_2,\ldots,m_{k-1}$。初始时设置 $c_j\leftarrow 0$，并设置 m_j 为空，$\forall j\in[k-1]$。

记 $A=(a_1,a_2,\ldots,a_n)$。依次枚举 $i\in[n]$，每次对 a_i 做如下判断

(1) 若存在 $j \in [k-1]$ 满足 $m_j = a_i$，则令 $c_j \leftarrow c_j + 1$；

(2) 否则，任意 $j \in [k-1]$ 均有 $m_j \neq a_i$：

(a) 若存在 $j \in [k-1]$ 使得 $c_j = 0$，则令 $c_j \leftarrow 1, m_j \leftarrow a_i$。

(b) 若任意 $j \in [k-1]$ 均有 $c_j > 0$，则令 $c_j \leftarrow c_j - 1, j = 1, 2, \ldots, k-1$。

最后，对储存的每个元素 $m_j, j \in [k-1]$，依次判断 m_j 在 A 中出现的次数是否大于 n/k 即可。时间复杂度为 $O(nk)$。

下证正确性。只需证明：若 A 中存在元素 x 出现次数大于 n/k，则算法结束时 x 必储存在某个 m_j 中。

考虑非负变量 $C = \sum_{j=1}^{k-1} c_j$。注意到每个 a_i 对 C 的贡献为 1 （情形 (1) (2a)）或 $-(k-1)$（情形 (2b)）。假设元素 x 贡献了 w^+ 次 1，w^- 次 $-(k-1)$；其他元素一共贡献 v^+ 次 1，v^- 次 $-(k-1)$。特别地 $w^+ + w^- + v^+ + v^- = n$。由于元素 x 出现次数大于 n/k，故

$$w^+ + w^- > \frac{1}{k}(w^+ + w^- + v^+ + v^-)$$

另一方面，注意到 $C \geq 0$，故

$$w^+ + v^+ - (k-1)(w^- + v^-) \geq 0$$

以上两式消去 w^-, v^+，可得 $w^+ > v^-$；表明元素 x 对 C 的正贡献不会被其他元素的负贡献抵消，故 x 最终储存在某个 m_j 中。 □

例 2.2. 设数组 $A = (x_1, x_2, \ldots, x_n)$ 由 n 个互不相等的整数组成。设计一个比较次数尽可能少的算法，找到 A 中的一个极小值点。（称下标 j 为极小值点，若 $x_j < \min\{x_{j-1}, x_{j+1}\}$；当 $j = 1, n$ 时只考虑一侧。）

解. 首先判断 $x_{n/2}$ 是否为极小值点。若是，则已经找到。否则不妨设 $x_{n/2-1} < x_{n/2}$。容易证明此时 $A[1 : n/2] = (x_1, x_2, \ldots, x_{n/2})$ 上的极小值点也是 A 上的极小值点。故只需在 $A[1 : n/2]$ 上求极小值点即可。

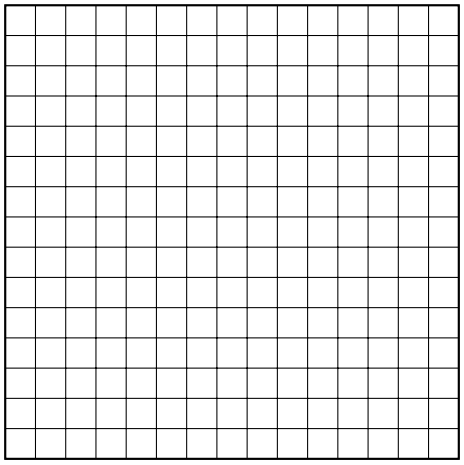
时间复杂度 $T(n) = T(n/2) + O(1)$，即 $T(n) = O(\log n)$。 □

