算法导论　第十一周作业　11 月 28 日　周六

PB18151866 龚小航

7.1 假定我们对一个数据结构执行一个由 n 个操作组成的操作序列，当 i 严格为 2 的幂时，第 i 个操作的代价为 i，否则代价为 1。使用聚合分析确定每个操作的摊还代价。

解：写出其第 i 个操作的代价，用 c_i 表示：

$$c_i = \begin{cases} i\,, & i = 2^k \\ 1\,, & i \neq 2^k \end{cases} \quad k \in \mathbb{Z}^*$$

需要求出对所有 n，一个 n 个操作的系列在最坏情况下所花费的总时间 $T(n)$，就是求出 n 步操作所用的时间开销的上界。此处令 $i = 2^k$ 时，$c_i = i + 1$：

$$\text{实际总开销} = \sum_{i=1}^{n} c_i \leq \sum_{k=0}^{\lfloor \lg n \rfloor} \frac{n}{2^k} + n = n + n\frac{1-\left(\frac{1}{2}\right)^{\lfloor \lg n \rfloor}}{1-\frac{1}{2}} < n + 2n - 1 < 3n$$

即使用聚合分析得出每个操作的摊还代价为：

$$\frac{O(3n)}{n} = O(1)$$

7.2．用核算法重做第一题。

解：由上一题得到的启发，可以为这个操作赋予如下的摊还代价：

$$\widehat{c_i} = 3$$

做第 i 个操作时可以这样：当 i 不是 2 的幂次时，实际消耗代价为 1，摊还代价为 3，因此可以把代价 2 存为信用；当 i 是 2 的幂次时，实际消耗代价为 i，摊还代价为 3，令 $2^k = i$，显然从 2^{k-1} 个操作到第 $i = 2^k$ 个操作之间的操作存储的信用为 $2 \cdot (2^k - 2^{k-1}) = 2^k = i$，恰好支付第 i 次操作的费用。因此信用额度始终非负。

因此使用核算法可知：

$$\sum_{i=1}^{n} \widehat{c_i} = 3n,\quad \sum_{i=1}^{n} c_i \leq \sum_{i=1}^{n} \widehat{c_i} = 3n,\quad \Rightarrow \quad \sum_{i=1}^{n} c_i = O(n)\text{，即每个操作的摊还代价为 O(1)}$$

7.3　使用势能法重做第一题。

解：由第一题得到的启发，可设第 i 个操作时，$i = 2^{\lfloor \lg i \rfloor} + j$，即 j 表示当前操作系列号离小于等于它的最大 2 的幂次系列号的距离。可令势能函数 $\phi(D_i) = 2j$。显然 $\phi(D_i) \geq \phi(D_0) = 0$

① $j = 0$ 时：此时 $c_i = i$，$i = 2^k$，计算摊还代价：

$$\widehat{c_i} = c_i + \phi(D_i) - \phi(D_{i-1}) = i + 0 - 2 \cdot \left(2^{\lg i} - 2^{\lg i - 1} - 1\right) = 2$$

② $j \neq 0$ 时：此时 $c_i = 1$，计算摊还代价：

$$\widehat{c_i} = c_i + \phi(D_i) - \phi(D_{i-1}) = 1 + 2j - 2(j-1) = 3$$

综上，每个操作的摊还代价都是 $O(1)$，因此 n 个操作的总摊还代价为 $O(n)$

因此最坏情况下每个操作的摊还代价为 $O(1)$

7.4 我们可以将一维离散傅里叶变换推广到 d 维上。这时输入是一个 d 维的数组 $A = (a_{j_1,j_2,\cdots,j_d})$，维数分别为 $n_1,n_2,\ldots,n_d$，其中 $n_1n_2\ldots n_d = n$。定义 d 维离散傅里叶变换如下：

$$y_{k_1,k_2,\cdots,k_d} = \sum_{j_1=0}^{n_1-1}\sum_{j_2=0}^{n_2-1}\cdots\sum_{j_d=0}^{n_d-1} a_{j_1,j_2,\cdots,j_d}\omega_{n_1}^{j_1k_1}\omega_{n_2}^{j_2k_2}\cdots\omega_{n_d}^{j_dk_d}$$

其中 $0 \le k_1 < n_1$，$0 \le k_2 < n_2$，…… $,0 \le k_d < n_d$

($\boldsymbol{a}$). 证明：我们可以依次在每个维度上计算一维的 DFT 来计算一个 d 维的 DFT。也就是说，首先沿着第 1 维计算 n/n_1 个独立的一维 DFT。然后，把沿着第 1 维的 DFT 结果作为输入，我们计算沿着第 2 维的 n/n_2 个独立的一维 DFT。利用这个结果作为输入，我们计算沿着第三维的 n/n_3 个独立的一维 DFT，如此下去，直到第 d 维。

($\boldsymbol{b}$). 证明：维度的次序并无影响，于是可以通过在 d 个维度的任意顺序中计算一维 DFT 来计算一个 d 维的 DFT。

($\boldsymbol{c}$). 证明：如果采用计算快速傅里叶变换计算每个一维的 DFT，那么计算一个 d 维的 DFT 的总时间是 $O(n\lg n)$，与 d 无关。

解:

($\boldsymbol{a}$)　利用 d 维向量定义把向量 y 由里及外展开，从第 n_d 维到第 n_1 维依次展开的过程是从子项仅有 n_d 项到子项有 $n_2n_3\ldots n_d$ 项，项数逐渐增加的过程，每次展开一个维度就要保存这个维度的向量 y，然后以 y 作为下一维度基础值。若展开定义式是由外及里，那么首先展开第 n_1 维，n_1 维每项是由 $n_2n_3\ldots n_d$ 个子项组成，而由于最初不知道这些子项数据，所以不可以由外至里展开。所以从第 1 维开始计算就是第 n_d 维。计算 d 维 FFT 的过程就是一个展开多重求$\sum$式过程。每展开一个求$\sum$式就是求一维 DFT 的过程。

($\boldsymbol{b}$)　将 d 维 DFT 展开后，相当于求这个多重$\sum$式通项元素的全排列，$a_{j_1,j_2,\cdots,j_d}\omega_{n_1}^{j_1k_1}\omega_{n_2}^{j_2k_2}\cdots\omega_{n_d}^{j_dk_d}$ 这个元素， ω_{n_1} 有 n_1 种数据，ω_{n_2} 有 n_2 种数据，ω_{n_d}有 n_d 种数据，所以此全排列有 $n_1n_2\ldots n_d = n$ 种数据，所以向量 y 一共有 n 项，其中的每一项 y_i 也是由 n 个子元素组成。而通项 y_i 中的 n 个通项子元素 $a_{j_1,j_2,\cdots,j_d}\omega_{n_1}^{j_1k_1}\omega_{n_2}^{j_2k_2}\cdots\omega_{n_d}^{j_dk_d}$ 是固定的，调整维度顺序仅仅是子元素出现顺序有变化，但是整体的数值是不变的不受维度排列顺序影响，所以通项 y_i 也是固定的，从而得证。

($\boldsymbol{c}$) 由第一问可知:

$$\begin{aligned}&(n/n_1)O(n_1\lg n_1) + (n/n_2)O(n_2\lg n_2)+\ldots.(n/n_d)O(n_d\lg n_d)\\ &= n\cdot O(\lg n_1) + n\cdot O(\lg n_2)+\ldots..+n\cdot O(\lg n_d) = n\cdot O(lgn_1n_2..n_d) = O(n\lg n)\end{aligned}$$