算法基础 HW6

PB18111697 王章瀚

2020 年 12 月 8 日

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.

可以考虑, n 次执行需要代价为

$$\begin{aligned} n - \lfloor\log_2 n\rfloor - 1 + \sum_{k=0}^{\lfloor\log_2 n\rfloor} 2^k &= n - \lfloor\log_2 n\rfloor - 1 + \frac{2^{\lfloor\log_2 n\rfloor} \times (2^{\lfloor\log_2 n\rfloor} + 1)}{2} \\ &= n - \lfloor\log_2 n\rfloor - 1 + \frac{n \times (n+1)}{2} \\ &= O(n^2) \end{aligned}$$

因此, 平均下来代价为:

$$O(n^2)/n = O(n)$$

2

用核算法重做第一题

则此时需要赋予费用, 可以这样赋予:

第 i 个操作	实际代价	摊还代价
i 不严格为 2 的幂	1	3
$i = 2^k$	i	2

其意为: 每次执行一个非严格为2的幂的操作时, 除了自己的代价 1 外, 需要额外付出一个代价. 这样后面每当操作严格为 2 的幂的操作时, 前面必然已经累积了相应的摊还代价.
具体而言, 假设第 2^k 个操作完成后, 信用余额为2(包含了该操作的摊还代价),
那么从第 2^k+1 到第 $2^{k+1}-1$共有$(2^{k+1}-1)-(2^k+1)+1 = 2^k-1$个实际代价为1的操作
其总共需要支付$2^{k+1}-2$的摊还代价, 加上第 2^{k+1} 个本身需要的2个代价单位, 总共就完成了对 2^{k+1} 个实际代价的支付
因此总有摊还代价为实际代价的上界.
这样一来, 总的代价就是$\underline{O(3n) = O(n)}$的.

3

用势能法重做第一题

定义第 i 次操作后势能为 $\Phi(D_i) = i - \lfloor i/2 \rfloor$, 那么必然有 $\Phi(D_n) \geq \Phi(D_0)$
此时若 i 不严格为 2 的幂, 则总代价为

$$\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) \leq 1 + 1 = 2$$

此时若 i 严格为 2 的幂, 则总代价为

$$\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) \leq i + (-i+1) = 1$$

从而有

$$\sum_{i=0}^{n} c_i = \sum_{i=0}^{n} \hat{c}_i - \Phi(D_n) + \Phi(D_0) \leq \sum_{i=0}^{n} \hat{c}_i \leq 2n$$

因此, 摊还代价为 $O(n)$

4

我们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 $n_1, n_2, \cdots, n_d$, 其中 $n_1 n_2 \cdots n_d = n$, 定义 d 维离散傅里叶变换如下:

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

其中 $0 \leq k_1 < n_1, 0 \leq k_2 < n_2, \cdots 0 \leq k_d < n_d$

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 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 3 维的 n/n_3 个独立的一维 DFT. 如此下去直到第 d 维.

考虑恒等变换:

$$\begin{aligned}
y_{k_1,k_2,\cdots,k_d} &= \sum_{j_d=0}^{n_d-1} \sum_{j_{d-1}=0}^{n_{d-1}-1} \cdots \sum_{j_1=0}^{n_1-1} a_{j_1,j_2,\cdots,j_d} \omega_{n_d}^{j_d k_d} \omega_{n_{d-1}}^{j_{d-1} k_{d-1}} \cdots \omega_{n_1}^{j_1 k_1} \\
&= \sum_{j_d=0}^{n_d-1} \sum_{j_{d-1}=0}^{n_{d-1}-1} \cdots \sum_{j_2=0}^{n_2-1} \omega_{n_d}^{j_d k_d} \omega_{n_{d-1}}^{j_{d-1} k_{d-1}} \cdots \omega_{n_2}^{j_2 k_2} \sum_{j_1=0}^{n_1-1} a_{j_1,j_2,\cdots,j_d} \omega_{n_1}^{j_1 k_1} \\
&= \cdots \\
&= \sum_{j_d=0}^{n_d-1} \omega_{n_d}^{j_d k_d} \sum_{j_{d-1}=0}^{n_{d-1}-1} \omega_{n_{d-1}}^{j_{d-1} k_{d-1}} \cdots \sum_{j_2=0}^{n_2-1} \omega_{n_2}^{j_2 k_2} \sum_{j_1=0}^{n_1-1} a_{j_1,j_2,\cdots,j_d} \omega_{n_1}^{j_1 k_1}
\end{aligned}$$

因此按上式来看, 我们确实可以按照题述方法来依次计算各维而得到最终 d 维离散傅里叶变换的结果.

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

按照原式, 由于求和号的上下界之间相互无依赖(如上题第一步), 因此可以互换顺序, 然后再按照第一题的拆解变换方法就能够得到不同顺序的计算方法, 因此维度的次序并无影响.

c

证明: 如果采用计算快速傅里叶变换计算每一个一维的 DFT, 那么计算一个 d 维的 DFT 的总时间是 $O(n\lg n)$, 与 d 无关.

每一个维度都是

$$\Theta(n_i\log n_i)$$

故总的为

$$\Theta(\sum_{i=0}^{d} n_i\log n_i)$$

其中有

$$\sum_{i=1}^{d} n_i\log n_i \le n\log n_i \le n\log\prod_{i=1}^{d} = n\log n$$

因此总时间为$O(n\log n)$, 与 d 无关