例 2.3. n 个互不相等的整数排列在圆环上。设计一个比较次数尽可能少的算法，找到圆环上的一个极小值点

解. 取圆环的三等分点 x, y, z。不妨设 $x = \min\{x, y, z\}$。则容易证明弧 yxz 上一定存在一个不同于 y, z 的极小值点。从而归约到规模为 $2n/3$ 的子问题。

时间复杂度 $T(n) = T(2n/3) + O(1)$，即 $T(n) = O(\log n)$。 □

例 2.4. 在带权网格图中，mn 个顶点排列为 m 行 n 列，每个顶点的权值互不相等。如果一个顶点的权值比其相邻的 4 个顶点的权值都小，则称该顶点为一个极小值点。请设计一个尽可能高效的算法，在带权网格图中找出一个极小值点



解. 注意到极小值点或者在方格表边界，或者在其内部。因此可以首先枚举边界上的方格，若找到极小值点则返回；否则可以断定所有极小值点均在内部，进而寻找内部极小值点。

我们下面叙述怎样寻找方格表 A 的一个内部极小值点。首先考察 A 的第 $m/2$ 行和第 $n/2$ 列的所有 $m+n-1$ 个元素。不妨设这些元素中的最小值是 $A(m/2,j)$，且 $1<j<n/2$。若 $A(m/2,j)$ 是 A 的一个内部极小值点，则直接返回；否则其上下两个元素之一有更小的数值，不妨设 $A(m/2-1,j)<A(m/2,j)$。我们接下来说明：只需在 A 的左上 1/4 方格表 $A_1=A\binom{1,2,\ldots,m/2-1}{1,2,\ldots,n/2-1}$ 寻找极小值点。

具体而言，我们仍然首先枚举 A_1 的边界。若 A_1 的某个边界方格是 A 的内部极小值点，则直接返回。否则，A_1 的任何一个边界方格都不是 A 的内部极小值点。此时我们断言：A_1 必存在内部极小值点。从而只需递归地求出 A_1 的一个内部极小值点，它也是 A 的一个内部极小值点。

下面我们证明上述断言：若 A_1 的任何一个边界方格都不是 A 的内部极小值点，则 A_1 必存在内部极小值点。采用反证法：假设 A_1 不存在内部极小值点，则 A_1 的最小值点必在其边界上。不妨设 A_1 的最小值点为 $A(i,n/2-1)$。则有以下不等式关系

$$A(i,n/2-1)<A(m/2-1,j)<A(m/2,j)<A(i,n/2)$$

其中第一个 $<$ 是因为 $A(i,n/2-1)$ 是 A_1 上的最小值点；第二个 $<$ 是因为我们先前的假设；第三个 $<$ 是因为 $A(m/2,j)$ 是 A 的第 $m/2$ 行和第 $n/2$ 列的所有 $m+n-1$ 个元素中的最小值。至此我们有 $A(i,n/2-1)<A(i,n/2)$，这表明 A_1 的边界方格 $A(i,n/2-1)$ 正是 A 的一个内部极小值点，矛盾。

综上，我们的算法分成两部分：首先枚举 A 的边界，试图寻找边界极小值点；若不存在则寻

找 A 的内部极小值点（分治）。时间复杂度为：

$$T(m,n) = 2(m+n) - 4 + T_1(m,n)$$
$$T_1(m,n) = (m+n-1) + (\frac{m}{2} + \frac{n}{2} - 1) + T(m/2, n/2)$$

解得 $T(m,n) = O(m+n)$。 □

3 动态规划

3.1 知识点

基本格式

- 列递推方程
- 确定初始/边界条件
- 标记函数：考试如果要求求出最优解，必须写出标记函数
- 时间复杂度：通常可以先确定空间复杂度（状态的总数），然后看每一步递推的时间复杂度

