Homework 6

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.（1）使用聚合分析确定每个操作的摊还代价。（2）使用核算法确定每个操作的摊还代价。（3）使用势能法确定每个操作的摊还代价。

2. V.Pan 发现一种方法，可以用 132464 次乘法操作完成 68×68 的矩阵相乘，发现另一种方法，可以用 143664 次乘法操作完成 70×70 的矩阵相乘，还发现一种方法，可以用 155424 次乘法操作完成 72×72 的矩阵乘法。当用于矩阵乘法的分治算法时，上述哪种方法会得到最佳的渐近运行时间？与 Strassen 算法相比，性能如何？

3. 我们可以将一维离散傅里叶变换 (DFT) 推广到 d 维上。这时输入是一个 d 维的数组 $A=(a_{j_1,j_2,\ldots,j_d})$，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：

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

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

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。

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。

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。

n个操作

(1) 最坏情况：$(1+2+\cdots+2^{\lfloor\log_2 n\rfloor})+1\times(n-\lfloor\log_2 n\rfloor-1) = O(n)$

每个操作：$\dfrac{O(n)}{n}=O(1)$

(2) 核算法：赋予以下代价：$\begin{cases} 2i & i=2^k \\ 0 & \end{cases}$

即当 $i=2^k$ 时 预支后面 $2^k\sim 2^{k+1}$ 之间的代价

故每个操作：$\dfrac{O(n)}{n}=O(1)$

(3) 设 $i=2^j+k$ $\quad\phi(D_i)=2^k$

$k=0$ 时 实际代价为 i $\quad\hat{c}_i=c_i+(\phi(D_i)-\phi(D_{i-1}))=2$

$k \neq 0$时 实际代价为1 $\hat{c}_i = c_i + (\Phi(D_i) - \Phi(D_{i-1})) = 3$

故每个操作平摊代价为 $O(1)$

2. ① $\log_{68} 132464 \approx 2.7951285$

② $\log_{70} 143664 \approx 2.7951620$ 故①最佳

③ $\log_{72} 155424 \approx 2.7951459$

strassen : $O(n^{\log_2 7}) \approx O(n^{2.81})$

故 ①②③ 皆比 strassen 算法性能好

3. $y_{j_1 \cdots j_d} = \sum_{j_1=0}^{n_1-1} \omega_{n_1}^{j_1 k_1} \left(\sum_{j_2=0}^{n_2-1} \cdots \sum_{j_d=0}^{n_d-1} a_{j_2 \cdots j_d} \omega_{n_2}^{j_2 k_2} \cdots \omega_{n_d}^{j_d k_d} \right) = \sum_{j_1=0}^{n_1-1} \omega_{n_1}^{j_1 k_1} y_{j_2 \cdots j_d}$

$\forall i = 1, \cdots, d$

$$= \sum_{j_i=0}^{n_i-1} \omega_{n_i}^{j_i k_i} y_{j_1 \cdots j_{i-1} j_{i+1} \cdots j_d} \qquad (*)$$

(1) $d=1$时 结论成立

假设 $d = l-1$时 也成立 ($l \geq 2$)

$d = l$ 时 $y_{j_1 \cdots j_l} = \sum_{j_l=0}^{n_l-1} \omega_{n_l}^{j_l k_l} y_{j_1 \cdots j_{l-1}}$

$y_{j_1 \cdots j_{l-1}}$ 即为前 $l-1$ 维的输出 由归纳假设 得证.

(2) 由(*)式 用 $1', 2', \cdots l'$ 代替(1)中 $1, 2, \cdots l$

表示 $1, \cdots l$ 的一个重排, 与(1)同理可证

(3) ① 1维情况显然成立

② 假设 k 时成立 即 $\forall n_1 \cdots n_k$, $n = n_1 \cdots n_k$ 复杂度为

$$T(n_1 \cdots n_k) = \frac{n}{n_1}\lg\frac{n}{n_1} + \frac{n}{n_1 n_2}\lg\frac{n}{n_1 n_2} + \cdots + 1\lg 1 = O(n_1 \cdots n_k \lg n_1 \cdots n_k)$$

则

$$T(n_1 \cdots n_{k+1}) = \frac{n}{n_1}\lg\frac{n}{n_1} + \cdots + \frac{n}{n_1 \cdots n_{k+1}}\lg\frac{n}{n_1 \cdots n_{k+1}} + \frac{n}{n_1 \cdots n_k}\lg\frac{n}{n_1 \cdots n_k}$$

$$\leq C(n_1 \cdots n_k \lg n_1 \cdots n_k) + n_{k+1}\lg n_{k+1}$$

$$< C(n - n_{k+1})(\lg n - \lg n_{k+1}) + n_{k+1}\lg n_{k+1}$$

$$= Cn\lg n - Cn_{k+1}\lg n - Cn\lg n_{k+1} + (C+1)n_{k+1}\lg n_{k+1}$$

$$= Cn\lg n - Cn_{k+1}(\lg n - \lg n_{k+1}) - (Cn - 1)\lg n_{k+1}$$

$$< Cn\lg n \quad (C > 1 \text{ 时恒成立})$$

故 $T(n_1 \cdots n_{k+1}) = O(n\lg n)$ k+1 时也成立

由归纳假设，得证