清华大学软件学院

算法分析与设计基础

2020 年春季学期

作业 **6**

吴佳龙　班级：软件 83　学号：2018013418　　April 13, 2020

6.1. (CLRS Exercises 16.3-7)

三进制的哈夫曼编码算法为:

1. 若待编码字符集 C 的大小 n 为偶数，则往字符集内加入一个出现频率 $freq = 0$ 的元素构成 C'，此时 n' 为奇数。

2. 类似二进制的情形，建立以 $freq$ 为关键字的最小优先队列 Q，每次取出频率最低的 3 个对象将其合并，并将设置新对象的频率为三者之后重新插入队列中。

3. 不断重复 $(n' - 1)/2$ 次合并，最终创建为三叉树，对应一种三进制编码。

下证明其正确性。

Lemma 1. （类似 *CLRS Lemma 16.2*）若 n 为奇数，令 x, y, z 是 C 中频率最低的三个字符。那么存在一个 C 的最优前缀编码，x, y, z 的编码长度相同，且只有最后一个三进制位不同。

证明. 任取一颗 C 的最优三叉哈夫曼树 T。n 为奇数，则 T 的每个内部节点都有 3 个儿子。令 T 上深度最深的三个互为兄弟的叶子结点为 a, b, c，将 x, y, z 与他们分别互换，由于

$$depth(x) \le depth(a), depth(y) \le depth(b), depth(z) \le depth(c)$$

$$freq(x) \le freq(a), freq(y) \le freq(b), freq(z) \le freq(c)$$

则类似 CLRS Lemma 16.2 的证明，换完之后的树 $\hat{T}$ 有 $B(\hat{T}) \le B(T)$。由于 T 是最优的，则 $\hat{T}$ 也是最优的。在 $\hat{T}$ 上，x, y, z 的编码长度相同，且只有最后一个三进制位不同。 □

Lemma 2. （类似 *CLRS Lemma 16.3*）若 n 为奇数，从 C 中去掉 x, y, z，加入一个频率为三者之和的新字符 w 构成 $\hat{C}$，那么可以将 $\hat{C}$ 的一个最优三叉哈夫曼树 $\hat{T}$ 中的 w 结点替换为以 x, y, z 为儿子的内部结点，成为 C 的一个最优三叉哈夫曼树 T。

证明. 类似 CLRS Lemma 16.3 的证明，

$$B(T) = B(\hat{T}) + freq(x) + freq(y) + freq(z)$$

恒成立，因此 $B(T)$ 与 $B(\hat{T})$ 同时取到最优。 □

Lemma 3. 若 n 为偶数，往字符集 C 内加入一个出现频率 $freq = 0$ 的元素构成 C'，得到的最优哈夫曼树 T' 去掉加入的元素，就是 C 的一颗最优哈夫曼树 T。

证明. 类似 Lemma 2 的证明，

$$B(T) = B(\hat{T}) - 0 = B(\hat{T})$$

恒成立，$B(T)$ 和 $B(\hat{T})$ 同时取到最优。 □

Theorem 1. 以上三进制的哈夫曼编码算法能得到最优的三进制前缀码。

证明. 若 n 为奇数，由以上 Lemma 1 和 Lemma 2 ，可得该算法的正确性。

若 n 为偶数，结合 Lemma 3 和 n 为奇数的情形，也可得算法的正确性。 □

该算法还可推广到 k 进制最优前缀编码，只要往字符集里加入频率为 0 的元素，直到 $n \bmod (k-1) = 1$ 即可。

6.2. (CLRS Exercises 16.4-1)

证明.
- （遗传性）$\forall E \in \mathcal{I}_k, |E| \le k, \forall A \subset E, |A| \le |E| \le k$, 则 $A \in \mathcal{I}_k$
- （交换性）$\forall A, B \in \mathcal{I}_k, |A| < |B|$, 则 $|A| \le k-1$ 且 $B - A$ 非空，任取 $x \in B - A$，$|A \bigcup \{x\}| = |A| + 1 \le k$，则 $A \bigcup \{x\} \in \mathcal{I}_k$

因此 $(S, \mathcal{I}_k)$ 为拟阵。 □

6.3. (CLRS Problems 16-2)

(a) 算法为：将任务按照 p_i 从小到大排序，然后按照这个顺序依次完成。
这是最优的

证明. 这是因为若存在一种最优的执行顺序 $\hat{a}_1, \hat{a}_2, \cdots, \hat{a}_n$，其中 $\exists i < j$ 使得 $\hat{p}_i > \hat{p}_j$，那么交换 $\hat{a}_i, \hat{a}_j$，交换完之后 $\hat{c}_k, k = i, i+1, \cdots, j-1$ 将不大于原来的值。因此总的平均完成时间将不大于原来的值，由于原来的方案的最优性，交换之后的方案也是最优的。反复交换，直到所有任务按照 p 有序，仍然是最优方案。 □

算法的运行时间为 $O(n \lg n)$，这是排序所需的时间。

(b) 算法描述为：维护一个最小优先队列 Q，以队列中的任务的剩余完成时间为关键字。对于时刻 $t = 0, 1, 2, \cdots$

1. 若此时刻有任务刚好释放，将其放入 Q 中，剩余完成时间就是该任务的所需完成时间。

2. 若上一时刻有正在执行的任务，若该时刻该任务恰好完成，将该时刻计入最终答案；否则，将该任务拿出，更新剩余完成时间，放入 Q。

3. 从 Q 中取出剩余完成时间最少的任务，在该时刻执行。

该算法是最优的。

证明. 首先，任意时刻 t 处理完 1,2 两步之后，目前的最优答案为当前已记录的完成时间之和加上当前 Q 中所有任务的最优完成时间之和，满足最优子结构性质。

而且，若存在某一最优方案 S，在某一时刻 t_0，未按照 3 步中的步骤选取这一时刻执行的任务，令该时刻被选择执行的任务为 a_i，实际该时刻 Q 中剩余完成时间最少的任务为 a_j，令 S 中 a_j 完成时刻为 t_1。重新调整时刻区间 $[t_0, t_1]$ 中执行任务 a_i 或 a_j 的时刻 $\{\hat{t}_1, \hat{t}_2, \cdots, \hat{t}_k\}$，将这些时刻执行的任务调整为：$a_j$ 全部安排在 a_i 的前面，也就是说 $\{\hat{t}_1, \hat{t}_2, \cdots, \hat{t}_k\}$ 时刻执行的任务形如 $\{a_j, a_j, \cdots, a_j, a_i, a_i, \cdots, a_i\}$ 。这样调整过后 S' 完成时间 $c'_j \le c_i$，$c'_i \le c_j$，因此 S' 的平均完成时间不差于 S，而 S 为最优方案，因而 S' （满足算法步骤 3 中的贪心策略）也为最优方案。 □

算法的运行时间为 $O(n \lg n)$，这是按照释放时间排序以及维护优先队列的时间。