注. 如果题目没要求写伪代码，就不用写。

3.2 例题

例 3.1. 设正整数序列 $\{a_i\}_{i=1}^n$ 各项互不相等，求 $\{a_i\}_{i=1}^n$ 的最长单调递增子序列的长度。

解. 用 $f(k)$ 表示前 k 项中最长单调递增子序列的长度。动态规划时间复杂度为 $O(n^2)$。下面讨论时间复杂度更低的算法。

用 $g_i(j)$ 表示前 i 项中，长度为 j 的单调递增子序列末尾元素的最小可能值。若不存在长为 j 的单调递增子序列，则定义 $g_i(j) = \infty$。不难看出 $g_i(j)$ 关于 j 是单调的，即 $g_i(1) \le g_i(2) \le \cdots \le g_i(n)$。我们考虑如何维护 $\{g_i(j)\}$。初始时，$g_0(j) = \infty, j = 1, 2, \ldots, n$。而后枚举 $i = 1, 2, \ldots, n$，每次对 a_i 执行如下操作

- **查找**：找到 $j^* \in [n]$，满足 $g_{i-1}(j^* - 1) < a_i < g_{i-1}(j^*)$。
- **更新**：令 $g_i(j^*) \leftarrow a_i$; $g_i(j) \leftarrow g_{i-1}(j), \forall j \neq j^*$。

最后，求出 $\ell = \max\{j \colon g_n(j) < \infty\}$，即为 $\{a_n\}$ 的最长单调递增子序列长度。

正确性几乎是显然的。事实上，如果 $g_i(j) < g_{i-1}(j)$，则必有 $g_{i-1}(j-1) < g_i(j) = a_i < g_{i-1}(j)$。因此只有唯一一个 $g_i(j)$ 的值需要更新。

时间复杂度：对每个 a_i，查找 j^* 可以使用二分查找，复杂度为 $O(\log n)$。更新 $\{g_i(j)\}$ 需要 $O(n)$ 时间。但是注意到发生实质变化的只有 $g_i(j^*)$，其他均为复制。因此，如果不储存 i，则可以免去从 $\{g_{i-1}(j)\}$ 到 $\{g_i(j)\}$ 的 $n-1$ 次冗余复制，更新 $\{g(j)\}$ 的复杂度降为 $O(1)$。综上，时间复杂度为 $O(n\log n)$。 □

例 3.2. 五一期间，小江同学计划开车从 A 地出发，到相距为 L 公里的 B 地探险。在 A 地到 B 地的途中有 n 个加油站，它们分别分布在距离 A 地 $d_i, i = 1, 2, \ldots, n$ 公里的位置，每个加油站可以提供 V_i 升的汽油。一开始小江同学的汽车上有 S 升汽油，汽车每行驶一公里需要耗油一升，假设汽车油箱容量无穷。

小江同学希望在从 A 地和 B 地的途中用汽油换购一些土特产。在 A 地到 B 地的途中有 m 个售卖土特产的商店，它们分别分布在距离 A 地 $e_j, j = 1, 2, \ldots, m$ 公里的位置。在每一个商店，小江同学可以使用 p_j 升的汽油换购一些价值为 c_j 的商品，或者选择什么都不换购。在小江同学能够顺利到达 B 地的前提下，请问她最多能换购多少价值的土特产？

解. 用 $\varepsilon_j \in \{0, 1\}$ 表示小江是否在商店 j 换购土特产。则目标函数为

$$\text{maximize} \sum_{j=1}^{m} \varepsilon_j c_j.$$

约束条件是：行驶途中任意位置油量 ≥ 0。事实上，只需约束到达每个加油站的瞬间，剩余油量 ≥ 0，即

$$S + \sum_{\ell=1}^{i-1} V_\ell - d_i - \sum_{j\colon e_j < d_i} \varepsilon_j p_j \geq 0, \qquad \forall i \in [n+1].$$

这里为了统一记号，我们补充定义 B 地为第 $n+1$ 个加油站，且 $d_{n+1} := L$。

将以上两式稍加整理，我们得到此问题的数学描述：

$$\begin{aligned} &\underset{\varepsilon_1,\ldots,\varepsilon_m \in \{0,1\}}{\text{maximize}} \sum_{j=1}^{m} \varepsilon_j c_j, \\ &\text{s.t.} \qquad \sum_{k=1}^{j} \varepsilon_k p_k \leq R_j, \qquad \forall j \in [m]. \end{aligned}$$

其中

$$R_j := S + \sum_{\ell=1}^{i(j)-1} V_\ell - d_{i(j)}, \qquad i(j) := \underset{i}{\text{argmin}}\{d_i \colon d_i > e_j\}.$$

把 c_j 视为价值，p_j 视为体积，则容易看出此问题等价于背包问题。递推方程为

$$F(j,R)=\begin{cases} f(j-1,R), & R<p_j;\\ \max\{f(j-1,R),c_j+f(j-1,R-p_j)\}, & p_j\le R\le R_j;\\ f(j,R_j), & R>R_j. \end{cases}$$

$$F(0,R)=0.$$

时间复杂度 $O(mR_m)$。 □

例 3.3. 某大学有 n 个职员，编号为 $1,2,\ldots,n$。他们之间有从属关系，也就是说他们的关系就像一棵以校长为根的树，父结点就是子结点的直接上司。现在有个周年庆宴会，宴会每邀请来一个职员都会增加一定的快乐指数 r_i，但是呢，如果某个职员的直接上司来参加舞会了，那么这个职员就无论如何也不肯来参加舞会了。

所以，请你计算，邀请哪些职员可以使快乐指数最大，求最大的快乐指数。

解. 对于职员 i，用 $f(i)$ 表示以 i 为根节点的子树所能获得的最大快乐指数。用 $f_0(i)$ 表示 i 不参加舞会的条件下，以 i 为根节点的子树所能获得的最大快乐指数。用 $f_1(i)$ 表示 i 参加舞会的条件下，以 i 为根节点的子树所能获得的最大快乐指数。则有

$$\begin{aligned} f(i)&=\max\{f_0(i),f_1(i)\};\\ f_0(i)&=\sum_{j\in\mathrm{Child}(i)} f(j);\\ f_1(i)&=r_i+\sum_{j\in\mathrm{Child}(i)} f_0(j). \end{aligned}$$

标记函数

$$t(i)=\begin{cases} 0, & f_0(i)\ge f_1(i);\\ 1, & f_0(i)<f_1(i). \end{cases}$$

要确定邀请哪些职员，只需从树根开始，自顶向下地根据 i 的父节点和标记函数 $t(i)$ 共同决定是否邀请职员 i。

时间复杂度 $O(n)$。 □

4 贪心

4.1 知识点

几个经典的贪心算法 Huffman, Prim, Kruskal, Dijkstra 正确性证明、时间复杂度

贪心算法的设计与分析 对步骤/规模归纳，交换论证，模仿经典算法的设计/分析思路

注. 贪心算法必须证明正确性。

4.2 例题

例 4.1. 某公司计划面试 $2n$ 人，每个人面试一次。面试的地点有 A 和 B 两个城市可选。假设第 i 人飞往 A 市的费用为 a_i，飞往 B 市的费用为 b_i。请设计一个面试方案，使得每个城市都有 n 个人面试，并且总飞行费用最低。

解. 问题归结为：求 $[2n]$ 的一个划分 $[2n] = A \cup B$，满足 $|A| = |B| = n$，且 $\sum_{i\in A} a_i + \sum_{j\in B} b_j$ 最小。上式可以等价转化为：

$$\sum_{i=1}^{2n} a_i - \sum_{j\in B} a_j + \sum_{j\in B} b_j = \sum_{i=1}^{2n} a_i + \sum_{j\in B} (b_j - a_j).$$

因此，只需选择最小的 n 个 $b_j - a_j$ 对应的下标，记为 $j_1, j_2, \ldots, j_n$；并令 $B = \{j_1, j_2, \ldots, j_n\}$，$A = [2n] \setminus B$。时间复杂度为 $O(n)$。 □

例 4.2. 五一期间，小江同学计划开车从 A 地出发，到相距为 L 公里的 B 地探险。在 A 地到 B 地的途中有 n 个加油站，它们分别分布在距离 A 地 $d_i, i = 1, 2, \ldots, n$ 公里的位置，每个加油站可以提供 V_i 升的汽油。一开始小江同学的汽车上有 S 升汽油，汽车每行驶一公里需要耗油一升，假设汽车油箱容量无穷。

请问小江同学在途中至少要加几次油？她要在哪些加油站加油？

解. 用 $\varepsilon_i \in \{0, 1\}$ 表示是否在第 i 个加油站处加油，则目标函数为

$$\underset{\varepsilon_1,\ldots,\varepsilon_n\in\{0,1\}}{\text{minimize}} \sum_{i=1}^{n} \varepsilon_i.$$

约束条件为：到达每个加油站的瞬间，剩余油量 ≥ 0，即

$$S + \sum_{j=1}^{i-1} \varepsilon_j V_j - d_i \geq 0, \qquad \forall i \in [n+1].$$

这里为了统一记号，我们补充定义 B 地为第 $n+1$ 个加油站，且 $d_{n+1} := L$。

考虑以下贪心算法：初始时令 $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n \leftarrow 0$。依次枚举每个 $i \in [n+1]$，若当前 $\sum_{j=1}^{i-1} \varepsilon_j V_j < d_i - S$，则考察那些“尚未加油”的加油站 $\{1 \leq j \leq i-1 \colon \varepsilon_j = 0\}$。按照 V_j 从大到小的顺序依次置 $\varepsilon_j \leftarrow 1$，直至 $\sum_{j=1}^{i-1} \varepsilon_j V_j \geq d_i - S$（油量充足）。

正确性：对算法的步骤归纳。对于 $i \in [n+1]$，用 F_i 表示考虑了加油站 i 处的约束条件之后，所有满足 $\varepsilon_j = 1$ 的加油站 j 组成的集合。特别地，有 $\emptyset = F_1 \subseteq F_2 \subseteq \cdots \subseteq F_{n+1}$，

且 $F_i \subseteq \{1, 2, \ldots, i-1\}$。我们将对 i 归纳证明：存在一个最优解（加油站的集合） F^*，使得 $F_i \subseteq F^*$。

当 $i = 1$ 时，结论是平凡的。假设结论对 $i-1$ 成立，即存在最优解 F^*，使得 $F_{i-1} \subseteq F^*$。若 $F_i = F_{i-1}$，则结论对 i 也成立。否则 $F_i \supset F_{i-1}$，表明 F_{i-1} 中的加油站无法满足加油站 i 处的约束条件。记 $F_i^* := F^* \cap [i-1]$，则 $F_i^* \setminus F_{i-1} \neq \emptyset$。用 $F_i \setminus F_{i-1}$ 中的元素替换掉 $F_i^* \setminus F_{i-1}$ 中 V_j 最大的那 $|F_i \setminus F_{i-1}|$ 个元素，得到的解 F' 仍然满足所有约束条件（回忆 $F_i \setminus F_{i-1}$ 是如何构造的），且 $|F'| = |F^*|$。从而存在一个最优解 F' 包含 F_i，结论对 i 也成立。

时间复杂度：我们的贪心算法中涉及到求 $\max\{V_j \colon 1 \leq j \leq i-1, \varepsilon_j = 0\}$。这可以通过最大堆实现。具体地，当考虑加油站 i 处的约束条件时，堆中存放着 $\{(V_j, j) \colon 1 \leq j \leq i-1, \varepsilon_j = 0\}$，且按照 V_j 排序。因此每次更新 ε_j 对应着从堆顶弹出 (V_j, j)，并置 $\varepsilon_j \leftarrow 1$。当加油站 i 处的约束条件满足后，向堆中插入 (V_i, i)。这样每个 i 总计入堆一次，出堆一次。时间复杂度 $O(n \log n)$。 □

例 4.3. 每座公路桥梁都有其最大限重，超过限重会导致安全事故发生。已知每座桥梁的最大限重值，以及相应的公路网络，问能否将一件超重设备从 A 地通过公路网络送到 B 地，使得经过的路线上每座桥梁的最大限重值均不低于超重设备和运输车辆合在一起的总重量。请设计算法，给出一条“超重设备和运输车辆合在一起的总重量”最大的路线。

解. 将公路网络视为无向带权图 $G = (V, E, w)$。给定顶点 $A, B \in V$，用 $\mathcal{P}_{A \to B}$ 表示 A 到 B 的所有路径组成的集合。只需求

$$\operatorname*{argmax}_{P \in \mathcal{P}_{A \to B}} \min_{e \in P} w(e).$$

模仿 Kruskal 算法的思路。维护 G 的一个子图 $H \subseteq G$。初始时令 $E(H) = \emptyset$。随后，按照权值从大到小的顺序依次将 G 中的边 e 加入 H；直至 A, B 在 H 中连通。此时，在 H 中求出任意一条 A, B 之间的路径即可。

假设最后求出的路径为 $P^* \in \mathcal{P}_{A \to B}$，并记加入 H 的最后一条边为 e^*。则必有 $e^* \in P^*$，且 $\min_{e \in P^*} w(e) = w(e^*)$。以下证明：$G$ 中任意路径 $P \in \mathcal{P}_{A \to B}$，均满足 $\min_{e \in P} w(e) \leq w(e^*)$。

如若不然，则存在路径 $P \in \mathcal{P}_{A \to B}$，使得 $\min_{e \in P} w(e) > w(e^*)$。根据算法向 H 加边的顺序知，必有 $P \subseteq H$。这表明在向 H 添加边 e^* 之前，顶点 A, B 在 H 中已经连通。故 $e^* \notin H$，矛盾。

算法的复杂度与 Kruskal 算法相同，为 $O(m \log m)$。 □

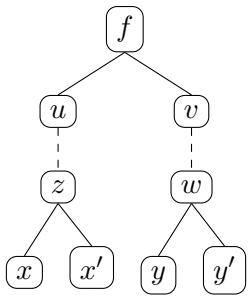
例 4.4. 在黑板上写了 N 个正整数组成的一个数列，进行如下操作：每一次擦去其中的两个数 a 和 b，然后在数列中加入一个数 $ab+1$，如此下去直至黑板上剩下一个数。请设计一个算法求黑板上剩余数的最大可能值，并分析算法的正确性和时间复杂度。

解. 类似于 Huffman 算法，每次取当前数列中最小的两个数操作。时间复杂度为 $O(n\log n)$。下证正确性。

操作过程形成一棵有 n 个叶节点的二叉树 T。对于每个内部节点 c，假设它的两个子节点分别为 a,b，则满足关系 $c=ab+1$。根节点的值即为黑板上剩余的最后一个数。

假设初始数列中最小的两个数为 x,y。我们说明，当根节点取最大值时，x,y 在 T 上有相同的父节点。若不然，设 x,y 的父节点分别为 $z,w\,(z\neq w)$，并假设 x,y 的兄弟结点分别为 x',y'，满足 $z=xx'+1, w=yy'+1$。假设 z,w 的最近共同祖先为 f，设 f 的两个子节点为 u,v，则 u,v 分别为 z,w 的祖先。注意到 u 可以表示为 z 的线性函数，即 $u=Az+B$，同理可设 $v=Cw+D$。则有

$$\begin{aligned} f=f(x,x',y,y')&=(Az+B)(Cw+D)+1\\ &=(Axx'+A+B)(Cyy'+C+D)+1\\ &=ACxx'yy'+(AC+AD)xx'+(AC+BC)yy'+(A+B)(C+D)+1 \end{aligned}$$



我们证明：在 T 上交换节点 x,y' （或 x',y）之后，节点 f 的值不减，进而可知根节点的值也不减。注意到

$$\begin{aligned} f(x,y,x',y')-f(x,x',y,y')&=(x'-y)((AC+BC)y'-(AC+AD)x) &&\text{(交换 } x',y)\\ f(y',x',y,x)-f(x,x',y,y')&=(y'-x)((AC+AD)x'-(AC+BC)y) &&\text{(交换 } x,y') \end{aligned}$$

当 $AC+BC\geq AC+AD$ 时，有 $f(x,y,x',y')\geq f(x,x',y,y')$；当 $AC+BC<AC+AD$ 时，有 $f(y',x',y,x)\geq f(x,x',y,y')$。总之 $\max\{f(y',x',y,x),f(x,y,x',y')\}\geq f(x,x',y,y')$。故存在一个最优解，在其对应的二叉树 T^* 上，x,y 有相同的父节点。

最后，仿照 Huffman 算法的正确性证明，对规模归纳即可证明算法正确性。 □

5 回溯与分支限界

5.1 知识点

描述清楚解向量，搜索树，约束条件，代价函数，界函数，用节点个数 × 单个节点处理时间估算时间复杂度

6 线性规划

6.1 知识点

问题的线性规划模型，线性规划标准形（怎样转化），单纯形法，对偶线性规划（怎样转化），整数线性规划

注. 不需要掌握教材上单纯形法部分的证明。

6.2 例题

例 6.1. 有一个长度为 n 的非负整数序列 $a_1, a_2, \ldots, a_n$。每一步可以选择一个区间 $[\ell, r]$，将此区间内的所有数减 1。试求将整个序列变成 0 的最小步数。

写出问题的线性规划模型及其对偶。

解. 对区间 $I \subseteq [1, n]$，用 $x_I \in \mathbb{N}$ 表示对区间 I 操作的次数。线性规划模型为

$$\begin{aligned} &\text{minimize} \sum_{I \subseteq [1,n]} x_I, \\ \text{s.t.} \quad &\sum_{I \subseteq [1,n]} \mathbb{I}(i \in I) \cdot x_I = a_i, \quad i = 1, 2, \ldots, n \\ &x_I \in \mathbb{N}, \quad I \subseteq [1, n]. \end{aligned}$$

其中使用记号

$$\mathbb{I}(i \in I) = \begin{cases} 1, & i \in I; \\ 0, & i \notin I. \end{cases}$$

对偶线性规划为

$$\begin{aligned} &\text{maximize} \sum_{i=1}^{n} a_i (y_i - z_i) \\ \text{s.t.} \quad &\sum_{i=1}^{n} \mathbb{I}(i \in I) \cdot (y_i - z_i) \leq 1, \quad I \subseteq [1, n], \\ &y_i, z_i \in \mathbb{N}, \quad i = 1, 2, \ldots, n \end{aligned}$$

□

注. 这是一个整数线性规划问题，但仍然存在线性时间的算法。

设向量 $\alpha := (a_1, a_2, \ldots, a_n)$。补充定义 $a_0 = a_{n+1} = 0$。对 $1 \leq i \leq n+1$，定义 $b_i := a_i - a_{i-1}$. 则易知向量 $\beta = (b_1, b_2, \ldots, b_{n+1})$ 满足以下条件

- 任意 $k \in [n]$，有 $\sum_{i=1}^{k} b_i = a_k \geq 0$。
- $\sum_{i=1}^{n+1} b_i = 0$。

另一方面，每次对向量 α 区间 $[\ell, r]$ 的操作等价于对向量 β 的第 ℓ 和第 $r+1$ 个坐标的操作：$b_\ell \leftarrow b_\ell - 1, b_{r+1} \leftarrow b_{r+1} + 1$。且 $\alpha = \mathbf{0}$ 当且仅当 $\beta = \mathbf{0}$。容易证明，将向量 β 变为 $\mathbf{0}$ 所需的操作次数为

$$\frac{1}{2}\|\beta\|_1 = \frac{1}{2}\sum_{i=1}^{n+1}|b_i| = \sum_{i:\, b_i>0} b_i.$$

计算上式的时间复杂度为 $O(n)$。

例如，考虑向量 $\alpha = (3,2,1,2)$，对应向量 $\beta = (3,-1,-1,1,-2)$。将向量 β 变为 $\mathbf{0}$ 的操作为

$$(3,-1,-1,1,-2) \to (2,0,-1,1,-2) \to (1,0,0,1,-2) \to (0,0,0,1,-1) \to (0,0,0,0,0).$$

对应于向量 α 的操作为

$$(3,2,1,2) \to (2,2,1,2) \to (1,1,1,2) \to (0,0,0,1) \to (0,0,0,0)$$

7 平摊分析

7.1 知识点

平摊分析的三种方法 聚集分析、记账法、势能法

- 常见的势函数定义
- 势函数 ≥ 0
- 每步开销 $c_i \lesssim \Phi_{i-1} - \Phi_i + O(1)$

7.2 例题

例 7.1. 队列是一个先进先出线性表，支持 $O(1)$ 时间复杂度的进队、出队操作。但是在队列中查找当前的最小元素需要 $O(n)$ 的时间复杂度，即遍历当前的整个队列。

下面设计一个辅助的单调队列完成求最小值操作。

在原有的队列基础上增加一个单调队列 $A_1, A_2, \ldots, A_m$，保证 $A_1 < A_2 < \cdots < A_m$。下面讨论三种操作时辅助队列的相应操作;

(a) 元素入队：设入队元素为 x，原有队列按照队列基本操作入队。比较辅助队列的最后一个元素 A_m 与 x 的大小关系，若 $x < A_m$，则 A_m 从辅助队列中删除；继续比较 $A_{m-1}\ldots$，直至当前辅助队列不存在大于 x 的队尾元素为止。将 x 加入辅助队列末尾。（原理：当前元素比从辅助队列中删除的元素小，且比它们后出队，因此在 x 出队以前，这些元素都不会成为最小值元素，可以直接删除）

(b) 元素出队：原有队列按照队列基本操作出队。若当前出队元素是辅助队列队首元素，则辅助队列队首元素出队。

(c) 求最小值：当前辅助队列队首元素即为当前队列的最小值。

请证明上述队列操作的均摊时间复杂度为 $O(1)$。

解. 记辅助队列为 Q'。定义势函数 $\Phi(Q') := 2|Q'|$ 为当前辅助队列长度的 2 倍。

元素入队：假设元素 x 淘汰了 Q' 中的 k 个元素，则 x 共进行 $k+1$ 次比较，并进行 2 次插入，且 Q' 上发生 k 次删除。故 $c_i = 2k+3$。另一方面，$\Phi(Q'_i) - \Phi(Q'_{i-1}) = 2(|Q'_i| - |Q'_{i-1}|) = 2(1-k)$。

元素出队：元素 x 进行至多 2 次出队和 1 次比较，故 $c_i \le 3$。另一方面，$\Phi(Q'_i) - \Phi(Q'_{i-1}) \le 0$。

求最小值：$c_i = 1$。另一方面 $\Phi(Q'_i) - \Phi(Q'_{i-1}) = 0$。

综上，

$$c_i + \Phi(Q'_i) - \Phi(Q'_{i-1}) \le \begin{cases} 5, & \text{元素入队} \\ 3, & \text{元素出队} \\ 1, & \text{求最小值} \end{cases}$$

$$\Longrightarrow \sum_{i=1}^{n} c_i \le 5n + \Phi(Q'_0) \le O(n).$$

故均摊复杂度为 $O(1)$。 □